Non-Linear Time Lower Bound for (Succinct) Quantified Boolean Formulas

Ryan Williams*
Carnegie Mellon University

Abstract

We prove a model-independent non-linear time lower bound for a slight generalization of the quantified Boolean formula problem (QBF). In particular, we give a reduction from arbitrary languages in alternating time $t(n)$ to QBFs describable in $O(t(n))$ bits by a reasonable (polynomially) succinct encoding. The reduction works for many reasonable machine models, including multitape Turing machines, random access Turing machines, tree computers, and logarithmic-cost RAMs. By a simple diagonalization, it follows that the succinct QBF problem requires superlinear time on those models. To our knowledge this is the first known instance of a non-linear time lower bound (with no space restriction) for solving a natural linear space problem on a variety of computational models.

*Email: ryanw@cs.cmu.edu.

1 Introduction

While our knowledge of time efficient algorithms has grown to a good level of sophistication, when it comes to proving time lower bounds we still have a long road ahead of us. It seems that we are far from settling frontier separation problems like $\mathsf{P} \neq \mathsf{PSPACE}$, although some modest progress has been made on $\mathsf{LOGSPACE} \neq \mathsf{NP}$ in the form of time-space tradeoffs for satisfiability (see *e.g.* [vM07, Wil07] for recent overviews of this line of work). However, these time-space tradeoff lower bounds critically rely on tiny space bounds. Even the task of proving *non-linear* time lower bounds for natural problems within PSPACE has become a surprisingly formidable challenge. Since the seminal result of Paul-Pippenger-Szemeredi-Trotter that $\mathsf{DTIME}[n] \neq \mathsf{NTIME}[n]$ for multitape Turing machines [PPST83], not much progress has been made on proving non-linear time lower bounds for NP problems. Progress has also stalled on lower bounds for PSPACE problems as well. For more details, see the survey by Regan [Reg95] (which is still up-to-date today).

In this paper we establish a non-linear time lower bound on a variant of the canonical PSPACE-complete problem: that of determining if an arbitrary quantified Boolean formula is true (abbreviated as QBF). Our result holds for a variety of computational models, both sequential-access and random-access. The proof interleaves several diagonalization-based components from prior work, and it yields new insight into how we can prove lower bounds on *natural* problems by bootstrapping from lower bounds for artificial problems (for example, those arising from a time hierarchy theorem). Such results are interesting since lower bounds for artificially constructed problems are plentiful, while bounds for natural well-studied problems are rare.

Here we give a reduction-based approach to lower bounds for QBF, by showing that if we choose a natural, polynomially succinct encoding for formulas, then we can reduce arbitrary languages in alternating time $t(n)$ to quantified Boolean formulas that use $O(t(n))$ bits to describe. This reduction, coupled with the fact that deterministic algorithms can be simulated asymptotically faster with alternating algorithms, yields a non-linear lower bound for solving these succinct QBFs. It is important to note that such a reduction is impossible to achieve with QBFs encoded in the traditional way (in prefix-normal form with a CNF predicate), since in that case the QBF instances of length n can be actually solved in alternating $O(n/\log n)$ time by machines having random access to the input.[1] Thus the existence of such a reduction would imply $\mathsf{ATIME}[t(n)] \subseteq \mathsf{ATIME}[t(n)/\log t(n)]$, an absurdity.

Why is this result new? It has been known for many years that deterministic algorithms can be simulated with faster alternating algorithms [PPR80, DT85, Mak94, Mak97]; in fact, multitape Turing machines can be simulated faster by alternating multitape Turing machines that take only a *constant* number of alternations [PPST83]. Clearly these results show that there is *some* language in alternating time $O(n)$ that cannot be solved in deterministic time $O(n)$. However, one cannot conclude a non-linear time lower bound for QBF (even our succinct QBF problem) because of the overheads in known reductions: the best known reductions from arbitrary alternating time $t(n)$ languages to quantified Boolean formulas result in QBFs of at least $\Omega(t(n)\log^2 t(n))$ size [Sch78, Coo88, Rob91]. (Also, one cannot conclude lower bounds for QBF from results like $\mathsf{DTIME}[t(n)] \subseteq \mathsf{SPACE}[t/\log t]$ [HPV77], as the best known reduction from $\mathsf{SPACE}[n]$ to QBF via Savitch's theorem [Sav70] outputs a formula of $\Theta(n^2 \log n)$ bits [SM73].) In this paper, we eliminate the $\log^2 t$ blowup in these reductions by choosing a good encoding and exploiting the power of alternation. The result is that we provide a time lower bound for a natural variant of QBF that allows for groups of $\log n$ variables to "address" one of the n variables. (For more details, see the next section.)

[1] To see this, first note that any formula with N variable occurrences requires $\Omega(N \log N)$ bits to write down, so a QBF of length n can have only $O(n/\log n)$ variables under the normal encoding. Secondly, note that a single clause can be (universally) chosen in $O(\log n)$ time, and a single literal of a clause can be (existentially) chosen in $O(\log n)$ time. An alternating algorithm for QBF could then (in $O(n/\log n)$ time) guess bits for all the relevant variables in its quantifier prefix, universally guess a clause, existentially guess a literal (to be true), existentially guess where the corresponding variable is located in the quantifier prefix, then check all its guesses.

2 Preliminaries

We use the abbreviations: TM for "Turing machine", ATM for "alternating Turing machine", RAM for "random access machine", and ARAM for "alternating random access machine." In this paper, all our TMs are allowed to have access to multiple input tapes. We refer to a *one-tape TM* as a Turing machine that is allowed random access and two-way access to multiple read-only input tapes, and one read-write tape that is two-way sequentially accessed. The random accesses can be up to polylogarithmic cost. A very similar model has been studied before in context of time lower bounds [vMR05].

Related Work. In the past, there have been several demonstrations of time lower bounds for natural problems with no space restriction. Adachi *et al.* [AIK84] showed that a problem called the *cat and k-mouse game* requires at least $n^{\Omega(k)}$ time (infinitely often) to solve on a multitape TM, using a reduction from $\mathsf{DTIME}[n^k]$ to the game. Grandjean [Gra88, Gra90] applied the result that $\mathsf{DTIME}[n] \neq \mathsf{NTIME}[n]$ for multitape TMs [PPST83] to prove that a satisfiability problem over the natural numbers and the NP-complete problem *Reduction of Incompletely Specified Automata* ([GJ79], Problem AL7) both require non-linear time on a multitape TM. While the above work also uses reductions to obtain their results, our work differs significantly in both techniques and implications: we utilize alternations in a novel way to keep our reduction's runtime low, and as a result, our reduction works for many more computational models than multitape TMs.

Our Encoding of QBFs. Let us describe one encoding of Boolean formulas that suffices for our results. We restrict ourselves to QBFs in prenex normal form, where the underlying Boolean formula F is written in conjunctive normal form, for example

$$(\exists\ x_1)(\forall\ x_2)\cdots(\exists\ x_n)F.$$

Here our atoms shall be the literals (variables x_i and their negations $\neg x_i$) as well as the *address functions* $X(x_{j_1},\ldots,x_{j_{\lceil\log n\rceil}})$, where $j_1,\ldots,j_{\lceil\log n\rceil} \in [n]$. For binary values v_{j_k}, the meaning of $X(v_{j_1},\ldots,v_{j_{\lceil\log n\rceil}})$ is the variable x_i, where the bit string $v_{j_1}\cdots v_{j_{\lceil\log n\rceil}}$ represents the binary encoding of i. The X-function allows us to refer to a range of variables without actually writing down all of their indices. For example, the following is a QBF in our encoding:

$$(\exists x_1)(\forall x_2)(\exists x_3)(\exists x_4)[X(x_2,x_1)\vee X(x_4,x_3)].$$

The formula is true, since by setting $x_1 = 1$, $x_3 = 1$, and $x_4 = 1$, $X(0,1) = x_1$ and $X(1,1) = x_4$ are both true.

It is important to observe that formulas encoded with the X-function are only "polynomially succinct" representations, and *not* exponentially succinct: any instance of the X-function can be replaced by a collection of $O(n)$ clauses that encode a circuit for the *storage access* function $SA_n(a_0,\ldots,a_{k-1},x_0,\ldots,x_{2^k-1}) = x_a$. We remark that our use of X could be exchanged for other encoding mechanisms as well. For example, Gurevich and Shelah [GS89] have defined *generalized Boolean formulas*, which also use a form of variable addressing. Quantified versions of generalized Boolean formulas would also be amenable to our reduction.

For a QBF in prenex normal form, we use $O(n)$ bits to describe the variable prefix, where the ith bit describes the quantification of variable x_i. For a formula F with ℓ occurrences of literals (as atoms) and at most $O(\ell/\log n)$ occurrences of the X-function, we use $O(n+\ell\log n)$ bits to describe F, taking $O(1)$ bits to write ORs and ANDs, $O(\log^2 n)$ bits to write down an instance of the X-function (as it takes $O(\log n)$ literals as input), and $O(\log n)$ bits to write variables and their negations.

The Models Studied. We assume the reader is familiar with the multitape Turing machine model. This work also discusses a few more models that may require a brief recollection:

- *Logarithmic-Cost RAM:* Here we have a random access machine with the usual addition, load/store, and simple branching instructions. The memory storage is a collection of registers, each holding an integer. Any operations on a register holding integer i take $\Theta(\log i)$ steps, so *e.g.* a random access to a location among s registers takes $O(\log s)$ steps.
- *Random Access TM:* Same functionality as the multitape TM, except that accesses to each tape are performed by writing bits to a special sequential "index tape" of $O(\log t)$ length, after which the tape head jumps to the location specified by the index tape content.
- *Tree Computer:* The auxiliary tapes of a multitape TM are replaced with binary trees. Each node of the tree contains a single symbol (like a tape cell), and the finite control's directions specify whether to move to the parent, left child, or right child of the current node.
- *Multi-dimensional TM:* The auxiliary tapes are replaced by k-dimensional cubes, for a fixed constant k. Finite control directions specify which of $2k$ directions to move to, from the current cell of the cube.

We assume all the above models receive their input on an initial tape of n separate cells (so that every non-trivial problem requires at least linear time to solve). We remark that, on any of the alternating computational models that we study, our succinct QBFs can be solved in linear time.

2.1 Tools We Need

Three prior results are utilized in our reduction. The first says that we can speed up the simulation of a one-tape TM by introducing alternations.

Theorem 2.1 (Maass-Schorr [MS87], Van Melkebeek-Raz [vMR05]) *Every deterministic linear time one-tape TM M can be simulated by a Σ_2 random access TM that runs in $O(n^{2/3}\log^2 n)$ time. In particular, the Σ_2 computation works by first existentially guessing $O(n^{2/3}\log n)$ bits on an auxiliary tape T, universally writing $O(\log n)$ bits to T, then executing a deterministic $O(n^{2/3}\log^2 n)$ time computation on a one-tape TM that takes the original input tapes of M and T as its own (random-access, read-only) input tapes.*

Proof. (Sketch) We combine a crossing sequence argument with a result that shows how to speed up small space computations with an alternating machine. Let M fit the conditions of the theorem statement and let x be an input. Let $CS(x)$ be the set of crossing sequences for $M(x)$ on its read-write sequential tape. We associate each sequence with a cell of the tape. Notice that each element of a crossing sequence for a particular cell is of $O(\log n)$ size (by storing the current position of the input heads in each element).

For every $i = 1, \ldots, n^{1/3}$, define $CS(x, i)$ to be the subset of sequences from $CS(x)$ that range over the tape cells numbered $i + k \cdot n^{1/3}$, for $k = 0, 1, \ldots, n^{2/3} - 1$. Observe that the union of all $CS(x, i)$ is exactly $CS(x)$, and that $CS(x, i) \cap CS(x, j) = \varnothing$ for $i \neq j$. Since the total sum of the lengths of all crossing sequences for $M(x)$ is $O(n)$, there is a j where the total length (in bits) of all crossing sequences in $CS(x, j)$ is at most $O(n^{2/3}\log n)$. If we existentially guess $CS(x, j)$ upfront (on a tape T), then our computation of $M(x)$ reduces to an $O(n \log n)$ time and $O(n^{1/3}\log n)$ space one-tape TM computation which checks that the computation between the crossing sequences of $CS(x, j)$ is valid and accepting.

We have now a nondeterministic computation that guesses $O(n^{2/3}\log n)$ bits, then runs an $O(n \log n)$ time and $O(n^{1/3}\log n)$ space one-tape TM computation. To obtain a Σ_2 random access TM of the desired form, we use a fast simulation of space-bounded computation that dates back to Kannan [Kan84]. Start by existentially guessing a list of $O(n^{1/3})$ configurations of the one-tape TM on input x. Such a guess can be written down on the tape T using $O(n^{2/3}\log n)$ bits. We then universally choose a pair of adjacent configurations on the list, by writing $O(\log n)$ bits on T. Finally, we move the tape head to the guessed pair, and verify that from the first configuration in the pair, the second configuration can be reached within

$O(n^{2/3} \log n)$ steps, on the one-tape TM with input x. On a one-tape TM, this simulation will have $O(\log n)$ running time overhead for each simulated step. □

The second result we use expresses multitape TM computations as short instances of the satisfiability problem.

Theorem 2.2 (Cook [Coo88]) *Let M be a $t(n)$ time bounded multitape Turing machine. For each input x there is a CNF F having $O(t(n) \log t(n))$ literals such that F is satisfiable if and only if $M(x)$ accepts. Moreover, the CNF can be constructed in $O(t(n) \log^2 t(n))$ time on a multitape TM.*

Cook actually proves the result for nondeterministic Turing machines, pointing out that Pippenger and Fischer's $O(t(n) \log t(n))$-size circuits for time $t(n)$ machines [PF79] yield the result easily. The idea behind the construction of the circuits is to take an arbitrary time $t(n)$ Turing machine and make it *oblivious* to its inputs: that is, the tape head movements of the Turing machine on an input x depends only on $|x|$, the input length. The oblivious simulation yields a Turing machine that runs in $O(t(n) \log t(n))$ time.

The third result we require is that, in the alternating machine setting, many different storage structures can be reduced to a one-tape TM with no asymptotic loss in time efficiency, by introducing more alternations. (This is similar to the space bounded case, where most storage structures that use $O(s)$ space can be efficiently simulated using only a sequential tape of $O(s)$ cells—this is the so-called Invariance Thesis [vEB90].) For example, Paul, Prauss and Reischuk show that one only needs to introduce a constant number of new alternations in order to simulate a multitape ATM with a one-tape ATM.

Theorem 2.3 (Paul-Prauss-Reischuk [PPR80], Theorem 1) *Multitape ATMs running in time $t(n)$ can be simulated by one-tape ATMs in time $O(t(n))$. In particular, if the multitape ATM uses k alternations, then the one-tape ATM uses ck alternations, for some universal constant $c \geq 1$.*

The above can be used to simulate several stronger ATM models. Mak claimed the following but only gave a scant proof outline.

Theorem 2.4 (Mak [Mak94]) *Every alternating tree computer, alternating logarithmic-cost RAM, random access ATM, and k-dimensional ATM that runs in $t(n)$ time can be simulated by a one-tape ATM in $O(t(n))$ time.*

We sketch how to obtain Theorem 2.4. As far as we know, the following argument is new. It draws from ideas in Paul, Prauss, and Reischuk's proof of Theorem 2.3, as well as the author's prior work on parallelizing algorithms [Wil05].

Proof. (Sketch) Let M be a machine of one of the above kinds, and let x be an input. By Theorem 2.3, it suffices to give a simulation of M with a multitape ATM M'. The simulation has basically two phases. First, M' uses alternations to build a chronological list L of the instructions/transitions executed on a particular run of $M(x)$. For every block of $\log t$ transitions in list L, the head positions (requiring $\Theta(\log t)$ bits) after those transitions are also stored. Such a list L can be easily stored with $O(t)$ bits. Secondly, M' verifies that the list L corresponds to a valid run of $M(x)$ on a fixed sequence of existential and universal choices.

The list L is generated as follows. Let s be the number of transitions in L so far. M' existentially guesses the bits $b_1, \ldots, b_k$ that are read in the $(s+1)$st step, and writes them to the end of L. If M is in a universal/existential mode, then M' switches to a universal/existential mode and writes one of the possible transitions according to the bits read and the current state of M. For every block of $\log t$ transitions written, the head positions $h_1, \ldots, h_k$ are existentially guessed and written to L.

Now that a list of transitions (and $t/\log t$ different head positions) have been written to tape, M' verifies the list's correctness. It universally guesses an integer $s \in [t]$ and intends to verify that the sth transition in

L is *correct*: namely, that the state and read bits $b_1, \ldots, b_k$ asserted for the transition are valid. (If $s = t$, then M' accepts iff the final state of that step is accepting and the transition is correct.)

To verify that the sth transition is correct, M' universally branches into k threads where each thread verifies that b_i is the correct bit read by the ith head of M at step s. Each such thread can be implemented quickly as follows. For a particular $i = 1, \ldots, k$, M' existentially guesses h_i, the position of the ith head of M at step s, as well as the most recent step $t_i < s$ in which h_i was the ith head's position, and universally performs two computations:

- The first examines the block of $O(\log t)$ transitions of L in which step t_i occurs, deterministically verifying that h_i is last accessed during this block at step t_i, and the last bit written at position h_i was b_i. On a multi-tape TM, this takes $O(\text{poly}(\log t))$ time for all of the above models.
- The second computation checks that all steps prior to t_i do not have h_i as the ith head's position. This is done by universally guessing an integer u in $[t_i + 1, s]$, and using the head position information stored in L, verifying that h_i is not the head position in the uth time step.

Since each of these threads simulate at most $O(\log t)$ steps of the original machine M, each of them can be performed easily in $O(\text{poly}(\log t))$ time. □

Note the proof introduces a constant factor of new alternations into the computation. In principle, the technique above can be applied to any model of computation that reads and writes at most a constant number of bits in each step, and has finite size programs.

Corollary 2.1 (Of Theorems 2.3 and 2.4) *Random access ATMs, multitape ATMs, tree computers, log-cost ARAMs, and k-dimensional ATMs can be simulated by a one-tape ATM in $O(t(n))$ time.*

The corollary implies that, in order to establish the initial claim of the abstract, it suffices to give a linear reduction from arbitrary languages accepted by one-tape ATMs to quantified Boolean formulas.

3 Reduction to Succinct QBF

We are ready to give a linear time reduction from alternating time $t(n)$ to succinct QBFs of size $t(n)$. After the reduction, the lower bound follows readily.

Theorem 3.1 *For every language L recognized in alternating time $t(n)$ by any tree computer, k-dimensional TM, random access TM, multitape TM, or logarithmic-cost RAM, there is a linear time reduction from L to quantified Boolean formulas (QBF) describable in $O(t(n))$ bits. The reduction can be performed on any of the above models.*

Let us first outline some of the subtleties involved. The typical reduction from an alternating time machine to a QBF would need at least $\Omega(t(n) \log^2 t(n))$ size. One $\log t(n)$ factor comes from an overhead in reorganizing the machine's reads and writes (for multitape TMs, the machines are made *oblivious* in the sense of Theorem 2.2; for RAMs, cf. Van Melkebeek's survey [vM07, p.14–15] for details). Another $\log t(n)$ factor comes from the simple fact that, without any form of compression, any variable in a formula of $O(t)$ variables requires $\Theta(\log t(n))$ bits to write down, so any reduction that has at least one variable for each step of a time t computation would have to require $\Omega(t \log t)$ bits with a typical encoding. Hence there are at least two obstacles to overcome.

The first obstacle is circumvented by exploiting properties of alternating machines: in particular, with Corollary 2.1, we can reduce all the alternating models mentioned in the theorem statement to one-tape

alternating machines with essentially the same running time. This reduction is very powerful, as it allows us to exploit the restrictiveness of a one-tape machine. That is, we can draw on arguments along the lines of the time-space tradeoff literature, speed up a deterministic portion of the one-tape computation with Theorem 2.1, and then make that fast deterministic portion oblivious. In that way, the $\log t(n)$ slowdown incurred by obliviousness is only applied to an asymptotically small portion of the computation. The second obstacle, that of representing each of t variables with $\Theta(\log t)$ bits, is inherently more difficult to overcome. We can obviate this hindrance using the X-function, which allows us to refer to a range of variables (over an appropriate range of quantified variables) without having to explicitly write down all variables in the range.

Proof of Theorem 3.1. By Corollary 2.1, it suffices for us to give the reduction for an alternating *one-tape* machine M that runs in t time. For notational convenience, we assign $\ell = ct^{2/3}\log t$ for a sufficiently large constant $c \geq 1$ in the following. Let x be an input. We may assume without loss of generality that M first guesses one bit b_1 existentially, then one bit b_2 universally, then one bit b_3 existentially, and so on, for $O(t)$ steps, then M runs a deterministic $O(t)$ time one-tape TM M' that reads input $\langle x, b_1 b_2 \cdots b_{ct}\rangle$. By Theorem 2.1, M' can be replaced by a Σ_2 machine M'' that existentially guesses a string y of ℓ bits, universally guesses a $c\log t$ bit string z, then executes a deterministic $\ell \log t$ time one-tape TM M''' that takes $\langle x, b_1 b_2 \cdots b_{ct}, y, z\rangle$ as its (randomly-accessed) inputs.

We are now ready to convert the computation on x into a succinct quantified Boolean formula. First we hardcode the input bits $x = x_1 \cdots x_n$ into the CNF formula, by including the clauses (x_i) or $(\neg x_i)$ depending on the sign of the ith bit, and including $x_1, \ldots, x_n$ as existentially quantified variables in the prefix. Converting the guesses $b_1 b_2 \cdots b_{ct}$ is straightforward: each (existential/universal) bit b_i is represented by an (existentially/universally) quantified variable x_{n+i}. Similarly, we have existential variables $x_{n+ct+1}, \ldots, x_{n+ct+\ell}$ for the bits of y, universal variables $x_{n+ct+\ell+1}, \ldots, x_{n+ct+\ell+c\log t}$ for the bits of z, and the initial prefix of the QBF is described in $O(t)$ bits. All in all, we have $m = n + ct + \ell + c\log t$ variables of the form x_i.

To convert the one-tape computation $M'''(\langle x, b_1 b_2 \cdots b_{ct}, y, z\rangle)$ into a short CNF formula, we use the X-function, and include some additional existentially quantified variables. For convenience in notation (though we intend for all variables to be of the form x_i) we give these new variables the names:

$$\begin{aligned} u_1^j, \ldots, u_{\log m}^j, &\qquad U^j, \\ v_1^j, \ldots, v_{\log m}^j, &\qquad V^j, \\ \text{and } \; w_1^j, \ldots, w_{\log m}^j, &\qquad W^j, \end{aligned}$$

for all $j = 1, \ldots, \ell \log t$. The capital-letter variables U^j, V^j and W^j shall represent the bits of x, $b_1 \cdots b_{ct}$, and yz (respectively) that are read during the jth step of the computation of M''', and the lowercase-letter variables represent the appropriate indices (among $x_1, \ldots, x_m$) of those bits.

For all $j = 1, \ldots, \ell \log t$, we include the clauses:

$$\begin{aligned} (\neg X(u_1^j, \ldots, u_{\log m}^j) \vee U^j) \quad &\wedge \quad (X(u_1^j, \ldots, u_{\log m}^j) \vee \neg U^j) \\ (\neg X(v_1^j, \ldots, v_{\log m}^j) \vee V^j) \quad &\wedge \quad (X(v_1^j, \ldots, v_{\log m}^j) \vee \neg V^j) \\ (\neg X(w_1^j, \ldots, w_{\log m}^j) \vee W^j) \quad &\wedge \quad (X(w_1^j, \ldots, w_{\log m}^j) \vee \neg W^j) \end{aligned}$$

The above clauses can be represented with $O(\log^2 t)$ bits each, so in total these clauses contribute only $O(t^{2/3}\log^4 t)$ bits to the length of the QBF. The above clauses guarantee that the variables U^j, V^j, and W^j correspond to the appropriate bits of x, $b_1 \cdots b_{ct}$, and yz, respectively. Therefore the remaining CNF formula capturing the computation of M''' can be expressed solely over the $O(t^{2/3}\log^2 t)$ variables U^j, V^j, and W^j. Note M''' reads the bits (U^1, V^1, W^1), (U^2, V^2, W^2), …, $(U^{\ell\log t}, V^{\ell\log t}, W^{\ell\log t})$ in precisely that order (after having written $u_1^j, \ldots, u_{\log m}^j$,$v_1^j, \ldots, v_{\log m}^j$, and $w_1^j, \ldots, w_{\log m}^j$ on its input index tapes).

We claim that the computation of M''' can be expressed as a fast multitape TM computation. Let $\hat{u^j} = u^j_1, \ldots, u^j_{\log m}$, and define $\hat{v^j}$ and $\hat{w^j}$ analogously. Our multitape TM has the following content on the input tape T_0:

$$(U^1, V^1, W^1, \hat{u^1}, \hat{v^1}, \hat{w^1}), (U^2, V^2, W^2, \hat{u^2}, \hat{v^2}, \hat{w^2}), \quad \ldots$$

and the input is read by sweeping the head from left to right. There is also a $O(\log t)$ length tape T_1 that keeps track of the indices that the original one-tape M''' would have kept, and there is an additional read-write tape used just as M''' would use it. Performing this simulation on a multitape TM takes at most $O(\ell \log^2 t)$ time: there are $O(\ell \log t)$ steps to simulate, and each simulation of a step in M''' is done in $O(\log t)$ time, by checking that the addresses claimed on T_0 match those on T_1 and updating the states and tapes appropriately.

Recall Theorem 2.2 says that a multitape TM computation taking T time can be written as an instance of satisfiability, taking $O(T \log^2 T)$ bits to describe. It follows that the computation of M''' can be expressed with a CNF formula of size $O(\ell \log^2 t \cdot \log(\ell \log^2 t)^2) \leq O(t^{2/3} \log^4 t) \leq O(t)$ that introduces at most that many new existentially quantified variables. Finally, observe that every component of the above reduction can be performed in $O(t + t^{2/3}\text{poly}(\log t))$ time on a tree computer, k-dimensional TM, random access TM, multitape TM, or logarithmic-cost RAM. Note all of the models can simulate a time t multitape TM in at most $t \cdot \text{poly}(\log t)$ time. □

Remark 1 *In the above, we introduce $O(t^{2/3} \log^2 t)$ occurrences of the X-function, so the most naïve description of our formula without the X-function would have $\tilde{O}(t^{5/3})$ size. While this may appear unsettling, it is important to note that other succinct encodings are also possible. Essentially we just require an encoding which can efficiently state (in $o(t)$ bits) that a particular subset of $\tilde{O}(t^{2/3})$ bits from $\langle x, b_1 b_2 \cdots b_{ct}, y, z \rangle$ corresponds to the $\tilde{O}(t^{2/3})$ subset read by M'''.*

Following the proof of the above theorem and applying Theorem 2.3 instead of Corollary 2.1, we also obtain a linear time reduction from bounded-alternating time to bounded-quantifier QBFs in the multitape setting, with a constant factor increase in the number of new quantifiers.

Corollary 3.1 *There is a constant $c \geq 1$ such that for all $k \geq 1$ and for every language L recognized by a alternating multitape TM in $t(n)$ time taking at most k alternations, there is a linear time reduction from L to quantified Boolean formulas describable in $O(t(n))$ bits, having at most ck quantifier blocks.*

4 The Lower Bound

For an assortment of algorithm models, one can show that deterministic algorithms in that model running in time $O(t)$ can be simulated by alternating one-tape TMs that run in time $t/\beta(t)$, for some unbounded function $\beta(n)$. This observation stems from a result of Mak:

Theorem 4.1 (Mak [Mak94]) *Any deterministic tree computer running in time $t(n)$ can be simulated by an alternating one-tape TM in time $O(t(n)/\log t(n))$.*

Corollary 4.1 *Any deterministic k-dimensional TM running in time $t(n)$ can be simulated by an alternating one-tape TM in time $O(t(n)5^{k \log^* t}/\log n)$.*

Proof. Reishuk [Rei82] showed how to simulate a k-dimensional TM by an $O(5^{k \log^* t(n)} t(n))$ time bounded tree computer, so the result follows from Theorem 4.1. □

Corollary 4.2 *Any deterministic multitape TM, random access TM, or log-cost RAM running in time $t(n)$ can be simulated by an alternating one-tape TM in time $O(t(n)/\log t(n))$.*

Proof. All three can be simulated by tree computers with no asymptotic slowdown, by simulations of Paul and Reischuk [PR81]. □

By the reduction of Theorem 3.1, a linear time algorithm for QBF in the appropriate machine model implies that every alternating algorithm running in t time can be simulated by a deterministic algorithm in $O(t)$ time. But this deterministic algorithm can be simulated in alternating $t/\beta(n)$ time for some unbounded $\beta(n)$, contradicting the time hierarchy for alternating time in that machine model. In particular, we have

Theorem 4.2 *The succinct QBF problem requires $\Omega(n \log n)$ time (infinitely often) on multitape TMs, random access TMs, log-cost RAMs, and tree computers; the problem requires $\Omega(n \frac{\log n}{5^{k \log^* t}})$ time (infinitely often) on any k-dimensional TM.*

Proof. We prove just one instance of the theorem; the others are analogous. Suppose for contradiction that there is a log-cost RAM M that solves the succinct QBF problem in $o(n \log n)$ time, almost everywhere. Let L be an arbitrary language in $\mathsf{ATIME}[t(n)]$ (under the log-cost RAM model), and let x be an instance. By Theorem 3.1, we know that x can be reduced into a succinct QBF ϕ_x of $O(t(n))$ size. But ϕ_x can be solved in $o(t(n) \log t(n))$ time by M, and in turn, Corollary 4.2 implies that the computation of $M(\phi_x)$ (for all x) can be simulated by a particular alternating log-cost RAM in $o(t(n))$ time. But this implies that $\mathsf{ATIME}[t(n)] \subseteq \mathsf{ATIME}[o(t(n))]$, a contradiction with the alternating time hierarchy. □

We remark that the reduction also implies a non-linear lower bound for succinct quantified Boolean formulas with k quantifier blocks on multitape Turing machines, for some constant k.

Theorem 4.3 *There is a constant k such that quantified Boolean formulas with at least k quantifier blocks requires $\Omega(n \log^* n)$ time to solve (infinitely often) on a multitape TM.*

Proof. Suppose for every $k \geq 1$ that there is an $o(n \log^* n)$ time multitape TM M recognizing the problem. Then by Corollary 3.1, $\Pi_4\mathsf{TIME}[n] \subseteq \mathsf{DTIME}[o(n \log^* n)]$, since we can reduce a $\Pi_4\mathsf{TIME}[n]$ computation into a QBF_{4c} instance of size $O(n)$ and running M on it, for some constant $c \geq 1$. But this implies

$$\Pi_4\mathsf{TIME}[n \log^* n] \subseteq \mathsf{DTIME}[o(n(\log^* n)^2)] \subseteq \Sigma_4\mathsf{TIME}[o(n \log^* n)],$$

where the second inclusion was proved by Paul, Pippenger, Szemeredi, and Trotter [PPST83]. This contradicts the fact (provable by a simple diagonalization) that $\Sigma_\ell\mathsf{TIME}[t] \not\subseteq \Pi_\ell\mathsf{TIME}[o(t)]$, for all $\ell \geq 1$. □

5 Conclusion

We conclude with two related problems of great interest. First, it would of course be desirable to prove a non-linear time lower bound on the QBF problem, as it is traditionally encoded. As mentioned in the Introduction, we cannot do this by giving a linear time reduction from $\mathsf{ATIME}[n]$ to QBF in the random access model, since typical encodings of QBF admit an $O(n/\log n)$ alternating time algorithm. It may be possible to obtain a QBF lower bound by assuming there is a fast QBF algorithm, and using that to build a fast reduction (leading to a contradiction).

Secondly, our approach cannot reduce nondeterministic $O(n)$ time languages to SAT instances described in $O(n)$ bits, even those with a succinct encoding, since we relied on the fact that robust computational models can be optimally simulated by one-tape machines when *additional alternations* are introduced. A linear time reduction from arbitrary languages in $\mathsf{NTIME}[n]$ to SAT would imply non-linear time lower bounds for SAT on multitape Turing machines, a long-sought result.

References

[AIK84] A. Adachi, S. Iwata, and T. Kasai. Some Combinatorial Game Problems Require $\Omega(n^k)$ Time. *Journal of the ACM* 31(2):361–376.

[Coo88] S. A. Cook. Short Propositional Formulas Represent Nondeterministic Computations. *Information Processing Letters* 26(5): 269-270, 1988.

[DT85] P. W. Dymond and M. Tompa. Speedups of Deterministic Machines by Synchronous Parallel Machines. *Journal of Computer and System Sciences* 30(2):149–161, 1985.

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

[GJ79] M. Garey and D. Johnson. *Computers and Intractability.* W.H. Freeman, San Francisco, 1979.

[Gra88] E. Grandjean. A Natural NP-Complete Problem with a Nontrivial Lower Bound. *SIAM Journal on Computing* 17(4):786–809, 1988.

[Gra90] E. Grandjean. A Nontrivial Lower Bound for an NP Problem on Automata. *SIAM Journal on Computing* 19(3):438–451, 1990.

[GS89] Y. Gurevich and S. Shelah. Nearly Linear Time. *Logic at Botik'89, Symposium on Logical Foundations of Computer Science*, Springer LNCS 363, 108–118, 1989

[HPV77] J. Hopcroft, W. Paul, and L. Valiant. On Time Versus Space. *Journal of the ACM* 24(2):332–337, 1977.

[Kan84] R. Kannan. Towards Separating Nondeterminism from Determinism. *Mathematical Systems Theory* 17(1):29–45, 1984.

[MS87] W. Maass and A. Schorr. Speed-Up of Turing Machines with One Work Tape and a Two-Way Input Tape. *SIAM Journal on Computing* 16(1):195–202, 1987.

[Mak94] L. Mak. Are Parallel Machines Always Faster than Sequential Machines? (Preliminary Version). In *Proceedings of STACS*, LNCS 775:137–148, Springer, 1994.

[Mak97] L. Mak. Parallelism Always Helps. *SIAM J. Comput.* 26(1):153–172, 1997.

[vMR05] D. van Melkebeek and R. Raz. A Time Lower Bound for Satisfiability. *Theoretical Computer Science* 348(2-3):311–320, 2005.

[vM07] D. van Melkebeek. A Survey of Lower Bounds for Satisfiability and Related Problems. To appear in *Foundations and Trends in Theoretical Computer Science*, 2007.

[PF79] N. Pippenger and M. J. Fischer. Relations Among Complexity Measures. *Journal of the ACM* 26(2):361–381, 1979.

[SM73] L. J. Stockmeyer and A. R. Meyer. Word Problems Requiring Exponential Time (Preliminary Report). In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, 1–9, 1973.

[PPR80] W. J. Paul, E. J. Prauss, and R. Reischuk. On Alternation. *Acta Informatica* 14:243–255, 1980.

[PR81] W. Paul and R. Reischuk. On Time Versus Space II. *Journal of Computer and System Sciences* 22:312–327, 1981.

[PPST83] W. Paul, N. Pippenger, E. Szemeredi, and W. Trotter. On Determinism Versus Nondeterminism and Related Problems. In *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS)*, 429–438, 1983.

[Reg95] K. Regan. On Super-Linear Lower Bounds in Complexity Theory. In *Proceedings of IEEE Conference on Structure in Complexity Theory*, 50–64, 1995.

[Rei82] R. Reischuk. Fast Implementation of Multidimensional Storage into Tree Storage. *Theoretical Computer Science* 19:253–266, 1982.

[Rob91] J. M. Robson. An $O(T \log T)$ Reduction from RAM Computations to Satisfiability. *Theoretical Computer Science* 82(1):141–149, 1991.

[Sav70] W. J. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *Journal of Computer and System Sciences* 4(2):177–192, 1970.

[Sch78] C. Schnorr. Satisfiability is Quasilinear Complete in NQL. *Journal of the ACM* 25(1):136–145, 1978.

[vEB90] P. van Emde Boas. Machine Models and Simulation. *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, J. van Leeuwen (ed.), 1–66, 1990.

[Wil05] R. Williams. Parellizing Time With Polynomial Circuits. In *Proceedings of ACM Symposium on Parallel Algorithms (SPAA)*, 171–175, 2005.

[Wil07] R. Williams. Algorithms and Resource Requirements for Fundamental Problems. Ph.D. Thesis, Carnegie Mellon University, CMU-CS-07-147, August 2007.