TIME-SPACE LOWER BOUNDS FOR SATISFIABILITY AND RELATED PROBLEMS ON RANDOMIZED MACHINES

by

Scott Diehl

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2008

ABSTRACT

Computational complexity studies the resources required for computers to solve certain problems; deciding satisfiability of Boolean formulas is one of the most fundamental such problems. This thesis proves lower bounds on the time and memory space required to solve satisfiability and related problems on randomized machines.

We establish the first randomized time-space lower bounds for the problem Σ_ℓSAT of deciding the validity of quantified Boolean formulas with at most $\ell-1$ quantifier alternations. We show that for any integer $\ell \geq 2$ and constant $c < \ell$, there exists a positive constant d such that Σ_ℓSAT cannot be computed by randomized machines with two-sided error in time n^c and space n^d. This result also applies to other natural complete problems in the polynomial-time hierarchy, such as maximizing the number of literals that can be removed from a disjunctive normal form formula without changing its meaning.

As for $\ell = 1$, we argue that similar results for satisfiability or tautologies would follow from a simulation that swaps Arthur and Merlin in two-round Merlin-Arthur games at subquadratic cost. We prove that the latter requires non-black-box techniques: any Arthur-Merlin game requires time $\Omega(t^2)$ to black-box simulate Merlin-Arthur games running in time t. This proves that known simulations of the latter class by the former with quadratic overhead are tight within the black-box setting.

In the case of the tautologies problem, we obtain nontrivial time-space lower bounds for randomized machines with *one-sided error*, i.e., randomized machines that can only err on tautological formulas. In fact, we prove new results for *nondeterministic* machines solving tautologies, which relates to proof complexity and the NP versus coNP problem: for

every $c < \sqrt[3]{4} \approx 1.587$ there exists a positive d such that tautologies cannot be decided by nondeterministic machines in time n^c and space n^d.

We can do better in the one-sided error setting: we prove that for every constant $c < 2\cos(\pi/7) \approx 1.801$, there exists a positive constant d such that tautologies cannot be decided by randomized machines with one-sided error in time n^c and space n^d.

ACKNOWLEDGMENTS

There really isn't any way that I can thank my advisor, Dieter van Melkebeek, enough, but I'll try anyway. His enthusiasm for research is infectious; his drive for perfection is inspiring. I would say that both are infectious, but that would incorrectly imply that his work ethic can be passively transfered from advisor to student like a benevolent virus; such a connotation would undermine the vast quantities of time and patience required to instill even a crude approximation of Dieter's virtues in a mere mortal such as myself. I graduate knowing that I've had the privilege of learning from the best, even if I may not be the best learner.

I'd also like to thank Kevin Compton for being my teacher, mentor, and friend during my latter years at the University of Michigan. The guidance he gave was vital to my development as a student, not to mention getting into and choosing the right graduate school. I am grateful for the times that his support prevented me from being lost in the academic shuffle. He's truly an all-around awesome guy.

My time in Madison would not have been nearly as fun without the company of some great friends. I'd like to thank all of them for the great times I've had playing frisbee, going camping, drinking beer at the terrace, ice skating, polar plunging, watching our hockey team win a national championship, etc, etc. Most of all, I cherish my time with Erica, whether we're out causing hijinx or in watching teevee. After a rough day, I know I can count on her support and sunny demeanor to cheer me up.

I'm indebted to my collaborators, classmates, and officemates for many fruitful discussions. The latter two groups were also great sources of commiseration during end-of-semester crunches, qualifying exams, and paper deadlines. I'd also like to thank my thesis committee

of Eric Bach, Shuchi Chawla, Somesh Jha, Steffen Lempp, Rod Downey, and Lance Fortnow. The work appearing in this thesis was supported by NSF Career Award CCR-0133693 and the Cisco Systems Distinguished Graduate Fellowship.

Finally, the biggest thanks go out to my family, both extended and nuclear, whom I love very deeply. I don't think there are many people that look forward to a week-long family reunion as much as I do. With apologies to everyone else, I know that I have the best parents and sister out of the billions of families on earth. I dedicate this thesis to my parents, the original Dr. (Beske-)Diehls, for their endless love, support, and trust. Thanks, Mom and Dad!

TABLE OF CONTENTS

Chapter 1

Introduction

Edwin sits in the campus library, trying to work on his last project for the semester. He's not having much luck, though. The record-setting snowfall from the last few months is finally melting off of the ground, people are shedding jackets for t-shirts, and the smell of spring from outside still lingers in his nose. He can't focus, so he does what all undergraduates do in his situation: procrastinate.

He opens his laptop, logs on to the social-networking site-of-the-moment, and clicks on his friend Imoen's page. He glances at her newest profile picture (a flattering helicopter shot, of course) before clicking on one of her friends, then on one of that person's friends, and so on without much aim. His mind wanders as he's doing this, and before long, he realizes that he's ended clicking back to his own page.

"Huh, I clicked on ten different pages without repeats and ended up back at my own page. I wonder if I can do the same thing for one hundred pages," he idly ponders. Intrigued by this thought, he tries a few different routes to see if they solve his one-hundred-click quandary. The first time he always chooses the alphabetically first friend not yet visited, but this route ends on Branwen's page. Next, he tries always choosing the person with the most self-aggrandizing profile picture, but this route ends on Skie's page.

After a few more unsuccessful strategies, he begins to think that maybe he won't find a solution just by following a simple rule: maybe the secret is to choose his moves at random. Although this idea sounds promising, several trials all end up on non-Edwin pages— fail!

Edwin stops to take stock of his situation: how many routes would he have to try before concluding that none of them work? Like most students, he has hundreds of social-network friends; each of these friends, in turn, has hundreds of friends, and so on. After one hundred clicks, the number of possibilities accumulates to something like 100^{100} friend-routes to explore— that's a one followed by two hundred zeroes! Edwin blinks a couple of times, logs out of the social-networking site, and starts working on something slightly less intractable: his project.

Edwin's quandary actually shares a property with many computational problems throughout the sciences: it is easy to check whether or not a potential solution is correct, but actually discovering one is like looking for a needle in a haystack the size of the entire universe. But does it have to be that difficult? If we are a little bit more clever, can we find solutions much more efficiently than by exhaustive search? After all, the entire history of algorithm design teaches us that a large search space does not necessarily indicate difficulty. Clever algorithms often take advantage of structure underlying the problems on hand to rapidly find solutions. However, some problems, including Edwin's, have long resisted attempts to engineer such an efficient algorithm.

The infamous P versus NP question asks if the ability to efficiently verify a solution always implies the ability to efficiently discover a solution. The poster-child representative of this question is *satisfiability*, the problem of deciding if a Boolean formula has at least one satisfying assignment. This thesis studies requirements on the computational resources — time and memory space — needed to solve satisfiability even when allowed the power of randomness.

1.1 Satisfiability

Satisfiability is a problem of central importance in computer science. Its applications throughout the field are numerous and fundamental. As the seminal NP-complete problem, satisfiability is the most prominent representative of a class that captures the complexity

of a large number of important problems from graph theory, circuit optimization, artificial intelligence, network design, databases, game theory, cryptography, logic, and so on. Understanding the computational resources required to solve satisfiability yields the same knowledge for these scores of other problems.

It is widely conjectured that any algorithm for satisfiability requires exponential time; in other words, that the answer to the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ problem is strongly negative. Confirming this belief involves proving lower bounds stating that no algorithm can solve satisfiability within a certain amount of time; we wish to make these time bounds as large as possible. A trivial linear time bound follows by observing that, in the worst case, the least that an algorithm must do to determine satisfiability is look at the entire input formula. Surprisingly, this bound stands as the best we can currently achieve up to constant factors. Several decades of effort have yielded no success in proving super-linear time lower bounds for satisfiability on general machine models. This underscores an enormous gulf between belief — an exponential time lower bound — and knowledge — a linear time lower bound — and stands as a testament to the difficulty of proving time lower bounds.

The situation improves if we focus on a large class of important algorithms: those that are restricted to use a small amount of workspace. A decade ago, Fortnow [For00] established nontrivial lower bounds for space-bounded algorithms solving satisfiability. Fortnow's technique has its roots in earlier work by Kannan [Kan84] and has been further developed since then [LV99, FvM00, FLvMV05, Wil06, Wil07b]. For algorithms that use only a subpolynomial amount of space, the current best time lower bound for satisfiability is $n^{2\cos(\pi/7)-o(1)} \approx n^{1.801}$.

However, these lower bounds apply only to deterministic algorithms. In practice, we would be just as happy with a randomized algorithm for satisfiability provided that it has small probability of error, so our lower bounds should say something about these algorithms as well. *This thesis presents results towards proving lower bounds for satisfiability on randomized machines with two-sided error bounded away from 1/2.*

- Section 1.2 describes time-space lower bounds on randomized machines for problems that are related to, but harder than, satisfiability, namely quantified Boolean formulas with few quantifier alternations.

- The task of extending these results to satisfiability is connected to that of swapping turns in a class of interactive protocols known as Arthur-Merlin games. Section 1.3 explores the latter setting and states a property needed for techniques to realize a lower bound for satisfiability on randomized machines.

- Finally, Section 1.4 states lower bounds for problems whose hardness is closer to satisfiability than those in Section 1.2, but on more restricted models. In particular, we prove nontrivial lower bounds for the complement of satisfiability, tautologies, on randomized algorithms that can only err on tautological formulas.

1.2 Lower Bounds for the Second Level and Higher

Randomized algorithmic techniques have been employed throughout computer science with great success. Some novel applications include finding primes for the RSA cryptosystem, polynomial identity testing, approximation algorithms for NP-hard optimization problems, secure multiparty computation, and powerful search heuristics such as simulated annealing (cf. [MR97]). Taking these applications as evidence for the power of randomness, it is natural to wonder if we could harness randomness to construct an efficient algorithm for satisfiability. Most experts believe that the answer is no: satisfiability requires exponential time even on randomized machines that can err (with small probability) on all inputs, both satisfiable and unsatisfiable; we refer to this general case of error behavior as *two-sided error*.

Lower bounds for randomized machines become even more difficult to prove than in the deterministic setting due to the addition of randomness. No nontrivial lower bounds for satisfiability have been established on randomized random-access machines with two-sided error even when the workspace of such machines is restricted to be logarithmic. In fact, prior

to the work contained within, this task remained open even for some important generalizations of satisfiability— here we are referring to the problem Σ_ℓSAT that augments a Boolean formula with ℓ alternating quantifiers. Recall that satisfiability can be characterized by one existential quantifier — does there *exist* a satisfying assignment to a formula. Similarly, Σ_ℓSAT can be characterized by ℓ quantifiers, beginning with an existential — does there exist an assignment to some variables such that for all assignments to another set, and so on for $\ell - 1$ quantifier alternations, such that a formula is satisfied. In this manner, Σ_ℓSAT is precisely the satisfiability problem for $\ell = 1$, but becomes more difficult and captures richer complexity classes for larger values of ℓ.

Time-space lower bounds for Σ_ℓSAT have been previously considered for deterministic machines. For example, Fortnow and Van Melkebeek [FvM00, FLvMV05] show an $n^{\ell - o(1)}$ time lower bound for deterministic machines solving Σ_ℓSAT in subpolynomial space, for $\ell \geq 2$. We match these bounds for *randomized* machines running in subpolynomial space; a more careful analysis yields lower bounds for sublinear polynomial space bounds.

Theorem 1.1. *For any integer $\ell \geq 2$ and constant $c < \ell$, there exists a positive constant d such that Σ_ℓSAT cannot be solved by randomized random-access machines with two-sided error running in time n^c and space n^d. Moreover, d approaches $1/2$ from below as c approaches 1 from above for $\ell = 2$, and d approaches 1 from below as c approaches 1 from above for $\ell \geq 3$.*

The randomized machines in Theorem 1.1 and in the rest of this thesis refer to the natural coin flip model, in which the machine has one-way read-only access to a tape with random bits. Viola [Vio07] extended Theorem 1.1 to the model in which the randomized machines have two-way access to the random bit tape, although his approach yields weaker lower bounds and only works for $\ell \geq 3$.

Recall that the polynomial-time hierarchy is the hierarchy of complexity classes extending P and NP by allowing acceptance conditions with any constant number of quantifier alternations. Alternatively, we can define these classes as the set of languages recognized by polynomial-time Turing machines that alternate between existential and universal states a

constant number of times. For any integer $\ell \geq 1$, Σ_ℓSAT is complete for the ℓ^{th} existential level of the polynomial-time hierarchy. As such, Theorem 1.1 establishes the first polynomial-strength time-space lower bounds for natural complete problems in the polynomial-time hierarchy on two-sided error randomized machines. By time-space lower bounds of "polynomial strength" we mean time lower bounds of the form $\Omega(n^c)$ for some constant $c > 1$ under non-trivial space upper bounds. Previous works establish randomized time-space lower bounds but they either consider problems believed to be harder than the polynomial-time hierarchy, or involve time lower bounds that are only slightly super-linear. Allender et al.'s [AKR$^+$01] time-space lower bounds for problems in the counting hierarchy on probabilistic machines with unbounded error fall within the first category; Beame et al.'s [BSSV03] nonuniform time-space lower bound for a binary quadratic form problem in P falls within the second category.

Time-space lower bounds for satisfiability carry over with no loss in quantitative strength to virtually every known NP-complete problem, such as subset sum, traveling salesperson, vertex cover, etc. This follows by virtue of extremely efficient reductions from satisfiability to these problems. However, the situation is different in higher levels of the polynomial-time hierarchy. For example, Umans [Uma01] showed that several variants of minimizing the size of a disjunctive normal form (DNF) formula are complete for the second level of the polynomial-time hierarchy. However, the reductions he gave from Σ_2SAT run in at least quartic time, which is too inefficient to allow Theorem 1.1 to extend to DNF minimization. We improve on the known reductions to build a very time- and space-efficient reduction from Σ_2SAT to one of the versions of DNF minimization. Recall that a DNF formula is a disjunct of *terms*, each of which is a conjunct of *literals*— a variable or its complement. The version of DNF minimization that we consider allows minimization only by removing occurrences of literals— in other words, we wish to maximize the number of literal occurrences that can be removed from a DNF formula without changing its meaning. In practice, this corresponds to removing connections from a programmable logic array (PLA); hence, we refer to this problem as PLA-simplification. Our reduction allows the results of Theorem 1.1 to carry

over to PLA-simplification, establishing the first nontrivial time-space lower bounds for a practical circuit minimization problem.

Corollary 1.2. *For any constant $c < 2$, there exists a positive constant d such that* PLA-simplification *cannot be solved by randomized random-access machines with two-sided error running in time n^c and space n^d. Moreover, d approaches $1/2$ from below as c approaches 1 from above.*

Because of the tight connection between Σ_ℓSAT and linear time in the ℓ^{th} level of the polynomial-time hierarchy, Theorem 1.1 can be stated equivalently as a time-space lower bound for simulations of the latter complexity class on randomized machines with two-sided error. In fact, we can strengthen Theorem 1.1 to establish time-space lower bounds for simulations of linear-time alternating machines that only use space n^a for constant $a \leq 1$.

Theorem 1.3. *For any integer $\ell \geq 2$ and any constants $a \leq 1$ and $c < 1 + (\ell - 1)a$, there exists a positive constant d such that linear-time alternating machines using space n^a and $\ell - 1$ alternations cannot be simulated by randomized random-access machines with two-sided error running in time n^c and space n^d. Moreover, d approaches $a/2$ from below as c approaches 1 from above for $\ell = 2$, and d approaches a from below as c approaches 1 from above for $\ell \geq 3$.*

Note that when $a = 1$, the space restriction on the alternating machines disappears and Theorem 1.3 becomes Theorem 1.1.

The $\ell \geq 2$ restriction in Theorem 1.1 implies that it does not give any bounds for the first level of the polynomial-time hierarchy, i.e., for satisfiability or its complement, tautologies, the problem of deciding if a Boolean formula is true under all assignments. At first glance, this situation may seem strange. On the one hand, techniques from derandomization allow space-bounded randomized machines to be transformed into deterministic ones. In particular, assuming that we can solve satisfiability on a randomized machine in logarithmic space and time n^c, Nisan's deterministic simulation [Nis94] yields a deterministic algorithm for satisfiability that runs in polylogarithmic space and polynomial time. However, even

for $c = 1$, the degree of the latter polynomial is far too large for this simulation to yield a contradiction with known time-space lower bounds for deterministic machines. On the other hand, lower bounds for Σ_2SAT often yield lower bounds for satisfiability. Indeed, in the *deterministic* setting, we know that a time-n^c algorithm for satisfiability yields a time-n^{c^2} algorithm for Σ_2SAT; thus, a lower bound of n^2 for Σ_2SAT, approximately matching that in Theorem 1.1, implies one of $n^{\sqrt{2}}$ for satisfiability. However, this is not necessarily the case when we consider *randomized* algorithms. To identify the main culprit, we launch an investigation into the field of interactive protocols.

1.3 Swapping Arthur and Merlin

Interactive protocols extend the traditional, static notion of a proof by allowing the verifier to enter into a conversation with the prover, wherein the goal is to convince the computationally limited verifier in a probabilistic sense. Traditional polynomial-time proof systems (with a deterministic verifier) capture the class NP, whereas interactive protocols capture the presumably larger class of languages recognizable in polynomial space [LFKN92, Sha92].

Of particular interest is the subclass of (public-coin) protocols in which the number of communication rounds between the prover and the verifier is bounded by a constant: we know such protocols as *Arthur-Merlin games*. This name comes about by envisioning the prover as the omnipotent (nondeterministic) magician Merlin and the verifier as the impatient (polynomial-time) King Arthur. Merlin tries to teach Arthur about his kingdom, but Arthur does not quite trust Merlin's advice. To restore his confidence, Arthur tosses coins to add unpredictability to his questions in such a way that he discovers any deception with high probability. Merlin's magical powers allow him to discover the outcome of these coin tosses even if Arthur tries to keep them secret, so Arthur must be clever enough to ask his questions in a way that Merlin does not gain any advantage from this knowledge. The latter condition represents the *public-coin* setting of interactive protocols, where the verifier reveals all of his coin tosses to the prover. Our results refer to this public-coin setting.

A two-round Arthur-Merlin game involves each party sending a single message and deciding on the outcome by an underlying polynomial-time predicate acting on the input and these messages. Graph nonisomorphism, which is not known to be in NP, has a two-round Arthur-Merlin game [GS89, GMW91]. In fact, any language that has an Arthur-Merlin game with any constant number of rounds can be decided by an Arthur-Merlin game with only two rounds [Bab85, BM88]. Thus, there are two classes of languages associated with Arthur-Merlin games, determined by the order in which the parties act: those recognized by two-round games where Merlin acts first, MA, and those recognized by two-round games where Arthur acts first, AM.

We know that the latter type of game is at least as powerful as the former: any MA-game can be transformed into an AM-game recognizing the same language. Thus, we can recognize at least as many languages when Arthur goes first as when Merlin goes first. This transformation does come at a cost, though: we must allow for some polynomial factor more time to verify Merlin's claim in the resulting AM-game than in the original MA-game. For example, in Babai's transformation [Bab85], one first reduces the error probability of the MA-game to exponentially small (by taking the majority vote of parallel trials) and then applies a union bound to show that, since the probability of accepting an invalid claim is so small, switching the parties' order does not give Merlin an unfair advantage. Since the error reduction necessitates verifying a number of trials of the MA-game that is linear in the number of bits sent by Merlin, this transformation incurs a quadratic time overhead in general.

Herein lies the culprit of our inability to extend Theorem 1.1 by transforming a randomized algorithm for satisfiability into one for Σ_2SAT: the arguments known to carry out such a translation utilize a simulation of MA by AM, such as the one outlined above, that comes at the cost of squaring the running time. In this manner, a randomized algorithm for satisfiability running in time n^c yields one for Σ_2SAT running in time n^{2c^2}; thus, an n^2 lower bound for Σ_2SAT gives only the trivial linear lower bound for satisfiability. To obtain a nontrivial lower bound for satisfiability on randomized machines by this line of argument,

we require either a stronger lower bound for Σ_2SAT or a subquadratic simulation of MA by AM.

We investigate the possibility of the latter path, and prove that *non-black-box* techniques are required to establish such a simulation.

Theorem 1.4. *For any time function t, there is no black-box simulation of* MA*-games running in time t by* AM*-games running in time $o(t^2)$.*

Informally, a simulation is black-box if it decides the same language as the MA-game by running the game's underlying predicate on a number of inputs; the key point is that it ignores the computational details of *how* the predicate arrives at its outputs. Babai's simulation of MA by AM is indeed black-box, since it makes its decision by the majority vote of a linear number of trials of the MA-game. The same holds for all other known simulations, such as those using pseudorandom generators [NW94] and those using extractors [GZ97]. Thus, Theorem 1.4 implies that those with a quadratic overhead are optimal within this setting; no tweaking or optimization of known techniques — even by improvements to extractor or pseudorandom generator parameters — can possibly yield better performance. This directs us to look for new, non-black-box techniques to establish nontrivial time-space lower bounds for satisfiability on randomized machines via Theorem 1.1.

As far as we know, this result is the first to compare the running-time of MA-games to equivalent AM-games. Santha compared the power of MA and AM by showing that there is an oracle relative to which AM is not contained in MA [San90], but this result does not concern the overhead of the opposite inclusion. A lower bound for black-box samplers by Canetti et al. [CEG95] yields a corollary that the error reduction underlying the transformations of [Bab85] and [GZ97] cannot be done more efficiently, but does not say anything about those that do not use error reduction, such as [NW94]. Our result applies to *all* known techniques.

We point out that removing the black-box restriction in Theorem 1.4 would solve another long-standing open problem: an unrestricted version of Theorem 1.4 would imply a time hierarchy for both MA and AM. Currently there are no nontrivial time hierarchies for any

semantic class [vMP07]. For example, in the case of AM, recall that the known simulation of MA by AM guarantees that any language with a linear-time MA-game also has a quadratic-time AM-game. On the other hand, an unrestricted version of Theorem 1.4 would provide such an MA-language that has no subquadratic-time AM-game, witnessing a hierarchy for AM.

1.4 Lower Bounds Below the Second Level

Theorem 1.4 identifies a roadblock to achieving nontrivial time-space lower bounds for satisfiability on randomized machines. Although we have yet to surmount this obstacle, we can get closer to satisfiability than where Theorem 1.1 left us.

One approach is to derive lower bounds for classes that lie in between linear time in the first and second levels of the polynomial-time hierarchy, such as the class of languages recognized by linear-time MA-games with perfect completeness, i.e., MA-games in which Merlin can convince Arthur to accept a valid claim with probability one. We argue that Theorem 1.1 yields time-space lower bounds for simulations of such MA-games by randomized machines.

Corollary 1.5. *For any constant $c < \sqrt{2}$, there exists a positive constant d such that linear-time* MA*-games with perfect completeness cannot be simulated by randomized random-access machines with two-sided error running in time n^c and space n^d.*

Corollary 1.5 was independently discovered by Emanuele Viola and also by Thomas Watson [personal communication]. In fact, Watson later achieved a lower bound for any $c < \phi \approx 1.618$ by a direct argument [vM07].

Corollary 1.5 follows because a time-n^c randomized algorithm for linear-time MA-games yields a time-n^{c^2} randomized algorithm for linear time in the second level of the polynomial-time hierarchy; thus, a lower bound of $n^{2-o(1)}$ for the second level implies one of $n^{\sqrt{2}-o(1)}$ for the class of linear-time MA-games. The argument requires some additional technical steps to achieve the lower bound for MA-games with perfect completeness.

Another approach towards time-space lower bounds for problems in the first level of the polynomial-time hierarchy is to prove results for randomized algorithms that are somewhat restricted. One particularly interesting and natural restriction found "in the wild" is that the algorithms be *one-sided*, i.e., that they do not err on "no" instances. For example, Rabin's test for integer compositeness [Rab80] is one-sided.

This setting has been considered before, albeit in a broader context: Fortnow and Van Melkebeek previously showed a time lower bound of $n^{\sqrt{2}-o(1)}$ for the tautologies problem on subpolynomial-space nondeterministic machines [FvM00, FLvMV05]; since randomized machines with one-sided error are special cases of nondeterministic machines, these bounds also apply in the former setting.

More importantly, lower bounds for tautologies on nondeterministic machines are interesting on their own merit, as they relate to proof complexity and the NP versus coNP problem. Typical results in proof complexity deal with specific types of nondeterministic machines that implement well-known proof systems, such as resolution. They establish strong (superpolynomial or even exponential) lower bounds for the size of any proof of certain families of tautologies within that system, and thus for the running time of the corresponding nondeterministic machine deciding tautologies [BP01]. A more generic approach to the NP versus coNP problem follows the approach described above: proving lower bounds for tautologies on nondeterministic machines. These results apply to *any* proof system. As an aside to our quest for randomized lower bounds, we improve the lower-bound exponent for tautologies on nondeterministic machines from $\sqrt{2} \approx 1.414$ to $\sqrt[3]{4} \approx 1.587$.

Theorem 1.6. *For any constant $c < \sqrt[3]{4}$ there exists a positive constant d such that tautologies cannot be decided by nondeterministic random-access machines running in time n^c and space n^d.*

Theorem 1.6 yields the same lower bounds for one-sided error randomized machines as a special case. However, this reasoning ignores the extra structure provided by the bounded error of randomized machines. Using ideas from the proof of Theorem 1.1, we take advantage

of this structure to derive better lower bounds for randomized machines with one-sided error, boosting the exponent further to $2\cos(\pi/7) \approx 1.801$.

Theorem 1.7. *For every constant $c < 2\cos(\pi/7) \approx 1.801$ there exists a positive constant d such that tautologies cannot be solved by randomized random-access machines with one-sided error running in time n^c and space n^d.*

Notice that Theorem 1.7 applies as a special case to deterministic machines. By the closure of deterministic classes under complement, Theorem 1.7 matches the deterministic time-space lower bounds for satisfiability mentioned earlier [Wil07b]. Our contribution consists of a generic technique that upgrades existing time-space lower bound proof techniques for deterministic machines to handle one-sided randomness.

1.5 Organization

The rest of this thesis is organized as follows.

- Chapter 2 discusses our notational and computational conventions. It also states robust completeness results for Σ_ℓSAT and PLA-simplification. The latter result yields the lower bound for PLA-simplification (Corollary 1.2) as a corollary to the one for Σ_2SAT (Theorem 1.1); it is currently unpublished.

- Chapter 3 gives an overview of the indirect diagonalization paradigm that we adapt to prove our time-space lower bounds.

- Chapter 4 discusses our contributions that build on existing time-space lower bound techniques in order to handle the randomized setting, and proves our lower bounds for Σ_ℓSAT on randomized machines (Theorem 1.1). A preliminary version of these results, which focused on achieving large time lower bounds at the small-space end of the spectrum, was presented at the 32nd International Colloquium on Automata, Languages,

and Programming (Lisbon, July 2005) [DvM05]. The full version extended these results to yield nontrivial lower bounds even at the large-space end of the spectrum; it was published in the SIAM Journal on Computing [DvM06].

- Chapter 5 concerns Arthur-Merlin games, proving our lower bounds for swapping Arthur and Merlin (Theorem 1.4) as well as our time-space lower bound for MA-games (Corollary 1.5). These results were presented at the 11th International Workshop on Randomization and Computation (Princeton, August 2007) [Die07].

- Chapter 6 proves our lower bounds for tautologies on nondeterministic machines (Theorem 1.6) and on randomized machines with one-sided error (Theorem 1.7). The former result is currently published as a University of Wisconsin technical report [DvMW07]. The latter result is a yet unpublished culmination of techniques from [DvM05, DvM06] and [Wil07b].

Chapter 2

Preliminaries

In this chapter we lay down the notational and computational conventions followed throughout this thesis. We then establish tight completeness results relating these computational models to the problems studied in this thesis.

2.1 Machine Model

Our lower bounds are robust with respect to the choice of machine model. For concreteness, we state and prove our results for a random-access machine model. This model is very powerful: for example, a deterministic random-access machine can simulate any deterministic machine model that we know of with at most a polylogarithmic time and space overhead. These efficient simulations allow our main results to carry over with little loss to other models.

The specific random-access machine model we use is somewhat nonstandard, so we provide a short discussion of its key points. The random-access machine model augments the standard sequential-access multitape Turing machine model by allowing some tapes to be *random-access*. The tape head movements on these tapes are no longer sequential, left-or-right moves; rather they are dictated by the contents of an *index tape* associated with each random-access tape: after writing an address on the index tape, the machine can relocate the random-access tape head to that location, erasing the contents of the index tape in the process. Such a step costs only one time unit; otherwise, time measurement is as usual. Measuring space usage is a little bit more tricky. As is standard, we count only the *auxiliary*

space used by the machine: the input tape is read-only and the output tape is one-way and write-only. We measure space usage for each tape by the largest index visited (assuming the tapes are infinite in just one direction). This convention is more natural than the alternative that measures space by number of tape cells modified. For example, a program that addresses a gigabyte of memory requires that the computer running it have at least that much memory, regardless of how much the program actually ends up modifying. However, measuring space usage by index has the unfortunate side effect of allowing for space usage that is exponential in the running time, contrary to our usual notion that space is bounded by running time. This pitfall can be avoided by use of appropriate memory-access data structures at only a polylogarithmic cost in running time; as such factors do not affect the quantitative strength of our main theorems, we assume throughout that space usage is bounded by time.

This machine model allows easy extensions to nondeterministic, alternating, randomized, and interactive protocol versions. The nondeterministic model is standard: we let the transition function become a relation and define acceptance by the existence of an computation path leading to an accept state. The alternating model is similar, except we label each state as existential or universal. The accepting condition for the machine can be characterized as a quantified statement over machine moves, expressing that these moves lead to an accept state. The statement has as many quantifier alternations as the machine has state type alternations: portions of the computation spent in existential states naturally translate to existential quantification of these moves, while universal states translate to universal quantification.

For randomized machines models, we follow the natural convention that random bits are presented on a one-way read-only worktape. If the machine wishes to re-read random bits, it must copy them down on a worktape at the cost of space. This contrasts the more powerful model that allows two-way access to the random tape. Except where stated otherwise, our results about randomized machines hold only for the former machine model. The standard error model for randomized machines is two-sided and bounded. In other words, the machine

can err on *all* inputs, both inside and outside the language, with a probability bounded away from 1/2 by a constant. We often discuss machines with *one-sided error*, which can only err on inputs that are in the language.

We form interactive protocols by a hybrid nondeterministic/randomized model, where the machine has states that are either existential (corresponding to the prover) or randomized (corresponding to the verifier). The machines must also satisfy a *completeness* condition that, on inputs in the language, the prover can convince the verifier to accept with high probability, and a *soundness* condition that, on inputs outside of the language, the verifier accepts with low probability no matter what the prover does.

Arthur-Merlin games are public-coin interactive protocols with a constant number of rounds (the number of *rounds* is one more than the number of alternations between state types). We say that Arthur goes first if the machine starts in a randomized state; otherwise, Merlin goes first. Our results mostly concern games with only two rounds, MA-games and AM-games.

- An MA-game decides a language L if for all strings $x \in L$, there is a proof Merlin can send to Arthur that causes him to accept with high probability; on the other hand, for $x \notin L$, any proof sent by Merlin is rejected by Arthur with high probability.

- An AM-game decides a language L if for all strings $x \in L$, the probability is high that Arthur's coin tosses sent to Merlin allow the latter party to respond with a valid proof; on the other hand, this probability is low for all $x \notin L$.

We refer to Section 5.1.1 for more formal definitions.

The standard error model for Arthur-Merlin games is two-sided and bounded, i.e., the completeness and soundness conditions hold for probabilities bounded away from each other by a constant. The exceptions are games with *perfect completeness*, where an honest Merlin causes acceptance with probability one for inputs in the language (this contrasts with the "one-sided" setting of randomized machines where the error is on the membership side).

We require that the time and space bounds of machines are constructible functions from natural numbers to natural numbers which are at least logarithmic, and refer to them as time and space functions. We often discuss *subpolynomial* space functions, which refer to space functions in $n^{o(1)}$. Many of our results ultimately apply to computations with polynomial time and space bounds, which certainly meet the required constructibility conditions. Note that machines running in sublinear time or sublogarithmic space trivially cannot solve problems like satisfiability where the answer can depend on the entire input.

2.2 Complexity Classes

A complexity class is defined as the class of languages decided by a type of machine running within certain resource bounds. Much of the notation we use is standard [BDG95, Pap94], but we recap it here for completeness. We consider many different machine types, abbreviated as follows:

Definition 2.1. *Our notation for complexity classes refers to machines of type X, where X is is an abbreviation for one of the following machine types:*

- $X = D$*: Deterministic machines.*
- $X = N$*: Nondeterministic machines.*
- $X = \Sigma_\ell$*: Alternating machines making at most* $\ell - 1$ *alternations and starting in an existential state.*
- $X = \Pi_\ell$*: Alternating machines making at most* $\ell - 1$ *alternations and starting in a universal state.*
- $X = BPP$*: Randomized machines with two-sided error of at most 1/3.*
- $X = RP$*: Randomized machines with one-sided error — on the membership side — of at most 1/2.*
- $X = MA$*: Two-round Arthur-Merlin games where Merlin goes first, with two-sided error at most 1/3.*

- $X = \mathrm{MA}_1$*: Same as MA except with perfect completeness. In other words, for* $x \in L$ *there is a proof that causes acceptance with probability 1.*
- $X = AM$*: Two-round Arthur-Merlin games where Arthur goes first, with two-sided error at most 1/3.*
- $X = \mathrm{AM}_1$*: Same as AM except with perfect completeness. In other words, for* $x \in L$*,* all *of Arthur's coin tosses can be answered by Merlin with a valid proof.*

For any X as in Definition 2.1, time function t, and space function s we define XTIME(t) to be the class of languages decided by machines of type X that run in time $O(t)$, and XTISP(t, s) to be the class of of languages decided by machines of type X that run in simultaneous time $O(t)$ and space $O(s)$. We prefix "co" to these to represent the complementary classes. The default model is understood to be deterministic if we omit X. We often use the same notation to refer to classes of machines rather than classes of languages.

We often replace TIME in the above notation by P or a superscript p when we allow the running time to be any polynomial. In this manner, we define the classes

$$\mathrm{P} = \mathrm{TIME}(\mathrm{poly}(n)) \text{ and } \mathrm{BPP} = \mathrm{BPTIME}(\mathrm{poly(n)}).$$

The first level of the polynomial-time hierarchy contains the classes

$$\mathrm{NP} = \mathrm{NTIME}(\mathrm{poly}(n)) \text{ and } \mathrm{coNP} = \mathrm{coNTIME}(\mathrm{poly}(n)).$$

More generally, the ℓ^{th} level of the polynomial-time hierarchy consists of

$$\Sigma_\ell^p = \Sigma_\ell \mathrm{TIME}(\mathrm{poly}(n)) \text{ and } \Pi_\ell^p = \Pi_\ell \mathrm{TIME}(\mathrm{poly}(n)).$$

We often refer to the linear-time hierarchy, which corresponds to alternating machines with linear, rather than polynomial, running times.

Many of our arguments involve alternating computations in which the numbers of bits guessed at each stage are bounded by explicitly given small functions. We use the following notation to describe such computations:

Definition 2.2. *Given a complexity class $\mathcal{C}$ and a function f, we inductively define the class $\exists^f\mathcal{C}$ to be the set of languages that can be described as*

$$\{x|\exists y \in \{0,1\}^{O(f(|x|))} P(x,y)\},$$

where P is a predicate accepting a language in the class $\mathcal{C}$ when its complexity is measured in terms of $|x|$ (not $|x|+|y|$). We analogously define $\forall^f\mathcal{C}$.

For example, $\exists^f\mathrm{DTIME}(n)$ and $\forall^f\mathrm{DTIME}(n)$ are subsets of NP and coNP for $f(n) = n^{O(1)}$. The requirement that the complexity of P be measured in terms of $|x|$ allows us to express the running times in terms of the original input length, which allows for much cleaner accounting in many of our arguments.

A subtlety arises when we consider space-bounded classes $\mathcal{C}$. Computations corresponding to $\exists^f\mathcal{C}$ and $\forall^f\mathcal{C}$ explicitly write down their guess bits y and then run a space-bounded machine on the combined input consisting of the original input x and the guess bits y. Thus, the space-bounded machine effectively has two-way access to the guess bits y. For example, although machines corresponding to $\exists^n\mathrm{DTISP}(n, n^{o(1)})$ and to $\mathrm{NTISP}(n, n^{o(1)})$ both use only a subpolynomial amount of space to verify their guesses, they do not necessarily have the same computational power. This is because the former machines have two-way access to the guess bits written on a separate tape that does not count towards its space bound, whereas the latter machines only have one-way access to these bits and do not have enough space to write them down on their work tape.

2.3 Robust Completeness in the Polynomial-Time Hierarchy

All of the known time-space lower bounds for satisfiability and its generalizations to higher levels hinge on a tight connection between these problems and the class of languages recognized by linear-time alternating machines, $\Sigma_\ell\mathrm{TIME}(n)$. The Cook-Levin Theorem, the seminal result showing that satisfiability is NP-complete, can be interpreted as saying that satisfiability captures the time complexity of all of NP up to polynomial factors; this statement generalizes to $\Sigma_\ell\mathrm{SAT}$ and the ℓ^{th} level of the polynomial-time hierarchy, Σ_ℓ^p. We

make use of a stronger versions of Cook-Levin that show that Σ_ℓSAT *tightly* captures — up to polylogarithmic factors — the simultaneous time *and* space complexity of the ℓ^{th} level of the linear-time hierarchy, $\Sigma_\ell\text{TIME}(n)$.

2.3.1 Robust Completeness of Σ_ℓSAT

At the first level of the polynomial-time hierarchy, we know that satisfiability can be solved in nondeterministic quasi-linear time, so time-space lower bounds for satisfiability imply the same lower bounds for nondeterministic linear time, $\text{NTIME}(n)$, up to polylogarithmic factors. Conversely, various strengthenings [FvM00, FLvMV05, vM07] give a reduction from $\text{NTIME}(n)$ to satisfiability that is efficient in both time and space, showing that if satisfiability can be solved in time n^c and space n^d, then $\text{NTIME}(n)$ can be solved in time $n^c \operatorname{polylog}(n)$ and space $n^d \operatorname{polylog}(n)$. Thus, time-space lower bounds for $\text{NTIME}(n)$ and for satisfiability are equivalent up to polylogarithmic factors.

At higher levels of the polynomial-time hierarchy, we know that Σ_ℓSAT can be solved in quasi-linear time on a machine that makes $\ell - 1$ alternations, so time-space lower bounds for Σ_ℓSAT imply the same lower bounds for $\Sigma_\ell\text{TIME}(n)$ up to polylogarithmic factors. Conversely, just as the Cook-Levin theorem generalizes from NP to higher levels of the polynomial-time hierarchy to show that Σ_ℓSAT is complete for Σ_ℓ^p, the stronger reductions generalize to give time- and space-efficient reductions from $\Sigma_\ell\text{TIME}(n)$ to Σ_ℓSAT.

Theorem 2.3. *For any integer $\ell \geq 1$ and positive reals c and d, if*

$$\Sigma_\ell\text{SAT} \in \text{XTISP}(n^c, n^d),$$

then

$$\Sigma_\ell\text{TIME}(n) \subseteq \text{XTISP}(n^c \operatorname{polylog}(n), n^d \operatorname{polylog}(n)),$$

where X is any machine type described in Definition 2.1.

This establishes an equivalence of time-space lower bounds for $\Sigma_\ell\text{TIME}(n)$ and Σ_ℓSAT up to polylogarithmic factors. In particular, polynomial-strength time-space lower bounds for $\Sigma_\ell\text{TIME}(n)$ yield essentially the same lower bounds for Σ_ℓSAT.

Corollary 2.4. *For any integer $\ell \geq 1$ and positive reals c and d, if*

$$\Sigma_\ell \mathrm{TIME}(n) \not\subseteq \mathrm{XTISP}(n^c, n^d),$$

then for any reals $c' < c$ and $d' < d$,

$$\Sigma_\ell \mathrm{SAT} \notin \mathrm{XTISP}(n^{c'}, n^{d'}),$$

where X is any machine type described in Definition 2.1.

The same holds if we replace Σ_ℓSAT and Σ_ℓTIME(n) by their complements; in particular, for $\ell = 1$, the connection is between tautologies and coNTIME(n). In all, these connections allow us to prove our time-space lower bounds for Σ_ℓSAT and tautologies by instead proving lower bounds for the corresponding linear-time class.

We include a proof sketch for Theorem 2.3 for completeness.

Proof sketch of Theorem 2.3. Let L be a language decided by a random-access linear-time alternating Turing machine that makes $\ell - 1$ alternations, beginning in an existential stage. The acceptance condition for $x \in L$ can be written as

$$(\exists y_1 \in \{0,1\}^{O(n)})(\forall y_2 \in \{0,1\}^{O(n)}) \ldots (Q y_{\ell-1} \in \{0,1\}^{O(n)}) R(x, y_1, y_2, \ldots, y_{\ell-1}),$$

where $Q = \forall$ and R is a linear-time nondeterministic machine if ℓ is odd; otherwise $Q = \exists$ and R is a linear-time conondeterministic machine. Let $P = R$ if ℓ is odd and $P = \neg R$ otherwise, so that P is nondeterministic regardless of ℓ. All that remains to represent the acceptance condition of L as a quantified Boolean formula is to reduce P to satisfiability. The original Cook-Levin reduction produces a formula of quadratic size, which is too large for our purposes. Cook [Coo88] showed how to leverage the oblivious simulations of Hennie and Stearns [HS66] to obtain a formula of quasilinear size in the case of multitape machines. Robson [Rob91] built on this technique to obtain the reduction for random-access machines. Alternatively, Van Melkebeek [vM07] formulated a simpler approach using sorting networks— as sorting networks are inherently oblivious, this replaces the need for the Hennie-Stearns simulation. The net effect of either approach is that we can construct a formula φ depending only on P and n such that φ:

- has size $O(n \operatorname{polylog}(n))$, where each bit can be constructed in time $O(\operatorname{polylog}(n))$ and space $O(\log(n))$,
- involves the bits of $x, y_1, \ldots, y_{\ell-1}$ input to P as well as $O(n \operatorname{polylog}(n))$ additional Boolean variables z, and
- is satisfiable in z on input $x, y_1, \ldots, y_{\ell-1}$ if and only if P accepts $x, y_1, \ldots, y_{\ell-1}$.

Defining $\varphi' \doteq \varphi$ if ℓ is odd and $\varphi' \doteq \neg\varphi$ otherwise, this shows that the Σ_ℓ-formula

$$\psi \doteq \exists y_1 \forall y_2 \ldots Q y_{\ell-1} \overline{Q} z \, \varphi',$$

is in Σ_ℓSAT if and only if $x \in L$. The size of ψ is only $O(n \log n)$ more than φ: the log factor is required to write down the bit indices of the quantified variables. The easy nature of these extensions to φ endows ψ with the same constructibility properties as φ.

Now, suppose that $\Sigma_\ell\text{SAT} \in \text{XTISP}(n^c, n^d)$ and that M is a machine of type X deciding Σ_ℓSAT in time $O(n^c)$ and space $O(n^d)$. Notice that we can decide L by running M on input ψ. However, computing ψ and writing down the result on a worktape requires too much space. Instead, when M needs a bit of ψ, the simulation computes this bit from scratch. As φ is of size $O(n \operatorname{polylog}(n))$, M runs for time $O(n^c \operatorname{polylog}(n))$ and space $O(n^d \operatorname{polylog}(n))$ on input ψ. Computing the bits of ψ on the fly adds a multiplicative overhead of $O(\operatorname{polylog}(n))$ to the time and an additive overhead of $O(\log n)$ to the space. This simulation is still of type X and decides L within the desired time and space bounds. □

2.3.2 Robust Completeness of Other Problems

There are reductions of similar time and space efficiency from satisfiability to virtually all known NP-complete problems. Thus, existing time-space lower bounds for satisfiability also apply to these problems. However, efficient reductions to Σ_ℓSAT are not known for some natural problems that are complete for higher levels of the polynomial-time hierarchy, i.e., Σ_ℓ^p for $\ell \geq 2$. One particularly interesting problem is that of disjunctive normal form (DNF) formula minimization, MIN-DNF.

Umans [Uma01] showed that MIN-DNF and several of its variants are complete for the second level of the polynomial-time hierarchy. Unfortunately, the reductions he gives from Σ_2SAT to these problems are rather rather inefficient; in particular, the reduction to MIN-DNF blows up the instance size by the fourth power. This reduction does not establish the tight relationship between Σ_2SAT and MIN-DNF necessary to transfer lower bounds for the former to the latter.

We improve the efficiency of the reduction for a natural version of MIN-DNF that remains Σ_2-complete. Recall that a DNF formula is the disjunction of *terms*, each of which is the conjunction of *literals*— variables or their complements. The version of MIN-DNF that we consider allows minimization only by removing literals occurrences from the input formula; we refer to this variant as PLA-simplification. Formally, this problem is defined as follows:

Definition 2.5. *The* PLA-simplification *problem takes as input a DNF formula ψ on m variables and an integer k and asks whether or not there is a subformula ψ' of ψ with at most k literal occurrences such that $\psi(x) = \psi'(x)$ for all $x \in \{0,1\}^m$.*

Our reduction from Σ_2SAT to PLA-simplification produces an instance of the latter problem of size quasi-linear in the Σ_2SAT instance size. Furthermore, each bit can be computed on the fly in polylogarithmic time and logarithmic space. This results in a tight connection between the two problems similar to that exhibited in Theorem 2.3.

Theorem 2.6. *For any positive reals c and d, if*

$$\text{PLA-simplification} \in \text{XTISP}(n^c, n^d),$$

then

$$\Sigma_2\text{SAT} \in \text{XTISP}(n^c \ \text{polylog}(n), n^d \ \log(n)),$$

where X is any machine type described in Definition 2.1.

Theorem 2.6 shows that time-space lower bounds for PLA-simplification are equivalent to those for Σ_2SAT, along the same lines as Corollary 2.4; therefore, the lower bounds of

Corollary 1.2 for PLA-simplification follow directly from the time-space lower bounds for Σ_2SAT of Theorem 1.1.

The reduction from Σ_2SAT to PLA-simplification works as follows on input

$$(\exists x \in \{0,1\}^n)(\forall y \in \{0,1\}^n)\varphi(x,y),$$

where we assume without loss of generality that φ is a DNF. The reduction converts φ into another DNF ψ so that every term but one, denoted by T, must appear in its entirety in any equivalent DNF. Thus, the only way to arrive at an equivalent DNF by literal removal — as allowed for PLA-simplification — is to remove literals from T. The goal is to design ψ and T so that valid removal of n literals from T encodes a satisfying assignment to the existential variables of the Σ_2SAT instance, and conversely, that the existence of a valid assignment implies that n literals can be removed from T without changing the function computed by ψ.

Proof of Theorem 2.6. Consider an instance

$$(\exists x_1, \cdots, x_n)(\forall y_1, \cdots, y_n)\varphi(x_1, \cdots, x_n, y_1, \cdots, y_n) \tag{2.1}$$

of Σ_2SAT where φ is a DNF. The reduction tweaks φ to allow a critical term T as above in two steps: we first construct a circuit to evaluate φ in a slightly nonstandard way and then convert the circuit to a DNF formula by introducing auxiliary variables.

With this in mind, we envision the circuit C that evaluates an assignment to φ, and make some minor modifications. We have C take an input for each possible literal of x_i, say, $a_1, \cdots, a_n$ corresponding to positive literals, and $b_1, \cdots, b_n$ corresponding to negative literals (the y variables remain unchanged) and evaluate φ at the implied assignment. C handles a, b-assignments that do not correspond to valid x-assignments in a specific way to enable properties needed later for the correctness of our reduction. C is described in Figure 2.3.2.

We then transform C into a DNF formula by the (the complement of) Cook's transformation of a circuit to a CNF formula [Coo71]. This reduction replaces each circuit gate by a term enforcing that a new auxiliary variable z_i represents the gate's output. In this

$C(a, b, y)$ sequentially checks the following conditions:

(1) **Reject** if $a_i = b_i = 1$ for all i.

(2) **Reject** if $a_i = b_i = 0$ for some i.

(3) **Accept** if $a_i = b_i = 1$ for some i.

(4) **Evaluate** φ by substituting a_i where the literal x_i appears, and b_i where the literal $\overline{x_i}$ appears; the y variables are taken directly from the input to C. Output the result of this evaluation.

Figure 2.1 Description of the circuit C to evaluate φ.

manner, we achieve a DNF formula φ' such that $\varphi'(a, b, y, z)$ is a tautology under a setting to $a_1, \ldots, a_n, b_1, \ldots, b_n$ if and only if $C(a, b, y)$ accepts every input to $y_1, \ldots, y_n$ under the same a, b-setting.

We are now ready to form the desired instance of PLA-simplification by adding the critical term T. We also add new variables w that allow us to "turn on" and "turn off" individual terms of φ'. More concretely, if

$$\varphi' = \bigvee_{i=1}^{m'} t_i,$$

then our reduction produces the formula

$$\psi = \left(\bigvee_{i=1}^{m'} t_i w_i \right) \vee \underbrace{a_1 \cdots a_n b_1 \cdots b_n w_1 \cdots w_{m'}}_{T}.$$

We assume without loss of generality that t_i contain no redundant or contradictory literals, and $m' > 1$. The target size is

$$k = \#(\text{literal occurrences in } \psi) - n. \tag{2.2}$$

The purpose of the w variables is to make each term $t_i w_i$ necessary in any minimization of ψ. Consider an assignment that lets $w_i = 1$ and sets all other w variables to 0: all that remains of ψ is t_i. Since t_i is not contradictory, this term cannot be completely missing in

any equivalent subformula. Furthermore, any missing literals from t_i cause acceptance on more assignments than ψ, while missing w_i causes acceptance of assignments with $w_j = 0$ for all j, neither of which matches the behavior of ψ.

Therefore, any equivalent subformula of ψ is the result of removing literals from T, but could come by *completely* removing T. As we plan to have the remains of T encode a satisfying assignment to the Σ_2SAT instance, the latter option is problematic. This is where the check in step (1) of Figure 2.3.2 is crucial: it ensures that a nonempty subset of T is necessary in any equivalent subformula, and in fact, the literals $w_1, \ldots, w_{m'}$ must all appear. This follows because step (1) guarantees that φ' rejects some extension of the all-one assignment to the a and b variables. If we further set $w_i = 1$ for all i, we see that T is the only term of ψ that accepts such an assignment, so removal of T incorrectly rejects. On the other hand, if we set $w_i = 0$ for some i and $w_j = 1$ for $j \neq i$, then T is violated and ψ rejects; however, removing w_i from T causes this assignment to be incorrectly accepted. Thus, none of the $w_1, \ldots, w_{m'}$ literals from T can be removed to form an equivalent subformula, so any valid removal must come from the a and b variables in T.

Our check in step (2) of figure 2.3.2 ensures that valid removal of at least n of these variables encodes an assignment to the x variables of the Σ_2SAT instance. C rejects any assignment with $a_i = b_i = 0$ for some i, so φ' also rejects some extension of this assignment to the z variables. As T rejects any assignment with an a_i or b_i set to zero, ψ rejects such an assignment even when all the w_j variables are set to one. However, if we were to remove both a_i and b_i, T would incorrectly accept such an assignment. Therefore, any equivalent subformula of ψ that removes at least n variables contains a subset of T that has *exactly* one of a_i and b_i for every i. This naturally encodes an assignment A to the x variables by setting x_i to one if a_i is present and to zero if b_i is present.

With these observations in mind, we prove the correctness of the reduction: there is an equivalent subformula of ψ with at most k literal occurrences, where k is defined in (2.2), if and only if the Σ_2SAT instance (2.1) is positive. We start with the forward direction. Let A be the assignment to the x variables implied, as above, by the valid removal of

n literals from ψ to form ψ'. We claim that A satisfies the Σ_2SAT instance, i.e., that $\varphi(A_1, \ldots, A_n, y_1, \ldots, y_n)$ is a tautology. Notice that ψ' accepts any assignment where the literals remaining from T are set to one and the other a_i's and b_i's to zero, so ψ also accepts these assignments. Tracing back through our construction, we see that the satisfied term of ψ cannot be T, because some a_i's and b_i's are set to zero, so it must be a term from φ'. Thus, φ' is a tautology under this setting to the a_i's and b_i's, which implies that C accepts all y under this setting. Since exactly one of each a_i and b_i is set to one, checks (1–3) of Figure 2.3.2 are passed and its output must be the evaluation of φ under the assignment A. We conclude that φ accepts every assignment to the y variables under A, i.e., it is a tautology.

Conversely, suppose that (2.1) is a positive instance of Σ_2SAT. Let A be an assignment to the x variables such that $\varphi(A_1, \ldots, A_n, y_1, \ldots, y_n)$ is a tautology. Then we claim that the DNF

$$\chi = \left(\bigvee_{i=1}^{m'} t_i w_i \right) \vee T' w_1 \cdots w_{m'},$$

where T' contains the literal a_i if $A_i = 1$ and b_i otherwise for $1 \leq i \leq m'$, is equivalent to ψ. This follows because removing literals from a term of a DNF is a benign operation so long as no new assignments are accepted due to such a removal. To this end, we consider an assignment A' that satisfies $T' w_1 \cdots w_{m'}$ but not T, and show that $\psi(A') = 1$. Notice that A' cannot satisfy step (1) of Figure 2.3.2, since T accepts these assignments. It also cannot satisfy step (2), because acceptance of $T' w_1 \cdots w_{m'}$ implies that at least one of each a_i and b_i is set to one. So suppose that A' satisfies step (3): then we must have that $\psi(A') = 1$ since C accepts such assignments. Thus, all that remains is the case where A' does not satisfy steps (1–3). This implies that A' has exactly one of a_i and b_i set to one for each i, and furthermore, that the output of C is decided by evaluating φ. This evaluation substitutes the assignment implied by A' for the x variables, which is exactly the assignment A under which we assumed φ is a tautology. This causes φ', and therefore ψ, to accept A', finishing the proof that χ is an equivalent subformula of ψ of appropriate size.

We have shown that the reduction is correct, so it remains to verify its complexity. C can be constructed to have size $O(n)$, where n is the size of the Σ_2SAT instance (2.1); the

formula φ' has size linear in the size of C; ψ is larger by another constant factor. Describing ψ requires an additional $O(\log n)$ factor increase in size due to the necessity of writing down the indices of auxiliary variables, for a total description size of $O(n \log n)$. Furthermore, the easy form of the reduction allows each bit of the reduction to be computed on the fly in polylogarithmic time and logarithmic space. □

We point out that our proof diverges from that of Umans's in a couple ways. His more general reduction to MIN-DNF squares the formula size in two different steps: one that forces any minimization to occur within T, and another that ensures any minimization leaves behind a nontrivial subset of T. We avoid the squaring from the former step by adding only one new variable to each term of φ', as opposed to $m' - 1$, while still ensuring the desired property. The latter step involves making T necessary by having every term from φ' reject the all-one assignment to the a and b variables. This can be done while maintaining that the formula is a DNF by multiplying such a restriction into φ, increasing its size by a linear factor. One way to avoid the size increase is to enforce the restriction using auxiliary variables instead; this is what we accomplished by checking the all-one condition via a circuit (i.e., step (1) of Figure 2.3.2) and then converting back to a DNF. Unfortunately, auxiliary variables can be used to aid minimization of the DNF in unexpected ways; namely, by *adding* literals over these new variables to the special term T, we enable removal of many more literals than before and spoil our ability to extract a valid assignment. This type of minimization is not permissible by literal removal alone, which allows this idea to work in the restricted case of PLA-simplification; it remains open to establish tight completeness results for MIN-DNF.

Chapter 3

Indirect Diagonalization

The proofs of our time-space lower bounds follow a paradigm known as *indirect diagonalization*, which we describe in this chapter. Indirect diagonalization establishes a lower bound by contradiction: assuming the lower bound does not hold, we derive a sequence of progressively more unlikely inclusions of complexity classes until we reach one that contradicts a known diagonalization result. Kannan [Kan84] used the paradigm avant la lettre to investigate the relationship between deterministic linear time and nondeterministic linear time. This technique has been further developed by recent time-space lower bounds for satisfiability and problems higher up in the polynomial-time hierarchy [For00, LV99, FvM00, FLvMV05, Wil06, Vio07, Wil07b]. Allender et al. [AKR$^+$01] also employed it to establish time-space lower bounds for problems in the counting hierarchy.

As an example, consider Theorem 1.1 for $\ell = 2$, that Σ_2SAT cannot be solved by randomized machines running in time n^c and n^d for certain interesting values of c and d. By the tight completeness of Σ_2SAT for the second level of the polynomial-time hierarchy — Theorem 2.3 — it suffices to prove such a lower bound for simulations of $\Sigma_2\text{TIME}(n)$. The indirect diagonalization paradigm does this in three steps:

1. Assume the inclusion that we wish to rule out actually holds. In our case, we assume that $\Sigma_2\text{TIME}(n) \subseteq \text{BPTISP}(n^c, n^d)$.

2. Using the hypothesis, derive inclusions of complexity classes that are increasingly unlikely.

3. Eventually, one of these inclusions contradicts a known diagonalization result, proving the desired result.

There is a myriad of ways to derive new inclusions from the hypothesis in step 2, with different approaches yielding different results. Often, the inclusions derived in step 2 are obtained by a combination of two opposing processes that can be loosely described as

(2a) Speed up a space-bounded computation at the cost of adding alternations, and

(2b) Eliminate these alternations at a moderate increase in running time.

To envision the utility of these items, notice that the assumption in step 1 allows the simulation of an alternating machine by a space-bounded machine. Item (2a) allows us to simulate the latter machine by an alternating machine that runs in less time, but may make more alternations than we started with. Item (2b) allows us to eliminate some of these alternations at the cost of increasing the running time modestly. Under the right combination of parameters c and d, this process allows a speedup within the second level of the polynomial-time hierarchy that contradicts the time hierarchy for this class (stated in Section 3.3).

With this outline in mind, we present the ingredients used by existing arguments in each of these steps. Unfortunately, not all of these are suited for the setting of randomized lower bounds; we leave the task of bridging the gap to randomized machines to Chapter 4.

3.1 Speedups

To satisfy step (2a), we simulate a space-bounded machine of the type for which we are attempting to derive a lower bound (i.e., deterministic or randomized) in significantly less time. We must pay a price to achieve this task, and we choose to pay in the currency of alternations. Specifically, we introduce a small number of alternations to the computation and carry out a fast simulation on an alternating machine.

This goal is realized by the divide-and-conquer strategy underlying Savitch's Theorem [Sav70]. Briefly, the idea is to divide the computation tableau of a space-bounded deterministic machine M into b blocks. Observe that M accepts x in time t if and only if there are $b-1$

configurations $C_1, C_2, \ldots, C_{b-1}$ at the boundaries of these blocks such that for every block i, $0 \leq i \leq b-1$, the configuration at the beginning of that block, C_i, reaches the configuration at the end of that block, C_{i+1}, in t/b steps, where C_0 is the initial configuration and C_b is the accepting configuration. We can check this condition quickly with an alternating machine: first existentially guess $b-1$ configurations of M, universally guess a block number i, and conclude by deciding if C_i reaches C_{i+1} via a simulation of M for t/b steps. This yields the inclusion:

$$\mathrm{DTISP}(t,s) \subseteq \exists^{bs}\forall^{\log b}\mathrm{DTISP}(t/b,s) \subseteq \Sigma_2\mathrm{TIME}(bs+t/b). \tag{3.1}$$

By the closure of DTISP under complement, this inclusion can also be stated for the Π-side of the polynomial time hierarchy, which will be more convenient to use in some of our arguments. If we choose b to optimize the running time of the resulting Π_2-computation, the result is a square-root speedup for small space bounds s:

$$\mathrm{DTISP}(t,s) \subseteq \forall^{\sqrt{ts}}\exists^{\log t}\mathrm{DTISP}(\sqrt{ts},s) \subseteq \Pi_2\mathrm{TIME}(\sqrt{ts}). \tag{3.2}$$

Recursively applying (3.1) while exploiting DTISP's closure under complementation to conserve alternations yields

$$\mathrm{DTISP}(t,s) \subseteq \underbrace{\forall^{bs}\exists^{bs}\ldots\overline{Q}^{bs}}_{k-1}\ Q^{\log b}\mathrm{DTISP}(t/b^{k-1}+bs,s) \subseteq \Pi_k\mathrm{TIME}(t/b^{k-1}+bs) \tag{3.3}$$

for any integer $k \geq 2$, where $Q = \exists$ if k is even and $Q = \forall$ otherwise, and $\overline{Q}$ denotes the quantifier complementary to Q. Choosing b to optimize the resulting running time, (3.3) achieves a k^{th}-root speedup for small space bounds s.

$$\begin{aligned}\mathrm{DTISP}(t,s) &\subseteq \underbrace{\forall^{(ts^{k-1})^{1/k}}\exists^{(ts^{k-1})^{1/k}}\ldots\overline{Q}^{(ts^{k-1})^{1/k}}}_{k-1}\ Q^{\log(t/s)}\mathrm{DTISP}((ts^{k-1})^{1/k},s)\\ &\subseteq \Pi_k\mathrm{TIME}((ts^{k-1})^{1/k}).\end{aligned} \tag{3.4}$$

Our lower bound for tautologies on nondeterministic machines (Theorem 1.6) requires the speedup of a space-bounded *nondeterministic* machine. Luckily, the divide-and-conquer approach underlying Savitch's theorem works in this case as well (in fact, it was designed

for it, to show that nondeterministic and deterministic polynomial space coincide). In this setting, the final step of the simulation checks that C_i reaches C_{i+1} in a *nondeterministic* sense. This requires simulating the space-bounded nondeterministic machine for t/b steps, adding one more alternation to the simulation:

$$\mathrm{NTISP}(t,s) \subseteq \exists^{bs}\forall^{\log b}\mathrm{NTISP}(t/b,s) \subseteq \Sigma_3\mathrm{TIME}(bs+t/b). \tag{3.5}$$

Choosing b optimally results in the same square-root speedup as in the deterministic case at the cost of one more alternation. Furthermore, recursive applications of (3.5) add *two* alternations each time instead of one as in the deterministic case, (3.3), since nondeterministic classes are not known to be closed under complement. Thus, our speedups for nondeterministic machines are less alternation-efficient than for deterministic machines.

For future use, we point out one important fact about the simulation underlying (3.1) and (3.5): the final phase of this simulation, that of running M for t/b steps, does not need access to all of the configurations guessed during the initial existential phase— it only reads the description of two configurations, C_i and C_{i+1}, in addition to the original input x. Thus, the input size of the final stage is $O(n+s)$ as opposed to $O(n+bs)$ as the complexity-class inclusions of (3.1) and (3.5) suggest in general. This fact has a subtle but key impact on our analyses, especially in Chapter 6.

3.2 Eliminating Alternations

In the other direction, the process (2b) of removing alternations involves simulating an alternating machine by another that makes fewer alternations and runs for only slightly more time. In many cases, this is accomplished by deriving a statement such as

$$\Sigma_\ell\mathrm{TIME}(t) \subseteq \Pi_\ell\mathrm{TIME}(g(t)),$$

for a small function g. When g is a polynomial, such an inclusion results in a collapse of the polynomial-time hierarchy to the ℓ^{th} level, and we refer to it as an *efficient complementation*. Note that a complementation can be derived unconditionally by using exhaustive search in

lieu of an alternating step, but this causes g to be exponential in t. In our arguments, we derive efficient complementations that are conditional on the unlikely assumption of step 1, and inductively derive more efficient complementations in step 2.

For example, when proving satisfiability lower bounds in the deterministic setting, the assumption in step 1 is of the form $\mathrm{NTIME}(n) \subseteq \mathrm{DTISP}(n^c, n^d)$; in the case of tautologies lower bounds on nondeterministic machines, it is $\mathrm{coNTIME}(n) \subseteq \mathrm{NTISP}(n^c, n^d)$. Either case gives an efficient complementation of the first level of the polynomial-time hierarchy, $\mathrm{NTIME}(n) \subseteq \mathrm{coNTISP}(n^c)$ that allows us to eliminate alternations at the cost of raising the running time to the power of c.

Proposition 3.1. *Suppose that*

$$\mathrm{coNTIME}(n) \subseteq \mathrm{NTIME}(n^c)$$

for some real $c \geq 1$. Then for any time functions t and t',

$$\exists^{t'}\mathrm{coNTIME}(t) \subseteq \mathrm{NTIME}((t + t' + n)^c).$$

Proof. Consider a machine M recognizing a language in $\exists^{t'}\mathrm{coNTIME}(t)$. Its acceptance condition on input x can be written as

$$\exists y \in \{0,1\}^{O(t')} P(x, y),$$

where $P(\cdot, \cdot)$ is a predicate recognized by a conondeterministic machine running in time $O(t)$ on input $\langle x, y\rangle$. Since P takes input of size $O(n + t')$, a padding argument applied to the assumption allows P to be recognized by a nondeterministic machine running in time $O((t + t' + n)^c)$. In this way, we characterize the acceptance of M by two consecutive existential guesses, which is a nondeterministic machine overall. It requires time $O(t')$ for the guess of y and $O((t + t' + n)^c)$ for the part recognizing P, for a total of $O((t + t' + n)^c)$ since $c \geq 1$. □

In a typical setting of $t = t' = n^{1+\Omega(1)}$, Proposition 3.1 exhibits the exponent cost of c as described above. A finer point to make is that although the argument only applies the

hypothesis to the final conondeterministic phase, Proposition 3.1 indicates that, in general, the t' guess bits of the initial phase factor into the cost of eliminating the alternation as much as the running time of the final phase, even when the latter is much smaller. This point is where the special input-size property of the speedups (3.1) and (3.5) mentioned earlier become important: the input to the final stage of the divide-and-conquer speedup is only a small portion of the bits guessed in the initial stage, dramatically reducing the effect just described when we eliminate some of these alternations. This observation is crucial to achieve the quantitative strength of our results in Chapter 6.

3.3 Diagonalization Results

Our indirect diagonalization arguments use speedups and alternation removal to derive more and more efficient complementations of the polynomial-time hierarchy. In fact, we know how to rule out complementations that are *too* efficient, namely, those that speed up as well as complement. This supplies us with the aforementioned direct diagonalization result with which we ultimately derive a contradiction.

Lemma 3.2 (Folklore). *Let c and c' be positive reals such that $c' < c$. Then for any integer $\ell \geq 1$,*

$$\Sigma_\ell \mathrm{TIME}(n^c) \not\subseteq \Pi_\ell \mathrm{TIME}(n^{c'}).$$

Our lower bounds for space-bounded alternating linear time (Theorem 1.3) require a stronger diagonalization result that is both time- and space-sensitive.

Lemma 3.3 (cf. [FLvMV05]). *Let c, c', d, d' be positive reals such that $c' < c$ and $d' < d$. Then for any integer $\ell \geq 1$,*

$$\Sigma_\ell \mathrm{TISP}(n^c, n^d) \not\subseteq \Pi_\ell \mathrm{TISP}(n^{c'}, n^{d'}).$$

3.4 A Concrete Example

Now that we have outlined ingredients for each of the steps, let us give an example of an indirect diagonalization argument. In particular, we prove the result of [LV99] that

satisfiability cannot be solved by deterministic random-access machines running in time n^c and space $n^{o(1)}$ for constants $c < \sqrt{2}$. By the tight completeness given by Theorem 2.3, it suffices to prove the lower bound for NTIME(n). The first step of an indirect diagonalization argument for this is to assume that

$$\text{NTIME}(n) \subseteq \text{DTISP}(n^c, n^{o(1)}). \tag{3.6}$$

By padding, this allows a simulation of NTIME(t) by DTISP($t^c, t^{o(1)}$) for some polynomial t. The speedup given by the inclusion (3.2) yields a square-root speedup of the latter class at the cost of one alternation. The net result is

$$\text{NTIME}(t) \subseteq \text{DTISP}(t^c, t^{o(1)}) \subseteq \Pi_2\text{TIME}(t^{c/2+o(1)}), \tag{3.7}$$

which represents a speedup of NTIME(t) for $c < 2$ by using one more alternating stage. To contradict Lemma 3.2, we need to arrive at a speedup using just a universal stage. The hypothesis (3.6) provides us with the efficient complementation required to carry out this task: Proposition 3.1 eliminates one alternation from the right-hand side of (3.7) to yield

$$\text{NTIME}(t) \subseteq \text{coNTIME}(t^{c^2/2+o(1)}),$$

provided that $t(n) \geq n^{2/c}$. This contradicts Lemma 3.2 when $c^2/2 < 1$, proving the desired result.

Chapter 4

Time-Space Lower Bounds on Randomized Machines

In this chapter we adapt the indirect diagonalization paradigm to prove our time-space lower bounds for Σ_ℓSAT on randomized machines.

Theorem 1.1 (restated). *For any integer $\ell \geq 2$ and constant $c < \ell$, there exists a positive constant d such that* Σ_ℓSAT *cannot be solved by randomized random-access machines with two-sided error running in time n^c and space n^d. Moreover, d approaches $1/2$ from below as c approaches 1 from above for $\ell = 2$, and d approaches 1 from below as c approaches 1 from above for $\ell \geq 3$.*

We also obtain such time-space lower bounds for PLA-simplification.

Corollary 1.2 (restated). *For any constant $c < 2$, there exists a positive constant d such that* PLA-simplification *cannot be solved by randomized random-access machines with two-sided error running in time n^c and space n^d. Moreover, d approaches $1/2$ from below as c approaches 1 from above.*

We prove these results by focusing on the class Σ_ℓTIME(n). Lower bounds for the latter class carry over to Σ_ℓSAT and PLA-simplification (for $\ell = 2$) by way of the tight connections given by Theorems 2.3 and 2.6, respectively.

The critical ingredient in our proof is a time- and space-efficient alternating simulation of space-bounded randomized computations. This follows from a careful combination of Nisan's partial space-bounded derandomization [Nis93], randomness-efficient error reduction by a random walks in an expander [CW89, IZ89], and a version of Lautemann's proof

that randomized machines with two-sided error can be simulated in the second level of the polynomial-time hierarchy with a polynomial-time overhead [Lau83]. This construction enables both ingredients needed to fill in the second step of the indirect diagonalization argument in the randomized setting: a speedup of small-space randomized computations with two-sided error and an efficient complementation of the ℓ^{th} level of the polynomial-time hierarchy, for $\ell \geq 2$. The complementation is conditional on the assumption of the indirect diagonalization argument, that

$$\Sigma_\ell \text{TIME}(n) \subseteq \text{BPTISP}(n^c, n^d). \tag{4.1}$$

Combining this hypothesis with our speedups and complementations, we derive a new statement showing that computations in the ℓ^{th} level of the polynomial-time hierarchy that run in time t (where t is a sufficiently large polynomial) can be complemented in time $g(t)$, where g is some function depending on c and d. For sufficiently small values of c and d, $g(t)$ becomes $t^{1-\epsilon}$ for positive ϵ, which contradicts the direct diagonalization argument of Lemma 3.2. For somewhat larger values, we do not obtain a contradiction right away, but rather a more efficient complementation of the ℓ^{th} level of the polynomial-time hierarchy than we previously had. We then run the argument again using the new complementation in place of the old, yielding an even more efficient complementation. Bootstrapping in this way eventually improves the complementations to the point that they contradict Lemma 3.2. This allows us to rule out larger values of c and d, leading to Theorem 1.1.

A careful analysis shows that the largest space bound that this argument can handle is of the form n^d where d is a positive constant depending on c. For $\ell = 2$, d approaches $1/2$ from below when c approaches 1 from above. For $\ell \geq 3$, we achieve a better value of d for such small values of c by deriving a more efficient simulation of randomized computations in the *third* level of the polynomial-time hierarchy.[1] This follows by exploiting the structure of the second-level simulation described above and adding an alternation to reduce the time overhead. The savings in time are more substantial for large values of d. When c approaches

[1] Viola [Vio07] independently obtained the same simulation using a somewhat more complicated argument.

1 from above, the modified argument can handle values of d approaching 1 from below. For larger values of c, the cost of the additional alternation obviates the savings in running time and makes the earlier argument the better one. Paying close attention to the space used by the simulations involved, we obtain the strengthening to lower bounds for randomized simulations of small-space alternating machines given in Theorem 1.3.

With the high level overview in mind, we now construct our time- and space-efficient alternating simulation of space-bounded randomized computations.

4.1 Alternating Simulations and Derandomization

Our investigation of alternating simulations of randomized machines is motivated by the desire to use the assumption of the indirect diagonalization argument (4.1) to realize an efficient complementation of the polynomial-time hierarchy. Existing proofs that BPP lies in the second level of the polynomial-time hierarchy provide just that when combined with (4.1) for $\ell \geq 2$. More specifically, they give a simulation of randomized machines with two-sided error by Π_2-machines at a polynomial overhead in time.

Assuming the Π_2-simulation is sufficiently time- and space-efficient, we can also use it for the other ingredient we need, namely the speedup. This is because the divide-and-conquer strategy for DTISP-computations from Section 3.1 applies to Σ_kTISP-computations as well. However, it turns out that the known Π_2-simulations of randomized two-sided error machines are not time- and space-efficient enough to obtain any lower bounds this way. Moreover, in order to achieve the quantitative strength of our lower bounds, we need to save alternations by applying the speedup as in (3.1) to the final deterministic phase of the simulation as opposed to the Π_2-simulations as a whole. For that approach to yield an overall speedup, we need the number of guess bits in the alternating phases of the simulation to be small — otherwise, the time needed for the guesses would obviate the speedup obtained in the final deterministic phase. The known simulations use too many guess bits from that perspective. Therefore, in this section, we develop a new Π_2-simulation of randomized machines with two-sided error that meets all the above efficiency requirements.

We start by analyzing Lautemann's proof that any language L in BPP is also in $\Sigma_2^p \cap \Pi_2^p$. The proof assumes a randomized algorithm using r random bits to decide L with error ϵ. When ϵ is small enough in comparison to r, there is a $v \geq 1$ so that membership of x in L can be characterized by the existence of v shifts of the set of random strings accepting x that together cover the universe of all random strings. If $x \in L$, the set of random strings accepting x is large enough to guarantee that such shifts exist as long as $\epsilon^v < 2^{-r}$. On the other hand, if $x \notin L$, the set of accepting random strings is small enough so that v shifts cannot cover the universe of random strings as long as $\epsilon < \frac{1}{v}$. For such ϵ and v, these complementary conditions are expressed by a Σ_2^p-predicate. Since BPP is closed under complement, this shows that BPP $\subseteq \Sigma_2^p \cap \Pi_2^p$. Specifically, we are interested in the Π_2^p-side of the inclusion.

Theorem 4.1 (Lautemann [Lau83]). *Let L be a language recognized by a randomized machine M that runs in time t, space s, and uses r random bits with error bounded on both sides by ϵ. Then for any $v \geq 1$ such that $\epsilon < \min(2^{-r/v}, \frac{1}{v})$, we have that*

$$L \in \forall^{vr}\exists^{r}\mathrm{DTISP}(vt, s + \log v). \tag{4.2}$$

In Lautemann's proof that BPP $\subseteq \Sigma_2^p \cap \Pi_2^p$, one starts from an algorithm deciding $L \in$ BPP that has error less than the reciprocal of the number r' of random bits it uses. Such an error probability can be achieved from a standard BPP algorithm deciding L that uses r random bits by taking the majority vote of $O(\log r)$ independent trials, which results in $r' = O(r \log r)$. For $\epsilon < \frac{1}{r'}$, choosing $v = r'$ satisfies the conditions of Lemma 4.1. This shows that if L can be decided by a BPTISP(t, s) machine using r random bits, then

$$L \in \forall^{(r \log r)^2}\exists^{r \log r}\mathrm{DTISP}(rt \log r, s). \tag{4.3}$$

We can reduce the number of guess bits in the alternating stages of the simulation by using more efficient methods of amplification. With such methods, the error can be made as small as 2^{-r} while using only $O(r)$ random bits. Specifically, the algorithm runs $O(r)$ trials which are obtained from the labels of vertices on a random walk of length $O(r)$ in

an easily constructible expander graph, and accepts if a majority of these trials accept. For our purposes, we choose the Gabber-Galil family of expanders [GG81], a construction based on the Margulis family [Mar73] where the vertices are connected via simple affine transformations on the labels. The easy form of the edge relations ensures that the walk is efficiently computable in time $O(r^2)$ and space $O(r)$.

Theorem 4.2 ([CW89, IZ89]). *Let M be a randomized machine with constant error bounded away from $\frac{1}{2}$ that runs in time t, space s, and uses r random bits. Then M can be simulated by another randomized machine M' that runs in time $O(rt)$ and space $O(r+s)$, while using only $O(r)$ random bits to achieve error 2^{-r}.*

An overview of Theorem 4.2 can be found in Section 4.2.4. When an algorithm has had its error reduced as in Theorem 4.2, $v = O(1)$ shifts suffice for Theorem 4.1. This shows that if L can be decided by a $\mathrm{BPTISP}(t, s)$ machine using r random bits, then

$$L \in \forall^r \exists^r \mathrm{DTISP}(rt, r+s). \tag{4.4}$$

The efficiency of this simulation depends on the number of random bits r for all the criteria we mentioned: the number of bits guessed in the alternating stages, the multiplicative time overhead, and the additive space overhead for the final deterministic stage are all $O(r)$. However, r can be as large as t, so we need an additional ingredient to do better. That ingredient exploits the fact that we are dealing with *space-bounded* BPP-computations. In that setting, we know of techniques to reduce the number of random bits without increasing the time or space by much, which in turn increases the efficiency of the Π_2-simulation in (4.4). The means by which we achieve the needed reduction in randomness is the space-bounded derandomization of Nisan [Nis93]. We state a version here, and leave the proof to Section 4.1.1.

Theorem 4.3. *Any randomized machine M running in time t and space s with error ϵ can be simulated by another randomized machine running in time $O(t \operatorname{polylog}(t))$, space $O(s \log t)$, and using only $O(s \log t)$ random bits. The error of the simulation is $\epsilon + 2^{-s}$, and is one-sided if M has one-sided error.*

Note that we do not apply Theorem 4.3 to deterministically simulate the randomized machine. Instead, we use it to reduce the randomness required by a BPTISP(t, s) machine to $O(s \log t)$. If we subsequently efficiently amplify using Theorem 4.2, then the overhead of the alternating simulation given by Theorem 4.1 becomes acceptable for polynomial t and small s. More precisely, we have:

Theorem 4.4.

$$\text{BPTISP}(t, s) \subseteq \forall^{s \log t} \exists^{s \log t} \text{DTISP}(ts \operatorname{polylog}(t), s \log t). \tag{4.5}$$

Proof. Let M be the randomized time t, space s machine for recognizing $L \in \text{BPTISP}(t, s)$. By the derandomization of Theorem 4.3, we obtain a simulation using $r \doteq O(s \log t)$ random bits, time $O(t \operatorname{polylog}(t))$, and space $O(s \log t)$, with error $\frac{1}{3} + 2^{-s}$. Theorem 4.2 gives a machine deciding L with error 2^{-r} while using $O(r)$ random bits. The time increases to $O(ts \operatorname{polylog}(t))$, while the space is still $O(s \log t)$. Applying Theorem 4.1 for $v = O(1)$ yields the desired alternating simulation of M. □

We point out that using the instantiation of Theorem 4.1 given by (4.3) instead of (4.4) in the proof of Theorem 4.4 yields a simulation similar to (4.5). The main difference is that the initial universal phase takes time $O((s \log t)^2)$ rather than $O(s \log t)$. This version is still efficient enough to yield time-space lower bounds as in Theorem 1.1, but the dependence of the space parameter d on c becomes worse.

4.1.1 Nisan's Pseudorandom Generator

We now describe our time- and space- efficient implementation of Nisan's pseudorandom generator that yields Theorem 4.3. In fact, we prove a somewhat stronger version of Theorem 4.3.

Theorem 4.5. *Any randomized machine M running in time t and space s with error ϵ can be simulated by another randomized machine running in time $O(t \log^2 s \log \log s)$, space $O(s \log t)$, and using only $O(s \log t)$ random bits. If two-way access to the random bits is*

allowed, the space requirement is reduced to $O(s)$. The error of the simulation is $\epsilon + 2^{-s}$, and is one-sided if M has one-sided error.

Theorem 4.5 gives a tighter time bound than the bound of $O(t\,\mathrm{polylog}(t))$ stated in Theorem 4.3. Our arguments are not noticeably improved by using the tighter bound stated in Theorem 4.5, so we use the simpler bounds of Theorem 4.3 there for clarity. We state the tighter bound here because it may be of independent interest.

Before proving Theorem 4.5, we introduce Nisan's pseudorandom generator [Nis93] and discuss some of its properties. Define

$$G_{m,k} : \{0,1\}^m \times H_m^k \to (\{0,1\}^m)^{2^k},$$

where H_m is a family of two-universal hash functions $h : \{0,1\}^m \to \{0,1\}^m$ [CW79]. The evaluation of $G_{m,k}$ is defined recursively as

$$G_{m,k}(y, h_1, \ldots, h_k) = \begin{cases} y & \text{if } k = 0 \\ G_{m,k-1}(y, h_1, \ldots, h_{k-1}) \circ G_{m,k-1}(h_k(y), h_1, \ldots, h_{k-1}) & \text{otherwise,} \end{cases}$$

where "$\circ$" denotes concatenation. Given a randomized machine M running in time t and space s, we define $G \doteq G_{m,k}$ for $k = \log \frac{t}{m}$ where $m = \Theta(s)$ will be determined later. The simulation of M proceeds with the output of G as the random string, one block of length m at a time. Nisan proves that G fools M in the following sense:

Theorem 4.6 (Nisan [Nis93]). *There exists a constant ν such that if M is a randomized machine running in time t and space s on input x, then for $m \geq \nu \cdot s$ and $k = \log \frac{t}{m}$,*

$$\left| \Pr_{\rho}[M(x,\rho) \text{ accepts}] - \Pr_{y,h_1,\ldots,h_k}[M(x, G_{m,k}(y, h_1, \ldots, h_k)) \text{ accepts}] \right| \leq 2^{-s},$$

where $M(x,\rho)$ denotes the outcome of running M on input x and random string ρ.

This satisfies the error requirements of Theorem 4.5, so all that remains to prove Theorem 4.5 is to show how to simulate M on the random string $G(y, h_1, \ldots, h_k)$ within the correct bounds on the time, space, and randomness.

We start with the standard simulation that computes G in a block-wise fashion, where each subsequent m-bit block is computed after m simulation steps of M using the current block. One way to do this is to compute each block from scratch, namely, apply the appropriate sequence of at most k hash functions to the m-bit seed y. Although this technique is good enough to derive our main results, we can do slightly better, namely by a factor of $O(\frac{\log s}{\log(t/s)})$. This improvement follows by computing each block in a recursive manner, which avoids the calculations that the "from scratch" method does over and over again. Another key ingredient in achieving the desired efficiency is the use of fast Fourier transform multiplication methods to quickly evaluate and invert the hash functions involved.

Proof of Theorem 4.5. To accommodate the approach described above, we choose $m = \Theta(s)$ such that $m \geq \nu \cdot S$ and is of the form $2 \cdot 3^q$ for some integer $q \geq 0$. The latter guarantees a simple explicit formula for an irreducible polynomial of degree m over GF(2), namely $z^{2 \cdot 3^q} + z^{3^q} + 1$ [vL91, Theorem 1.1.28 on page 13]. We choose H_m to be the set of all invertible linear mappings from GF(2^m) to GF(2^m), i.e., all functions of the form $x \mapsto ax + b$ where $a, b \in$ GF(2^m) and $a \neq 0$ [CW79].[2] Such functions can be described by $2m$ bits, so that the input to G can be described by $(2k + 1)m = O(s \log t)$ bits. This meets the requirements on the randomness.

To reach the desired time and space bounds, we must be able to evaluate the functions in H_m much faster than the naïve bound of $O(m^2)$. Using fast Fourier transform multiplication techniques based on those of Schönhage and Strassen and exploiting the sparseness of the above irreducible polynomial, we can evaluate $h \in H_m$ in time $O(m \log m \log \log m)$ and space $O(m)$. These fast multiplication techniques can be combined with the extended Euclidean algorithm to invert $h \in H_m$ in time $O(m \log^2 m \log \log m)$ and space $O(m)$. See [vzGG03, Corollary 11.8 on page 319] for more details. We use these algorithms to output the blocks of $G_{m,k}(y, h_1, \ldots, h_k)$ recursively with small space overhead. Specifically, we

[2]Excluding the non-invertible functions ($a = 0$) introduces a small bias. Although our family H_m is not perfectly two-universal, it is close enough for our purposes.

define the procedure P_m which uses global registers containing $k \geq 0$, $y \in \{0,1\}^m$, and $h_1, h_2, \ldots, h_k \in H_m$ to perform the following steps:

1. **If** $k > 0$:
2. $k \leftarrow k - 1$;
3. Recursively **call** P_m;
4. $y \leftarrow h_{k+1}(y)$;
5. Recursively **call** P_m;
6. $y \leftarrow h_{k+1}^{-1}(y)$; $k \leftarrow k + 1$ and **return**.
7. **Else output** y and **return**.

The output of $P_m(k, y, h_1, \ldots, h_k)$ is exactly $G_{m,k}(y, h_1, \ldots, h_k)$. Evaluating P_m takes 2^k recursive calls, each accompanied by an evaluation of h_i or h_i^{-1}. Thus, the overall time complexity to output $G_{m,k}$ is $O(2^k \cdot m \log^2 m \log\log m)$. By replacing y by $h_k(y)$ instead of writing down $h_k(y)$ separately on the work tape, only a constant amount of space overhead is required for each level. Thus, the space requirement is $O(m + k)$. For the settings of G, $m = O(S)$ and $k = \log \frac{t}{s} + O(1)$, the time becomes $O(t \log^2 s \log\log s)$, while the space is $O(s + \log \frac{t}{s})$, which is $O(s)$ since $t \leq 2^s$ without loss of generality. This assumes that the hash functions can be accessed repeatedly, so there is an additional space cost of $O(s \log t)$ to copy the hash functions from the random tape to the work tape, bringing the space bound to $O(s \log t + s) = O(s \log t)$. However, if the simulation has two-way access to the random tape, this cost is avoided.

Our simulation runs M for $O(s)$ steps every time P_m outputs a block of G, for a total of t steps while using space s. Thus, the above time and space bounds for computing G hold for the simulation as a whole. □

4.2 Indirect Diagonalization Components

Theorem 4.4 presents an alternating simulation of randomized machines that is sufficiently time- and space-efficient enough to allow for the necessary components of an indirect diagonalization argument: a space-bounded speedup of randomized machines and a conditional efficient complementation.

4.2.1 Speedup

Theorem 4.4 has the desired nice properties that allow us to derive a speedup for BPTISP. The simulation spends only time $O(s \log t)$ in its alternating phases, which is small when s is small. In this case, the running time is dominated by the final deterministic computation, so a speedup of the final stage results in a speedup of the computation as a whole. Since the final deterministic computation of the simulation given by (4.5) is space-bounded, we can achieve this by applying the divide-and-conquer speedup of (3.1) or (3.3). For example, applying (3.1) adds one alternation, and by merging adjacent existential stages we obtain a simulation given by:

$$\begin{aligned} \mathrm{BPTISP}(t,s) &\subseteq \forall^{s\log t}\exists^{s\log t}\exists^{bs\log t}\forall^{\log b}\mathrm{DTISP}(ts\,\mathrm{polylog}(t)/b, s\log t) \\ &= \forall^{s\log t}\exists^{bs\log t}\forall^{\log b}\mathrm{DTISP}(ts\,\mathrm{polylog}(t)/b, s\log t). \end{aligned}$$

Choosing b to optimize the running time of this simulation up to a $\mathrm{polylog}(t)$ factor, we get

$$\mathrm{BPTISP}[t,s] \subseteq \forall^{s\log t}\exists^{\sqrt{t}s}\forall^{\log t}\mathrm{DTISP}(\sqrt{t}s\,\mathrm{polylog}(t), s\log t). \tag{4.6}$$

For small s, we obtain a speedup for space-bounded randomized machines similar to what (3.2) gives for deterministic machines. However, the simulation uses two alternations rather than one to realize the same speedup.

4.2.2 Complementation

We also use Theorem 4.4 to derive an efficient complementation under the assumption of the indirect diagonalization argument. In the case of lower bounds for Σ_2SAT, this

assumption is

$$\Sigma_2\text{TIME}(n) \subseteq \text{BPTISP}(n^c, n^d).$$

This gives a space-bounded randomized computation that can in turn be simulated by the Π_2-machine given by Theorem 4.4. Thus, we derive

$$\Sigma_2\text{TIME}(n) \subseteq \forall^{n^d \log n}\exists^{n^d \log n}\text{DTISP}(n^{c+d}\,\text{polylog}(n), n^d \log n), \tag{4.7}$$

which gives the desired complementation for small enough c and d. Thus, we can remove alternations — down to the second level — at the cost of raising the running time to the power of $c + d$; this compares to the deterministic setting of Proposition 3.1, where we can remove alternations — down to the first level — at the exponent cost of c. In this manner, (4.7) can be used analogously to the complementation given by Proposition 3.1 in the deterministic setting, although there is a dependence on the space parameter d that did not exist before.

We note that the bottleneck causing our lower bounds for Σ_ℓSAT to hold only for $\ell \geq 2$ arises right here. In the case $\ell = 1$, the hypothesis becomes $\text{NTIME}(n) \subseteq \text{BPTISP}(n^c, n^d)$; combining with Theorem 4.4 as above, we obtain $\text{NTIME}(n) \subseteq \Pi_2\text{TIME}(n^{c+d}\,\text{polylog}(n))$, which is trivial for $c \geq 1$ and does not represent an efficient complementation.

4.2.3 Higher Levels of the Polynomial-Time Hierarchy

The discussion in the previous section developed complementations useful in proving lower bounds for $\Sigma_2\text{TIME}(n)$. These readily generalize to higher levels, where the hypothesis becomes

$$\Sigma_\ell\text{TIME}(n) \subseteq \text{BPTISP}(n^c, n^d) \tag{4.8}$$

for $\ell \geq 3$. Theorem 4.4 then allows for an efficient simulation of $\Sigma_\ell\text{TIME}(n)$ by a Π_2-machine, which can still be used to eliminate alternations in this case. This allows us to establish Theorem 1.1 for values of $c < \ell$, where d approaches some small value depending on ℓ from below as c approaches 1 from above.

For $\ell \geq 3$, we can get a better dependence of d on c when c approaches 1. In this setting, a Π_3-simulation of BPTISP suffices to achieve an efficient complementation. The ability to use an additional alternation allows us to achieve a more time-efficient simulation than the one given by Theorem 4.4. Specifically, we add an alternation to the latter Π_2-simulation and eliminate the time blowup incurred by running the $O(s \log t)$ trials required by the amplification of Theorem 4.2. Rather than deterministically simulate all of these trials, we use the power of alternation to efficiently verify that a majority of these trials accept.

Lemma 4.7. *Let M be a randomized machine with constant error bounded away from $\frac{1}{2}$ that runs in time t, space s, and uses r random bits. Then M can be simulated by another randomized machine M' that uses $O(r)$ random bits to achieve error 2^{-r}. Furthermore, the acceptance of M' on input x and random string ρ can be decided in*

$$\exists^r \forall^{\log r} \mathrm{DTISP}(t + r \operatorname{polylog}(r), r + s).$$

We defer the proof to Section 4.2.4. Notice that the final deterministic stage of the simulation represented by (4.5) can be replaced by the Σ_2-verification given by Lemma 4.7. Merging the resulting adjacent existential phases results in a simulation using one more alternation but running in less time.

Theorem 4.8.

$$\mathrm{BPTISP}(t, s) \subseteq \forall^{s \log t} \exists^{s \log t} \forall^{\log s} \mathrm{DTISP}(t \operatorname{polylog}(t), s \log t). \tag{4.9}$$

As in Sections 4.2.1 and 4.2.2, Theorem 4.8 admits a speedup of BPTISP to Π_4 as well as an efficient complementation of Σ_3; the latter now eliminates alternations essentially at the cost of raising the running time to the power of c, rather than $c + d$ as before. For values of c close to 1, this cost is small enough to outweigh the negative effects of the extra alternation in (4.9). In this case, the better dependence on the space parameter allows us to derive contradictions for larger values of d with Theorem 4.4. On the other hand, for larger values of c, eliminating alternations becomes costlier, so the extra alternation in (4.9) has a greater impact and eventually prevents us from reaching a contradiction. In this case,

switching to the more alternation-efficient simulation given by Theorem 4.4 allows us to derive a contradiction for such larger values of c. However, we must restrict d to smaller values in order to counteract the worse dependence of (4.5) on the space bound. Therefore, to derive Theorem 1.1, we focus on using Theorem 4.4 to obtain the bounds for large values of c first, and then show how Theorem 4.8 yields the larger values of d when c is small.

4.2.4 Randomness-Efficient Error Reduction

It remains to prove Lemma 4.7 regarding the complexity of randomness-efficient error reduction. We begin with a brief discussion of some properties of the error reduction given by Theorem 4.2.

Theorem 4.2 (restated). *Let M be a randomized machine with constant error bounded away from $\frac{1}{2}$ that runs in time t, space s, and uses r random bits. Then M can be simulated by another randomized machine M' that runs in time $O(rt)$ and space $O(r+s)$, while using only $O(r)$ random bits to achieve error 2^{-r}.*

Let M be a machine as in Theorem 4.2. We may assume that M has error at most some small constant δ to be determined later, since this can be achieved with only a constant overhead from a machine with any error bounded away from $1/2$. The amplified machine M' given by Theorem 4.2 interprets its random string ρ' of length $O(r)$ as an initial vertex in a Gabber-Galil graph followed by $O(r)$ edge labels (of constant size) specifying the edges on a walk in the graph. Formally, this graph is described as follows:

Definition 4.9 (Gabber-Galil graph [GG81]). *The Gabber-Galil graph* $\mathrm{GG}(m)$ *is a graph with vertices $\mathbb{Z}_m \times \mathbb{Z}_m$ of degree 5 where the vertices adjacent to $(x, y) \in \mathbb{Z}_m \times \mathbb{Z}_m$ are the five pairs (x', y') obtained from the matrix multiplication $[x', y', 1]^T = A_i[x, y, 1]^T$ (over*

$\mathbb{Z}_m$) *for* $1 \leq i \leq 5$, *where*

$$A_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, A_3 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix},$$

$$A_4 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, A_5 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

We now state a useful lemma showing that the vertex at the end of a path of length p in $\mathrm{GG}(2^k)$ can be found in time quasi-linear in p and k, an improvement on the naïve bound of $O(pk)$. We leave the proof for later.

Lemma 4.10. *Given a vertex* (x, y) *of the graph* $\mathrm{GG}(2^k)$ *and a path* π *of length* p *indicated by edge labels* $(e_1, e_2, \ldots, e_p)$, *where* $1 \leq e_i \leq 5$ *for* $1 \leq i \leq p$, *the vertex connected to* (x, y) *by* π *can be determined in time* $O(m \operatorname{polylog}(m))$ *and space* $O(m)$ *where* $m = k + p$.

For the purposes of Theorem 4.2, the vertices of the Gabber-Galil graph must be described by r bits corresponding to the possible random strings for M. To this end, we choose $\mathrm{GG}(2^{r/2})$. Given input x and random string ρ', M' proceeds by deterministically carrying out the walk of length $p = O(r)$ on $\mathrm{GG}(2^{r/2})$ indicated by ρ'. Every β steps, where β is some constant, M' simulates M on input x and random string corresponding to the label of the current vertex on the walk. M' accepts if a majority of these trials accept. As each edge relation can be computed by simple arithmetic in time $O(r)$ and space $O(r)$, and running M requires time t and space s, M' runs in time $O(r^2 + rt) = O(rt)$ and space $O(r + s)$. Cohen and Wigderson [CW89] and Impagliazzo and Zuckerman [IZ89] show that when β is large enough and M has error smaller than some constant δ, the trials specified by a randomly chosen ρ' are close enough to uniform so that M' only errs with probability 2^{-r}. This establishes Theorem 4.2.

We now prove Lemma 4.7, giving a more time-efficient manner to determine if M' accepts on x and ρ' at the cost of using alternations.

Lemma 4.7 (restated).

$$\mathrm{BPTISP}(t,s) \subseteq \forall^{s\log t}\exists^{s\log t}\forall^{\log s}\mathrm{DTISP}(t\,\mathrm{polylog}(t), s\log t).$$

Proof. To arrive at Lemma 4.7, we show how to use alternations to verify if there is a majority of trials on the walk given by the random string ρ' where M accepts, in such a way that the final deterministic phase only needs to simulate M once. Specifically, given input x and random string ρ', we can express the acceptance condition of M' as

$$(\exists Z \subseteq \{1,2,\ldots,R'\}, |Z| = \lceil r'/2\rceil)(\forall i \in Z) M(x,\rho_i) \text{ accepts}, \tag{4.10}$$

where r' is the number of trials of M specified by ρ', and ρ_i is the random string produced for the i^{th} trial. Observe that this describes a Σ_2-computation which accepts if and only if M' accepts. The initial existential phase guesses the characteristic string of the set Z consisting of $\lceil r'/2\rceil$ indices of the r' trials, for a total of $r' = O(r)$ bits. The universal phase guesses $\log r' = O(\log r)$ bits to determine the index of a trial to verify. The final deterministic stage must first determine ρ_i and then run M on input x and random string ρ_i. Once the former has been computed, the latter task takes time t and space s. Since ρ_i corresponds to the label of the $(\beta i)^{\text{th}}$ vertex on the walk in $\mathrm{GG}(2^{r/2})$ specified by ρ', Lemma 4.10 shows that ρ_i can be computed in time $O(r\,\mathrm{polylog}(r))$. Therefore, the final deterministic stage takes time $O(t + r\,\mathrm{polylog}(r))$ and space $O(r+s)$. All told, we have shown that (4.10) is a computation in

$$\exists^{r}\forall^{\log r}\mathrm{DTISP}(t + r\,\mathrm{polylog}(r), r+s)$$

which accepts if and only if M' accepts. This completes the proof. □

All that remains is to establish Lemma 4.10, which follows from a divide-and-conquer strategy to efficiently evaluate a product of p matrices A_i.

Proof of Lemma 4.10. Throughout this proof, we use the fact that multiplication of b bit integers can be done in time $O(b\,\mathrm{polylog}(b))$ and space $O(b)$. This follows from the fast

Fourier transform techniques of Schönhage and Strassen [vzGG03, Theorem 8.24 on page 240]. Let

$$A \doteq A_{e_p} A_{e_{p-1}} \cdots A_{e_1}.$$

Then the vertex (x', y') connected to (x, y) by π satisfies $[x', y', 1]^t = A[x, y, 1]^t \bmod 2^k$. Therefore, computing (x', y') reduces to computing A and multiplying by the vector $[x, y, 1]^t$ modulo 2^k.

We accomplish the latter with a divide-and-conquer strategy. Namely, we split the product approximately in half and recursively compute the subproducts $A_{e_p} A_{e_{p-1}} \cdots A_{e_{\lfloor p/2 \rfloor + 1}} \doteq B$ and $A_{e_{\lfloor p/2 \rfloor}} A_{e_{\lfloor p/2 \rfloor - 1}} \cdots A_{e_1} \doteq C$. It can be shown by induction that any product of p matrices A_i has entries bounded by 2^{p-1}, since each column of any matrix A_i has at most two non-zero entries, which are ones. This shows that once B and C are computed, A can be computed as $A = BC$ by $O(1)$ multiplications and additions of integers of bit length $p/2$, so we require time $O(p \operatorname{polylog}(p))$ in addition to the recursive calls. Thus, by following this strategy, we can see that at each recursive call to compute the product of q matrices, we solve two subproblems of size $q/2$ and do an additional amount of work which is quasi-linear in q and uses space $O(q)$. Thus, A can be computed in total time $O(p \operatorname{polylog}(p))$ and space $O(p)$.

By our observation on the size of the product matrix entries, we also know that the entries of the matrix A are at most 2^{p-1}. Therefore, computing the product $A[x, y, 1]^t$ and reducing modulo 2^k can be done in time $O(m \operatorname{polylog}(m))$ and space $O(m)$ as desired. □

We point out that another natural way to arrive at Theorem 4.8 is to use expander walks to generate the shift vectors for Lautemann's simulation in lieu of amplifying the confidence of the simulated algorithm as above. While the effect of the latter is to reduce the *number* of shift vectors needed, the former allows a large number of "good" shifts to be described by *very few bits*. Briefly, the hitting property of expanders guarantees that the shifts satisfy the needed property (i.e., the shifts of the accepting set cover the entire set of random strings) with approximately the same probability when they are chosen by a random walk on a Gabber-Galil graph as when they are chosen independently. Thus, we can generate a set of $O(r)$ good shifts in the initial stage of the simulation with only $O(r)$ bits. Furthermore, each

shift can be computed efficiently by Lemma 4.10, so we can avoid the $O(r)$ blowup in the running time of the final deterministic stage by using an additional alternation to verify that M accepts on some shift. This approach leads to a simulation that matches the parameters of Theorem 4.8.

4.3 Lower Bound for Σ_2SAT and PLA-simplification

We now use the techniques discussed in the previous sections to formulate an indirect diagonalization argument for the case $\ell = 2$ of Theorem 1.1. For clarity, we present the following exposition in terms of subpolynomial space bounds, and generalize these techniques to polynomial space bounds in the subsequent formal proof. Thus, to obtain a lower bound for $\Sigma_2\text{TIME}(n)$, we start with the assumption that

$$\Sigma_2\text{TIME}(n) \subseteq \text{BPTISP}(n^c, n^{o(1)}).$$

Consider a $\Sigma_2\text{TIME}(t)$ computation for some time function $t(n) = n^{O(1)}$. We adopt the approach outlined in Section 3.4, namely speeding up $\Sigma_2\text{TIME}(t)$ at the cost of adding alternations and then removing these alternations via an efficient complementation to arrive at a $\Pi_2\text{TIME}(t^{1-\epsilon})$ computation for positive ϵ. We begin by padding the assumption to give a simulation of $\Sigma_2\text{TIME}(t)$ in $\text{BPTISP}(t^c, t^{o(1)})$. We then apply the square-root speedup of (4.6) to obtain a simulation in Π_3.

$$\begin{aligned}\Sigma_2\text{TIME}(t) &\subseteq \text{BPTISP}(t^c, t^{o(1)}) \\ &\subseteq \forall^{t^{o(1)}} \underbrace{\exists^{t^{\frac{c}{2}+o(1)}} \forall^{\log t} \text{DTISP}(t^{c/2+o(1)}, t^{o(1)})}_{(\alpha)}.\end{aligned}$$

We have arrived at a simulation that makes one more alternation than we started with; we need to eliminate one to balance the number of alternations. Notice that the part of the computation indicated by (α) can be seen as a computation in Σ_2^p taking input of size $n + t^{o(1)}$ and running in time $t^{\frac{c}{2}+o(1)}$. When t is large enough, this running time is at least linear in the input size, and we can pad our complementation (4.7) to allow us to switch (α)

to Π_2. Merging the resulting adjacent universal stages yields the desired Π_2-simulation:

$$\Sigma_2\text{TIME}(t) \subseteq \Pi_2\text{TIME}(t^{\frac{c^2}{2}+o(1)}). \tag{4.11}$$

For $c < \sqrt{2}$, this results in a net speedup that is a contradiction to the direct diagonalization argument of Lemma 3.2. Thus we have derived a lower bound of $n^{\sqrt{2}-o(1)}$ for Σ_2SAT on subpolynomial-space randomized machines. However, we can do better by observing what (4.11) represents for values of c that do not immediately contradict Lemma 3.2. Specifically, (4.11) gives a complementation of Σ_2^p of the same form as (4.7) but with exponent cost of $c^2/2$ rather than c. Thus, we have derived a *more efficient* complementation for sufficiently large polynomial time bounds t when $c < 2$.

We now reiterate the above argument, except using (4.11) to eliminate alternations more efficiently than we did with (4.7). This yields an even more efficient complementation for sufficiently large polynomials t:

$$\Sigma_2\text{TIME}(t) \subseteq \Pi_2\text{TIME}(t^{\frac{c^3}{4}+o(1)}).$$

This can in turn be used to derive another more efficient complementation, and so on. In this manner, we derive a series of complementations, each one more efficient than the previous one. Specifically, each iteration multiplies the exponent cost of the complementation by $\frac{c}{2}$, so after k iterations we obtain

$$\Sigma_2\text{TIME}(t) \subseteq \Pi_2\text{TIME}(t^{c\cdot e_k+o(1)}),$$

where $e_k = (\frac{c}{2})^k$. Note that for $c < 2$, $e_k \to 0$ as $k \to \infty$. Thus, by choosing k large enough so that $c \cdot e_k < 1$, we arrive at a contradiction to Lemma 3.2, which proves the desired lower bound of $n^{2-o(1)}$.

The following lemma precisely derives the complementations of Σ_2^p for larger space bounds, n^d, giving the running times of the resulting Π_2-simulations in terms c, d, and k, the number of times the argument is recursively applied.

Lemma 4.11. *Suppose that*

$$\Sigma_2\mathrm{TIME}(n) \subseteq \mathrm{BPTISP}(n^c, n^d) \tag{4.12}$$

for some constants $c \geq 1$ and $d > 0$ where $c + 2d \leq 2$. Then for any time function t and integer $k \geq 0$ such that $d \leq f_k$,

$$\Sigma_2\mathrm{TIME}(t) \subseteq \Pi_2\mathrm{TIME}\left((t^{f_k} + n)^{c+d} \operatorname{polylog}(t+n)\right),$$

where

$$f_k \;=\; (\tfrac{c+2d}{2})^k. \tag{4.13}$$

Proof. We give a proof by induction on k. For $k = 0$, a padded version of the initial complementation given by (4.7) offers a Π_2-simulation of $\Sigma_2\mathrm{TIME}(t)$ running in the desired time.

We now show the inductive step by using the k^{th} complementation to derive the $(k+1)^{\text{st}}$. Padding the hypothesis (4.12) gives a simulation of $\Sigma_2\mathrm{TIME}(t)$ in $\mathrm{BPTISP}((t+n)^c, (t+n)^d)$. Note that the addition of the term n to t ensures the validity of this step for arbitrary t, in particular for sublinear t. Applying (4.6) gives a speedup of the BPTISP simulation at the cost of three alternations, yielding

$$\Sigma_2\mathrm{TIME}(t) \subseteq \forall^{(t+n)^d \log(t+n)} \underbrace{\Sigma_2\mathrm{TIME}\left((t+n)^{\frac{c+2d}{2}} \operatorname{polylog}(t+n)\right)}_{(\alpha)}. \tag{4.14}$$

The complementation given by the inductive hypothesis switches (α) to a Π_2-computation, eliminating one alternation. Specifically, (α) represents a Σ_2-machine running in time

$$\tilde{t} \doteq O\left((t+n)^{\frac{c+2d}{2}} \operatorname{polylog}(t+n)\right)$$

on inputs comprised of the original n-bit input in addition to the $(t+n)^d \log(t+n)$ bits guessed in the preceding universal stage, for a total input size of

$$\tilde{n} \doteq O\left(n + (t+n)^d \log(t+n)\right).$$

The inductive hypothesis allows us to simulate (α) by a Π_2-machine running in time

$$O\left((\tilde{t}^{f_k} + \tilde{n})^{c+d} \operatorname{polylog}(\tilde{t} + \tilde{n})\right).$$

Using this Π_2-machine for (α) and accounting for the time spent in the initial universal stage of the simulation given by (4.14) results in a Π_2 simulation of $\Sigma_2\mathrm{TIME}(t)$ running in time big-O of

$$(t+n)^d \log(t+n) + \left((\tilde{t}^{f_k} + \tilde{n})^{c+d}\right) \operatorname{polylog}(\tilde{t} + \tilde{n}).$$

All that remains is to show that the above running time is of the desired form under the conditions on c and d. To simplify this expression, note that $\operatorname{polylog}(\tilde{t}+\tilde{n}) = \operatorname{polylog}(t+n)$. Collecting all of the other polylog terms, the running time can be written as big-O of

$$\left[(t+n)^d + \left(\left((t+n)^{\frac{c+2d}{2}}\right)^{f_k} + n + (t+n)^d\right)^{c+d}\right] \operatorname{polylog}(t+n).$$

The terms depending on t in the big-O expression have exponents d, $\frac{c+2d}{2} f_k (c+d)$, and $d(c+d)$ (putting aside the $\operatorname{polylog}(t+n)$ factor for a moment). Thus, when $d \leq \frac{c+2d}{2} f_k = f_{k+1}$ and $c + d \geq 1$, the dominating term is $t^{\frac{c+2d}{2} f_k (c+d)} = t^{f_{k+1}(c+d)}$.

Similarly, the terms depending on n have exponents d, $\frac{c+2d}{2} f_k (c+d), c+d$, and $d(c+d)$. Under the same conditions on c and d, the first and last terms are subsumed by the second one, which can be rewritten as $f_{k+1}(c+d)$. Additionally, when $c + 2d \leq 2$, $f_k \leq 1$ for all $k \geq 0$, so that the n^{c+d} term dominates. Thus, we simplify the running time to big-O of

$$\left((t^{f_{k+1}} + n)^{c+d}\right) \operatorname{polylog}(t+n),$$

which is of the desired form. □

The series of complementations given by Lemma 4.11 lead to a contradiction with Lemma 3.2 for certain values of c and d, which proves the desired lower bound.

Theorem 4.12. *For any constant $c < 2$, there exists a positive constant d such that $\Sigma_2\mathrm{TIME}(n)$ cannot be simulated by randomized random-access machines with two-sided error running in time n^c and space n^d. Moreover, d approaches $1/2$ from below as c approaches 1 from above.*

Proof. For $c < 1$, the Theorem holds for any d by standard techniques. Namely, take the parity function, which is surely in $\Sigma_2\text{TIME}(n)$ and consider any $\text{BPTIME}(n^c)$ machine M which purportedly computes it. On input 0^n, we must have that

$$\sum_{i=1}^{n} \Pr[M \text{ looks at } i^{th} \text{ bit on input } 0^n] \leq n^c.$$

For $c < 1$, this implies that there is a bit position that M looks at very rarely. Namely, there exists an i such that

$$\Pr[M \text{ looks at } i^{th} \text{ bit on input } 0^n] = o(1).$$

From this, we can deduce that M has approximately the same probability of accepting 0^n as it does 0^n with the i^{th} bit flipped. Therefore, M cannot compute parity.

We prove the case for $c \geq 1$ via indirect diagonalization. Suppose, by way of contradiction, that

$$\Sigma_2\text{TIME}(n) \subseteq \text{BPTISP}(n^c, n^d),$$

for some constant $d > 0$ to be determined later. Then for any time function $\tau(n)$, Lemma 4.11 gives us the complementations

$$\Sigma_2\text{TIME}(\tau) \subseteq \Pi_2\text{TIME}\left((\tau^{f_k} + n)^{c+d} \operatorname{polylog}(\tau + n)\right),$$

when $c + 2d \leq 2$ and $d \leq f_k$. Choosing τ so that $\tau(n)^{f_k} \geq n$ allows us to simplify this to

$$\Sigma_2\text{TIME}(\tau) \subseteq \Pi_2\text{TIME}(\tau^{(c+d)f_k} \operatorname{polylog}(\tau)). \tag{4.15}$$

The inclusion (4.15) gives a contradiction with Lemma 3.2 for any k with $f_k < \frac{1}{c+d}$. Note that $f_k \to 0$ as $k \to \infty$ if $c + 2d < 2$. Therefore, all that remains is to show that the latter condition is compatible with the other ones, i.e., that we can pick a constant $d > 0$ and an integer $k > 0$ such that

$$c + 2d < 2, \tag{4.16}$$

$$d \leq f_k, \text{ and} \tag{4.17}$$

$$f_k < \frac{1}{c+d}. \tag{4.18}$$

For any c and d satisfying (4.16), consider choosing $k \geq 1$ to be the smallest integer such that (4.18) is satisfied. Observe that $f_k \geq \frac{c+d}{2} f_{k-1}$, and by how we chose k, $f_{k-1} \geq \frac{1}{c+d}$. This shows that $f_k \geq 1/2$, so (4.17) is satisfied when $d \leq 1/2$. From (4.16), we have $d < \frac{2-c}{2}$, which is at most $1/2$ when $c \geq 1$. Therefore, choosing d such that $d < \frac{2-c}{2}$ and then calculating k as described above yields a d and k satisfying all of the constraints, leading to the desired contradiction. As c approaches 1 from above, $\frac{2-c}{2}$ approaches $1/2$ from below, so the largest value of d that yields a contradiction approaches $1/2$ as well. This proves that $\Sigma_2\text{TIME}(n) \not\subseteq \text{BPTISP}(n^c, n^d)$ for such c and d. □

We point out that, although we can handle the same values of c as in the deterministic setting, the dependence of d on c in Theorem 4.12 is worse. In particular, the proofs of the time-space lower bounds for deterministic machines show that d approaches 1 from below as c approaches 1 from above [FvM00, FLvMV05], while in our result d approaches $1/2$ from below as c approaches 1 from above.

4.4 Lower Bound for Σ_ℓSAT

The proof of Theorem 4.12 generalizes to $\Sigma_\ell\text{TIME}(n)$ for any $\ell \geq 3$. In this setting, the hypothesis becomes

$$\Sigma_\ell\text{TIME}(n) \subseteq \text{BPTISP}(n^c, n^d). \tag{4.19}$$

Along with the Π_2-simulation of $\text{BPTISP}(t, s)$ of Theorem 4.4, this yields a collapse of the form $\Sigma_\ell^p \subseteq \Pi_2^p$, which allows us to eliminate more than one alternation at the same cost of removing one alternation in the setting of Theorem 4.12 where $\ell = 2$. Therefore, we can afford to use more alternations using (3.4) and achieve a greater speedup. In a manner analogous to the proof of Theorem 4.12, we derive a series of increasingly efficient complementations of Σ_ℓ^p to Π_ℓ^p for $c < \ell$, eventually reaching a contradiction as long as $d \leq \frac{1}{\sqrt{\ell}}$.

Alternatively, we can use extra alternations to achieve a dependence of d on c where d approaches 1 from below as c approaches 1 from above, as in the deterministic case. For example, when $\ell = 3$, this follows by deriving a complementation of Σ_3^p following the same

technique that derives the complementation of Σ_2^p given by (4.11) when $\ell = 2$, modulo the replacement of the Π_2-simulation given by Theorem 4.4 with the Π_3-simulation given by Theorem 4.8. Specifically, hypothesis (4.19) and Theorem 4.8 give a complementation of Σ_3:

$$\Sigma_3\mathrm{TIME}(n) \subseteq \forall^{n^d \log n}\exists^{n^d \log n}\forall^{\log n}\mathrm{DTISP}(n^c \,\mathrm{polylog}(n), n^d \log n). \tag{4.20}$$

Consider a $\Sigma_3\mathrm{TIME}(\tau)$ computation for some time function τ to be determined. Applying the speedup of (3.2) to the final deterministic stage of the simulation given by (4.20) yields

$$\Sigma_3\mathrm{TIME}(\tau) \subseteq \forall^{\tau^d \log \tau} \underbrace{\exists^{\tau^d \log \tau}\Pi_2\mathrm{TIME}(\tau^{\frac{c+d}{2}} \,\mathrm{polylog}(\tau))}_{(\alpha)}.$$

We can now use (4.20) to simulate (α) by a Π_3-machine, yielding a more efficient complementation than (4.20) for certain values of c and d:

$$\Sigma_3\mathrm{TIME}(\tau) \subseteq \forall^{\tau^d \log \tau}\Pi_3\mathrm{TIME}\left((\tau^{\frac{c+d}{2}} + n + \tau^d)^c \,\mathrm{polylog}(\tau)\right).$$

When $\tau(n) \geq n^{\frac{2}{c+d}}$ (and $d \leq c$), this simplifies to

$$\Sigma_3\mathrm{TIME}(\tau) \subseteq \Pi_3\mathrm{TIME}(\tau^{c\frac{c+d}{2}} \,\mathrm{polylog}(\tau)), \tag{4.21}$$

yielding a contradiction to Lemma 3.2 when $c < \sqrt{2}$ and $d < \frac{2-c^2}{c}$. Therefore, as c approaches 1 from above, the upper bound on d approaches 1 from below when $\ell = 3$. Indeed, this analysis establishes the desired behavior of d as c approaches 1 for any level $\ell \geq 3$, since if (4.19) holds for $\ell > 3$, it must also hold for $\ell = 3$.

We point out that we can augment the strategy leading to (4.20) with a bootstrapping argument similar to the one in the proof of Lemma 4.11 and obtain increasingly efficient complementations of Σ_3^p. This approach results in a contradiction for $c < 2$, whereas the analogous approach using Theorem 4.4 leads to a contradiction for $c < 3$. More generally, at level $\ell \geq 3$, we can modify the above approach to take full advantage of the stronger hypothesis and arrive at complementations of Σ_ℓ^p. However, we only reach a contradiction for $c < \ell - 1$ whereas the strategy based on Theorem 4.4 results in a contradiction for $c < \ell$. Therefore, we need *both* approaches to prove Theorem 1.1 – the latter achieves the lower

bound for large values of c, while the former establishes the dependence of d on c for small values of c.

Since much of the proof of Theorem 1.1 closely follows the outline of Theorem 4.12, we give only a brief sketch of it here. In fact, Theorem 1.1 also follows from the more general Theorem 1.3, which gives time-space lower bounds for simulations of space-bounded linear-time alternating machines. A complete proof of Theorem 1.3 appears in the next section.

Proof of Theorem 1.1. The case for $c < 1$ follows by standard techniques, as in the proof of Theorem 4.12.

For the case $c \geq 1$, assume that $\Sigma_\ell \mathrm{TIME}(n) \subseteq \mathrm{BPTISP}(n^c, n^d)$ for some constant $d > 0$ to be determined later. Using the speedup given by (3.4) rather than (3.2), we can step through an argument similar to that of Lemma 4.11 to show

$$\Sigma_\ell \mathrm{TIME}(t) \subseteq \Pi_\ell \mathrm{TIME}\left((t^{g_k} + n)^{c+d} \operatorname{polylog}(t+n)\right), \tag{4.22}$$

as long as $c + \ell d \leq \ell$ and $d \leq g_k$, where

$$g_k = (\frac{c + \ell d}{\ell})^k. \tag{4.23}$$

We can now use (4.22) as we used Lemma 4.11 in the proof of Theorem 4.12. For a time function τ such that $\tau(n)^{g_k} \geq n$, (4.22) gives that

$$\Sigma_\ell \mathrm{TIME}(\tau) \subseteq \Pi_\ell \mathrm{TIME}\left(\tau^{(c+d)g_k}\right),$$

which is a contradiction with Lemma 3.2 when $g_k < \frac{1}{c+d}$. Therefore, it remains to show that it is possible to choose a positive d and integer k satisfying the latter condition as well as those on (4.22). More specifically, for $c < \ell$ we can choose $d < \frac{\ell - c}{\ell}$ so that there exists a smallest positive integer k such that $g_k < \frac{1}{c+d}$. Since $g_k = \frac{c+\ell d}{\ell} \cdot g_{k-1} \geq \frac{c+\ell d}{\ell} \cdot \frac{1}{c+d}$, we can guarantee that the only constraint possibly left unsatisfied, namely $d \leq g_k$, is met by restricting our choice of d to $d \leq \frac{c+\ell d}{\ell} \cdot \frac{1}{c+d}$.

When $\ell = 2$, the upper bound on d approaches $\frac{1}{2}$ as c approaches 1. For $\ell \geq 3$, the upper bound on d approaches $\frac{1}{\sqrt{\ell}}$ as c approaches 1 but we can improve it using the combination

of the hypothesis and Theorem 4.8 that leads to the complementation of Σ_3^p represented by (4.21). For large enough τ, this gives a contradiction for values of d approaching 1 from below as c approaches 1 from above when $\ell \geq 3$.

The tight completeness result of Theorem 2.3 transfers the lower bounds to Σ_ℓSAT. □

4.5 Lower Bound for Small-Space Alternating Machines

By paying close attention to the space used by the simulations in the proof of Theorem 1.1, we can actually obtain time-space lower bounds for randomized simulations of linear-time alternating machines using space n^a for $a < 1$, given by Theorem 1.3. The main task is to use the weaker assumption that $\Sigma_\ell\text{TISP}(n, n^a) \subseteq \text{BPTISP}(n^c, n^d)$ to eliminate the alternations introduced by the speedup of (3.3). This requires that the Σ_ℓ-simulation after the speedup uses an amount of space which is at most the a^{th} power of its running time. Since the simulation guesses (and stores) $O(bs)$ bits in each alternating stage, this restricts us to choose a small value of b, which in turn grants a smaller speedup. Therefore, our bounds become weaker as a becomes smaller.

To prove Theorem 1.3, we first prove an analog of Lemma 4.11 which gives a series of complementations of Σ_ℓTISP.

Lemma 4.13. *Suppose that*

$$\Sigma_\ell\text{TISP}(n, n^a) \subseteq \text{BPTISP}(n^c, n^d) \tag{4.24}$$

for some integer $\ell \geq 2$ and constants $0 < a \leq 1$, $c \geq 1$, and $d > 0$ with $c + \ell d \leq 1 + (\ell - 1)a$ and $(1 - a)d \leq ac$. Then for any time function t and integer $k \geq 0$ such that $d \leq h_k$,

$$\Sigma_\ell\text{TISP}(t, t^a) \subseteq \Pi_\ell\text{TISP}\left((t^{h_k} + n)^{c+d}\,\text{polylog}(t + n), (t + n)^d\,\text{polylog}(t + n)\right), \tag{4.25}$$

where

$$h_k \;=\; \left(\tfrac{c+\ell d}{1+(\ell-1)a}\right)^k. \tag{4.26}$$

Proof. The base case $k = 0$ is given by combining the hypothesis (4.24) and the efficient Π_2-simulation of BPTISP given by Theorem 4.4. We now prove the inductive step $k \to$

$k+1$. Consider a $\Sigma_\ell\text{TISP}(t, t^a)$ computation. From the hypothesis (4.24), Theorem 4.4, and the speedup of (3.3) (to Σ_ℓ rather than Π_ℓ), we obtain a simulation which (neglecting the polylog$(t+n)$ factors) is in

$$\forall^{(T+n)^d} \underbrace{\Sigma_\ell\text{TISP}\left(b(t+n)^d + \frac{(t+n)^{c+d}}{b^{\ell-1}}, b(t+n)^d\right)}_{(\alpha)}.$$

In order to apply the inductive hypothesis to complete the complementation to Π_ℓTISP, the space bound of (α) must be at most the a^{th} power of its running time. This is the case when b satisfies

$$b(t+n)^d = \left(\frac{(t+n)^{c+d}}{b^{\ell-1}}\right)^a,$$

which offers the choice

$$b = (t+n)^{\frac{ac+(a-1)d}{1+(\ell-1)a}}$$

as long as $ac + (a-1)d \geq 0$. For such a value, (α) is a computation in

$$\Sigma_\ell\text{TISP}\left((t+n)^{\frac{c+\ell d}{1+(\ell-1)a}}, (t+n)^{a\frac{c+\ell d}{1+(\ell-1)a}}\right),$$

taking an input of size $O(n + (t+n)^d)$, which is in $O(n+t^d)$ when $d \leq 1$. Thus, instead of achieving an ℓ^{th} root speedup as we do when we are not concerned about the space used by the simulation of (3.3), we instead achieve a $(1+(\ell-1)a)^{\text{th}}$ root speedup.

The inductive hypothesis gives a Π_ℓTISP-simulation of (α), and hence a Π_ℓTISP-simulation of $\Sigma_\ell\text{TISP}(t, t^a)$ running in time big-O of (neglecting the polylog$(t+n)$ terms)

$$t^* \doteq (t+n)^d + \left((t+n)^{h_k\frac{c+\ell d}{1+(\ell-1)a}} + n + t^d\right)^{c+d}$$

and using space big-O of

$$s^* \doteq (t+n)^d + \left((t+n)^{\frac{c+\ell d}{1+(\ell-1)a}} + n + t^d\right)^d.$$

When $c + \ell d \leq 1 + (\ell-1)a$ and $d \leq 1$, s^* simplifies to $(t+n)^d$. When we have the further constraint that $d \leq h_{k+1}$, t^* has the appropriate leading terms and simplifies to $(t^{h_{k+1}} + n)^{c+d}$. Accounting for the polylog factors, we have shown that the simulation is of the desired form. □

We note that the proof of Lemma 4.13 also yields a version in which the Π_ℓ on the right-hand side of (4.25) is replaced by Π_2. However, we do not see a way to exploit that fact to strengthen our final result. For the sake of clarity and consistency, we present Lemma 4.13 as stated.

When $c + \ell d < 1 + (\ell - 1)a$, $h_k \to 0$ as $k \to \infty$. In such a case, we can use Lemma 4.13 to derive a contradiction to Lemma 3.3 in an analogous fashion to how we used Lemma 4.11 to prove Theorem 4.12 for values of c as large as possible. For $\ell = 2$, this gives a value of d approaching $\frac{a}{2}$ from below as c approaches 1 from above. We can do better when $\ell \geq 3$ by using Theorem 4.8 to derive a space-bounded complementation analogous to that given by (4.21) in the unrestricted case.

Lemma 4.14. *Suppose that*

$$\Sigma_3\mathrm{TISP}(n, n^a) \subseteq \mathrm{BPTISP}(n^c, n^d) \tag{4.27}$$

for constants $0 < a \leq 1$, $c \geq 1$ and $0 < d \leq ac$ with $c + d \leq 1 + a$. Then for any time function t,

$$\Sigma_3\mathrm{TISP}(t, t^a) \subseteq \Pi_3\mathrm{TISP}\left((t^{\frac{c+d}{1+a}} + n)^c \operatorname{polylog}(t+n), (t+n)^d \operatorname{polylog}(t+n)\right).$$

Proof. The hypothesis (4.27) and Theorem 4.8 yield the complementation

$$\Sigma_3\mathrm{TISP}(n, n^a) \subseteq \forall^{n^d \log n} \exists^{n^d \log^2 n} \forall^{\log n} \mathrm{DTISP}(n^c \operatorname{polylog}(n), n^d \log n). \tag{4.28}$$

Consider a $\Sigma_3\mathrm{TISP}(t, t^a)$ computation. Padding (4.28) and applying the speedup of (3.1) (on the Π_2-side) to the final deterministic stage gives a simulation of $\Sigma_3\mathrm{TISP}(t, t^a)$ which (neglecting the polylog terms) is in

$$\forall^{(t+n)^d} \exists^{(t+n)^d} \underbrace{\Pi_2\mathrm{TISP}(b(t+n)^d + (t+n)^c/b, b(t+n)^d)}_{(\beta)}.$$

Choosing $b = (t+n)^{\frac{ac-d}{1+a}}$ so that the in the space bound of (β) is the a^{th} power of its running time places the simulation in

$$\forall^{(t+n)^d} \underbrace{\exists^{(t+n)^d} \Pi_2\mathrm{TISP}((t+n)^{\frac{c+d}{1+a}}, (t+n)^{a\frac{c+d}{1+a}})}_{(\gamma)},$$

when $d \leq ac$. Under the same condition, (γ) is a computation in $\Sigma_3\mathrm{TISP}((t+n)^{\frac{c+d}{1+a}}, (t+n)^{a\frac{c+d}{1+a}})$ taking an input of size $n+(t+n)^d$, so that (4.28) gives a simulation of (γ) in $\Pi_3\mathrm{TISP}$. Overall we have derived

$$\begin{aligned}
&\Sigma_3\mathrm{TISP}(t, t^a) \\
&\subseteq \Pi_3\mathrm{TISP}\left(((t+n)^{\frac{c+d}{1+a}} + n + (t+n)^d)^c, (t+n)^d + ((t+n)^{\frac{c+d}{1+a}} + n + (t+n)^d)^d\right) \\
&\subseteq \Pi_3\mathrm{TISP}\left((t^{\frac{c+d}{1+a}} + n)^c, (t+n)^d\right),
\end{aligned}$$

where the last inclusion follows as long as $d \leq ac$ and $\frac{c+d}{1+a} \leq 1$. Accounting for the polylog$(t+n)$ terms finishes the proof. □

We are now ready to prove Theorem 1.3.

Proof of Theorem 1.3. The proof for $c < 1$ follows from standard techniques, as in the proof of Theorem 4.12.

For the case $c \geq 1$, assume by way of contradiction that $\Sigma_\ell\mathrm{TISP}(n, n^a) \subseteq \mathrm{BPTISP}(n^c, n^d)$ for some constant $d > 0$ to be determined later. Then for a time function τ where $\tau(n)^{h_k} \geq n$, Lemma 4.13 gives the complementation

$$\Sigma_\ell\mathrm{TISP}(\tau, \tau^a) \subseteq \Pi_\ell\mathrm{TISP}\left(\tau^{(c+d)h_k}\,\mathrm{polylog}(\tau), \tau^d\,\mathrm{polylog}(\tau)\right) \tag{4.29}$$

when $c + \ell d \leq 1 + (\ell-1)a$, $(1-a)d \leq ac$, and $d \leq h_k$. When $h_k < \frac{1}{c+d}$, the time bound of the right-hand side of (4.29) is $\tau^{1-\epsilon}$ for positive ϵ. Provided that $c < 1 + (\ell-1)a$, we can choose a positive d and an integer k such that all the above conditions are met. More specifically, for any value of $d < \frac{1+(\ell-1)a-c}{\ell}$, there exists a smallest positive integer k such that $h_k < \frac{1}{c+d}$. Since $h_k = \frac{c+\ell d}{1+(\ell-1)a} \cdot h_{k-1} \geq \frac{c+\ell d}{1+(\ell-1)a} \cdot \frac{1}{c+d}$, we can guarantee that $d \leq h_k$ by imposing the condition

$$d \leq \frac{c+\ell d}{1+(\ell-1)a} \cdot \frac{1}{c+d}. \tag{4.30}$$

It follows that we can meet all constraints mentioned so far by choosing d below some positive threshold. All that remains to reach a contradiction with Lemma 3.3 is to ensure that the

space bound of the right-hand side of (4.29) is $\tau^{a-\epsilon}$ for some positive ϵ, which can be done with the additional constraint on d that $d < a$.

For $\ell = 2$ the upper bound on d approaches $\frac{a}{2}$ from below as c approaches 1 from above. In the case $\ell \geq 3$ the upper bound on d imposed by the conditions other than (4.30) approaches $\frac{\ell-1}{\ell}a$ as c approaches 1; condition (4.30) implies an upper bound of $1/\sqrt{\ell}$ for $a = 1$ and a somewhat weaker bound for smaller values of a. However, we can achieve a contradiction for larger d as c approaches 1 and $\ell \geq 3$ using the following argument. For $\tau(n) \geq n^{\frac{1+a}{c+d}}$, Lemma 4.14 gives

$$\Sigma_3\mathrm{TISP}(\tau, \tau^a) \subseteq \Pi_3\mathrm{TISP}(\tau^{c\frac{c+d}{1+a}}\,\mathrm{polylog}(\tau), \tau^d\,\mathrm{polylog}(\tau))$$

when $c + d \leq 1 + a$ and $d \leq ac$. When $c < \sqrt{1+a}$, we can choose $d < \min(\frac{(1+a)-c^2}{c}, a)$ and arrive at a contradiction with Lemma 3.3. As c approaches 1 from above, the upper bound on d approaches a from below, which gives the desired dependence. □

4.6 Conclusion and Open Problems

This chapter presents the first polynomial-strength time-space lower bounds for natural complete problems in the polynomial-time hierarchy on randomized machines. The key ingredient is a time- and space- efficient alternating simulation of randomized machines. We established such a simulation at the second level by exploiting the space bound, and at the third level by additionally harnessing the power of alternations to quickly carry out randomness-efficient error reduction. These simulations result in complexity-class inclusions that are interesting in their own right. In fact, Viola [Vio07] studied second-level simulations of BPP and showed that black-box techniques result in running times that are at least quadratic; thus, our exploitation of the space bound, which is a non-black-box consideration, is *necessary* to achieve our quasilinear-time simulation.

The results in this chapter do not seem to extend to the first-level problems of satisfiability or tautologies in a straightforward way. This can be intuitively understood by the fact that we know how to construct an efficient simulation of BPP in the second level of the

polynomial hierarchy, but not the first. This causes our inability to exploit the assumption $\mathrm{NTIME}(n) \subseteq \mathrm{BPTISP}(n^c, n^d)$ to obtain an efficient complementation at some level of the polynomial-time hierarchy. Thus, establishing time-space lower bounds for satisfiability on randomized machines with two-sided error remains an important open problem. We prove some results towards this goal in the next two chapters.

Another interesting open problem is to improve the quantitative strength of our lower bounds. Recent developments in the deterministic setting have resulted in time-space lower bounds for $\Sigma_\ell\mathrm{SAT}$ that lie somewhere in between n^ℓ and $n^{\ell+1}$, cf. [vM07]. It might be possible to use some of these techniques to obtain better lower bounds in the randomized setting as well. However, there is evidence that a lower bound of $n^{\ell+1}$ for $\Sigma_\ell\mathrm{SAT}$ is the best that current techniques can possibly achieve in the deterministic setting [Wil07a]; thus, any improvements to our randomized bounds are subject to the same limit.

Chapter 5

Lower Bounds for Arthur-Merlin Games

In this chapter we prove our lower bounds for simulations of MA-games: first those for black-box AM-simulations given by Theorem 1.4, and then the time-space lower bounds for randomized simulations of MA-games given by Corollary 1.5.

5.1 Lower Bounds for Swapping Arthur and Merlin

We restate Theorem 1.4 for convenience.

Theorem 1.4 (restated). *For any time function t, there is no black-box simulation of* MA*-games running in time t by* AM*-games running in time $o(t^2)$.*

The proof of Theorem 1.4 is inspired by work of Viola [Vio07], where he proves a quadratic time lower bound for black-box simulations of BPP in the second level of the polynomial-time hierarchy. Viola's result builds on a switching lemma of Segerlind, Buss, and Impagliazzo [SBI04] — later improved upon by Razborov [Raz03] — that uses random restrictions setting very few variables.

This lower bound for BPP yields interesting corollaries that black-box simulations of standard two-sided error AM- or MA-games by games of the respective type with perfect completeness require a quadratic time overhead. This is because the two-sided error games contain BPP, while the games with perfect completeness are (by definition) contained in the second level of the polynomial-time hierarchy. However, since the transformation to the second level requires a quadratic time overhead for *both* MA and AM, this does not rule

out a linear-overhead, black-box simulation of MA by AM in the standard two-sided error model. Therefore, something more is needed.

Our proof can be summed up as follows: first, we transform the setting from AM-games into systems of disjunctive normal-form (DNF) formulas. This follows by viewing the behavior of an AM-game after Arthur sends his coin tosses as a nondeterministic computation where each guess corresponds to a possible message from Merlin. The black-box setting enforces that the outcome after the nondeterministic guess is determined by some number of trials $k(n)$ of the MA-game being simulated. Thus, by the standard polynomial-time hierarchy-to-circuit connection [FSS84], the result of the game after Arthur's move can be viewed as the output of an exponential-size, depth-two OR-of-ANDs circuit with bottom fan-in k— i.e., a k-DNF.

Next, we show that the ideas underlying the switching lemma of Segerlind et al. [SBI04, Raz03] and its adaptation by Viola [Vio07] present us with a dichotomy for any such k-DNF: either (i) its terms are spread out over the input variables and it accepts very often on random inputs, or (ii) its terms are focused on a small set of variables, elevating their influence above the rest. Our proof takes advantage of this dichotomy by constructing two distributions on the outcomes of MA-games: one that generates mostly "no" instances and one that generates mostly "yes" instances. We show that, for small k, if a k-DNF is of type (i), then it accepts on most instances of the "no" distribution; if the k-DNF is of type (ii), then it depends (with non-negligible probability) on too few variables to discern the difference between the "yes" and "no" distributions. This means Arthur's actions either put Merlin in a position to wrongfully convince Arthur of false claims or prevent Merlin from helping Arthur separate fact from fallacy with bounded error— either case is a failure. Therefore, the number of trials k made by the AM-game cannot be small, which leads to the lower bound.

The remainder of this chapter focuses on proving Theorem 1.4. However, we first need to concretely establish what we mean by a "black-box" simulation.

5.1.1 Black-Box Simulations

We begin by formally defining the class of languages recognizable by a time-bounded AM-game. For any time function t, we say that a language $L \in$ AM-TIME(t) if there exists a deterministic Turing machine V that takes three inputs and runs in time $t(n)$ such that for any x,

$$x \in L \;\Rightarrow\; \Pr_{|z|=t(n)}[\exists y \in \{0,1\}^{t(n)} \text{ such that } V(x,y,z)=1] \geq 2/3 \tag{5.1}$$

$$x \notin L \;\Rightarrow\; \Pr_{|z|=t(n)}[\exists y \in \{0,1\}^{t(n)} \text{ such that } V(x,y,z)=1] \leq 1/3, \tag{5.2}$$

where $n = |x|$ and the probabilities are taken over the uniform distribution.

Similarly, the class of languages recognized by MA-games bounded by time t, MA-TIME(t), are the languages L where there is a deterministic, time $t(n)$ Turing machine V such that for any x,

$$x \in L \;\Rightarrow\; \exists y \in \{0,1\}^{t(n)} : \Pr_{|z|=t(n)}[V(x,y,z)=1] \geq 2/3 \tag{5.3}$$

$$x \notin L \;\Rightarrow\; \forall y \in \{0,1\}^{t(n)} : \Pr_{|z|=t(n)}[V(x,y,z)=1] \leq 1/3. \tag{5.4}$$

An AM- or MA-game has *perfect completeness* if the acceptance probability in the case that $x \in L$ (i.e., the right-hand side of (5.1) or, respectively, (5.3)) is replaced by 1 in the above definitions. The resulting classes are referred to as AM-TIME$_1$(t) and MA-TIME$_1$(t), respectively.

We often refer to the Turing machine V as the underlying predicate, the string x as the problem input, y as Merlin's message, and z as Arthur's message.

We now precisely define the notion of a black-box simulation. This is meant to capture any method of showing that MA is contained in AM by the design of a "modular" AM-game, into which one can plug *any* MA-game's underlying predicate V as a subroutine (i.e., it is oblivious to the MA-games it simulates[1]). Such a simulation uses only the results of queries to V to decide an answer and is indifferent to the computational details of how these answers

[1]We point out the difference to a relativizing inclusion, where we can have a different simulation for each oracle.

are obtained: it should arrive at the right answer provided only that V satisfies the promise of an MA-game given by conditions (5.3) and (5.4). Therefore, a black-box AM-simulation of MA is an AM-game that solves the *promise problem* of whether or not a given subroutine satisfies (5.3) or (5.4) on input x.

We cast this promise problem as one on the (exponentially long) characteristic vector of the given subroutine on a fixed x. To do so, we first define the promise problem ApprMaj corresponding to the acceptance condition of a probabilistic computation with bounded error. This allows us to define the desired promise problem $\exists$-ApprMaj as one on matrices, where each row corresponds to a possible proof from Merlin, and each column corresponds to a possibility for Arthur's coin tosses.

Definition 5.1. ApprMaj *is the promise problem on Boolean strings where*

$$\begin{aligned} \text{ApprMaj}_Y &= \{r \mid \textit{at least } 2|r|/3 \textit{ bits of } r \textit{ are set to 1}\}\ , \\ \text{ApprMaj}_N &= \{r \mid \textit{at most } |r|/3 \textit{ bits of } r \textit{ are set to 1}\}. \end{aligned}$$

$\exists$-ApprMaj *is the promise problem on Boolean matrices where*

$$\begin{aligned} \exists\text{-ApprMaj}_Y &= \{M \mid \textit{at least one row of } M \textit{ belongs to } \text{ApprMaj}_Y\}, \\ \exists\text{-ApprMaj}_N &= \{M \mid \textit{every row of } M \textit{ belongs to } \text{ApprMaj}_N\}. \end{aligned}$$

Thus, the definition of an AM-game that efficiently black-box simulates MA fitting our intuition is one that solves $\exists$-ApprMaj on appropriately-sized matrices with few queries. We charge the simulation t time units for each query, where t is the running-time of the simulated MA-game, to capture the fact that a black-box simulation must calculate the bits of the $\exists$-ApprMaj instance on the fly by running trials of the MA-game.

Definition 5.2. *We say that an* AM*-game black-box q-simulates* MA*-games running in time $t(n)$ if there is a polynomial $p(n)$ and an oracle machine S such that for any input $M \in \{0,1\}^{2^t \times 2^t}$,*

$$\begin{aligned} M \in \exists\text{-ApprMaj}_Y &\Rightarrow \Pr_{|z|=p(n)}[\exists y \in \{0,1\}^{p(n)} \textit{ such that } S^M(y,z)=1] \geq 2/3 \\ M \in \exists\text{-ApprMaj}_N &\Rightarrow \Pr_{|z|=p(n)}[\exists y \in \{0,1\}^{p(n)} \textit{ such that } S^M(y,z)=1] \leq 1/3, \end{aligned}$$

where S queries M in at most $q(n)$ locations.

We say that an AM*-game running in time $t'(n)$ black-box simulates* MA*-games running in time $t(n)$ if there is an* AM*-game that black-box q-simulates such* MA*-games for $q = t'/t$ and $p = t'$.*

We prove the lower bound of Theorem 1.4 by showing that any AM-game black-box q-simulating MA-games running in time t must have $q = \Omega(t)$; therefore, the running time of such a simulation must be at least $\Omega(t^2)$. In fact, the other computational aspects of the AM-simulation, such as the bounds on the message lengths or the running-time of the underlying predicate V, turn out to be irrelevant to the number of required queries. Therefore, we simplify the situation by proving the query lower bound via a lower bound on the *bottom fan-in* of depth-3 circuits for $\exists$-ApprMaj (where by bottom fan-in we mean the fan-in of the gates adjacent to the input literals). In particular, the circuits we consider have the same correspondence to AM-games as standard constant-depth circuits have to the polynomial-time hierarchy [FSS84].

Definition 5.3. *An* AM*-circuit is a depth-3 circuit with literals as inputs (variables or their complements) whose output gate is an oracle gate for* ApprMaj *and each depth-2 subcircuit is an OR of AND's (i.e., a DNF); furthermore, we are guaranteed that the input to the* ApprMaj *gate formed by the depth-2 subcircuits on any input X is either in* ApprMaj_Y *or* ApprMaj_N.

The aforementioned connection to AM-games is given by the following proposition.

Proposition 5.4. *If an* AM*-game black-box q-simulates* MA*-games running in time t, then there is a family of* AM*-circuits with bottom fan-in q computing* $\exists$-ApprMaj *on matrices of size $2^t \times 2^t$.*

Proof. We model Arthur's probabilistic move by the top ApprMaj gate, and Merlin's non-deterministic move by the middle-layer's OR gate. The verification that takes place after Merlin's message is sent is a function that depends on at most q oracle bits; such a function

can be decided by a depth-two OR-of-ANDs circuit with bottom fan-in q. Furthermore, the guarantee that the inputs to the top gate are either in ApprMaj_Y or ApprMaj_N is satisfied by the conditions imposed on the game by Definition 5.2. □

5.1.2 Proof of the Lower Bound

Proposition 5.4 allows us to prove Theorem 1.4 by showing a lower bound on the bottom fan-in of AM-circuits for ∃-ApprMaj.

Theorem 5.5. *Any* AM*-circuit solving* ∃-ApprMaj *on* $2^t \times 2^t$ *matrices has bottom fan-in* $\Omega(t)$.

As outlined, the main idea behind the proof of Theorem 5.5 stems from an analysis of the k-DNF subcircuits of an AM-circuit on different input distributions. For an AM-circuit with bottom fan-in k to compute ∃-ApprMaj, Definition 5.3 requires most of its k-DNF subcircuits to accept when the input is drawn from a distribution producing mostly "yes" instances of ∃-ApprMaj and most to reject when the input is drawn from a distribution producing mostly "no" instances. By the union bound, a nontrivial fraction of the subcircuits output the correct answer most of the time for *both* the "yes" and "no" distributions. We construct specific "yes" and "no" distributions such that any k-DNF with $k = o(t)$ cannot do both at the same time; therefore, such an AM-circuit for ∃-ApprMaj cannot exist.

We begin by defining the distributions producing mostly "yes" instances and "no" instances of ∃-ApprMaj that we use to prove the lower bound.

Definition 5.6. *Let* $\mathcal{D}_N^{n\times m}$ *be the distribution on* $n \times m$ *matrices obtained by assigning each entry independently to 1 with probability 1/6 and 0 with probability 5/6.*

Furthermore, let $\mathcal{D}_Y^{n\times m}$ *be the distribution on* $n \times m$ *matrices obtained by choosing one row uniformly at random and setting each entry within this row to 1, while the remaining entries are set as in* $\mathcal{D}_N^{n\times m}$.

When n *and* m *are clear from context, we simply use* $\mathcal{D}_Y$ *and* $\mathcal{D}_N$.

We claim that when $m = \Omega(\log n)$, $\mathcal{D}_Y$ generates instances in $\exists$-ApprMaj$_Y$ and $\mathcal{D}_N$ generates instances in $\exists$-ApprMaj$_N$ with high probability.

Claim 5.7. *There exists a constant $\alpha > 0$ such that for large enough n and $m \geq \alpha \log n$,*

$$\Pr_{M \in \mathcal{D}_Y^{n \times m}}[M \in \exists\text{-ApprMaj}_Y] = 1 \text{ and } \Pr_{M \in \mathcal{D}_N^{n \times m}}[M \in \exists\text{-ApprMaj}_N] \geq (1 - 1/n).$$

Proof. The case of $\mathcal{D}_Y$ is trivial, since we always set some row to all 1's. For the case of $\mathcal{D}_N$, we expect there to be $m/6$ 1's in each row. Therefore, a row of M has more than $m/3$ 1's (i.e., is in not ApprMaj$_N$) with probability at most $2^{-\beta m}$ for some $\beta > 0$ by the Chernoff bound. Then the probability that *some* row is not in ApprMaj$_N$ (and therefore, $M \notin \exists$-ApprMaj$_N$) is at most $n2^{-\beta m}$ by the union bound; The claim follows for $\alpha = 2/\beta$. □

To analyze the relationship between the probability that a given k-DNF φ accepts on $\mathcal{D}_Y$ and the probability that it accepts on $\mathcal{D}_N$, we introduce a "lazy" procedure, Eval, that tries to evaluate $\varphi(M)$ (see Figure 5.1). Given φ, input M, and a parameter s, Eval attempts to reduce the problem of evaluating a k-DNF to that of evaluating a $(k-1)$-DNF by querying the values of a set of variables that *cover* the terms of φ (a cover, or hitting set, of a DNF ψ is a subset Γ of the variables such that each term of ψ contains at least one variable in Γ). Eval then restricts φ appropriately to obtain a $(k-1)$-DNF φ', whose evaluation on the remaining variables is the same as the evaluation of φ, and recurses. However, Eval is "lazy" in the sense that it refuses to query more than s variables at each step; if at any step it becomes impossible to achieve the above term-size reduction in such few queries, Eval simply gives up and outputs the current subformula. Thus, Eval has three possible outputs: 0, 1, or a restriction of φ that has a large minimum cover. We point out that if Eval outputs 0, then $\varphi(M) = 0$, and if Eval outputs 1, then $\varphi(M) = 1$; however, the converse does not hold in either case due to Eval's laziness.

Our proof requires the cover size s to balance two desires: (i) We need s large enough so that if Eval outputs a formula ψ, then ψ is almost certainly satisfied by an input from $\mathcal{D}_N$, and (ii) we need s small enough so that Eval has not queried many variables when it arrives at a 0/1 answer. Property (i) ensures that the non-Boolean output case of Eval

Procedure **Eval**(φ, M, s)

If $\varphi \equiv 0$, **output** 0.

Else if $\varphi \equiv 1$, **output** 1.

Else if every cover of φ has size greater than s, **output** φ.

Else

$\Gamma \leftarrow$ a cover of φ of size at most s.

Query the variables in Γ to obtain the partial assignment $\Gamma(M)$.

$\varphi' \leftarrow \varphi|_{\Gamma(M)}$.

Eval(φ', M, s).

Figure 5.1 The "lazy" procedure, Eval.

cannot happen very often for φ a k-DNF in an AM-circuit computing $\exists$-ApprMaj on $\mathcal{D}_N$, so the output of Eval(φ, M, s) almost always matches the evaluation of $\varphi(M)$ when $M \in \mathcal{D}_N$. Property (ii) guarantees that the event of Eval(φ, M, s) answering either 0 or 1 depends very weakly on most rows, which is not enough for Eval to distinguish the one-row difference between samples from $\mathcal{D}_Y$ and $\mathcal{D}_N$. Therefore, if φ behaves well on $\mathcal{D}_N$ and outputs 0 most of the time, then Eval(φ, M, s) not only outputs 0 most of the time for $M \in \mathcal{D}_N$ by (i), but also for $M \in \mathcal{D}_Y$ by (ii), so φ rejects too often on $\mathcal{D}_Y$.

We first quantify the effect of property (i) on the correctness of Eval on $\mathcal{D}_N$.

Lemma 5.8. *If φ is a k-DNF, then*

$$\Pr_{M \in \mathcal{D}_N}[\text{Eval}(\varphi, M, s) \neq 0] \leq \frac{\Pr_{M \in \mathcal{D}_N}[\varphi(M) = 1]}{1 - e^{-\frac{s}{k6^k}}}.$$

The proof begins by showing that any DNF ψ output by Eval must have many variable-disjoint terms due to its large cover size. These terms present too many independent events that must align in order for ψ to reject reliably on $\mathcal{D}_N$.

Lemma 5.9. *Let φ be a k-DNF. Then*

$$\Pr_{M \in \mathcal{D}_N}[\varphi(M) = 0 | \text{Eval}(\varphi, M, s) \notin \{0, 1\}] \leq e^{-\frac{s}{k6^k}}.$$

Proof. Suppose that Eval outputs some formula ψ. Notice that $\psi \not\equiv 0$ is a restriction of φ, so it is a k'-DNF for $k' \leq k$. Thus, a term of ψ is satisfied by $\mathcal{D}_N$ with probability at least $1/6^k$. We claim that ψ must have more than s/k variable-disjoint terms Γ. The lemma follows if this is the case, since the events that a term in Γ is satisfied are independent, so

$$\Pr_{M \in \mathcal{D}_N}[\psi(M) = 0] \leq \Pr_{M \in \mathcal{D}_N}[\text{Each clause in } \Gamma \text{ is not satisfied}] \leq (1 - 1/6^k)^{s/k} \leq e^{-\frac{s}{k6^k}}.$$

To prove the claim, consider a maximal set Γ of variable-disjoint terms of ψ. There are at most $|\Gamma|k$ variables in these terms, and they must form a cover of ψ; if they do not, then there is a term t that is not covered by the variables of Γ, i.e., t does not contain any of the variables in Γ. Thus, $\Gamma \cup \{t\}$ is a set of variable-disjoint terms, which contradicts the maximality of Γ. Furthermore, since ψ is a formula output by Eval, it cannot have a cover of size at most s. Therefore, we have that $|\Gamma|k > s$, so $|\Gamma| > s/k$ as claimed. □

Thus, since the formulas on which Eval does not give a 0/1 output are very likely to accept $\mathcal{D}_N$ by Lemma 5.9, Eval cannot give such an output often on a formula φ that largely rejects $\mathcal{D}_N$. Therefore, Eval must correctly output 0 for φ on most of these instances despite its laziness, yielding Lemma 5.8.

Proof of Lemma 5.8. We have noted that if $\varphi(M) = 1$, then $\text{Eval}(\varphi, M, s)$ must output either 1 or a formula, so

$$\Pr_{M \in \mathcal{D}_N}[\varphi(M) = 1] = \Pr_{M \in \mathcal{D}_N}[\text{Eval} = 1] + \Pr_{M \in \mathcal{D}_N}[\text{Eval} \notin \{0,1\}] \cdot \underbrace{\Pr_{M \in \mathcal{D}_N}[\varphi(M) = 1 | \text{Eval} \notin \{0,1\}]}_{(*)}.$$

By Lemma 5.9, we have that $(*)$ is at least $(1 - e^{-\frac{s}{k6^k}})$, so we conclude that

$$\begin{aligned} \Pr_{M \in \mathcal{D}_N}[\varphi(M) = 1] &\geq \left(\Pr_{M \in \mathcal{D}_N}[\text{Eval} = 1] + \Pr_{M \in \mathcal{D}_N}[\text{Eval} \notin \{0,1\}] \right) (1 - e^{-\frac{s}{k6^k}}) \\ &= \Pr_{M \in \mathcal{D}_N}[\text{Eval} \neq 0](1 - e^{-\frac{s}{k6^k}}). \end{aligned}$$

□

We now turn to quantify the effect of property (ii) on the relationship between Eval's behavior on $\mathcal{D}_Y$ and $\mathcal{D}_N$.

Lemma 5.10. *Let φ be a k-DNF. Then for any $\delta \geq \frac{sk}{n}$,*

$$\Pr_{M \in \mathcal{D}_Y}[\text{Eval}(\varphi, M, s) = 0] \geq (1 - \frac{sk}{\delta n}) \left(\Pr_{M \in \mathcal{D}_N}[\text{Eval}(\varphi, M, s) = 0] - \delta \right).$$

Proof. Recall that when the formula φ passed to Eval has small cover size, Eval reduces the term size of φ by one to obtain φ' (since Γ is a cover of φ). Therefore, Eval can make at most k recursive calls on a k-DNF. At each such call, Eval queries at most s variables, for a total of at most sk variables queried during the entire run. Different outcomes for the sample M cause Eval to form different subformulas φ' with (possibly) different covers, so the variables queried by Eval follow some probability distribution determined by the input distribution. We show that most variables, and in fact most rows, are queried with very small probability.

For a fixed k-DNF φ, s, and $\delta > 0$, define B as

$$B \doteq \{i | \Pr_{M \in \mathcal{D}_N}[\text{Eval}(\varphi, M, s) \text{ queries row } i] \geq \delta\}, \tag{5.5}$$

the set of rows queried by Eval with probability at least δ. Since any individual run of Eval queries at most sk variables, we have that

$$\sum_i \Pr_{M \in \mathcal{D}_N}[\text{row } i \text{ is queried by Eval}(\varphi, M, s)] \leq sk.$$

Therefore, B cannot be too large:

$$|B| \leq \frac{sk}{\delta}. \tag{5.6}$$

So (5.5) and (5.6) tell us that, for most rows i, Eval queries a variable in row i with very small probability when $\delta > \frac{sk}{n}$.

We would like to isolate the cases of $\mathcal{D}_Y$ that choose a row outside of B to set to 1. Towards this end, define D_Y^i to be the distribution on $n \times m$ matrices where each entry in row i is assigned 1 and each other entry is assigned 0 with probability 5/6 (and 1 otherwise). Then sampling M from $\mathcal{D}_Y$ is equivalent to choosing $i \in \{1, \ldots, n\}$ at random and sampling M from $\mathcal{D}_Y^i$. This allows us to divide the probability that Eval rejects on inputs from $\mathcal{D}_Y$ into cases by row. Discarding the rows in B, we have

$$\Pr_{M \in \mathcal{D}_Y}[\text{Eval}(\varphi, M, s) = 0] \geq \sum_{i \notin B} \Pr_{M \in \mathcal{D}_Y^i}[\text{Eval}(\varphi, M, s) = 0]/n. \tag{5.7}$$

Now consider how $\text{Eval}(\varphi, M, s)$ differs when M is drawn from $\mathcal{D}_N$ and when it is drawn from $\mathcal{D}_Y^i$ where $i \notin B$. Notice that the only time that Eval queries variables which are distributed differently in $\mathcal{D}_N$ and $\mathcal{D}_Y^i$ is when it queries a variable in row i. We know that this happens with probability less than δ under $\mathcal{D}_N$ by (5.5); this is also the case under $\mathcal{D}_Y^i$, since the distribution is identical to $\mathcal{D}_N$ outside of row i. Thus, the outcome of Eval on $\mathcal{D}_N$ and $\mathcal{D}_Y^i$ can be different on at most a δ fraction of inputs for $i \notin B$,

$$\Pr_{M \in \mathcal{D}_Y^i}[\text{Eval}(\varphi, M, s) = 0] \geq \Pr_{M \in \mathcal{D}_N}[\text{Eval}(\varphi, M, s) = 0] - \delta. \tag{5.8}$$

The lemma follows by combining (5.6), (5.7), and (5.8). □

The two properties captured by Lemma 5.8 and Lemma 5.10 formalize the contradictory dichotomy that allows us to prove Theorem 5.5.

Proof of Theorem 5.5. Let $n = m = 2^t$ and C be an AM-circuit with bottom fan-in k purported to solve $\exists$-ApprMaj. Without loss of generality, we assume that 9/10 of C's DNF subcircuits accept on $M \in \exists\text{-ApprMaj}_Y$ while 9/10 reject on $M \in \exists\text{-ApprMaj}_N$—this can be achieved by naïve error reduction by majority vote while increasing the bottom fan-in by only a constant factor.

Consider sampling M from $\mathcal{D}_Y$ or $\mathcal{D}_N$ and a random depth-2 subcircuit φ connected to the output gate of C. Then by Claim 5.7,

$$\Pr_{M \in \mathcal{D}_Y, \varphi \in_U C}[\varphi(M) = 1] \geq \frac{9}{10} \text{ and } \Pr_{M \in \mathcal{D}_N, \varphi \in_U C}[\varphi(M) = 0] \geq \frac{9}{10}(1 - \frac{1}{n}) > 5/6$$

for large enough n. Therefore, the probability that a random subcircuit is correct on most instances of $\mathcal{D}_Y$ is greater than 1/2 (and similarly for $\mathcal{D}_N$):

$$\Pr_{\varphi \in_U C}[\Pr_{M \in \mathcal{D}_Y}[\varphi(M) = 1] > 2/3] > 1/2 \text{ and } \Pr_{\varphi \in_U C}[\Pr_{M \in \mathcal{D}_N}[\varphi(M) = 0] > 2/3] > 1/2.$$

Thus, by a union bound, there is a k-DNF φ^* where

$$\Pr_{M \in \mathcal{D}_Y}[\varphi^*(M) = 0] \leq 1/3 \text{ and } \Pr_{M \in \mathcal{D}_N}[\varphi^*(M) = 1] \leq 1/3 \tag{5.9}$$

Our proof proceeds by showing that such a k-DNF $\varphi*$ cannot exist when $k = o(\log n)$. By Lemma 5.8, we have that

$$\Pr_{M \in \mathcal{D}_N}[\text{Eval}(\varphi^*, M, s) = 0] \geq 1 - 1/\left(3(1 - e^{-\frac{s}{k6^k}})\right). \tag{5.10}$$

Plugging (5.10) into Lemma 5.10 gives

$$\Pr_{M \in \mathcal{D}_Y}[\text{Eval}(\varphi^*, M, s) = 0] \geq \underbrace{(1 - \frac{sk}{\delta n})}_{(*)}\left(1 - 1/\left(3\underbrace{(1 - e^{-\frac{s}{k6^k}})}_{(**)}\right) - \delta\right). \tag{5.11}$$

We claim that for small enough k, we can set δ and s so that the right-hand side of (5.11) is at least $1/2$. Since Eval outputs 0 no more often than $\varphi^*(M) = 0$, this contradicts that $\Pr_{M \in \mathcal{D}_Y}[\varphi^*(M) = 0] \leq 1/3$ (from (5.9)), completing the lower bound for such k.

All that remains is to find the largest k so that we can choose δ and s appropriately. In order for the right-hand side of (5.11) to be large enough, we need $(*)$ and $(**)$ to be close to 1 and δ to be small. The condition on $(*)$ requires $s = O(\frac{\delta n}{k})$, while the condition on $(**)$ requires $s = \Omega(k6^k)$. Choosing δ to be a small constant, say, $\delta = 1/100$, we see that such a setting is possible when $k^2 6^k = O(n)$— i.e., when $k = c \log n$ for small enough c. □

We now highlight a few details of the preceding analysis. First, notice that the time complexity of the simulated MA-game's verification procedure V enters into the running-time of the AM-simulation only when accounting for the resources needed to compute each query in the black-box setting of Definition 5.2— the circuit lower bound of Theorem 5.5 depends solely on the dimensions of the instance of ∃-ApprMaj, which correspond to the number of bits sent by Arthur and Merlin in the MA-game.

Additionally, notice that the only place that the number of columns m in the instance of ∃-ApprMaj (corresponding to the number of coins tossed by Arthur in an MA-game) enters into the proof is in Proposition 5.7, where a lower bound on m is needed to ensure that the distribution $\mathcal{D}_N$ behaves properly. Thus, Theorem 5.5 actually gives a lower bound of $\Omega(n)$ for $n \times m$ instances of ∃-ApprMaj, where $m \geq \alpha \log n$ for α as defined in Proposition 5.7.

This implies that the black-box lower bound of Theorem 1.4 holds even when simulating MA-games where Arthur tosses very few coins.

The above two observations show that the crucial factor in determining the number of queries needed by an AM-game black-box simulating MA is the *number of bits sent by Merlin*. We point out that this behavior is shared by the known simulations of MA by AM (such as Babai's).

Furthermore, the instances produced by $\mathcal{D}_Y$ satisfy an even stricter promise than $\exists$-ApprMaj$_Y$, since *every* bit in some row is set to 1. This corresponds to the promise on the "yes" side fulfilled by MA-games with *perfect completeness*. Therefore, Theorem 1.1 holds even for simulations of the weaker class of time-t MA-games with one-sided error [2]

Finally, we take a moment to compare our techniques to those of Viola's [Vio07]. Our approach is similar in that it uses ideas inspired by recent small-restriction switching lemmas [SBI04, Raz03] to show that k-DNF's behave similarly enough on certain "yes" and "no" instances of the problem in question. In the setting of normal depth-3 circuits that Viola considers (for the problem ApprMaj), it is sufficient to give an argument that explicitly constructs a "yes" instance rejected by one particular depth-2 subcircuit (in the case where the top gate is AND) assuming that the subcircuit behaves correctly on "no" instances. However, one incorrect subcircuit is insufficient to fool an AM-circuit (as required by our setting), since the top gate only rejects if *most* subcircuits reject. Our approach addresses this issue via the aforementioned *probabilistic* construction of the bad "yes" instances. By producing instances that are independent of the specific subcircuits they are designed to fool, we simultaneously violate many more of them than we could by an explicit construction.

The bounded error of the AM-circuit setting also leads to a quantitatively better statement of the lower bound than in the depth-3 setting: the latter result states that no depth-3 circuit can compute ApprMaj with small bottom fan-in unless it is very large; in contrast,

[2] Since the usual definition of an MA-game has two-sided error and applies the same bound to the message bits and the time bound of the verification procedure, we chose not to draw any of these distinctions in Theorem 1.4.

Theorem 5.5 states that no AM-circuits with small bottom fan-in can compute ∃-ApprMaj, *regardless of their size.*

5.2 Time-Space Lower Bounds for Merlin-Arthur Games

We now turn to proving lower bounds for simulations of MA-games by randomized machines, i.e., trying to decide the same language without the aid of Merlin.

Corollary 1.5 (restated). *For any constant $c < \sqrt{2}$, there exists a positive constant d such that linear-time* MA*-games with perfect completeness cannot be simulated by randomized random-access machines with two-sided error running in time n^c and space n^d.*

As mentioned in Section 1.4, Corollary 1.5 follows from our lower bounds for the second level of the polynomial-time hierarchy; in particular, we refer to the lower bound of Theorem 4.12 for $\Sigma_2\text{TIME}(n)$. We extend this lower bound to MA in a proof by contradiction: if the desired lower bound does not hold for MA, then we show that $\Sigma_2\text{TIME}(n)$ can be computed efficiently enough to contradict Theorem 4.12, i.e. in time n^c and space n^d for $c < 2$ small enough d. This requires a slight improvement of the efficient alternating simulation of space-bounded randomized computations with two sided error given by Theorem 4.4. In particular, we need the simulation to be an MA-game with perfect completeness.

Lemma 5.11. *For any time function t and space function s,*

$$\text{BPTISP}(t, s) \subseteq \text{MA-TIME}_1(ts \text{ polylog}(t)).$$

Proof. Theorem 4.4 shows that $\text{BPTISP}(t, s) \subseteq \Sigma_2\text{TIME}(ts \text{ polylog}(t))$. Recall that Lautemann's proof argues that a randomized algorithm with small enough error accepts an input if and only if a few number of shifts of the set of accepting random strings covers the entire universe. By requiring that the error be smaller than required to satisfy this property by a factor of 1/3 (which can be achieved while only increasing the running time by a constant factor), we strengthen the condition on the rejecting side so that every small set of shifts cannot cover *even a third* of the universe of random strings. This ensures bounded error

on the "no" side of the simulation, allowing us to portray it as an MA-game with perfect completeness. The improvements of Theorem 4.4 retain this property, placing the simulation in MA-TIME$_1(ts \operatorname{polylog}(t))$. □

With this simulation in hand, we are ready to prove the lower bound for MA.

Proof of Corollary 1.5. Assume that linear-time MA-games with perfect completeness can indeed be simulated by small-space randomized machines with two-sided error running in time $O(n^c)$. Specifically, the assumption is that

$$\text{MA-TIME}_1(n) \subseteq \text{BPTISP}(n^c, n^d), \tag{5.12}$$

for some constants $c \geq 1$ and $d > 0$. Since randomized classes are closed under complement, the hypothesis (5.12) gives that co-MA can be efficiently simulated by space-bounded randomized machines; as a special case, this provides such simulations for conondeterministic computations. In turn, the simulation given by Lemma 5.11 yields efficient MA-games with perfect completeness error for coNP:

$$\text{coNTIME}(n) \subseteq \text{BPTISP}(n^c, n^d) \subseteq \text{MA-TIME}_1(n^{c+d} \operatorname{polylog}(n)). \tag{5.13}$$

Consider an arbitrary Σ_2-machine N running in linear time. We view this as a game where the prover goes first and, instead of a randomized verifier (as in MA), we have a conondeterministic "verifier". On input x, the prover sends a string y of length n, and the outcome of the game is determined by a linear-time conondeterministic computation performed by the verifier on input (x, y). However, (5.13) allows the conondeterministic verification to be done by a one-sided error MA-game at the cost of raising the running-time to the power of $c + d$ (modulo a polylog factor). Thus, by asking the prover for the original proof *and* that required by the MA-simulation of the conondeterministic stage, we have devised an MA-simulation of N with perfect completeness. In turn, the hypothesis (5.12) provides that this game can be simulated by a space-bounded randomized machine by again raising the running-time to the power of c. All in all, we have that

$$\Sigma_2\text{TIME}(n) \subseteq \text{MA-TIME}_1(n^{c+d} \operatorname{polylog}(n)) \subseteq \text{BPTISP}(n^{c^2+cd} \operatorname{polylog}(n), n^{cd+d^2}),$$

which contradicts Theorem 4.12 when $c^2 < 2$ and d is small enough. □

5.3 Conclusion and Open Problems

This chapter establishes a lower bound for black-box techniques to switch the order of Arthur and Merlin in two-round Arthur-Merlin games. These results connect in an important way to the task of proving time-space lower bounds for satisfiability on randomized machines: they show that some new, non-black-box techniques are needed to extend our results for Σ_2SAT (Theorem 1.1) to satisfiability.

The most compelling related open problem is to determine whether or not a subquadratic-overhead simulation of MA by AM is at all possible— i.e., to either prove a general quadratic lower bound for the overhead of *any* simulation of MA by AM or to use some non-black-box technique to actually achieve a subquadratic AM-simulation. Either case yields something interesting: in the former case, one gets a hierarchy for MA and AM; in the latter case, one makes progress towards time-space lower bounds for satisfiability on probabilistic machines.

Furthermore, we ask if there are any common restrictions on the verifier that a non-black-box approach could take advantage of to allow a subquadratic simulation of a restricted class of MA-games. Especially interesting is the case of a space-bounded verifier: a subquadratic AM-simulation that exploits the verifier's space bound could also be used to achieve time-space lower bounds for SAT on randomized machines. Recall that we have used space bounds previously to avoid black-box impossibility arguments: our efficient simulation of space-bounded randomized algorithms in the second level of the polynomial-time hierarchy, Theorem 4.4, exploits the space bounds in a crucial way, as Viola [Vio07] showed that black-box simulations cannot achieve this level of efficiency.

Another direction is to extend the black-box lower bound to relationships between other complexity class inclusions where the best known simulations have quadratic overhead. In particular, inclusions that proceed by reducing the error of a probabilistic computation to be exponentially small seem particularly amenable, since we already know that the error

reduction cannot be done faster in a black-box manner [CEG95]. We propose finding black-box lower bounds for the overhead of inclusions such as MA $\subseteq$ PP, $\oplus \cdot$ BPP $\subseteq$ BP $\cdot \oplus$P, etc. The latter is particularly interesting, since it applies to the time overhead of the collapse of the polynomial-time hierarchy to BP $\cdot \oplus$P in Toda's Theorem [Tod91]; it is currently unknown how to achieve such a collapse with a sublinear factor increase in time per alternation. Achieving such an efficient collapse could yield new time-space lower bounds for counting problems.

Chapter 6

Time-Space Lower Bounds for Tautologies

In this chapter we consider the task of proving time-space lower bounds for complete problems in the first level of the polynomial-time hierarchy; in particular, we focus on the coNP-complete language of tautologies. Our results improve the known time-space lower bounds for nondeterministic algorithms solving tautologies (Theorem 1.6) and give an even better lower bound for the special case of randomized machines with one-sided error (Theorem 1.7).

6.1 Lower Bounds for Proof Complexity

We begin by proving our generic results on proof complexity and the NP versus coNP problem. Our result boosts the exponent in known time-space lower bound for tautologies on nondeterministic machines from $\sqrt{2} \approx 1.414$ [FvM00, FLvMV05] to $\sqrt[3]{4} \approx 1.587$. More precisely, we obtain the following result.

Theorem 1.6 (restated). *For every real $d < \sqrt[3]{4}$ there exists a positive real e such that tautologies cannot be decided by nondeterministic algorithms running in time n^d and space n^e.*

The earlier result of Fortnow and Van Melkebeek [FvM00, FLvMV05] can be refined to rule out either nondeterministic algorithms solving tautologies in time n^c (regardless of space) *or* nondeterministic algorithms solving tautologies in simultaneous time n^d and space n^e for certain combinations of c, d, and e. The precise relationship between the parameters in [FvM00, FLvMV05] is that for every c and d such that $(c^2 - 1)d < c$, there

is a positive e such that the previous sentence holds. In particular, tautologies cannot have both a nondeterministic algorithm that runs in time $n^{1+o(1)}$ and a nondeterministic algorithm that runs in logarithmic space [For00]. Correspondingly, our argument yields the following refinement.

Theorem 6.1. *For all reals c and d such that $c^2 d < 4$, there exists a positive real e such that tautologies cannot be solved by both*

(i) a nondeterministic algorithm that runs in time n^c and

(ii) a nondeterministic algorithm that runs in time n^d and space n^e.

In order to compare the condition $c^2 d < 4$ in Theorem 6.1 with the corresponding condition $(c^2 - 1)d < c$ in [FvM00, FLvMV05], we include a plot of those bounds in Figure 6.1. Note that the interesting range of parameters satisfies $d \geq c \geq 1$. This is because an algorithm of type (ii) is a special case of an algorithm of type (i) for $d \leq c$, and that an algorithm of type (i) with $c < 1$ can be ruled out unconditionally by simple diagonalization. The condition due to this paper, $c^2 d < 4$, is less restrictive for values of d not too much larger than c. Thus, Theorem 6.1 gives a better lower bound in this range. In particular, for $c = d$, our condition requires $c < \sqrt[3]{4} \approx 1.587$, whereas that of [FvM00, FLvMV05] requires $c < \sqrt{2} \approx 1.414$; this setting yields the improvement stated in Theorem 1.6.

Our main technical contribution builds upon the reasoning underlying the $(c^2 - 1)d < c$ condition of [FvM00, FLvMV05]. Briefly, the argument for the latter follows an indirect diagonalization argument similar to that found in the proof of Theorem 1.1, inductively deriving a series of more and more efficient complementations of the first level of the polynomial-time hierarchy. Our new approach allows us to harness the complementation provided by the inductive hypothesis *twice* in each inductive step, whereas earlier approaches could only do so once per step. This yields better performance in an interesting range of c and d. Due to the double application of the inductive hypothesis, the resulting recurrence relation for the exponents in the complementations' running times becomes of degree two (rather than one as before) and has nonconstant coefficients, but we can still handle it analytically.

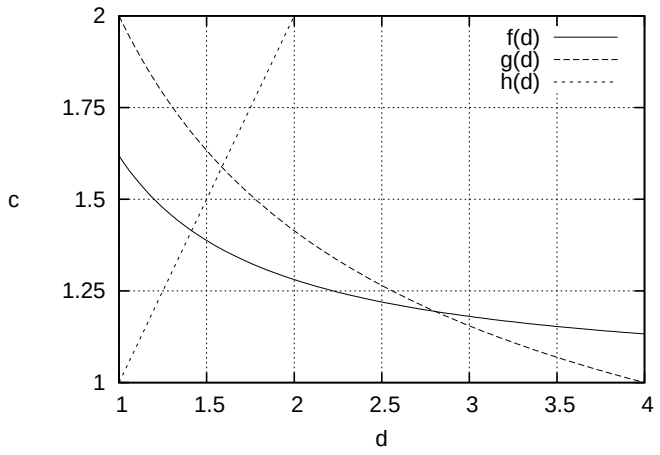

Figure 6.1 Tautologies cannot have both (i) a nondeterministic algorithm that runs in time n^c and (ii) a nondeterministic algorithm that runs in time n^d and space $n^{o(1)}$. The function $f(d)$ solves for the best bound on c due to the prior results of [FvM00, FLvMV05], and $g(d)$ does the same for Theorem 6.1; the function $h(d)$ marks the case $c = d$, illustrating the bound in Theorem 1.6.

6.1.1 Lower Bound for Nondeterministic Machines

We begin with a brief discussion of the techniques required to prove the result of [FvM00, FLvMV05] in the subpolynomial-space setting, namely, that tautologies cannot be solved by subpolynomial-space nondeterministic machines running in time n^d for any reals $d < \sqrt{2}$.

Recall that the tight connection between coNTIME(n) and tautologies, Theorem 2.3, allows lower bounds for the class coNTIME(n) to carry over to tautologies. Therefore, we focus our efforts on proving lower bounds for the former class. We do so by indirect diagonalization. This starts with the assumption, by way of contradiction, that

$$\text{coNTIME}(n) \subseteq \text{NTISP}(n^d, n^{o(1)}). \tag{6.1}$$

The relevant technical lemma from [FvM00, FLvMV05] can be thought of as trading space for time within NP under the indirect diagonalization assumption. More precisely, we try to establish

$$\text{NTISP}(t, s) \subseteq \text{NTIME}(g(t, s)) \tag{6.2}$$

for the smallest possible functions g, with the hope that $g(t,s) \ll t$. In particular, for sub-polynomial space bounds ($s = t^{o(1)}$) and sufficiently large polynomial t, [FvM00, FLvMV05] achieves $g = t^{d-1/d+o(1)}$,

$$\mathrm{NTISP}(t, t^{o(1)}) \subseteq \mathrm{NTIME}(t^{d-1/d+o(1)}), \tag{6.3}$$

which is smaller than t when $d < \phi \approx 1.618$.

To see the utility of this statement, we combine it with the indirect diagonalization assumption (6.1) padded to a sufficiently large polynomial time function t:

$$\begin{array}{rcll} \mathrm{coNTIME}(t) & \subseteq & \mathrm{NTISP}(t^d, t^{o(1)}) & \text{[by assumption (6.1)]} \\ & \subseteq & \mathrm{NTIME}(t^{d^2-1+o(1)}) & \text{[by trading space for time using (6.3)].} \end{array}$$

This is a contradiction with the direct diagonalization argument of Lemma 3.2 when $d < \sqrt{2}$, yielding the desired lower bound.

The space-for-time inclusion (6.3) is shown by an inductive argument that derives statements of the type (6.2) for a sequence of smaller and smaller running times $g = g_k$, $k = 1, 2, \ldots$. The idea can be summarized as follows: we start with a space-bounded nondeterministic machine and apply the divide-and-conquer speedup for space-bounded nondeterministic machines at the cost of two alternations, (3.5):

$$\mathrm{NTISP}(t,s) \subseteq \underbrace{\exists^{bs}\, \forall^{\log b}\, \underbrace{\mathrm{NTISP}(t/b, s)}_{(*)}}_{(**)}. \tag{6.4}$$

We then use the inductive hypothesis to trade the space bound of the final stage $(*)$ of this Σ_3-simulation for time:

$$\mathrm{NTISP}(t,s) \subseteq \exists^{bs}\forall^{\log b}\mathrm{NTIME}(g_k(t/b, s)).$$

We conclude the inductive argument by using the assumption (6.1) to eliminate the two alternations in this simulation, ending up with another statement of the form

$$\mathrm{NTISP}(t,s) \subseteq \mathrm{NTIME}(g_{k+1}(t,s)).$$

Notice that the above inductive argument does not rely on the space bound in (6.1); the weaker assumption that coNTIME(n) $\subseteq$ NTIME(n^d) is enough to eliminate the alternations introduced by the speedup. Our new argument exploits the fact that when we transform ($*$) using the assumption (6.1), we eliminate an alternation *and* re-introduce a space-bound. This allows us to apply the inductive hypothesis for a *second time* and trade that space bound for a speedup in time once more. This way, we hope to eliminate the alternation in ($**$) more efficiently than before, yielding a smaller g_k after completing the argument.

Some steps of our new argument exploit the space bound while others do not. In the analysis we allow for different parameters in those two assumptions, say we assume that

$$\text{coNTIME}(n) \subseteq \text{NTISP}(n^c) \cap \text{NTISP}(n^d, n^{o(1)}),$$

where $d \geq c \geq 1$. The success of our approach to eliminate the alternation in ($**$) now depends on how large d is compared with c: if d is not too large compared to c, then the increased cost of complementing via the space-bounded assumption is counteracted by the benefit of trading this space bound for time. That our approach works better in this range of c and d makes plausible the behavior illustrated in Figure 6.1.

Two key ingredients that allow the above idea to yield a quantitative improvement for certain values of c and d are (i) that the conondeterministic guess at the beginning of stage ($**$) is only over $\log b$ bits and (ii) the fact mentioned in Section 3.1 that ($*$) has input size only $O(n + s)$. Because of (i), the running time of ($**$) is dominated by that of ($*$), allowing us to reduce the cost of simulating ($**$) without an alternation by reducing the cost of simulating ($*$) in coNP. Item (ii) is important for the latter task because the effective input size for the computation ($*$) is much smaller than the $O(n + bs)$ bits taken by ($**$); in particular, it does not increase with b. This allows the use of larger block numbers b to achieve greater speedups while maintaining that the final stage runs in time at least linear in its input. The latter behavior is crucial in allowing alternation removal at the expected cost — raising the running time to the power of c or d — because we can pad the indirect diagonalization assumption up (to superlinear time) but not down (to sublinear time)— see Proposition 3.1.

Now that we have sketched the intuition and key ingredients, we proceed with the actual argument. The following lemma formalizes the inductive process of speeding up nondeterministic space-bounded computations on space-unbounded nondeterministic machines.

Lemma 6.2. *If*

$$\mathrm{coNTIME}(n) \subseteq \mathrm{NTIME}(n^c) \cap \mathrm{NTISP}(n^d, n^e)$$

for some reals c, d, and e then for every nonnegative integer k, time function t, and space function $s \leq t$,

$$\mathrm{NTISP}(t, s) \subseteq \mathrm{NTIME}\left((ts^k)^{\gamma_k} + (n+s)^{a_k}\right),$$

where $\gamma_0 = 1$, $a_0 = 1$, and γ_k and a_k are defined recursively for $k > 0$ as follows: Let

$$\mu_k = \max(\gamma_k(d + ek), ea_k), \tag{6.5}$$

then

$$\gamma_{k+1} \quad = \quad c\gamma_k\mu_k/(1 + \gamma_k\mu_k), \tag{6.6}$$

and

$$a_{k+1} \quad = \quad ca_k \cdot \max(1, \mu_k). \tag{6.7}$$

Proof. The proof is by induction on k. The base case $k = 0$ is trivial. To argue the inductive step, $k \to k+1$, we consider a nondeterministic machine M running in time t and space s and construct a simulation with the goal of achieving a speedup at the cost of sacrificing the space bound. We begin by simulating M in the third level of the polynomial-time hierarchy via the divide-and-conquer speedup (3.5) using $b > 0$ blocks (to be determined later); this simulation is in

$$\exists^{bs}\forall^{\log t}\underbrace{\mathrm{NTISP}(t/b, s)}_{(*)}. \tag{6.8}$$

We focus on simulating the computation of $(*)$ as described above. Recall that the input to $(*)$ consists of the original input x of M as well as two configuration descriptions of size $O(s)$, for a total input size of $O(n+s)$. The inductive hypothesis allows the simulation of $(*)$ in

$$\mathrm{NTIME}((\frac{t}{b}s^k)^{\gamma_k} + (n+s)^{a_k}). \tag{6.9}$$

In turn, this simulation can be complemented while simultaneously introducing a space bound via the assumption of the lemma; namely, (6.9) is in

$$\text{coNTISP}\left(((\frac{t}{b}s^k)^{\gamma_k}+(n+s)^{a_k})^d,((\frac{t}{b}s^k)^{\gamma_k}+(n+s)^{a_k})^e\right), \tag{6.10}$$

where here the $(n+s)^{a_k}$ term subsumes the $O(n+s)$ term from the input size because $a_k \geq 1$. The space bound allows for a simulation via the inductive hypothesis once more, yielding a simulation of $(*)$ in

$$\begin{aligned}\text{coNTIME}\left(((\tfrac{t}{b}s^k)^{\gamma_k}+(n+s)^{a_k})^{\gamma_k(d+ek)}+(n+s+((\tfrac{t}{b}s^k)^{\gamma_k}+(n+s)^{a_k})^e)^{a_k}\right)\\ \subseteq \text{coNTIME}\left((\tfrac{t}{b}s^k)^{\gamma_k\mu_k}+(n+s)^{a_k\mu_k}+(n+s)^{a_k}\right).\end{aligned} \tag{6.11}$$

Replacing $(*)$ in (6.8) by (6.11) eliminates an alternation, lowering the simulation of M to the second level:

$$\exists^{bs}\underbrace{\forall^{\log t}\text{coNTIME}\left((\frac{t}{b}s^k)^{\gamma_k\mu_k}+(n+s)^{a_k\mu_k}+(n+s)^{a_k}\right)}_{(**)} \tag{6.12}$$

We now complement the conondeterministic computation represented by $(**)$ via the lemma's assumption that $\text{NTIME}(n) \subseteq \text{coNTIME}(n^c)$, eliminating one more alternation in the simulation of M. Specifically, since $(**)$ takes input of size $O(n+bs)$, this places the simulation in

$$\begin{aligned}&\exists^{bs}\text{NTIME}\left(((\tfrac{t}{b}s^k)^{\gamma_k\mu_k}+(n+s)^{a_k\mu_k}+(n+s)^{a_k}+(bs+n))^c\right)\\ \subseteq\ &\text{NTIME}((\underbrace{(\frac{t}{b}s^k)^{\gamma_k\mu_k}}_{(\searrow)}+(n+s)^{a_k\mu_k}+(n+s)^{a_k}+\underbrace{bs}_{(\nearrow)})^c),\end{aligned} \tag{6.13}$$

where the inclusion holds by collapsing the adjacent existential phases (and the time required to guess the $O(bs)$ configuration bits is accounted for by the observation that $c \geq 1$).

Therefore, we have arrived at a simulation that gives rise to an inclusion of $\text{NTISP}(t,s)$ in $\text{NTIME}(\cdot)$; all that remains is to choose b to optimize the running time. Notice that the running time of the simulation in (6.13) has one term, $(\nearrow)$, that increases with b and one term, $(\searrow)$, that decreases with b. The running time is optimized up to a constant factor by

choosing b to equate the two terms, resulting in a choice of

$$b^* = \left(\frac{(ts^k)^{\gamma_k \mu_k}}{s} \right)^{1/(1+\gamma_k \mu_k)}.$$

When this value is at least 1, the running time of the nondeterministic simulation (6.13) is

$$O\left((ts^{k+1})^{c\gamma_k \mu_k/(1+\gamma_k \mu_k)} + (n+s)^{ca_k \mu_k} + (n+s)^{ca_k} \right),$$

resulting in the recurrences (6.6) and (6.7). If $b^* < 1$, then $b = 1$ is the best we can do; the desired bound still holds since in this case $(\nearrow) + (\searrow) = O(s)$, which is dominated by the $(n+s)^{a_{k+1}}$ term. □

Under the assumption of Lemma 6.2, we can further deduce that for a sufficiently large polynomial τ,

$$\text{coNTIME}(\tau) \subseteq \text{NTISP}(\tau^d, \tau^e) \subseteq \text{NTIME}(\tau^{(d+ek)\gamma_k} + \tau^{ea_k}) = \text{NTIME}(\tau^{\mu_k}), \qquad (6.14)$$

which is a contradiction with the direct diagonalization argument of Lemma 3.2 when $\mu_k < 1$. Therefore, the key question is for what values of c, d, and e does μ_k take on a value less than 1. Our analysis focuses on small values of e and shows how such a setting allows us to exhibit the desired behavior in μ_k.

Theorem 6.3. *For all reals c and d such that $c^2 d < 4$ there exists a positive real e such that*

$$\text{coNTIME}(n) \nsubseteq \text{NTIME}(n^c) \cap \text{NTISP}(n^d, n^e).$$

Proof. The case where either $c < 1$ or $d < 1$ is ruled out by Lemma 3.2. For $c \geq 1$ and $d \geq 1$, assume (by way of contradiction) that

$$\text{coNTIME}(n) \subseteq \text{NTIME}(n^c) \cap \text{NTISP}(n^d, n^e)$$

for a value of e to be determined later. As noted above, the theorem's assumption in conjunction with Lemma 6.2 yields the complementation (6.14) for any integer $k \geq 0$ and sufficiently large polynomial bound τ.

Our goal is now to characterize the behavior of μ_k in terms of c, d, and e. This task is facilitated by focusing on values of e that are small enough to smooth out the complex behavior of μ_k caused by (i) the appearance of the nonconstant term ek in the recurrence and (ii) its definition via the maximum of two functions.

We first handle item (i) by introducing a related, nicer sequence by substituting a real β (to be determined) as an upper bound for ek: Let

$$\mu'_k = \max(\gamma'_k(d+\beta), ea'_k), \tag{6.15}$$

where $\gamma'_0 = 1$, $a'_0 = 1$ and

$$\gamma'_{k+1} = c\gamma'_k\mu'_k/(1+\gamma'_k\mu'_k), \tag{6.16}$$

and

$$a'_{k+1} = ca'_k \cdot \max(1, \mu'_k). \tag{6.17}$$

As long as β behaves as intended, i.e., that $ek \leq \beta$, we can show by induction that $\gamma_k \leq \gamma'_k$, $a_k \leq a'_k$, and $\mu_k \leq \mu'_k$. Therefore, μ'_k upper bounds μ_k up to a value of k that depends on e, and this k-value becomes large when e is very small. This allows us to use μ'_k as a proxy for μ_k in our analysis.

To smooth out the behavior caused by issue (ii), we point out that the first term in the definition (6.15) of μ'_k is larger than the second when e is very small. Provided that this is the case, μ'_k equals the sequence ν_k defined as follows:

$$\begin{aligned} \nu_0 &= d+\beta \\ \nu_{k+1} &= \nu_k^2 c(d+\beta)/((d+\beta)+\nu_k^2). \end{aligned} \tag{6.18}$$

This delivers a simpler sequence to analyze. Notice that because the underlying transformation

$$\eta \to \eta^2 c(d+\beta)/((d+\beta)+\eta^2) \tag{6.19}$$

is increasing over the positive reals, the sequence ν_k is monotone in this range. It is decreasing if and only if $\nu_1 < \nu_0$, which is equivalent to $(c-1)(d+\beta) < 1$. Furthermore, when $c^2(d+\beta) < 4$, the transformation has a unique real fixed point at 0. Since the underlying transformation

is also bounded and starts positively, the sequence ν_k must decrease monotonically to 0 in this case.

Therefore, when $c^2 d < 4$ we can choose a positive β such that ν_k becomes as small as we want for large k. Provided that β, e, and k satisfy the assumptions required to smooth out items (i) and (ii), this also gives us that μ_k is small. More formally, let k^* be the first value of k such that $\nu_k < 1$. Then to satisfy item (i), we require that

$$ek^* \leq \beta. \tag{6.20}$$

For item (ii), we require that the first term in the definition (6.15) of μ'_k dominates the second up to this point, namely,

$$\gamma'_k(d + \beta) \geq ea'_k \text{ for all } k \leq k^*. \tag{6.21}$$

When all of these conditions are satisfied, we have that

$$\mu_{k^*} \leq \mu'_{k^*} = \nu_{k^*} < 1,$$

and the running time of the conondeterministic simulation represented by (6.14) for $k = k^*$ runs in time

$$O(\tau^{\mu_{k^*}}) = O(\tau^{\mu'_{k^*}}) = O(\tau^{\nu_{k^*}}). \tag{6.22}$$

Therefore, by choosing a small enough positive e to satisfy the finite number of constraints in (6.20) and (6.21), we arrive at our goal of exhibiting an exponent cost in the complementation of (6.14) that is smaller than 1. This is a contradiction, which proves the desired lower bound for coNTIME(n). □

The proof of Theorem 6.3 establishes $c^2 d < 4$ as a sufficient condition for our approach to work but it isn't clear that the condition is also necessary. For completeness we include an argument showing that our analysis in the proof is tight — our approach does not work for any setting with $c^2 d \geq 4$.

Referring to the notation used in the proof of Theorem 6.3, our approach works if we can find a value of $\mu_k < 1$ so that (6.14) contradicts Lemma 6.2. Without loss of generality, we

can assume that the space-efficient simulation only uses logarithmic space, in which case the sequence μ_k simplifies to ν_k given by (6.18), where the term β is negligible. Since $\nu_1 \geq 1$, ν_k has to decrease in order to obtain a contradiction; as this happens when $(c-1)d < 1$, we can rule out success in settings with $c^2 d \geq 4$ and $d = 1$, or $c \geq 2$. For $c^2 d = 4$ and $d > 1$, the underlying transformation (6.19) has a unique positive fixed point to which the sequence ν_k converges, namely $cd/2 = \sqrt{d}$, which is less than d. If we let c grow, the double fixed point separates into two positive fixed points that gradually diverge from but remain centered around $cd/2$. As long as the largest of the two fixed points remains less than d, the sequence converges to it in a monotone decreasing way. At the verge where that fixed point reaches d, the sequence remains constant, which means that $(c-1)d = 1$. Beyond that the sequence monotonically increases to the larger fixed point. This argument implies that at all times $\nu_k \geq cd/2$. In order to have $\nu_k < 1$, we would need $cd < 2$, which is incompatible with our assumption that $c^2 d \geq 4$ and $d > 1$.

6.2 Lower Bounds for Randomized Machines with One-Sided Error

Theorem 1.6 yields the same lower bounds for randomized machines with one-sided error as a special case. However, we can take advantage of the extra structure provided by such machines to establish a better lower bound in the latter setting.

Theorem 1.7 (restated). *For every constant $c < 2\cos(\pi/7) \approx 1.801$ there exists a positive constant d such that tautologies cannot be solved by randomized random-access machines with one-sided error running in time n^c and space n^d.*

In fact, we actually prove time-space lower bounds for a more powerful class of machines, namely nondeterministic machines that are *restricted to guess few bits*.

Theorem 6.4. *For any constant $c < 2\cos(\pi/7) \approx 1.801$ there exists positive reals a and d such that tautologies cannot be solved by nondeterministic random-access machines that run in time n^c, space n^d, and nondeterministically guess only n^a bits.*

By way of the space-bounded derandomization of Theorem 4.3, a space-bounded randomized machine with one-sided error can be made to use very few random bits. Since a randomized machine with one-sided error is also a special type of nondeterministic machine, we can view the one-sided error machines obtained by Theorem 4.3 as nondeterministic machines which guess very few random bits. Thus, the time-space lower bounds of Theorem 1.7 follow as a corollary to Theorem 6.4.

Proof of Theorem 1.7. By the derandomization of Theorem 4.3, if tautologies can be solved in $\text{RTISP}(n^{c'}, n^d)$, then it can also be solved in $\exists^{n^d \log n}\text{DTISP}(n^{c'} \operatorname{polylog}(n), n^d)$. This contradicts Theorem 6.4 as long as $c' < 2\cos(\pi/7)$ and $d < \min(a^*, d^*)$, where a^* and d^* are the values given by Theorem 6.4 for a and d, respectively, for some c such that $c' < c < 2\cos(\pi/7)$. □

Our main technical contribution for Theorem 1.7 is to upgrade existing indirect diagonalization arguments for the deterministic setting to handle a small number of guess bits. This follows by showing that these guesses can effectively be ignored, so that we can proceed almost as we do in the deterministic arguments. For example, consider the task of speeding up a subpolynomial-space nondeterministic machine that guesses only a subpolynomial number of bits. We can do this by way of the divide-and-conquer speedup for nondeterministic machines given by (3.5), but this adds one more alternation than the speedup for deterministic machines. Alternatively, we can view the machine as one that guesses a subpolynomial number of bits, writes them down on a worktape, and then verifies this guess with a subpolynomial-space deterministic machine. Since little time is spent in the guessing stage, we can speed up the entire machine by only focusing on the verification stage. The latter is deterministic and space-bounded, so it can be sped up by a Σ_2-simulation. This results in a computation that nondeterministically guesses a small number of bits and then verifies them with a Σ_2-machine, which is overall a faster Σ_2-machine. Thus, we've realized a speedup using only *one* alternation, just as in the deterministic case. The conditional

elimination of alternations works similarly. Some careful accounting is required to show that these ideas work for small polynomial numbers of guess bits.

We instantiate our techniques within the current state-of-the-art time-space lower bound for satisfiability [Wil07b]. This proof proceeds in two stages: first, we show that the indirect diagonalization assumption allows for a conditional speedup of space-bounded deterministic machines that is *better* than that given unconditionally by (3.1); we then use this improved speedup in a bootstrapping argument to derive more and more efficient complementations of the second level of the polynomial-time hierarchy, until one contradicts the known diagonalization argument of Lemma 3.2. The latter part takes a form similar to that of the main lemma for our randomized lower bounds, Lemma 4.11, although we must pay careful attention to the effect of input sizes in order to achieve the best quantitative result.

6.2.1 Lower Bound for Few Guess Bits

We set out to prove Theorem 6.4 by proving lower bounds for the class coNTIME(n); these transfer to tautologies via the tight connection given by Theorem 2.3. It is actually more convenient for us to prove results about the complementary classes, so we write the assumption of the indirect diagonalization argument as:

$$\mathrm{NTIME}(n) \subseteq \forall^{n^a}\mathrm{DTISP}(n^c, n^d). \tag{6.23}$$

This unlikely scenario certainly yields an efficient complementation with which we can eliminate alternations, and it is actually strong enough to allow for something more: under this hypothesis, we can improve the space-bounded speedup of (3.2) for certain values of a, c and d.

Lemma 6.5. *Suppose that*

$$\mathrm{NTIME}(n) \subseteq \forall^{n^a}\mathrm{DTISP}(n^c, n^d) \tag{6.24}$$

for positive reals a, $c \geq 1$, and d such that $c + d \leq 2$ and $a \leq c + d - 1$. Then for any time function t, space function s, and integer $j \geq 0$,

$$\mathrm{DTISP}(t, s) \subseteq \Sigma_2\mathrm{TIME}((ts)^{\gamma_j} + n + s), \tag{6.25}$$

where $\gamma_0 = \frac{1}{2}$ *and* $\gamma_{j+1} = \frac{(c+d)\gamma_j}{(c+d)\gamma_j+1}$.

Before proving the lemma, we analyze the sequence $(\gamma_j)_j$ in order to clearly assess the speedup represented by Lemma 6.5. In fact, we study sequences that have the general form of $(\gamma_j)_j$, as we encounter another one later.

Proposition 6.6. *For any positive reals* r *and* ξ_0*, the sequence*

$$\xi_{i+1} = \frac{r\xi_i}{r\xi_i + 1}$$

converges monotonically to $1 - \frac{1}{r}$ *if* $r \geq 1$ *and to 0 otherwise; it is decreasing if and only if* $\xi_0 > 1 - \frac{1}{r}$.

Proof. Notice that the transformation underlying the sequence $(\xi_i)_i$,

$$x \mapsto \frac{rx}{rx + 1}, \tag{6.26}$$

is increasing over the reals. It follows that the sequence $(\xi_i)_i$ is monotonic and that it decreases if and only if $\xi_0 < 1 - \frac{1}{r}$. Furthermore, (6.26) has fixed points at 0 and $1 - \frac{1}{r}$; since $(\xi_i)_i$ is continuous and bounded, it must converge to one of these fixed points. Notice that the condition for the sequence to be decreasing corresponds exactly to the start point being greater than the fixed point at $1 - \frac{1}{r}$. Thus, the sequence converges to $1 - \frac{1}{r}$ as long as this value is nonnegative, namely, when $r \geq 1$. Otherwise, the sequence converges to the other fixed point at 0. □

The sequence $(\gamma_j)_j$ is the same as $(\xi_i)_i$ in Proposition 6.6 for $r = (c + d)$. Since $\gamma_0 = \frac{1}{2}$ and $c \geq 1$, the proposition tells us that $(\gamma_j)_j$ decreases monotonically to $1 - \frac{1}{c+d}$. Thus, when s is small, Lemma 6.5 essentially offers a $(1 + \frac{1}{c+d-1})^{\text{th}}$-root speedup of a DTISP$(t, s)$ machine on a Σ_2-machine, provided this running time remains at least linear. Recall that the unconditional speedup offered by (3.2) gives a similar square-root speedup. Thus, Lemma 6.5 offers a greater speedup than we had unconditionally when $c + d < 2$.

We prove Lemma 6.5 by inductively deriving better and better speedups of DTISP(t, s) into Σ_2. The inductive step uses the unconditional speedup (3.1) and the assumption (6.23)

of the indirect diagonalization argument to go from a speedup in Σ_2 to one in NP with a space-bounded verification procedure. We apply the inductively derived speedup into Σ_2 to this verification, resulting in an overall simulation in Σ_2. Choosing the number of blocks used in the initial step to optimize the final running time results in each new speedup being superior to the previous for certain ranges of a, c and d.

Proof of Lemma 6.5. The proof is by induction on j. For the base case $j = 0$, (6.25) holds unconditionally by the standard square root speedup of (3.2). For the inductive step $j \to j+1$, consider a DTISP(t, s) computation. We speed up this computation using (3.1):

$$\mathrm{DTISP}(t,s) \subseteq \exists^{bs} \underbrace{\forall^{\log b} \mathrm{DTISP}(t/b, s)}_{(i)}.$$

We can see that (i) represents a computation in conondeterministic time $O(t/b)$ taking an input of length $O(n + bs)$. Thus, the assumption (6.24) gives a simulation of (i) which yields

$$\mathrm{DTISP}(t,s) \subseteq \underbrace{\exists^{bs}\exists^{(t/b+n+bs)^a}}_{(ii)} \underbrace{\mathrm{DTISP}((t/b+n+bs)^c, (t/b+n+bs)^d)}_{(iii)}.$$

Notice that the final space-bounded deterministic stage (iii) takes an input of size $O(n + bs + (t/b)^a)$ provided $a \leq 1$, so the inductive hypothesis yields a simulation of this stage in

$$\Sigma_2\mathrm{TIME}(((t/b+n+bs)^{c+d})^{\gamma_j} + n + bs + (t/b)^a + (t/b+n+bs)^d).$$

Merging the initial existential phase of this simulation with that represented by (ii), we see that we have arrived at a simulation of DTISP(t, s) on a Σ_2-machine running in time big-O of

$$bs + (t/b+n+bs)^{\max(a,d)} + ((t/b+n+bs)^{c+d})^{\gamma_j} + n.$$

To simplify this, notice that when $c + d \leq 2$, we have that $(c + d)\gamma_j \leq 1$ (since $\gamma_j \leq \frac{1}{2}$). If $c \geq 1$, we have $d \leq (c + d)\gamma_j$ (since $\gamma_j \geq 1 - \frac{1}{c+d}$). Furthermore, if $a \leq c + d - 1$, we have $a \leq (c+d)\gamma_j$ (since $\gamma_j \geq 1 - \frac{1}{c+d}$). Under these conditions, the above running time simplifies to big-O of

$$bs + (t/b)^{(c+d)\gamma_j} + n.$$

To minimize this running time up to a constant factor, we choose a value for b such that $bs = (t/b)^{(c+d)\gamma_j}$, namely

$$b^* \doteq (\frac{t^{(c+d)\gamma_j}}{s})^{1/((c+d)\gamma_j+1)}.$$

If $b^* \geq 1$, this choice results in a running time in big-O of

$$(ts)^{\frac{(c+d)\gamma_j}{(c+d)\gamma_j+1}} + n = (ts)^{\gamma_{j+1}} + n.$$

On the other hand, if $b^* < 1$, then $b = 1$ is the best we can do; this yields a running time of $O(s+n)$. In either case, $O((ts)^{\gamma_{j+1}} + n + s)$ is an upper bound on the running time. By induction, (6.25) holds for all $j \geq 0$ and a, c, and d as above. □

We point out that, under the conditions of Lemma 6.5, we can actually derive a complementation within the second level of the polynomial-time hierarchy that is at least as good as the one provided by the indirect diagonalization hypothesis, (6.23), alone. To see this, we write

$$\begin{aligned} \Pi_2\text{TIME}(n) &\subseteq \exists^{n^{ac}}\text{DTISP}(n^{c^2}, n^{cd}) \\ &\subseteq \Sigma_2\text{TIME}(n^{c(c+d)\gamma_j} + n^{cd} + n^{ac} + n), \end{aligned} \tag{6.27}$$

where the first inclusion holds by using (6.23) twice and the second holds by Lemma 6.5. Thus, the new complementation of the second level comes at the cost of raising the running time to the power of $c(c+d)\gamma_j$, compared to the cost of c in the hypothesis (6.23) (assuming the $n^{c(c+d)\gamma_j}$ term dominates the running time in (6.27); if this isn't the case, then we already have a better complementation for $a, d < 1$). Since $\gamma_j \to 1 - \frac{1}{c+d}$, the exponent cost $c(c+d)\gamma_j$ approaches $c(c+d-1)$, which is at most c when $c+d \leq 2$: the new complementation (6.27) is at least as good under these conditions.

This leads to an approach to derive even better speedups than those offered by Lemma 6.5: rather than staying within the second level and eliminating alternations with the hypothesis (6.23), we go beyond the second level where we can reap the benefits of greater speedups offered by using more alternations *and* where we can eliminate alternations more efficiently using (6.27).

Lemma 6.7. *Suppose that*

$$\mathrm{NTIME}(n) \subseteq \forall^{n^a}\mathrm{DTISP}(n^c, n^d) \tag{6.28}$$

for positive reals a, $c \geq 1$, and d such that $c + d \leq 2$ and $a \leq c + d - 1$. Then for any time function t, space function s, and integers $j, k \geq 0$,

$$\mathrm{DTISP}(t, s) \subseteq \Sigma_2\mathrm{TIME}\left((ts^k)^{\beta_{j,k}} + (n + s)^{c^k}\right), \tag{6.29}$$

where $\beta_{j,0} = \gamma_j$ and $\beta_{j,k+1} = \frac{(c+d)c\gamma_j\beta_{j,k}}{(c+d)c\gamma_j\beta_{j,k}+1}$ for $(\gamma_j)_j$ defined as in Lemma 6.5.

Before proving Lemma 6.7, we compare its speedup to that offered in Lemma 6.5 by observing the behavior of the exponent $\beta_{j,k}$ relative to γ_j. For a fixed j, the sequence $(\beta_{j,k})_k$ is of the form $(\xi_k)_k$ in Proposition 6.6 for $r = (c + d)c\gamma_j$. Since $(\beta_{j,k})_k$ begins at γ_j, the behavior we are interested in is if $(\beta_{j,k})_k$ is decreasing; this occurs when $\gamma_j > 1 - \frac{1}{c(c+d)\gamma_j}$ by Proposition 6.6. Recall that, when $c + d \leq 2$ for positive d, we have $1 - \frac{1}{c+d} \leq \gamma_j \leq \frac{1}{2}$. Thus,

$$1 - \frac{1}{c(c+d)\gamma_j} < 1 - \frac{1}{c+d} \leq \gamma_j,$$

so we conclude that $(\beta_{j,k})_k$ is decreasing when $c + d \leq 2$. Therefore, Lemma 6.7 represents an improvement of Lemma 6.5 for sufficiently large polynomials t and small enough s.

Proof of Lemma 6.7. The proof is by induction on k. The base case $k = 0$ follows directly from Lemma 6.5. For the inductive step $k \to k+1$, consider the speedup of $\mathrm{DTISP}(t, s)$ into Σ_2 given by (3.1):

$$\mathrm{DTISP}(t, s) \subseteq \underbrace{\exists^{bs}\,\forall^{\log b}\,\underbrace{\mathrm{DTISP}((t/b), s)}_{(i)}}_{(ii)}.$$

Recall that the computation represented by (i) takes as input the descriptions of two configurations in addition to the global input, for an input size of $O(n + s)$. The inductive hypothesis in combination with the assumption of the lemma allows us to simulate (i) in

$$\Pi_2\mathrm{TIME}\left((\frac{t}{b}s^k)^{\beta_{j,k}} + (n + s)^{c^k}\right) \subseteq \mathrm{coNTIME}\left((\frac{t}{b}s^k)^{\beta_{j,k}} + (n + s)^{c^k})^c\right).$$

Thus, we have that (ii) can be simulated by a computation in coNP that takes an input of size $O(n + bs)$. The assumption of the lemma allows us to simulate (ii) in

$$\exists^{(t^*)^a} \underbrace{\mathrm{DTISP}((t^*)^c, (t^*)^d)}_{(iii)},$$

where

$$t^* \doteq ((\frac{t}{b}s^k)^{\beta_{j,k}} + (n+s)^{c^k})^c + bs.$$

The input to (iii) includes the input to (ii) in addition to $(t^*)^a$ guess bits. Lemma 6.5 allows us to speed up (iii) by a simulation in

$$\Sigma_2\mathrm{TIME}((t^*)^{(c+d)\gamma_j} + n + bs + (t^*)^a).$$

Putting all of the simulations together and merging adjacent existential quantifiers, we have a Σ_2-simulation of $\mathrm{DTISP}(t, s)$ that runs in time

$$O\left((t^*)^{\max(a,(c+d)\gamma_j)} + n + bs\right).$$

Provided that $c + d \leq 2$, we have seen that the sequence $(\gamma_j)_j$ decreases monotonically from $\frac{1}{2}$ to $1 - \frac{1}{c+d}$. In this case, we note that the exponent of $\max(a, (c+d)\gamma_j)$ in the above running time simplifies to $(c + d)\gamma_j$ provided that $a \leq c + d - 1$; furthermore, this exponent is at most 1. In this case, expanding t^* shows that the running time is

$$O\left((\frac{t}{b}s^k)^{(c+d)c\gamma_j\beta_{j,k}} + bs + (n+s)^{c^{k+1}}\right).$$

We minimize this expression up to a constant factor by choosing b so that the two terms depending on b are equal. This occurs at the value

$$b^* \doteq (t^{(c+d)c\gamma_j\beta_{j,k}} s^{k(c+d)c\gamma_j\beta_{j,k}-1})^{\frac{1}{(c+d)c\gamma_j\beta_{j,k}+1}}.$$

When $b^* \geq 1$, this yields a running time of

$$O\left((ts^{k+1})^{\frac{(c+d)c\gamma_j\beta_{j,k}}{(c+d)c\gamma_j\beta_{j,k}+1}} + (n+s)^{c^{k+1}}\right) = O\left((ts^{k+1})^{\beta_{j,k+1}} + (n+s)^{c^{k+1}}\right)$$

If $b^* < 1$, then $b^* = 1$ is the best we can do, and the running time becomes $O(s+(n+s)^{c^{k+1}}) = O((n+s)^{c^{k+1}})$. Either case completes the inductive step, proving the lemma. $\square$

We now prove Theorem 6.4 by showing that the speedup offered by Lemma 6.7 actually cannot exist for certain values of a, c and d.

Proof of Theorem 6.4. Assume by way of contradiction (and Theorem 2.3) that

$$\mathrm{NTIME}(n) \subseteq \forall^{n^a}\mathrm{DTISP}(n^c, n^d),$$

where c is given and the positive reals a and d are to be determined later. We must have $c \geq 1$ or else we have an immediate contradiction with Lemma 3.2 by choosing $a \leq 1$. When $c + d \leq 2$ and $a \leq c + d - 1$, the hypothesis in addition to Lemma 6.7 yields

$$\begin{aligned}\Sigma_2\mathrm{TIME}(\tau) &\subseteq \forall^{\tau^{ac}}\mathrm{DTISP}(\tau^{c^2}, \tau^{cd}) \\ &\subseteq \Pi_2\mathrm{TIME}\left(\tau^{(c^2+kcd)\beta_{j,k}} + (n + \tau^{ac} + \tau^{cd})^{c^k}\right) \qquad (6.30)\end{aligned}$$

for integers j and k and time function $\tau \geq n$. We wish to determine the values of c where

$$\beta_{j,k} < 1/c^2 \qquad (6.31)$$

for large enough j and k and some positive d. In this case, choosing τ to be a large enough polynomial,

$$d < \frac{1/\beta_{j,k} - c^2}{kc}, \text{ and } a, d < 1/c^{k+1}, \qquad (6.32)$$

results in the running time of $\tau^{1-\epsilon}$ for positive ϵ in the right-hand side of (6.30). This contradicts the direct diagonalization argument of Lemma 3.2, yielding the lower bound for such c.

Of course, we must ensure that choosing parameters in this way is possible: the choice of j and k that satisfies (6.31) depends on the choice of d, which in turn depends on the choice of k in (6.32). We claim that it is valid to first determine j and k as in (6.31) for some initial value of d and then reduce d to comply with the upper bounds (6.32) that depend on k. This follows because $(\gamma_j)_j$ and $(\beta_{j,k})_k$ only become smaller when d is made smaller, making (6.31) and (6.32) easier to satisfy.

Therefore, we focus on satisfying (6.31). For fixed j, $(\beta_{j,k})_k$ converges to either 0 or $1 - \frac{1}{(c+d)c\gamma_j}$. We are done in the case of the former, as we can pick k large enough to

have $(\beta_{j,k})_k$ smaller than any positive constant. In the case of the latter, notice that this fixed point approaches $1 - \frac{1}{(c+d-1)c}$ as j grows, since $\gamma_j \to 1 - \frac{1}{c+d}$. We want this limit point to be less than $1/c^2$ in order to satisfy (6.31) for large enough j and k; provided that $1 - \frac{1}{(c-1)c} < 1/c^2$, we can choose a positive d so that this holds. The condition on c is equivalent to $c^3 - c^2 - 2c + 1 < 0$. Standard techniques to solve cubic equations show that this occurs in the range of $c \geq 1$ as long as $c < 2\cos(\pi/7)$.

In either case, whenever $c < 2\cos(\pi/7)$ the above method of setting a, d, j, and k given an initial positive value of d such that $1 - \frac{1}{(c+d-1)c} < 1/c^2$ and $c + d \leq 2$ leads to a setting that is guaranteed to satisfy (6.31) and lead to the desired contradiction. □

6.3 Conclusion and Open Problems

This chapter presents the current state-of-the-art time-space lower bounds for the tautologies problem. On the one hand, we improve the quantitative strength of time-space lower bounds for tautologies on nondeterministic machines, boosting the time exponent from $\sqrt{2} \approx 1.414$ to $\sqrt[3]{4} \approx 1.587$. On the other hand, we strengthen the best known time-space lower bounds for tautologies on deterministic machines by proving lower bounds of the same strength for randomized machines with one-sided error. The latter result is tantalizingly close to a full-fledged, nontrivial time-space lower bound for satisfiability on randomized machines with two-sided error.

Aside from bridging the gap to two-sided error algorithms, the most compelling avenue for future work is to improve the quantitative strength of the lower bounds for tautologies. There is no a priori reason to believe that the exponents involved for nondeterministic machines, $\sqrt[3]{4}$, or for one-sided error machines, $2\cos(\pi/7)$, could not be improved upon in the near future by making modifications to current arguments. For example, one might think that applying the inductive hypothesis more than twice per step in the case of nondeterministic machines or more than once per step in the case of one-sided error machines would lead to further progress, but this approach fails. In fact, an automated search exploiting the regularity within current indirect diagonalization practices revealed no evidence of *any* proof

doing better [Wil07a]. Therefore, we believe that an improvement to these lower bounds must involve novel ingredients or even a completely new approach to proving lower bounds for NP-complete and related hard problems.

One way that we might get more mileage out of current indirect diagonalization arguments is to find new ways to exploit their structure. This approach has already yielded dividends: our improvement to nondeterministic lower bounds relies on the input-size behavior of the divide-and-conquer speedup (3.5) in a key way. In fact, this speedup contains another promising detail that has yet to be fully exploited: one of the quantifiers is only over a logarithmic number of bits. It might be possible to achieve better lower bounds by finding a technique to remove such small quantifiers at a reduced cost.

LIST OF REFERENCES

[AKR+01] E. Allender, M. Koucky, D. Ronneburger, S. Roy, and V. Vinay. Time-space tradeoffs in the counting hierarchy. In *Proceedings of the 16th IEEE Conference on Computational Complexity*, pages 295–302. IEEE, 2001.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on the Theory of Computing*, pages 421–429. ACM, 1985.

[BDG95] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1995.

[BM88] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[BP01] P. Beame and T. Pitassi. Propositional proof complexity: Past, present, and future. In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science: Entering the 21st Century*, pages 42–70. World Scientific, 2001.

[BSSV03] P. Beame, M. Saks, X. Sun, and E. Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *Journal of the ACM*, 50(2):154–195, 2003.

[CEG95] R. Canetti, G. Even, and O. Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53:17–25, 1995.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on the Theory of Computing*, pages 151–158. ACM, 1971.

[Coo88] S. Cook. Short propositional formulas represent nondeterministic computations. *Information Processing Letters*, 26:269–270, 1988.

[CW79] L. Carter and M. Wegman. Universal hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[CW89] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 14–19. IEEE, 1989.

[Die07] S. Diehl. Lower bounds for swapping Arthur and Merlin. In *Proceedings of the 11th International Workshop on Randomization and Computation*, pages 449–463. Springer-Verlag, 2007.

[DvM05] S. Diehl and D. van Melkebeek. Time-space lower bounds for the polynomial-time hierarchy on randomized machines. In *Proceedings of the 32nd International Colloquium On Automata, Languages and Programming*, pages 982–993. Springer-Verlag, 2005.

[DvM06] S. Diehl and D. van Melkebeek. Time-space lower bounds for the polynomial-time hierarchy on randomized machines. *SIAM Journal on Computing*, 36(3):563–594, 2006.

[DvMW07] S. Diehl, D. van Melkebeek, and R. Williams. A new time-space lower bound for nondeterministic algorithms solving tautologies. Technical Report 1601, Department of Computer Sciences, University of Wisconsin-Madison, 2007.

[FLvMV05] L. Fortnow, R. Lipton, D. van Melkebeek, and A. Viglas. Time-space lower bounds for satisfiability. *Journal of the ACM*, 52:835–865, 2005.

[FvM00] L. Fortnow and D. van Melkebeek. Time-space tradeoffs for nondeterministic computation. In *Proceedings of the 15th IEEE Conference on Computational Complexity*, pages 2–13. IEEE, 2000.

[For00] L. Fortnow. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60:337–353, 2000.

[FSS84] M. L. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[GG81] O. Gabber and Z. Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences*, 22:407–420, 1981.

[vzGG03] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2003.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38:691–729, 1991.

[GS89] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 73–90. JAI Press, Greenwich, 1989.

[GZ97] O. Goldreich and D. Zuckerman. Another proof that BPP $\subseteq$ PH (and more). Technical Report TR-97-045, Electronic Colloquium on Computational Complexity, 1997.

[HS66] F. Hennie and R. Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13:533–546, 1966.

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 248–253. IEEE, 1989.

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

[Lau83] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17:215–217, 1983.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39:859–868, 1992.

[vL91] J. H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1991.

[LV99] R. Lipton and A. Viglas. On the complexity of SAT. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 459–464. IEEE, 1999.

[Mar73] G. A. Margulis. Explicit construction of concentrators. *Problems of Information Transmission*, 9:325–332, 1973.

[vM07] D. van Melkebeek. A survey of lower bounds for satisfiability and related problems. *Foundations and Trends in Theoretical Computer Science*, 2:197–303, 2007.

[vMP07] D. van Melkebeek and K. Pervyshev. A generic time hierarchy for semantic models with one bit of advice. *Computational Complexity*, 16:197–303, 2007.

[MR97] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1997.

[Nis93] N. Nisan. On read-once vs. multiple access to randomness in logspace. *Theoretical Computer Science*, 107:135–144, 1993.

[Nis94] N. Nisan. RL $\subseteq$ SC. *Computational Complexity*, 4:1–11, 1994.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Rab80] M. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12:128–138, 1980.

[Raz03] A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. Unpublished manuscript, 2002–2003.

[Rob91] J. Robson. An $O(T \log T)$ reduction from RAM computations to satisfiability. *Theoretical Computer Science*, 82:141–149, 1991.

[San90] M. Santha. Relativized Arthur-Merlin versus Merlin-Arthur games. *Information and Computation*, 80(1):44–49, 1990.

[Sav70] W. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[SBI04] N. Segerlind, S. Buss, and R. Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM Journal on Computing*, 33(5):1171–1200, 2004.

[Sha92] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39:869–877, 1992.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Uma01] C. Umans. The minimum equivalent DNF problem and shortest implicants. *Journal of Computer and System Sciences*, 63(4):597–611, 2001.

[Vio07] E. Viola. On approximate majority and probabilistic time. In *Proceedings of the 22nd IEEE Conference on Computational Complexity*, pages 155–167. IEEE, 2007.

[Wil06] R. Williams. Inductive time-space lower bounds for SAT and related problems. *Computational Complexity*, 15(4):433–470, 2006.

[Wil07a] R. Williams. *Algorithms and Resource Requirements for Fundamental Problems*. PhD thesis, Carnegie Mellon University, 2007.

[Wil07b] R. Williams. Time-space tradeoffs for counting NP solutions modulo integers. In *Proceedings of the 22nd IEEE Conference on Computational Complexity*, pages 70–82. IEEE, 2007.