Electronic Colloquium on Computational Complexity, Report No. 16 (2014)

The Complexity of Debate Checking

H. Gökalp Demirci[1], A. C. Cem Say[2] and Abuzer Yakaryılmaz[3,*]

[1]University of Chicago, Department of Computer Science, Chicago, IL, USA
[2]Boğaziçi University, Department of Computer Engineering, Bebek 34342 İstanbul, Turkey
[3]University of Latvia, Faculty of Computing, Raina bulv. 19, Rīga, LV-1586 Latvia
demirci@cs.uchicago.edu,say@boun.edu.tr,abuzer@lu.lv

Keywords: incomplete information debates, private alternation, games, probabilistic computation

Abstract

We study probabilistic debate checking, where a silent resource-bounded verifier reads a dialogue about the membership of a given string in the language under consideration between a prover and a refuter. We consider debates of partial and zero information, where the prover is prevented from seeing some or all of the messages of the refuter, as well as those of complete information. This model combines and generalizes the concepts of one-way interactive proof systems, games of possibly incomplete information, and probabilistically checkable debate systems. We give full characterizations of the classes of languages with debates checkable by verifiers operating under simultaneous bounds of $O(1)$ space and $O(1)$ random bits. It turns out such verifiers are strictly more powerful than their deterministic counterparts, and allowing a logarithmic space bound does not add to this power. PSPACE and EXPTIME have zero- and partial-information debates, respectively, checkable by constant-space verifiers for any desired error bound when we omit the randomness bound. No amount of randomness can give a verifier under only a fixed time bound a significant performance advantage over its deterministic counterpart. However, randomness does seem to help verifiers with simultaneous bounds on space and time. In the case of logspace and polynomial-time verifiers, we show that logarithmic randomness is sufficient to check complete- and partial-information debates for all languages in PSPACE. Any such language can be reduced to the quantified max word problem for matrices, which allows us to present a nonapproximability result for the optimization version of this problem.

1 Introduction

An alternating Turing machine for a language L can be viewed [24] as a system where a deterministic Turing machine (the "verifier") makes a decision about whether the input string w is a member of L upon reading a "complete-information" debate on this issue between a "prover" and a "refuter": There exists a strategy of the prover that guarantees that it will be declared as the winner (that is, w will be accepted) by the verifier if and only if $w \in L$. Reif [35] has shown that the class of languages with debates checkable by space-bounded verifiers is enlarged significantly when the debate format is generalized so that the refuter is allowed to hide some of its messages from the

*Abuzer Yakaryılmaz was partially supported by ERC Advanced Grant MQC.

ISSN 1433-8092

prover[1]: For constant-space verifiers, the class of languages with such partial-information debates is E, whereas the corresponding class for complete-information debates is just the regular languages. Zero-information debates, where the refuter is forced to hide all its messages from the prover, correspond to the class NSPACE(n) under this resource bound [34].

The case where the verifier is upgraded to a probabilistic Turing machine has first been studied by Condon et al. [11], who showed that polynomial-time verifiers that use logarithmically many random bits to read only a constant number of bits of a debate can handle every language in PSPACE. They considered only the complete-information case.

In this paper, we examine the power of debate checking under a wider variety of simultaneous resource bounds, and for all the different levels of information hiding described above. Space-bounded probabilistic verifiers, and the effects of using a constant (nonzero) number of random bits, are considered for the first time. The findings are used to establish a nonapproximability result for an optimization problem.

Similar computational models [8,20] have been used to study various questions in the intersection of computational complexity and game theory. Feigenbaum et al. [20] examined the complexity of two-person zero-sum games with varying properties (like perfect or imperfect recall, and perfect information) of the players. In this game-theoretical context, our model can be seen as involving two-person zero-sum games with perfect recall. We concentrate on players using only pure (e.g. deterministic) strategies, like [8,11,29,34,35]. Mixed (e.g. probabilistic) strategies were considered in [18,20].

The rest of the paper is structured as follows: We introduce our model of debate checking, give an overview of the related models mentioned above, and display the power of deterministic debate checking under various resource bounds in Section 2. Section 3 presents our results on probabilistic debate checking. In Section 3.1.1, we give full characterizations of the three classes of languages that have complete-, zero-, and partial-information debates checkable, for some error bound, by constant-space verifiers which use only a constant number of random bits, independent of the length of the input, as P, PSPACE, and EXPTIME, respectively. Adding logarithmic space to these constant-randomness verifiers does not change their power. We then show in Section 3.1.2 that, besides having debates checkable by constant-space constant-randomness verifiers for some error bound, PSPACE and EXPTIME also have zero- and partial-information debates, respectively, checkable by constant-space verifiers for any desired error bound when we loosen the randomness bound.

We start Section 3.2 by examining the power of debate checking by time-bounded verifiers, and show that no amount of randomness can help a verifier on which only a time bound is imposed to outperform its deterministic counterpart significantly. The classes corresponding to complete- and partial-information debates checked by polynomial-time verifiers coincide with PSPACE. However, we show that randomness does seem to help when we further constrain these time-bounded verifiers to use only logarithmic space in the order of their time bound. For example, complete-information debates checkable by logspace polynomial-time probabilistic verifiers exist for all languages in PSPACE, whereas logspace deterministic verifiers can check complete-information debates only for P. Additionally, in case of logspace and polynomial-time verifiers, we show that logarithmic randomness is sufficient to check complete- and partial-information debates for the languages in PSPACE. Finally, we use the structure of the setup of complete-information debate checking by logspace polynomial-time verifiers to show that any language having debates checkable by such verifiers can be reduced to the quantified max word problem for matrices, which allows us to present

[1]This may be likened to a debate where the prover is deaf, and can “hear” the refuter only when the refuter chooses to write her message on a board for him to read.

a result on the hardness of approximating this problem in Section 3.2.4. Section 4 is a conclusion.

2 Debate Checking

2.1 The Model

A *debate system* is a verifier checking a debate Π written on a one-way read-only accessible tape. The verifier is a probabilistic Turing machine. A debate Π is a string constructed by the prover and the refuter (conventionally named Player 1 and Player 0, and denoted P1 and P0, respectively). The ith symbol Π is determined by P1 (resp. by P0) if i is odd (resp. even), for $i > 0$. The symbols Π are selected from three alphabets named Γ_1, Γ_0, and Δ, such that $\Gamma_0 \cap \Delta = \emptyset$. For $i \in \{0, 1\}$, Γ_i is the set of symbols emittable by Pi that can be seen by the opposing player. Δ is the set of private symbols that P0 may choose to write on the debate without showing to P1. P0 can use any member of $\Gamma_0 \cup \Delta$ in its messages. Before preparing the ith symbol of Π, P1 is assumed to have seen the subsequence of all the public symbols (i.e. those in Γ_0 and Γ_1) and P0 is assumed to seen all the symbols (i.e. those in Γ_0, Γ_1 and Δ) among the first $i-1$ characters of Π.

The verifier declares either P1 or P0 as the winner upon reading some portion of the debate that is sufficient for arriving at a decision. When a language recognition problem is considered, P1 argues that the input string is in the language, while P0 claims the opposite, and the verifier's accepting (resp. rejecting) the input corresponds to P1 (resp. P0) "winning." We allow debates of infinite length, for a cheating player can try to make the verifier run forever, rather than to arrive at the correct decision.

Each round in a debate consists of a symbol from P1 and a symbol from P0. Note that the debate definition in [11] allows the players to alternate after emitting messages that are polynomially long in the input length. Our approach with single-symbol (or, at least, constant-length) messages is necessary for proper accounting of space bounds, since we will consider verifiers that are severely constrained in that regard. When the verifier has sufficient resources, our setup with constant-length messages can emulate the debates of [11] easily. Another difference between the model of [11] and our model with complete-information debates is that the verifiers of [11] have the capability of accessing a desired location in the debate directly, while our verifiers must read the debate sequentially.[2]

The following subsections give a more formal treatment of debate systems.

2.1.1 The Verifier

The verifier has access to a read-only input tape and a single read/write work tape, in addition to the one-way read-only debate tape. The input tape contains the input string between two occurrences of the end-marker ¢, and we assume that the machine's program never attempts to move the input head beyond the end-markers. The input tape head is on the left end-marker, and the debate tape head is on the first symbol of the debate at the start of the process. Let the members of the set $\Diamond = \{\leftarrow, \downarrow, \rightarrow\}$ represent head movements on tapes.

Formally, a verifier is a 12-tuple

$$V = (Q, R, \Sigma, \Gamma, \Gamma_1, \Gamma_0, \Delta, \delta_1, \delta_2, q_0, q_a, q_r), \tag{1}$$

where

[2]The relationship between the probabilistically checkable debate systems of [11] and our model is very similar to the one between probabilistically checkable proofs and interactive proof systems.

Q is the state set,

$R \subseteq Q$ is the finite set of *reading states*,

$q_0, q_a, q_r \in Q$ are the initial, accepting, and rejecting states,

Σ is the input alphabet ($\cent \notin \Sigma$),

Γ is the work tape alphabet, including the blank symbol $\sqcup$,

Γ_1 is the public alphabet of the prover, P1,

Γ_0 is the public alphabet of the refuter, P0,

Δ is the private alphabet of the refuter, with $\Gamma_0 \cap \Delta = \emptyset$,

$\delta_1 : (R - \{q_a, q_r\}) \times (\Sigma \cup \{\cent\}) \times \Gamma \times (\Gamma_1 \cup \Gamma_0 \cup \Delta) \longrightarrow Q \times (\Gamma - \{\sqcup\}) \times \Diamond^2$ is the first transition function, which applies only to reading states (except the halting states q_a and q_r), and,

$\delta_2 : (Q - R - \{q_a, q_r\}) \times (\Sigma \cup \{\cent\}) \times \Gamma \times Q \times (\Gamma - \{\sqcup\}) \times \Diamond^2 \longrightarrow \left\{0, \frac{1}{2}, 1\right\}$ is the second transition function defined for the states in $Q - R - \{q_a, q_r\}$.

The first transition function describes deterministic moves associated by reading a symbol from the debate, whereas the second transition function describes moves with no movement of the debate tape head, and possibly a probabilistic branching. $\delta_1(q, \zeta, \theta, \sigma) = (q', \theta', d_{ih}, d_{wh})$ means that the verifier will switch to state q', write θ' on the work tape, move the input head in direction $d_{ih} \in \Diamond$, the work tape head in direction $d_{wh} \in \Diamond$, and move the debate head on the next character of the debate, if it is originally in reading state q, scanning the symbols ζ, θ, and σ in the input and work tapes, and in the debate, respectively. $\delta_2(q, \zeta, \theta, q', \theta', d_{ih}, d_{wh}) = \rho$ means that the probability that the verifier will switch to state q', write θ' on the work tape, and change positions of input and work tape heads in directions d_{ih} and d_{wh}, respectively, when originally in state q, and reading ζ and θ as the input and work symbols, equals $\rho \in \left\{0, \frac{1}{2}, 1\right\}$ For any such triple of q, ζ and θ, the sum of the transitions triggered by that triple must equal 1. This can be formalized as follows.

$$\forall q \in Q - R - \{q_a, q_r\}, \forall \zeta \in \Sigma, \forall \theta \in \Gamma, \left[\sum_{\substack{q' \in Q, \theta' \in \Gamma, \\ d_{ih}, d_{wh} \in \Diamond}} \delta_2(q, \zeta, \theta, q', \theta', d_{ih}, d_{wh}) = 1 \right].$$

A *configuration* of the verifier is a 4-tuple containing its state, the positions of input and work tape heads, and work tape content. A configuration having a state in R will be called *reading configuration.* If δ_2 allows the machine on some configuration to move to two different configurations each having probability $\frac{1}{2}$, this can be seen as the machine *tossing a coin* to determine the next configuration. We use the terms "tossing coins" and "reading random bits" interchangeably. A verifier is said to be *deterministic* if the codomain of δ_2 is restricted to $\{0, 1\}$ (i.e. the verifier is not allowed to toss coins).

2.1.2 The Debate

A *debate tree* is an infinite tree whose nodes have $|\Gamma_1|$ children at even-numbered levels (including level 0, containing the root node) and $|\Gamma_0| + |\Delta|$ children at odd-numbered levels. The nodes at even-numbered levels of the debate tree (*P1-type nodes*) correspond to points in the debate where it is P1's turn to speak, and each one of the edges connecting such a node to its children corresponds to a different symbol that P1 can decide to be the next character of the debate at that point. Each node at odd-numbered levels (*P0-type nodes*) has a similar correspondence with P0. The edges connecting a P0-type node (resp. a P1-type node) to its children are called *P0-type edges* (resp. *P1-type edges*).

At any point in the debate, P1 can base its decision on what to say next on the sequence of all symbols determined by P0 and P1 up to that moment, except P0 symbols that cannot be "seen" by P1. Let

$$h(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in \Gamma_0 \\ \flat & \text{if } \sigma \in \Delta, \end{cases} \tag{2}$$

where $\flat$ is a "blank" symbol, not in $\Gamma_0 \cup \Delta$. For any P1-type node N, the *P0 sequence seen by N* is created by starting with the empty list, and adding $h(\sigma)$ for each symbol σ one encounters at P0-type edges while walking from the root node down to N.

A *debate subtree* is a subtree of a debate tree where each P1-type node has just one child, and each P0-type node has $|\Gamma_0|+|\Delta|$ children. A debate subtree is said to be *well-formed* if, for any two P1-type nodes N_1 and N_2 in the same level, and their only children N_1' and N_2', the edges connecting N_1 to N_1' and N_2 to N_2' correspond to different symbols from $|\Gamma_1|$ only if the P0 sequences seen by N_1 and N_2 are different. Intuitively, this well-formedness condition stands for our desire that P1's messages should reflect the level of ignorance that it has about the private messages of P0.

Formally, a *debate* Π is a sequence of symbols that labels an infinite path starting at the root node of a well-formed debate subtree.

The general definition we have given corresponds to *partial-information* debates. When $\Delta = \emptyset$, (i.e. P0 never emits private symbols), one has a *complete-information* debate. The other extreme, where P0 never emits public symbols ($\Gamma_0 = \emptyset$), corresponds to *zero-information* debates.

2.1.3 Language Recognition

We will associate languages with debate systems in two different ways. We start with the "strong definition."

The probability that a verifier V accepts an input string x (i.e. ends up in q_a) as the result of reading a debate Π is denoted by $P^{Acc}_{(V,\Pi)}(x)$ and calculated in a natural way over the coin tosses of the verifier. $P^{Rej}_{(V,\Pi)}(x)$ denotes the probability that V rejects x in such a scenario.

We say that language L *has debates checkable with error probability ε* if there exists a verifier V such that

1. for every $x \in L$, there is a well-formed debate subtree on which, for all debates Π labeling a path of this subtree, $P^{Acc}_{(V,\Pi)}(x) \geq 1-\varepsilon$, and,

2. for every $x \notin L$, on all well-formed debate subtrees, there exists a debate Π labeling some path of the subtree such that $P^{Rej}_{(V,\Pi)}(x) \geq 1-\varepsilon$.

In other words, the prover is able to convince the verifier to accept x with high probability, no matter what the refuter says (in public or private), if and only if $x \in L$.

When we replace item (2) above with the following condition, we obtain the "weak definition" of debate checking, where the verifier does not have to halt with high probability when $x \notin L$:

2'. for every $x \notin L$, on all well-formed debate subtrees, there exists a debate Π labeling some path of the subtree such that $P^{Acc}_{(V,\Pi)}(x) \leq \varepsilon$.

Note that we have defined language recognition in the *two-sided bounded error* setting. We leave the investigation of the languages having debates checkable in other error settings (e.g. unbounded-error) for future study.

We trust the truthful player[3] to put the best effort to defend its claim. In other words, if the input is in the language, P1 is assumed to follow the strategy of the debate subtree in item (1) above, whose debates make the verifier accept with high probability. Otherwise, P0 is expected to emit symbols which would cause the verifier to reject (as in item (2)) or at least avoid accepting states (as in item ($2'$)) for this specific P1. The behavior of the verifier on debates that do not meet this requirement is immaterial. Therefore, we define resource bounds on verifiers only for such "valid" debates.

A verifier is said to have space bound $s(n)$ (resp. time bound $t(n)$) if the work tape head scans at most the $s(n)$th work tape cell in any configuration reachable from the initial configuration (resp. if it halts in $t(n)$ steps) on any input with length n, any valid debate, and any sequence of random bits. It is said to have randomness bound $r(n)$ if it reads at most $r(n)$ random bits on any input with length n, and any valid debate.

2.1.4 The Notation

$\mathsf{CDEB}(s,t,r)$ is the class of languages that have complete-information debates checkable with some error probability $\varepsilon < \frac{1}{2}$, such that the verifier of the debate uses $O(s(n))$ space, $O(t(n))$ time, and $O(r(n))$ random bits. The classes of languages that have partial-information debates and zero-information debates with these restrictions have names of the form $\mathsf{PDEB}(s,t,r)$ and $\mathsf{ZDEB}(s,t,r)$, respectively. $\mathsf{CDEB}_w(s,t,r)$, $\mathsf{PDEB}_w(s,t,r)$, and $\mathsf{ZDEB}_w(s,t,r)$ are the corresponding classes of languages recognized with respect to the weak definition. Note that, since these "weak" debate systems are less constrained than the "strong" ones of the previous definition, the xDEB classes are always contained in the corresponding xDEB_w classes for $\mathsf{x} \in \{\mathsf{C},\mathsf{Z},\mathsf{P}\}$. We use notations *cons*, *log*, *poly*, *exp* to stand for functions in $O(1)$, $O(\log n)$, $O(n^c)$, $O(2^{n^c})$, respectively, for any constant c. If there is no restriction on a resource, we indicate this by ∞. The verifier is deterministic if the randomness parameter is set to 0.

We differentiate between the class of languages that have debates checkable with some error probability, and the class of languages that have debates checkable for *all* positive error probabilities $\varepsilon < \frac{1}{2}$. The latter is denoted by adding an $^{\forall\varepsilon}$ to the corresponding class name (e.g. $\mathsf{PDEB}_w^{\forall\varepsilon}(s,t,r)$ is the class of languages that have partial information debates according to the weak definition for all positive error bounds $\varepsilon < \frac{1}{2}$). By definition, the $\mathsf{xDEB}^{\forall\varepsilon}$ classes are always contained in the corresponding xDEB classes. Since a time-bounded verifier can be made to repeat itself and announce the majority result to meet any desired error probability with just a constant factor increase in its time and randomness costs, this distinction is needed only for the classes associated with debates checkable with verifiers having no time bounds.

2.2 Relation to Other Models

This section introduces the other computational models that are closely related to debate checking. Sections 2.2.1 and 2.2.2 give the definitions of alternating, private alternating, and blind alternating Turing machines, and show the exact correspondence between alternation and debate checking with deterministic verifiers. The rest of the section explores the relation between debate systems and the other proof systems in the literature.

[3]The truthful player can be either P1 or P0 depending on the membership situation of a specific input.

In this and the following sections, we use the notation $\mathsf{xTIME}(t(n))$ to denote the class of languages recognized by deterministic (resp. nondeterministic, alternating, private alternating, and blind alternating) Turing machines with an input tape and a separate work tape in time $O(t(n))$ if $\mathsf{x} = \mathsf{D}$ (resp. $\mathsf{x} = \mathsf{N}$, $\mathsf{x} = \mathsf{A}$, $\mathsf{x} = \mathsf{PA}$, and $\mathsf{x} = \mathsf{BA}$). The class of languages recognized by corresponding machines with space-bound $O(s(n))$ is denoted by $\mathsf{xSPACE}(s(n))$, for $\mathsf{x} \in \{\mathsf{D}, \mathsf{N}, \mathsf{A}, \mathsf{PA}, \mathsf{BA}\}$. xL stands for $\mathsf{xSPACE}(\log n)$ for $\mathsf{x} \in \{\mathsf{N}, \mathsf{A}\}$. xP, $\mathsf{xPSPACE}$ and $\mathsf{xEXPTIME}$ stand for $\cup_{c\geq 1}\mathsf{xTIME}(n^c)$, $\cup_{c\geq 1}\mathsf{xSPACE}(n^c)$, and $\cup_{c\geq 1}\mathsf{xTIME}(2^{n^c})$ for $\mathsf{x} \in \{\mathsf{N}, \mathsf{A}, \mathsf{PA}, \mathsf{BA}\}$, respectively. L, P, PSPACE, E, $\mathsf{EXPTIME}$, and $\mathsf{EXPSPACE}$ stand for $\mathsf{DSPACE}(\log n)$, $\cup_{c\geq 1}\mathsf{DTIME}(n^c)$, $\cup_{c\geq 1}\mathsf{DSPACE}(n^c)$, $\mathsf{DTIME}(2^{O(n)})$, $\cup_{c\geq 1}\mathsf{DTIME}(2^{n^c})$, and $\cup_{c\geq 1}\mathsf{DSPACE}(2^{n^c})$. $\mathsf{2AFA}(k)$ denotes the class of languages recognized by alternating multihead finite automata with k input heads [28]. $\Sigma_i\mathsf{TIME}(t(n))$ is the ith level of the $t(n)$-time hierarchy [2], and REG denotes the class of regular languages.

2.2.1 Alternating Turing Machines

An *alternating Turing machine* (ATM) is an 8-tuple

$$M = (S, Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r),$$

where

S is a finite set,

$Q \subseteq S \times \{0, 1\}$ is the finite state set,

$q_0, q_a, q_r \in Q$ are the initial, accepting and rejecting states,

Σ and Γ are the input and work tape alphabets, respectively, (with ¢ $\notin \Sigma$ as the input end-marker and $\llcorner\!\lrcorner \in \Gamma$ as the blank tape symbol),

$\delta : Q \times (\Sigma \cup \{¢\}) \times \Gamma \longrightarrow \mathcal{P}\left(Q \times (\Gamma - \{\llcorner\!\lrcorner\}) \times \Diamond^2\right)$ is the transition function.

A state $(s, i) \in Q$ of an ATM is said to be *existential* (resp. *universal*) if $i = 1$ (resp. $i = 0$).

A configuration of an ATM consists of the state, the positions of input and work tape heads, and the work tape content. The initial configuration has the initial state as its state component, with the input head on the first end-marker ¢, and the work tape head at the leftmost cell of the empty work tape. Accepting and rejecting configurations are the ones with accepting and rejecting states, respectively. A configuration is universal if its state is universal. Otherwise, it is existential. If a configuration β follows from another configuration α according to the transition function δ, we say that β is a *successor* of α, and denote this by $\alpha \vdash \beta$.

For an input string w, the function l labels configurations as either 1 or 0 as follows:

$$l(\alpha) = \begin{cases} 1, & \text{if } \alpha \text{ is an accepting configuration,} \\ 0, & \text{if } \alpha \text{ is a rejecting configuration,} \\ \bigvee\limits_{\alpha\vdash\beta} l(\beta), & \text{if } \alpha \text{ is existential,} \\ \bigwedge\limits_{\alpha\vdash\beta} l(\beta), & \text{if } \alpha \text{ is universal,} \end{cases}$$

where $\bigwedge\limits_{\alpha\vdash\beta} l(\beta)$ and $\bigvee\limits_{\alpha\vdash\beta} l(\beta)$ stand for the logical conjunction and disjunction of the labelings of all possible successors of α, respectively. Note that the labeling of one of its successors may be sufficient to determine the labeling of some configuration, i.e. an existential configuration is bound to be labeled 1 if one of its successors is already labeled 1, and a universal configuration should be labeled 0 without having to know the labelings of all the successors if just one of them is labeled 0. This fact enables one to label a configuration if the labeling of one of its successors that is sufficient to determine it is known, even if some infinite branches of successors hang from it.[4] An ATM is said to accept input w if and only if the initial configuration of the machine on w is labeled 1.

Nondeterministic Turing machines (NTMs) are ATMs with no reachable universal states. If NTMs are further restricted to have single successor on every configuration, we obtain *deterministic Turing machines* (DTMs).

The ATM is a computational model for games of perfect information between the *existential player*, who makes its moves on existential configurations, and the *universal player*, who determines the transition to be made on universal configurations, among the many choices permitted by the transition function of the ATM. In this analogy, the initial configuration corresponds to the initial position of the game, and the recursive labeling essentially formalizes the idea that the input is accepted only if the existential player has a *winning strategy* for this initial position, regardless of the strategy followed by the universal player.

Slightly different versions of ATM definitions are available in the literature, e.g. see [7, 21, 30].[5] We have given a simplified version of the definition in [7].[6] Alternatively to the labeling-based definition above, language recognition by ATMs can also be defined through the existence of *finite accepting subtrees* as follows: Any node N in a finite accepting subtree corresponds to a configuration α of the ATM. If α is existential, N has exactly one child, corresponding to a successor of α. Otherwise, for any successor β of α, a node corresponding to β is among the children of N. The root node corresponds to the starting configuration. All the leaves in a finite accepting subtree correspond to accepting configurations. The existence of a finite accepting subtree for an ATM on some input implies that the input is in the language of the ATM. Intuitively, an accepting subtree completely defines a winning strategy for the existential player. Note that this new language recognition definition for ATMs resembles our definition in Section 2.1.3 of a language having debates associated with it. This similarity lets us prove that the language recognition powers of ATMs and deterministic verifiers checking complete-information debates are the same under any resource bound in Lemmas 1 and 2.

Lemma 1. *For any time bound $t(n)$ and space bound $s(n)$,*

$$\mathsf{ATIME}(t(n)) \subseteq \mathsf{CDEB}(\infty, t(n), 0),$$

and

$$\mathsf{ASPACE}(s(n)) \subseteq \mathsf{CDEB}(s(n), \infty, 0).$$

Proof. Let L be a language recognized by ATM $M = (S, Q, \Sigma, \Gamma, \delta_M, q_0, q_a, q_r)$, which uses at most $s(|w|)$ work tape cells, and halts within $t(|w|)$ steps on any input w. Without loss of generality, we make certain assumptions about M. The initial and halting states of M are existential, and the transition function δ_M makes sure that the computation of the machine always alternates between universal and existential states (i.e. at any time during computation, if the current state is universal, then the next state is existential and vice versa). These assumptions are needed to match our debates where constant length messages are sent at each turn with alternation, and has no effect on

[4]See [7] for a more detailed and formal treatment.

[5]A quantum version was introduced recently in [42].

[6]The simplification is the elimination of negating states and multiple work tapes.

the language recognition power of alternating Turing machines (an ATM that does not meet this requirement can be modified by introducing dummy existential and universal states to make sure that the computation always alternates between existential and universal states, which only multiplies the runtime with a constant). We construct a verifier $V = (Q, Q, \Sigma, \Gamma, \Gamma_1, \Gamma_1, \emptyset, \delta_1, \delta_2, q_0, q_a, q_r)$ that checks complete-information debates for membership in L.

V has the same state set Q, the same initial, accept, and reject states, and the same input and tape alphabets as M. Since all the states of V are reading states, δ_2 is immaterial, and need not be defined specifically. We define Γ_1, the common public alphabet of P1 and P0, such that a bijection $b : Q \times \Gamma \times \Diamond^2 \longrightarrow \Gamma_1$ exists.

δ_1 is defined, using δ_M, as follows. If $\delta_M(q, \zeta, \theta) = C \subseteq Q \times \Gamma \times \Diamond^2$, then

$$\delta_1(q, \zeta, \theta, b(c)) = \begin{cases} c & \text{if } c \in C, \\ (q_r, \theta, \downarrow, \downarrow) & \text{if } c \notin C \text{ and } q \text{ is an existential state,} \\ (q_a, \theta, \downarrow, \downarrow) & \text{if } c \notin C \text{ and } q \text{ is a universal state.} \end{cases}$$

b defines a correspondence between transitions of M and the messages of P1 and P0. A configuration of M may have possibly multiple successors. V, simulating M, can realize only one of these successors, and the one to be realized is chosen with respect to the message of P1 or P0 at that point in the debate depending on the configuration being existential or universal, respectively. P1 and P0 are responsible for choosing a message which stands for a successor of the current configuration. Otherwise, V declares the opponent as the winner.

Using its next message in the debate, P1 tries to point out a universal configuration β of M labeled 1, such that $\alpha \vdash \beta$, which is sufficient to label the current existential configuration α as 1 too. On the other hand, if α is universal, P0 uses its corresponding message in the debate to show $l(\alpha) = 0$ by indicating a transition $\alpha \vdash \beta$ which leads to an existential configuration β labeled 0. Therefore, on any input w, labeling of the initial configuration of M as 1 is possible if and only if a well-formed debate subtree exists, such that for all debates on this subtree, V accepts. V clearly uses the same amount of space and time as M. □

Lemma 2. *For any time bound $t(n)$ and space bound $s(n)$,*

$$\mathsf{CDEB}(\infty, t(n), 0) \subseteq \mathsf{ATIME}(t(n)),$$

and

$$\mathsf{CDEB}(s(n), \infty, 0) \subseteq \mathsf{ASPACE}(s(n)).$$

Proof. Let $V = (Q, R, \Sigma, \Gamma, \Gamma_1, \Gamma_0, \emptyset, \delta_1, \delta_2, q_0, q_a, q_r)$ be a deterministic $s(n)$-space $t(n)$-time verifier that checks complete-information debates for a language L. We construct an ATM $M = (Q, Q \times \{0, 1\}, \Sigma, \Gamma, \delta_M, (q_0, 1), (q_a, 1), (q_r, 1))$ recognizing L.

A direct approach to define δ_M is to let it mimic δ_2 on nonreading states and δ_1 on reading states, using existential and universal moves to play the role of the debate. In other words, for each possible transition in δ_1 on a configuration with the state component q reading some symbol on the debate, δ_M should have a transition on the corresponding configuration with state component (q, i). Furthermore, it should be guaranteed that the state (q, i) is existential (resp. universal) if this configuration of V is reading a symbol in the debate written by P1 (resp. P0). Therefore, i in the state of M should be flipped on any transition corresponding to a transition of V on a reading state, and it should be kept unchanged until another state corresponding to a reading state of V is reached, so that the messages written in the debate alternately by P1 and P0 will be produced alternately by existential and universal states, respectively.

Formally, δ_M is defined as follows.

$$\delta_M((q,i),\zeta,\theta) =$$
$$\begin{cases} \{((q',1-i),\theta',d_{ih},d_{wh}) \mid \exists\sigma\in\Gamma_1 \text{ s.t. } \delta_1(q,\zeta,\theta,\sigma)=(q',\theta',d_{ih},d_{wh})\}, & \text{if } q\in R \text{ and } i=1,\\ \{((q',1-i),\theta',d_{ih},d_{wh}) \mid \exists\sigma\in\Gamma_0 \text{ s.t. } \delta_1(q,\zeta,\theta,\sigma)=(q',\theta',d_{ih},d_{wh})\}, & \text{if } q\in R \text{ and } i=0,\\ \{((q',i),\theta',d_{ih},d_{wh}) \mid \delta_2(q,\zeta,\theta,q',\theta',d_{ih},d_{wh})=1\}, & \text{if } q\notin R, \end{cases}$$

Since M does not introduce a transition or a state that does not have a direct correspondence in δ_1 or δ_2, or Q, it consumes the same amount of space and time as V. □

Lemmas 1 and 2 show the exact correspondence between alternation and complete-information debate checking by deterministic verifiers. In this correspondence, P1 and P0 act as existential and universal players, respectively. Since the strategies of the players of an ATM determine exactly which transition is to be made at each step during the computation, once the strategies of the players are specified by P1 and P0, what remains is only a deterministic Turing machine, the verifier.

Hence, we can use some well-known facts about the language recognition power of ATMs to characterize that of complete-information debate checking by deterministic verifiers.

Theorem 1 ([7]). *For any space bound $s(n) \geq \log n$,*

$$\mathsf{CDEB}(s(n),\infty,0) = \mathsf{ASPACE}(s(n)) = \mathsf{DTIME}(2^{O(s(n))}).$$

Theorem 2 ([7]). *For any time bound $t(n) \geq n$,*

$$\mathsf{NSPACE}(t(n)) \subseteq \mathsf{CDEB}(\infty,t(n)^2,0) = \mathsf{ATIME}(t(n)^2),$$

and

$$\mathsf{CDEB}(\infty,t(n),0) = \mathsf{ATIME}(t(n)) \subseteq \mathsf{DSPACE}(t(n)).$$

For well-known space and time bounds, these theorems yield:

Corollary 1. $\mathsf{CDEB}(log,\infty,0) = \mathsf{AL} = \mathsf{P}$,
$\mathsf{CDEB}(poly,\infty,0) = \mathsf{APSPACE} = \mathsf{EXPTIME}$,
$\mathsf{CDEB}(\infty,poly,0) = \mathsf{AP} = \mathsf{PSPACE}$, *and*
$\mathsf{CDEB}(\infty,exp,0) = \mathsf{AEXPTIME} = \mathsf{EXPSPACE}$.

Alternating finite automata (constant-space alternating Turing machines) have also been studied, and shown to have the same language recognition power as deterministic finite automata.

Theorem 3 ([30]). $\mathsf{CDEB}(cons,\infty,0) = \mathsf{ASPACE}(cons) = \mathsf{REG}$.

It is well known that multiple input heads are equivalent to logarithmic space [27], for instance, the class of languages recognized by deterministic multihead finite automata is precisely L. This equivalence carries over to the nondeterministic, probabilistic [32], and alternating versions of these machines: Alternating multihead finite automata with two-way access to the input (denoted 2afa(k) for k input heads) are known to characterize alternating logspace:

Theorem 4 ([28]). $\cup_{k\geq 1}\mathsf{2AFA}(k) = \mathsf{AL} = \mathsf{P}$.

2.2.2 Private and Blind Alternating Turing Machines

Reif [35] introduced a natural extension to alternation by giving computational models for games of imperfect information, where the existential player does not have access to all of the opponent's moves. To incorporate this notion to the alternating Turing machine framework, Reif augmented the ATM model with a *private work tape* and *private states* in addition to the usual tapes and states that are common to both players. In a *private alternating Turing machine* (PATM), only the universal states can execute moves that can see or change the content of the private work tape and states. This prevents the strategy of the existential player from depending on the private content, effectively enforcing the same condition mentioned in our definition of well-formed debate subtrees on the existential choices.

The formal definition of a PATM is different from that of an ATM only in its states and transition function. A state (s_c, s_p, i) of a PATM is an element of $Q \subseteq S_c \times S_p \times \{0,1\}$, where $s_c \in S_c$ and $s_p \in S_p$ are the common and private portions of the state, respectively, and $i \in \{0,1\}$ indicates the state being either existential or universal as in the case of ATMs. The transition function δ of a PATM maps $Q \times (\Sigma \cup \{¢\}) \times \Gamma^2$ to $\mathcal{P}\left(Q \times (\Gamma - \{␣\})^2 \times \Diamond^3\right)$, where the extra components corresponding to the tape symbol in the domain, and the tape symbol and the head direction in the codomain are added to introduce the private work tape. The transition function has the following additional constraint to make sure that the existential moves do not depend on private content:

$$\forall s_p, s'_p \in S_p, \forall \theta_2, \theta'_2 \in \Gamma \left[\delta((s_c, s_p, 1), \zeta, \theta_1, \theta_2) = \delta((s_c, s'_p, 1), \zeta, \theta_1, \theta'_2)\right]$$

should be satisfied on all existential configurations with common state component $s_c \in S_c$, reading ζ and θ_1 on input and common work tape, respectively, and

$$\text{if } ((s'_c, s'_p, 0), \theta'_1, \theta'_2, d_{ih}, d_{wh_1}, d_{wh_2}) \in \delta((s_c, s_p, 1), \zeta, \theta_1, \theta_2),$$

$$\text{then } s'_p = s_p,\ d_{wh_2} = \downarrow,\ \text{and } \theta'_2 = \theta_2$$

should be satisfied to make sure that existential moves do not change the content that is private to the universal player.

When the transitions from universal states of a PATM are forbidden to change the common memory elements, we get a *blind alternating Turing machine* (BATM). The equivalence of these two models to our setup with deterministic verifiers reading partial and zero-information debates, respectively, is demonstrated in Lemmas 3, 4, and 5.

Lemma 3. *For any space bound $s(n)$,*

$$\mathsf{PASPACE}(s(n)) \subseteq \mathsf{PDEB}(s(n), \infty, 0),$$

and

$$\mathsf{BASPACE}(s(n)) \subseteq \mathsf{ZDEB}(s(n), \infty, 0).$$

Proof. Let $M = (S_c, S_p, Q, \Sigma, \Gamma, \delta_M, q_0, q_a, q_r)$ be a private alternating Turing machine, where $Q \subseteq S_c \times S_p \times \{0,1\}$, and S_p and S_c are the private and common portions of the states, respectively. Similar to what we did in Lemma 1, we assume that M's initial and halting states are existential, and its computation alternates between existential and universal states. In the construction of Lemma 1, a message from P1 and P0 determined the next transition of the ATM simulated by the verifier at any point in the debate. We used the bijection b to define a one-to-one correspondence between the codomain of the transition function of the ATM and the alphabets of P1 and P0, so that they could inform the verifier about their decision on which transition, among the permitted ones, is to be made at each step. We will use the same idea again.

In the verifier we will build here, Γ_1, Γ_0, and Δ will denote the public alphabets of P1 and P0, and the private alphabet of P0, respectively, such that $\natural \in \Gamma_1$. A bijection b_1 is defined between $S_c \times \Gamma \times \Diamond^2$ and $\Gamma_1 - \{\natural\}$. On any existential configuration, P1 will tell the verifier V the transition to be made using the appropriate symbol from its alphabet. One complication with universal moves is that those may change both the public content (e.g. the public portion of the state) and the content private to universal player (e.g. the private work tape) at the same time. While the change on the latter should be kept as a secret between P0 and V, change on the public content should immediately be available to P1, who impersonates the existential player of M, so that it can use this information while making the future choice of moves as the existential player would do. To handle this, we let a pair of symbols $\beta\gamma$, where $\beta \in \Gamma_0$ and $\gamma \in \Delta$, to completely describe a universal move. We will need two bijections for P0: $b_0^c : S_c \times \Gamma \times \Diamond^2 \longrightarrow \Gamma_0$, which stands for the changes made on the common parts of the machine during a transition, and $b_0^p : S_p \times \Gamma \times \Diamond \longrightarrow \Delta$, for the changes made on the private portion of the state and the private work tape. On some universal state q, if $((s_c, s_p, 1), \theta'_1, \theta'_2, d_{ih}, d_{wh_1}, d_{wh_2}) \in \delta_M(q, \zeta, \theta_1, \theta_2)$ is the transition chosen by P0, $\beta\gamma$ will be the message of P0 for this particular transition, where $b_0^c(s_c, \theta'_1, d_{ih}, d_{wh_1}) = \beta \in \Gamma_0$, and $b_0^p(s_p, \theta'_2, d_{wh_2}) = \gamma \in \Delta$.

The debate format described in Section 2.1.2 requires P1 and P0 to build the debate using one symbol at a time alternately, however, it takes two symbols for P0 to describe a universal move in the construction here. In order to fit the format, V expects each turn in the debate to be communicated as follows: First, P1 emits the symbol for its choice of the next existential move. Then, it is P0's turn in the debate to designate the symbol corresponding to the change made in the common portion by its choice of move. P1 puts the dummy symbol $\natural$ next. Finally, P0 gives the symbol for changes made in private portion to complete the move. V can check if the debate is in this format easily. If a player violates the format, V declares the opponent as the winner.

Using the infrastructure described above, P1 and P0 will write their choice of transitions on the debate. V simulates M by taking the exact steps described in the debate. Since its computation is available to neither P0 nor P1 during the preparation of the debate, V does not need to worry about the confidentiality of the private work tape and private state portions during simulation. The space usage on V's single work tape equals the sum of the usages on M's two work tapes.

The construction for blind alternating Turing machines is very similar. Since common elements of a BATM are not altered during the transitions from universal states, the public alphabet Γ_0 of P0 is the empty set, and we do not need b_0^c. This is the only change needed to construct deterministic verifiers that check zero-information debates from BATMs. □

Since we do not know a method which would handle the simulation in Lemma 3 of a PATM with two work tapes by a deterministic verifier with one work tape within the same time bounds, we handle the time-bounded PATM simulation separately as follows.

Lemma 4. *For any time bound $t(n)$,*

$$\mathsf{PATIME}(t(n)) \subseteq \mathsf{PDEB}(\infty, t(n), 0),$$

and

$$\mathsf{BATIME}(t(n)) \subseteq \mathsf{ZDEB}(\infty, t(n)^2, 0).$$

Proof. We first simulate a $t(n)$-time PATM P with two work tapes by an $O(t(n))$-time ATM M with three work tapes.[7] This simulation is very similar to the one in [34] used to show that the power of time-bounded PATMs is the same with that of ATMs running under the same time bounds. In the first phase of the simulation, M writes down all the existential and public universal moves that the P would make on this input via existential and universal branchings, respectively. Whenever M needs to write a private universal move of P, it writes a symbol meaning "pass" for now. M makes no existential branching after the completion of the first phase. In the second phase, M writes down its choices of moves for each private move indicated as pass in the first phase via universal branchings. All these moves are written in the first work tape of M. Finally, M uses its second and third work tapes to simulate P by taking the exact steps specified on the first tape. The first and second phases and the simulation each take $t(n)$ time. Therefore, the three-tape ATM M simulates the P in $O(t(n))$ steps.

It is known that a $t(n)$ time multitape ATM can be simulated by a one-tape ATM in $O(t(n))$ steps [33]. Therefore, we may as well say that the PATM above can be simulated by an $O(t(n))$ time one tape ATM. Then, by Lemma 1, the simulation of this one tape ATM with a deterministic verifier checking partial-information debates can be done in $O(t(n))$ time.

One might think that the construction above can be made to work for the zero-information case as well, by first converting a BATM to a multitape ATM that makes all of its existential moves before the universal ones, then transforming that alternation-bounded multitape ATM to an equivalent one-tape ATM, and, finally, simulating the resulting one-tape ATM by a zero-information debate checker. However, the simulation of multitape ATMs by one-tape ATMs in [33] does not preserve the alternation bounds, and we do not know a method to simulate a general one-tape ATM that can alternate multiple times between universal and existential moves by a zero-information debate checker under the same time bounds.

Instead, we simply construct a deterministic verifier which stores the contents of the two work tapes of the given $t(n)$-time BATM in its single work tape. The simulation proceeds exactly as in the proof of Lemma 3, and the verifier's time complexity is $t(n)^2$, due to the well-known [39] quadratic slowdown involved in such constructions. □

Lemma 5. *For any time bound $t(n)$ and space bound $s(n)$,*

$$\begin{aligned}
&\mathsf{PDEB}(s(n),\infty,0) \subseteq \mathsf{PASPACE}(s(n)),\\
&\mathsf{PDEB}(\infty,t(n),0) \subseteq \mathsf{PATIME}(t(n)),\\
&\mathsf{ZDEB}(s(n),\infty,0) \subseteq \mathsf{BASPACE}(s(n)), \text{ and}\\
&\mathsf{ZDEB}(\infty,t(n),0) \subseteq \mathsf{BATIME}(t(n)).
\end{aligned}$$

Proof. We build a PATM (resp. BATM) M that simulates the given verifier V checking partial-information (resp. zero-information) debates by using universal and existential moves to select P0 and P1 symbols in the debate, respectively, to feed to V. Since both V's internal configuration during simulation and the universal moves that are supposed to produce the private messages of P0 can reveal information undesirably to the existential player who produces P1's messages, both V's simulation and the production of P0's symbols should be performed on the private work tape. If the state information of V is kept in private portion of M's states, the simulation can be done within time and space bounds of V. This is all that is needed for the construction of a BATM for zero information debates. For constructing a PATM for general partial-information debates, we also see to it that if a universal move produces a symbol from the public alphabet Γ_0 of P0, M lets the existential player know about this by writing a copy of that symbol in the common memory area (i.e. the common state component). □

[7]We defined ATMs with a single work tape in Section 2.2.1. Versions with multiple work tapes can be defined in the usual manner.

Lemmas 3 and 5, and the facts proven in [35] about the power of space-bounded PATMs and BATMs let us conclude

Theorem 5. *For any $s(n) \geq \log n$,*

$$\mathsf{ZDEB}(s(n), \infty, 0) = \mathsf{BASPACE}(s(n)) = \mathsf{DSPACE}(2^{O(s(n))}).$$

Theorem 6. *For any $s(n) \geq \log n$,*

$$\mathsf{PDEB}(s(n), \infty, 0) = \mathsf{PASPACE}(s(n)) = \mathsf{DTIME}(2^{2^{O(s(n))}}).$$

The following theorems show that the private information brings no significant language recognition power to time-bounded alternation. They are derived from Lemmas 4 and 5, and [36].

Theorem 7. *For any $t(n) \geq n$,*

$$\mathsf{ZDEB}(\infty, t(n), 0) = \Sigma_2\mathsf{TIME}(t(n)) \subseteq \mathsf{BATIME}(t(n)) \subseteq \mathsf{ZDEB}(\infty, t(n)^2, 0) = \Sigma_2\mathsf{TIME}(t(n)^2).$$

Theorem 8. *For any $t(n) \geq n$,*

$$\mathsf{NSPACE}(t(n)) \subseteq \mathsf{PATIME}(t(n)^2) = \mathsf{PDEB}(\infty, t(n)^2, 0),$$

and

$$\mathsf{PDEB}(\infty, t(n), 0) = \mathsf{PATIME}(t(n)) \subseteq \mathsf{DSPACE}(t(n)).$$

We have seen that giving the universal player private resources makes the new model recognize a wider class of languages than ATMs for space-bounded settings. When considering space bounds smaller than $\log n$, even the ownership of the input head turns out to be an important issue. Peterson and Reif [34] have shown that such private alternating machines recognize only the regular languages when the input head is defined as common resource, and restricted to move only from the left to the right. Since the position of a verifier's input head may encompass private information that was not available to P1 during the construction of the debate, debate checking has a stronger correspondence to a PATM definition where the input head is a private resource of the universal player. Hence, we set the input head as a private resource of the universal player in private and blind alternating machines with sublogarithmic space bounds. Constant-space versions of such private and blind alternating machines have also been studied:

Theorem 9 ([34])**.** *$\mathsf{ZDEB}(cons, \infty, 0) = \mathsf{NSPACE}(n)$.*

Theorem 10 ([34])**.** *$\mathsf{PDEB}(cons, \infty, 0) = \mathsf{E}$.*

We will use the concepts of blind and private alternating multihead finite automata (2bafa(k)s and 2pafa(k)s, respectively, for k input heads), defined analogously to alternating multihead finite automata, in subsequent sections. These have common and private portions of their states, as in the definition of BATMs and PATMs. As explained above, defining all the input heads as resources private to universal player is the appropriate choice. Universal moves are not allowed to change the common state portion in a 2bafa(k), and existential moves are prevented from depending on and changing the private state portions and the positions of the input heads in both models. We name the k-head classes associated with these machines $\mathsf{2BAFA}(k)$ and $\mathsf{2PAFA}(k)$, respectively. It can be shown, again with the technique of [27], that blind and private alternating multihead finite automata are equivalent to logspace blind and private alternating Turing machines, respectively. However, we will show an alternative approach in Section 3.1.2, to which we defer the proof of the following two theorems.

Theorem 11. *$\cup_{k\geq 1}\mathsf{2BAFA}(k) = \mathsf{PSPACE}$.*

Theorem 12. *$\cup_{k\geq 1}\mathsf{2PAFA}(k) = \mathsf{EXPTIME}$.*

2.2.3 Interactive Proof Systems

An *interactive proof system* (IPS) consists of a verifier and a prover exchanging messages with each other. The verifier is again a probabilistic Turing machine, and the prover is a (possibly uncomputable) function from the input and the messaging history to the next message. The verifier arrives at a decision about the membership of the input word upon interacting with the prover in this fashion. A verifier-prover pair is an IPS for a language L if, for any input w, the prover makes the verifier accept w with high probability if $w \in L$, and no such prover exists otherwise.

Interactive proof systems have led to many theoretical insights and practical results. For example, IPSs and one of their variations, probabilistically checkable proofs, are extensively used for proving the hardness of finding approximate solutions to many practical problems [3, 10, 16]. IPS concepts have also been shown to be essential in cryptography [15, 25].

A *one-way interactive proof system* (oneway-IPS)[8] is an IPS where the verifier does not send any message, so the prover can be assumed to be a function mapping the input string to a certificate about its membership. A oneway-IPS is a special case of debate checking where P0 does not participate in the debate (or, equivalently, where V ignores P0's messages). Therefore, the class of languages recognized by oneway-IPSs with some resource bounds constitutes a lower bound for the language recognition power of the verifiers checking debates with the same bounds.

We now give an exact characterization for the class of languages that have partial information debates checkable by logspace polynomial-time deterministic verifiers. Our proof makes use of a number of results from the IPS literature.

Theorem 13. $\mathsf{PDEB}(log, poly, 0) = \mathsf{PSPACE}$.

Proof. One direction of the equality is evident from Theorem 8. Let us show that $\mathsf{PSPACE} \subseteq \mathsf{PDEB}(log, poly, 0)$.

An *Arthur-Merlin* (AM) *game* [4] is an IPS variant in which the coins of the verifier (Arthur) are visible to the prover (Merlin) during computation. AM games with polynomial-time bounded verifiers exist for all languages in PSPACE [26, 38]. Furthermore, these verifiers can be assumed to have perfect completeness, that is, a guarantee that members of the language are accepted with probability 1 [23]. Condon [9] showed that an AM game with a polynomial-time verifier can be simulated by an IPS with a logspace polynomial-time verifier. In the construction in her proof, the IPS simulating an AM game rejects the inputs that are in the language under consideration only when the simulated AM rejects them. If Condon's technique is used to simulate an AM game with perfect completeness, one obtains an IPS with perfect completeness whose verifier uses logarithmic space and polynomial time.

For any language $L \in \mathsf{PSPACE}$, we can convert the probabilistic logspace polynomial-time interactive verifier V_i of the IPS that handles L with perfect completeness, which must exist as we argued above, to a deterministic silent verifier V_d that checks partial-information debates for membership in L within the same space and time bounds as follows: In the debate checked by V_d, P0 is expected to demonstrate that the input string is rejected by V_i for a particular combination of coin tosses of V_i. P1 is expected to show that, no matter what V_i says, the prover that talks to it can always lead it to acceptance. V_d uses bits sent to it by P0 as private messages whenever it needs to simulate a coin toss of V_i. At every point where V_i would send a symbol to its prover, P0 is supposed to emit that symbol publicly, and V_d accepts the input whenever P0 fails to do this. P1 plays the role of the prover in the IPS, trying to convince the verifier to accept by messages based on the input and the public messages of P0. Whenever the simulation of V_i reaches a halting state, V_d halts with the same decision. □

[8]See, for instance, [37] for a review of the power of oneway-IPSs under various resource bounds.

2.2.4 Multi-Prover Models with Interacting Verifiers

Ben-Or et al. [6] introduced *multi-prover interactive proof systems*, in which a verifier exchanges messages with multiple provers who cannot communicate with each other (i.e. cannot see the messages exchanged between the verifier and the other provers), and share a common goal to make the verifier accept the input. Later, Babai et al. [5] proved that the power of these systems equals NEXPTIME when the verifier is polynomial-time bounded.

Motivated by this work, Feige et al. [17] studied what they call *the noisy oracle problem*, which is modelled essentially as a multi-prover interactive proof system in which some of the provers are refuters, who claim that the input is not in the language and try to make the verifier reject, as in our debate checking model. The differences between the model of [17] with only two competing provers and our debate checking model are that the verifier in their model engages in two-way conversations with P1 and P0, who are not allowed to see the messages exchanged between the verifier and the other prover. Furthermore, one of the provers in their model (the one who is telling the truth) is allowed to toss coins. The verifier in their model is allowed to fail to halt with high probability for nonmembers of the language under consideration. Feige and Shamir [19] further investigate the power of the model in [17] under severe space bounds. Feige and Kilian studied "refereed games," [18] which is basically a variant of the model in [17], in which both provers are allowed to toss coins.

In this work, we refer to a version of the interactive models in [17–19] as a *competing-prover interactive proof system* (CIPS). A CIPS is exactly the same model studied in [17] and [19], except that the provers in a CIPS adopt deterministic (i.e. pure) strategies (e.g. cannot toss coins). We note that a CIPS is a broader system that captures both the power of the model in [18] and the power of debate checking for any level of information hiding. Although, in a CIPS, the provers cannot see any message exchanged between the verifier and the other prover, the verifier can reveal a subset of the messages of some prover to the other one using its capability to talk to both provers. For example, if a CIPS is needed to simulate partial-information debate checking, the verifier of the CIPS should be designed so that every message of P1 will be revealed to P0 before the next message of P0, and only the messages written in a pre-agreed subset of the communication alphabet between P0 and the verifier will be shown to P1. In order to simulate the model in [18], where both provers may play randomly, the verifier of a CIPS only needs to give random bits to the provers whenever they request one. Hence, the following result proven in [18] applies for CIPSs as well:

Theorem 14. *Every language in* EXPTIME *has a CIPS with polynomial-time verifier.*

Most of the results in [17] and [19] also hold for CIPSs (i.e. they make no use of the truthful prover's ability to toss coins). We will find it interesting to compare our debate checking model with the CIPS model using the following facts proven in [19] regarding the CIPS model with constant-space verifiers:

Theorem 15. *Every recursively enumerable language has a CIPS with a constant-space verifier.*

Theorem 16. *Every recursive language has a CIPS with a constant-space verifier which halts with probability 1 on every input.*

This huge power afforded to the verifier by the ability of speaking has led us to define our model with a silent verifier, to study the effects of that restriction.

3 The Complexity of Probabilistic Debate Checking

3.1 Verifiers with Small Space Bounds

This section deals with debates checkable by constant- and log-space probabilistic verifiers. In Section 3.1.1, we characterize the languages that have debates checkable by constant- and log-space verifiers that are allowed to use only constant amounts of randomness, for each degree of information. Our theorems stating these characterizations are

Theorem 17. $\mathsf{CDEB}(cons, \infty, cons) = \mathsf{CDEB}(log, poly, cons) = \mathsf{P}$.

Theorem 18. $\mathsf{ZDEB}(cons, \infty, cons) = \mathsf{ZDEB}(log, exp, cons) = \mathsf{PSPACE}$.

Theorem 19. $\mathsf{PDEB}(cons, \infty, cons) = \mathsf{PDEB}(log, exp, cons) = \mathsf{EXPTIME}$.

An immediate conclusion is that allowing constant-space verifiers to toss even some constant number of coins enlarges the class of languages having debates checkable by such verifiers. The increase is from REG to P for complete-information debates as can be seen in Theorems 3 and 17, and it is from $\mathsf{NSPACE}(n)$ to PSPACE (Theorems 9 and 18) and from E to $\mathsf{EXPTIME}$ (Theorems 10 and 19) for zero- and partial-information debates, respectively.

Section 3.1.2 describes protocols which show that languages in PSPACE and $\mathsf{EXPTIME}$ have zero- and partial-information debates checkable in constant space not only with some error probability, but with any desired nonzero error probability, if one allows more random bits to be used. The results are as follows.

Theorem 20. $\mathsf{PSPACE} \subseteq \mathsf{ZDEB}^{\forall\varepsilon}(cons, \infty, \infty)$.

Theorem 21. $\mathsf{EXPTIME} \subseteq \mathsf{PDEB}^{\forall\varepsilon}(cons, \infty, \infty)$.

3.1.1 Constant Randomness

Our results here are obtained by adapting a technique discovered by Say and Yakaryılmaz [37] for simulating nondeterministic logarithmic-space machines by verifiers that use only constant amounts of randomness and memory. Say and Yakaryılmaz [37] show that any language recognized by some multihead nondeterministic finite automaton also has a verifier with these strict resource bounds recognizing it upon reading a proof statement prepared by a prover. Their idea in constructing such a verifier can be summarized as follows: A list of k symbols that would be scanned by the k input heads of the multihead automaton will be given in the proof statement by the prover for every step of the automaton's computation, together with an indication of the nondeterministic choices that would eventually lead the automaton to acceptance. The verifier uses a constant number of coins to decide which input head of the simulated automaton to track, and then checks the validity of the claims in the proof about the symbols scanned by this specific input head throughout the computation.

We adapt this technique for debate checking by letting P1 play the role of the prover, and P0 make the universal choices, to simulate alternating multihead finite automata.

Lemma 6. $\mathsf{P} \subseteq \mathsf{CDEB}_w^{\forall\varepsilon}(cons, \infty, cons)$.

Proof. Let L be any language in P, and let M be the 2afa(k) recognizing L by Theorem 4. We will construct a constant-space verifier which tosses only a constant number of coins while checking complete-information debates about membership in L with a desired positive error probability ε according to the weak definition.

Without loss of generality, we make the following assumptions about M: The starting and halting states of M are existential. Each existential and universal state leads exactly to two different branches, and computation alternates between existential and universal states.

In the debate, P1 and P0 are supposed to be talking about the step-by-step execution of M on the input string w in a sequence of exchanges, each of which treats a pair of M-steps. Each exchange has the following structure:

- P1 announces its claim about which existential choice to be made by M at the present step guarantees reaching an accept state eventually. P1 also gives a list of k tape symbols of M, claiming that the ith head of M will be scanning the ith symbol in this list after taking the step just announced by P1, for all i, where $1 \leq i \leq k$.

- P0 announces the universal choice of M for the next step that it claims will lead to a rejection of w.

- P1 announces the list of k tape symbols which it claims that will be scanned by M's heads after the execution of the universal step according to the choice just specified by P0.

(Our definition in Section 2.1 stipulates that P1 and P0 should alternate after emitting each symbol. The protocol here can be made to fit that requirement, since the "packages" sent by P1 are of fixed length, and P0 can emit dummy symbols while listening to them. The verifier can measure the length of the packages using its constant-sized memory, and reject the input if it sees P1 giving a package of the wrong length.)

If P1 claims that the set of moves described up to that point in the debate will have caused M to halt with acceptance, it emits the special symbol $\circlearrowleft$ indicating this claim, and restarts the procedure by giving the first existential choice from the initial configuration of M on w.

We now describe the verifier V. V will read the debate to simulate the execution of M on the particular computation path indicated by the alternating messages of P1 and P0. It starts by initializing a counter to 1, and randomly selecting one of the k heads of M, using $r = \lceil \log k \rceil$ random bits. Each head has a probability of at least 2^{-r} to be selected. V uses its single input head to track the position of the selected head during its simulation of M. V needs to know what the other heads are scanning in order to compute M's next state at any step, and it depends on P1 for this information. After each simulated step, V checks if the symbol scanned by its input head is consistent with what P1 claims about the corresponding head of M. If it sees that P1 is lying in this regard, V rejects the input. Otherwise, it continues by reading the next exchange. If V sees M reaching a reject state during the simulation, it rejects. If P1's argument in the debate is a $\circlearrowleft$, V checks if its simulation of M has indeed reached an accept state. If V sees that M has not accepted despite P1 announcing that it has, it rejects. If M is seen to have accepted, V increments the counter. If the counter has exceeded c, whose value will be discussed below, V accepts. Otherwise, V moves its head to its original position, randomly picks a head of M to track, and starts processing the restarted simulation in the debate.

If $w \in L$, P1 can always find a way of responding to P0 that will lead M to acceptance, while giving correct information about the symbols scanned by the heads, so V will accept with probability 1. If $w \notin L$, P1 must lie about at least one head at some point to prevent V from rejecting. This lie will be caught, and V will therefore reject, with probability at least 2^{-r}, in every simulated computation of M. The probability that V will fail to pick the head that P1 is lying about in all c iterations of the loop is $(1 - 2^{-r})^c$, which can be tuned to be below ε by choosing a sufficiently large c.

An evil P1 can cause V to spend infinite time without ever restarting the computation of M with probability at most $(1 - 2^{-r})$. We conclude that V is a weak debate checker for L with the desired properties. □

Next, we show how to transform the verifier in Lemma 6 into a strong debate checker at the expense of the ability to tune it for any desired error probability.

Lemma 7. *$\mathsf{P} \subseteq \mathsf{CDEB}(cons, \infty, cons)$.*

Proof. Consider the verifier V from the proof of Lemma 6 with the number of iterations c set to 1. Recall that this V uses $r = \lceil \log k \rceil$ random bits to simulate one run of the 2afa(k) that recognizes a language L in P. V accepts input strings in the language with probability 1, and rejects strings not in the language with probability 2^{-r}.

To increase the rejection probability for nonmembers so that the resulting machine fits the definition in Section 2.1, we construct a new verifier V', which starts by using an additional $r + 1$ random bits to reject the input directly with probability $\frac{2^r-1}{2^{r+1}}$, and transfers control to V with the remaining probability. V' can be seen to accept inputs in L with probability at least $1 - \left(\frac{2^r-1}{2^{r+1}}\right) = \frac{2^r+1}{2^{r+1}}$, and rejects the inputs not in L with probability $2^{-r}\left(\frac{2^r+1}{2^{r+1}}\right) + \frac{2^r-1}{2^{r+1}} = \frac{2^{2r}+1}{2^{2r+1}}$. Thus, V' constitutes a debate checker for L according to the strong definition with error bound $\frac{2^{2r}-1}{2^{2r+1}}$. □

We now derive some time bounds that will be useful in our characterization theorems.

Lemma 8. *$\mathsf{CDEB}_w(log, \infty, cons) \subseteq \mathsf{CDEB}(log, poly, cons)$,*
$\mathsf{ZDEB}_w(log, \infty, cons) \subseteq \mathsf{ZDEB}(log, exp, cons)$, and
$\mathsf{PDEB}_w(log, \infty, cons) \subseteq \mathsf{PDEB}(log, exp, cons)$.

Proof. We start by showing that a complete-information debate checkable by a verifier using logarithmic space and a constant amount of random bits does not need to be longer than some polynomially bounded number of symbols.

Let V be the logspace verifier for a language $L \in \mathsf{CDEB}_w(log, \infty, cons)$ that uses at most r random bits for some constant r. V can be derandomized as a collection of 2^r deterministic logspace verifiers S=$\{V_1, V_2, \ldots, V_{2^r}\}$ for each different assignment to the random sequence of length r, such that, for inputs in L, the common debate that the verifiers in S are reading eventually makes more than half of them accept. Each V_i can be in one of polynomially many different possible configurations. Let E be the set of all possible combined states of the "ensemble" consisting of these verifiers. The cardinality C of E is itself polynomially bounded.

For any string $w \in L$, we know that there is a debate subtree, i.e. a "best strategy" for P1, where P1 "wins" in every branch by steering V toward an ensemble with a majority of accepting configurations. If a debate corresponding to a particular branch of that tree has been going on for more than C turns without reaching such an "accepting" ensemble, this means that P1 does not have a sequence of clever responses to prevent P0 from causing a loop of ensemble states, and so this debate will never end up with an accepting ensemble. This contradiction leads us to conclude that no complete-information debate that is checkable by a logspace constant-randomness verifier needs to have superpolynomial length. We construct a new verifier that simulates V, while using its logarithmic memory to clock this simulation, cutting off and rejecting when either V is seen to enter an infinite loop without reading debate symbols, or when the debate goes on for too long.

For zero- and partial-information debates, we modify the argument above to take the increased ignorance of P1 about the state of V into account. Since P1 does not see some messages of P0 in the debate, it does not know precisely what ensemble state V will be in at any point while reading the debate. From P1's point of view, V can be in this or that ensemble state, having received this or that message from P0, so P1's strategy has to be based on viewing V as being in a *set* of possible ensemble states that are consistent with what little P1 knows about P0's messages up to that point, and finding an argument that would lead all those ensembles to acceptance. We now see that any zero- or partial-information debate longer than the cardinality of the power set of E must involve a repetition of a set of ensembles, which means a failure for P1. By the same reasoning as above, no zero- or partial-information debate that is checkable by a logspace constant-randomness verifier needs to have superexponential length.

Blind and private alternating Turing machines can count up to $2^{2^{s(n)}}$ using $s(n)$ space for all $s(n) \geq \log n$ (see Theorem 1 in [34]). We can integrate this counting mechanism to any logspace constant-randomness verifier to reject the input if the zero- or partial-information debate that it has been reading has exceeded the exponential time bound derived above. $\square$

Since the containment $\mathsf{CDEB}(cons, \infty, cons) \subseteq \mathsf{CDEB}(log, \infty, cons)$ is trivial, and Lemma 8 proved that $\mathsf{CDEB}(log, \infty, cons) \subseteq \mathsf{CDEB}(log, poly, cons)$, the next lemma leads us to our first characterization in Theorem 17.

Lemma 9. $\mathsf{CDEB}(log, poly, cons) \subseteq \mathsf{P}$.

Proof. Let L be a language in $\mathsf{CDEB}(log, poly, cons)$. Let V be the polynomial-time, logspace verifier that checks complete-information debates about L with bounded error using r random bits. We construct an alternating logspace Turing machine M that recognizes L.

As in the proof of Lemma 8, we think of V as a collection of 2^r deterministic logspace verifiers S=$\{V_1, V_2, \ldots, V_{2^r}\}$. M simulates the elements of S in a time-sharing fashion. It first advances each V_i until it reaches its first reading configuration, or halts, or is determined to have run for so long without doing any of these so that it is in an infinite loop. When all the V_i that have not halted and are not looping yet are ready to read a symbol from the debate, M branches existentially to produce a symbol from Γ_1, and feeds this symbol to all those V_is. Again, it advances each V_i until all surviving ones reach the next reading configuration. M then branches universally to produce a symbol from Γ_0 to feed to the verifiers waiting to read a symbol from P0 in the debate. M keeps simulating each V_i in this fashion by feeding them alternately produced symbols as the next pair of messages of the debate. M accepts if more than half of the deterministic verifiers in the set S has accepted at the end of the simulation. Otherwise, M rejects the input string. $\square$

We have proven

Theorem 17. $\mathsf{CDEB}(cons, \infty, cons) = \mathsf{CDEB}(log, poly, cons) = \mathsf{P}$.

In the rest of this section, we modify the arguments above to show similar results for zero- and partial-information debates checkable by constant- and logarithmic-space verifiers using constant amounts of randomness.

Lemma 10. $\mathsf{PSPACE} \subseteq \mathsf{ZDEB}(cons, \infty, cons)$.

Proof. Let L be any language in PSPACE, and let M be the 2bafa(k) recognizing L by Theorem 11. We will first modify the construction in the proof of Lemma 6 to build a verifier V which rejects strings not in L with probability 1, and accepts the members of L with probability at least $2^{-\lceil \log k \rceil}$. We will then convert V to a debate checker that fits the strong definition, in a manner similar to what we did in Lemma 7.

V initially picks one of M's heads randomly, and simulates M according to existential and universal choices suggested by P1 and P0, respectively. In Lemma 6, P1 was the player who is supposed to send the symbols scanned by the heads of the multihead automaton that is being simulated. This is not acceptable in this case, since the configuration of M during its execution depends on the choices of universal moves made by P0, and P1 in the zero-information protocol that we are designing cannot have that information. P0, on the other hand, can see the full configuration of M, just as the universal player of a 2bafa(k), and it is therefore P0 who provides the symbols scanned by the heads of M. V accepts the input if it reaches an accept state of M, or detects an inconsistency between its own input and the claim made for the head that it is tracking by P0. Otherwise, V rejects.

Therefore, V accepts inputs in L with probability at least 2^{-r}, using $r = \lceil \log k \rceil$ random bits. It rejects inputs not in L with probability 1. We construct a new verifier V' that accepts directly with probability $\frac{2^r-1}{2^r+1}$, and transfers control to V with the remaining probability. Then, V' accepts inputs in L with probability at least $\frac{2^{2r}+1}{2^{2r+1}}$, and rejects the inputs not in L with probability $\frac{2^r+1}{2^{r+1}}$. Hence, V' checks zero-information debates for L according to strong definition with error bound $\frac{2^{2r}-1}{2^{2r+1}}$. □

Lemma 11. $\mathsf{ZDEB}(log, exp, cons) \subseteq \mathsf{PSPACE}$.

Proof. Let L be any language in $\mathsf{ZDEB}(log, exp, cons)$, and let V be the logspace exponential-time constant-randomness verifier of zero-information debates on L. We build a logspace blind alternating Turing machine M recognizing L. The construction is almost identical to that of the proof of Lemma 9. Since the debates read by V are of zero-information, M simulates the 2^r deterministic verifiers (the V_is) on its private tape, feeding them symbols from the private alphabet Δ, ensuring that the existential moves obey the zero-information condition. Logarithmic space is sufficient for this simulation. As in Lemma 9, M accepts if and only if more than half of the V_is accept, which happens if and only if the input is in L. □

Lemmas 10 and 11 form

Theorem 18. $\mathsf{ZDEB}(cons, \infty, cons) = \mathsf{ZDEB}(log, exp, cons) = \mathsf{PSPACE}$.

We conclude this section with a characterization for partial-information debates.

Theorem 19. $\mathsf{PDEB}(cons, \infty, cons) = \mathsf{PDEB}(log, exp, cons) = \mathsf{EXPTIME}$.

Proof. We will describe the necessary modifications to the proofs of Lemma 10 and 11.

First, we can design partial-information debates with constant-space verifiers allowed to use only a constant number of random bits to simulate the 2pafa(k) of any given language in $\mathsf{EXPTIME}$, thereby showing $\mathsf{EXPTIME} \subseteq \mathsf{PDEB}(cons, \infty, cons)$. The only difference with the construction in Lemma 10 is that P0 is allowed to use the public alphabet Γ_0 as well as the private alphabet Δ for suggesting the public and private universal moves of the simulated 2pafa(k).

Second, simulation of logspace exponential-time constant-randomness verifiers reading partial-information debates by logspace private alternating Turing machines implies $\mathsf{PDEB}(log, exp, cons) \subseteq \mathsf{EXPTIME}$. The simulation is similar to the one in Lemma 11, but the private alternating TM now produces symbols by universal branching from the set $\Gamma_0 \cup \Delta$ when it needs the next message of P0 in the debate. If the produced symbol is from Γ_0, the PATM lets the existential player know this symbol by writing it in a special memory cell on the common work tape. The task of simulating the finitely many exponential-time deterministic verifiers is completed in exponential time. □

3.1.2 Building Constant-Space Verifiers for all Error Bounds

In Section 3.1.1, we showed that complete-, zero- and partial-information debates checkable by constant-space verifiers exist for P, PSPACE and $\mathsf{EXPTIME}$, respectively, if one adopts the weak definition of debate checking (e.g. Lemma 6), or is willing to accept rather large error probabilities (e.g. Lemma 7). Note that the standard technique of error reduction by repetition does not work for the debates of Section 3.1.1, since we cannot avoid a significant probability of nonhalting in the basic protocols there. In this section, we show that PSPACE and $\mathsf{EXPTIME}$ also have zero- and partial-information debates, respectively, checkable according to the strong definition by constant-space verifiers for any desired error probability. However, we note that the verifiers in this section use significantly more randomness than the ones in Section 3.1.1.

Theorem 20. *$\mathsf{PSPACE} \subseteq \mathsf{ZDEB}^{\forall\varepsilon}(cons, \infty, \infty)$.*

Proof. Let L be a language recognized by an $s(n)$-space deterministic TM M with a single work tape and separate read-only input tape, for some polynomial $s(n)$. As in [14] (Theorem 3.12, p. 817), we assume without loss of generality that M keeps a counter in a separate track of its work tape, and that this counter is incremented after each action performed on the first track, even for configurations that are not reachable from the initial configuration. M halts when this counter reaches $2^{cs(n)}$, where c is a constant, selected so that $2^{cs(n)}$ is greater than the runtime of any halting computation of M on inputs of length n. Let Q_M denote the state set of M.

We will describe a debate system with verifier V for L. Prover P1 attempts to convince V that the input string w is in L by repeatedly presenting what it claims to be an accepting computation path (ACP) of M on w. An ACP is a sequence of configurations of M on w that starts with the initial configuration, ends with an accepting configuration, and has the property that every pair of adjacent configurations in it obeys M's deterministic transition relation. Each configuration in the ACP is preceded by the symbol \$, which we assume is not a member of M's work tape alphabet or state set. Configurations are strings of the form uq_dx, representing the work tape and current state of M, where u and x are strings over the work tape alphabet of M, and q_d is a *state/direction symbol* whose position is used for representing the work tape head position,. The value of q_d in the ith configuration in an ACP indicates both the state of M in that configuration, and the direction in $\Diamond$ traveled by the input head of M during the transition from $(i-1)$th configuration, for all $i > 1$. The initial configuration consists of the single state/direction symbol corresponding to the start state of M, and the head direction $\downarrow \in \Diamond$. The input head position is not represented, since V will use its own input head to track this head. P1 is supposed to write the separator symbol # between every two adjacent ACP in its messages.

P0's task is to show that something is wrong with the computation paths sent by P1. Since V can check that a purported ACP "begins right" (i.e. with the initial configuration), and "ends right" (i.e. contains the accept state symbol in its last configuration) on its own, P0 attempts to convince V that P1 is making a transition error between two configurations, or that P1 is trying to make V run forever by reading an endless "configuration." In the following, these two types of counterargument are called the *transition error* and *infinity* (or ∞) claims, respectively. Of course, V should also be wary of the possibility that P0 may be trying to trick it to falsely reject a valid transition, or to cause it to enter a loop.

P0's private alphabet Δ is $\{0, \varsigma, \tau, \upsilon, \infty\}$. P0 writes 0 to match each symbol written by P1 until it comes to a point where it wishes to indicate an error in P1's sequence, as follows.

If P0 wishes to claim that P1 is making a transition error between two adjacent configurations, say, α and β, it writes ς in the debate right after \$ message of P1 preceding α, and then indicates

$$P1 : ...\$\alpha_1\alpha_2...\alpha_j...\$\beta_1\beta_2...\beta_k...\$...$$
$$P0 : ...\varsigma\ 0\ 0\ \ ...\tau\ \ ...0\,0\ 0\ \ ...\upsilon\ ...0...$$

$$\mathit{Debate} : ...\$\varsigma\alpha_1 0\alpha_2 0...\alpha_j\tau...\$0\beta_1 0\beta_2 0...\beta_k\upsilon...\$0...$$

Figure 1: P0 claiming a transition error between configurations α and β

the positions within these configurations whose examination would lead to the discovery of the error by the symbols τ and υ, emitting 0's for the other positions.

For example, if P0 is claiming that the invalidity of the transition from α to β can be detected by comparing the j^{th}, $(j+1)^{st}$ and $(j+2)^{nd}$ symbols of α with the k^{th}, $(k+1)^{st}$ and $(k+2)^{nd}$ symbols of β, P1 and P0's individual messages and the debate will look as in Figure 1, where α_i and β_i stand for the i^{th} characters of configurations α and β, respectively. Of course, if P0 is honest, $j = k$, and dishonest P0's may well trick careless verifiers to reject the input by giving these signals for some unequal j and k values.

At this point, some readers may be worrying about how P0 "knows" that P1 will make a transition error at that particular pair of configurations even before P1 has written the first character of the first configuration in the pair. This is not a problem, since our definition of language recognition by debate systems just requires such a debate, where P0 points out the errors in ACPs, to exist in all well-formed debate subtrees, in case of a cheating P1. Since P1 does not see any messages of P0 in zero-information setting, the messages of P1 should be exactly the same across all the debates in a well-formed debate subtree which, of course, contains a particular debate where P0 points out the erroneous locations. In case of a cheating P0, there exists a well-formed debate subtree in which the messages of P1 form the same sequence of correct ACPs, and no combination of P0 messages in this subtree points out an error in ACPs, as long as the verifier makes sure that the locations pointed by P0 corresponds to each other (i.e. $k = j$).

If P0 wishes to claim that P1 is trying to make V run forever by giving an infinite-length "configuration" η, it does not use the method above to indicate the transition error between the preceding configuration and η. Instead, P0 writes the symbol ∞ in response to the \$ preceding the infinite-length configuration.

We now describe the verifier V reading such a debate. Let us call each segment of the debate delimited by P1 printing #'s a *round*. V accepts immediately if it sees that P0 did not report an error of P1 (by either one of the two methods described above) in any single round,[9] or if it notices similar easily detectable "syntax errors" by P0. V rejects immediately if it sees P1 starting any round with an incorrect initial configuration, or if it detects P1 committing simple syntax errors such as writing a configuration without exactly one state/direction symbol.

V performs a partial check of every transition between adjacent configuration pairs presented by P1 in the debate by just focusing on the three-symbol "window" that has a state/direction symbol in its middle in each configuration. If these two windows are consistent with the transition function of M and the presently scanned input symbol, V moves its input head (which is tracing the movement of the input head of M) in the specified direction on the tape, and continues execution. Otherwise, it rejects.

If P0 is reporting a transition error, V tries to make sure that P0 is indeed pointing out to

[9]Note that the last configuration presented in such a round must be accepting, since we assume that P1 must be honest if P0 is violating the protocol.

corresponding positions in the two successive configurations. V compares these values by using a modified version of a technique by Freivalds [13, 22, 43]. Figure 1 may be helpful during the following exposition, which refers to two numbers l and r, whose values will be fixed according to the desired error bound, as will be explained later.

When it sees that P0 has responded by ς to the configuration delimiter \$ from P1, V starts flipping $4l$ coins on each of the first j symbols of configuration α. V flips l additional coins on the jth symbol corresponding to τ. These $4lj+l$ coins will be called *the first set of accept coins*. When scanning the relevant portion of the messages of P1, V stores the symbols α_j, α_{j+1}, and α_{j+2} in its memory for use in a possible comparison. For the next configuration β, V similarly flips $4lk+l$ coins that we name *the second set of accept coins*. Parallelly to the tossing of the accept coins, V flips another set of $2l(j+k)$ coins, which will be called *the control coins*.

After reading the υ, V decides whether to accept, control or continue:

- If one or both of the sets of accept coins contains all heads, V accepts.
- If both sets of accept coins contain at least one tail, and the set of control coins contains all heads, V proceeds to check if the two triples (α_j, α_{j+1}, α_{j+2}) and (β_k, β_{k+1}, and β_{k+2}) provide evidence for an invalid transition of M. If the two triples are among those that can appear when two configurations in a legitimate run of M are properly aligned, V accepts. Otherwise, it rejects.
- If the set of control coins, as well as both sets of accept coins, all contain at least one tail, V continues execution without comparing the triples, discarding all information it collected for this purpose.

If V arrives at the end of a round, it rejects if the final configuration is not accepting. Otherwise, it continues to the next round.

The only case where V gives up simulating the input head of M is when P0's message is ∞ in response to some \$ of P1, claiming that P1's messages will be an infinite sequence of non-\$ symbols from this point on.

- If $s(n) = O(n)$, V inspects this claim by checking if the next mn symbols emitted by P1 contains a \$ for some constant m such that $mn > s(n)$ for all input lengths $n > 0$. (The empty input can be handled separately.) This check can be performed by using V's input tape as a ruler. V accepts if it sees a \$ in this scan. Otherwise, it rejects.
- If $s(n)$ is superlinear, V enters a loop where each iteration involves reading n symbols from P1, and flipping rn coins. V accepts if it sees any \$ during this process. V rejects if all of the rn outcomes for some iteration turn out to be heads.

This concludes the description of V. We now show that this constitutes a debate system for L. Let us establish first that V halts with probability 1. When $w \in L$, in which case P1 will be repeating a valid ACP in an infinite loop, this is evident from the facts that V terminates directly if P0 does not claim any error, the procedure associated with the P0 signal ∞ halts with probability 1, and each processing of the transition error claim involves a small but nonzero probability of halting. When $w \notin L$, P1 could hope to avoid the ∞ signal by not giving an infinitely long "configuration", but by making a single transition error to an infinite loop of M's computation, thereby keeping the halting probability low. But this is not possible because of the way we defined M in the beginning: Since we equipped M with a robust counter that causes M to halt when it maxes out, any infinite sequence of configurations purporting to describe a run of M must contain not one, but infinitely many transition errors, which will of course be reported by P0, and whose combined processing will lead the halting probability to accumulate to 1.

We will now analyze the probabilistic procedures for the transition error and ∞ claims, and show that the probability that they lead to an incorrect decision is much lower than the probability of a correct decision. Consider Figure 1 again. Let A denote the event of acceptance as the result of the processing of a single transition error claim. Let T denote the event that the two triples from the two successive configurations are indeed tested for consistency during this procedure. According to the algorithm described above,

$$Pr[\mathrm{A}] = 2^{-4lj-l} + (1 - 2^{-4lj-l})2^{-4lk-l}, \tag{3}$$

and

$$Pr[\mathrm{T}] = (1 - Pr[\mathrm{A}])2^{-2l(j+k)}. \tag{4}$$

If $w \in L$, P0 will make sure that $j \neq k$ during transition error claims. In this case,

$$Pr[\mathrm{A}|j \neq k] > 2^{-l(4j+1)} + 2^{-l(4k+1)-1} > 2^{-l(4j+1)-1} + 2^{-l(4k+1)-1}.$$

Since $j \neq k$, the value of one of j and k is at most $\frac{j+k-1}{2}$. Without loss of generality, we assume j to have the smaller value, and we get $2^{-l(4j+1)-1} \geq 2^{-l\left(4\frac{j+k-1}{2}+1\right)-1} = 2^{-l(2j+2k-1)-1}$. Then,

$$Pr[\mathrm{A}|j \neq k] > 2^{-l(2j+2k-1)-1}.$$

The probability that V will be tricked to test the incorrectly aligned triples indicated by P0, thereby erroneously rejecting the input, is

$$Pr[\mathrm{T}|j \neq k] < 2^{-l(2j+2k)}.$$

Therefore,

$$Pr[\mathrm{A}|j \neq k] > 2^{l-1}Pr[\mathrm{T}|j \neq k],$$

that is, each incorrect claim about a transition error leads to an acceptance with probability at least 2^{l-1} times greater than the probability that it leads to a rejection.

If $w \notin L$, P1 should make transition errors to end its rounds with accepting configurations, and P0 raises the transition error claim with $j = k$. In this case, it is easy to see that V accepts with probability

$$Pr[\mathrm{A}|\ j = k] < 2^{-4lj-l+1},$$

and tests the transition among the two triples with probability

$$Pr[\mathrm{T} \mid j = k] > 2^{-4lj-1},$$

leading us to conclude

$$Pr[\mathrm{T}|j = k] > 2^{l-2}Pr[\mathrm{A}|\ j = k].$$

Therefore, each correct claim about a transition error leads to rejection with probability at least 2^{l-2} times greater than the probability of an incorrect acceptance.

As for the infinity claim, the only way V can reach an incorrect decision in this regard is the possibility of rejecting a proper ACP when $s(n)$ is superlinear. The control loop in the ∞-checking procedure described above can complete at most $\lfloor \frac{s(n)}{n} \rfloor$ iterations,[10] and if V is unlucky enough to

[10]Since P1 is truthful in this case, a \$ will be encountered after at most $s(n)$ P1 symbols are read.

obtain "all 0's" for a set of rn coin flips in any one of these iterations, it will reject incorrectly. The probability of this is

$$\sum_{i=1}^{\lfloor \frac{s(n)}{n} \rfloor} (1-2^{-rn})^{i-1}2^{-rn} < s(n)2^{-rn},$$

which is exponentially small, since $s(n)$ is polynomially bounded.

It is clear that any error bound $\varepsilon < \frac{1}{2}$ can be achieved by tuning the constants l and r in these procedures.

Since, in any round, the verifier moves on to the next configuration if both sets of accept coins and the set of control coins each contain at least one tail, an unlucky verifier may continue its run for a very long time without halting. Therefore, the verifier described above has no time bound in the sense that we defined in Section 2.1.3. However, as a side note, we analyze its expected time complexity, which is the expectation value of the run time over the random sequences used by the verifier. Since j and k are at most $s(n)$ in Equations 3 and 4, the probability of halting in any single processing of a transition error claim is $2^{-O(s(n))}$. There can be at most $2^{O(s(n))}$ configuration descriptions of length $O(s(n))$ between any two adjacent transition error claims. The ∞ check takes linear time for $s(n) = O(n)$, and $2^{O(rn)}$ time for superlinear $s(n)$. We conclude that V halts in expected time $2^{O(s(n))}$ in all cases. □

Note that the proof of Theorem 20 depends crucially on the inability of P1 to see P0's messages. If P1 could see P0, it could detect when P0 claimed a transition error is about to start, and present a correct transition at that point, branding P0 a liar.

In the next theorem, we use a protocol very similar to the one described above to further examine the power of partial-information debate checking by constant-space verifiers.

Theorem 21. $\mathsf{EXPTIME} \subseteq \mathsf{PDEB}^{\forall\varepsilon}(cons, \infty, \infty)$.

Proof. We modify the construction of Theorem 20 by setting up a debate about the computation of an alternating, rather than deterministic, Turing machine M with space-bound $s(n)$ for some polynomially bounded function $s(n)$. The result then follows from Theorem 1.

As in Lemma 6, P1 will try to convince the verifier of the membership of w in the language L of M by showing that it can always pick a sequence of existential moves for M that end up in an accept state, even if the universal moves are selected by an adversary. Without loss of generality assume that the initial and halting states of M are existential and its computation always alternates between existential and universal states. The protocol used in constructing the debate begins with P1 writing a sequence of symbols representing the first two configurations in an ACP of M, thereby making the existential choice in the first transition. P0 then announces its choice about the next (universal) move of M to P1 by using one of the prespecified symbols from the public communication alphabet Γ. P1 replies with the third and fourth configurations in the ACP, and the conversation goes on in this manner. All through this public exchange between P0 and P1 about the path for M to follow, P0 is privately writing its claims about the errors that P1 is committing, using a suitably adapted version of the method in the proof of Theorem 20.

Error bound analyses of the previous proof need no modification here. □

Now that the protocols above have been introduced, we are ready to present our proofs on the language recognition power of multihead blind and private alternating finite automata defined in Section 2.2.2.

Theorem 11. $\cup_{k\geq 1}\mathsf{2BAFA}(k) = \mathsf{PSPACE}$.

Theorem 12. $\cup_{k\geq 1}\mathsf{2PAFA}(k) = \mathsf{EXPTIME}$.

Proof. In order to see the forward containments $\cup_{k\geq 1}\mathsf{2BAFA}(k) \subseteq \mathsf{PSPACE}$ and $\cup_{k\geq 1}\mathsf{2PAFA}(k) \subseteq \mathsf{EXPTIME}$, one just needs to notice that the locations of any constant number of private input heads can be represented using $O(\log n)$ private memory. Therefore, $O(\log n)$-space BATMs and PATMs can simulate blind and private multihead alternating finite automata, respectively. We show that the reverse inclusions also hold.

Let L be a language in PSPACE, and M be the deterministic Turing machine recognizing it using not more than n^k space on any input with length $n > 0$, for some constant k. We give a 2bafa$(k+1)$ N recognizing L.

N looks for an accepting computation path (ACP) of M on the given input, in almost the same format as in the proof of Theorem 20. One additional feature of the ACP's here is that each configuration is assumed to be exactly n^k symbols long, padded with blank symbols if necessary.

N branches existentially to parallelly consider all possible strings, and uses universal branchings to drop the strings which do not satisfy some condition expected in an ACP from consideration. The input head of the simulated machine M is tracked by the $(k+1)$st head of N in the same way that we described in Theorem 20. N also branches universally for every new symbol considered by an existential split. One of these universal branches continues to the next symbol, whereas the other branch saves a copy of this and the next two symbols of the alleged ACP in production in private memory. These symbols will be compared with the corresponding triple of symbols in the next configuration of the alleged ACP. To find the corresponding position in the next configuration, N uses its first k input heads as a counter with capacity n^k. It increments this counter on each new symbol produced by existential branchings. When this counter maxes out (i.e. after exactly the n^kth new symbol), N compares the triple in its private memory with the triple of upcoming new symbols. This universal branch accepts if and only if these two triples are consistent with the transition function of M.

Since the second triple is located as a result of a deterministic counting process, N never mistakenly compares noncorresponding parts of two successive configurations, and the production of a single ACP is sufficient to determine if the input is in L.

This proof for 2bafa(k)s can be modified easily to simulate an n^k-space ATM by a 2pafa$(k+1)$ which changes the common state components by universal branchings to determine the universal moves of the simulated ATM before the production of every other configuration of the produced ACP. This completes the proof. □

The main protocol used in the debate in the proofs of Theorems 20 and 21 is that P0 points out the errors in the accepting computation paths of the simulated Turing machine that are presented by P1. Since the method adapted from [22] to compare the positions pointed by P0 has no upper limit in terms of the lengths to be compared, the bottleneck in both of the proofs is the check against a cheating P1 who tries to make the verifier run forever (i.e. the infinity claim). This is basically what holds us back from simulating any Turing machine rather than some polynomial-space Turing machine, and achieving much larger lower bounds for debate checking by constant-space verifiers. However, if the verifier had the ability to talk to P1 and P0, we would be able to use another technique in that setting to handle the possibility of P1 making the verifier run forever and get rid of this bottleneck, thereby enabling us to simulate arbitrary Turing machines, as follows.

Recall that a competing-prover interactive proof system (CIPS) consists of a verifier and two competing deterministic provers who cannot see the messages exchanged between the verifier and the other prover (see Section 2.2.4). The proof of Theorem 16 in [19] makes crucial use of the feature of that model which forbids the provers from seeing each other's messages. We can show that a new model obtained by allowing the P0 of the CIPS model to see all of P1's messages would still have the capability of handling all recursive languages.

Theorem 22. *Every recursive language has a competing-prover interactive proof system where the verifier is a probabilistic finite automaton that halts with probability 1, P0 displays zero information to P1, and P1 displays complete information to P0.*

Proof. Let M be a deterministic Turing machine that halts for all inputs. We will use the protocol in the proof of Theorem 20. P1 will present ACP's of M on the input string w, P0 will point out transition errors in the messages of P1, and the verifier V will essentially process the transition error claims all in the same manner as in that proof. The difference is that P0 will not be making infinity claims, since the methods for processing those in Theorem 20 are only applicable if M has a polynomial space bound. Instead, we use the following technique from [19].

After each move, V tosses a coin, and uses its new right to talk to the provers to announce the outcome to both provers. If the outcome is a tail, V, P0 and P1 continue with the protocol as usual. Otherwise, they restart the protocol, with P1 emitting the first symbol of a string that it claims to be an ACP. Suppose that a correct description of the (accepting or rejecting) computation path of M for w in the proper format is t symbols long. Either P1 or P0 must then tell a lie within the first t symbols of their messages. There will be infinitely many contiguous subsequences of t tails appearing one after the other within the sequence of the announced coin outcomes. In each of these runs of tails, V will halt with a small probability, but the decision that it gives with that small probability will be overwhelmingly likely to be correct, as we saw in the analysis of the processing of the transition error claims in the proof of Theorem 20. Overall, V halts with probability 1, and its error bound can be tuned down to be any desired nonzero value. $\square$

Condon [8] defines a generic game automaton which could be interpreted as a competing prover interactive proof system in which some part of the verifier's coin tosses are visible to P1 and the rest of the coin tosses are visible to P0. She then continues with a restricted model where all the messaging between P1 and the verifier is seen by P0, and gives results only for this restricted version. While doing that, she raises the question of whether the power of this restricted version is the same with that of the generic one (see Section 2.1.7 of [8]). Theorems 16 and 22 constitute evidence for the claim that the power of such game-like models stays the same if P1 displays complete information.

3.2 Time-Bounded Verifiers

In this section, we will first prove that probabilistic verifiers on which the only restriction is a time bound have no significant advantage over deterministic ones. For example, Corollary 2 and Theorem 8 show that no amount of randomness can help a polynomial-time verifier reading a debate to "break the PSPACE barrier:"[11]

Theorem 23. *For any $t(n) \geq n$,*

$$\mathsf{NSPACE}(t(n)) \subseteq \mathsf{CDEB}(\infty, t(n)^2, \infty) \subseteq \mathsf{PDEB}(\infty, t(n)^2, \infty),$$

and

$$\mathsf{CDEB}(\infty, t(n), \infty) \subseteq \mathsf{PDEB}(\infty, t(n), \infty) \subseteq \mathsf{DSPACE}(t(n)).$$

[11]Note that, in this section, there is no need to distinguish the debates checkable for some error bound and those checkable for any error bound, since our definition in Section 2.1.3 ensures that all time-bounded verifiers halt with probability 1, and therefore can be made to run repeatedly to achieve any desired error bound.

Corollary 2. $\mathsf{CDEB}(\infty, poly, \infty) = \mathsf{PDEB}(\infty, poly, \infty) = \mathsf{PSPACE}$.

In both partial-information debate checking and competing-prover interactive proof systems (Section 2.2.4), the verifiers try to figure out the truthful one between two provers with opposite claims who have restricted access to the messages of the opponent. The only major difference between these two models is the ability of the verifiers of CIPSs to send messages (e.g. ask questions) to the provers. We already mentioned (Theorem 14) that CIPSs with polynomial-time verifiers exist for all languages in the class $\mathsf{EXPTIME}$. Corollary 2 shows that, when the verifier is restricted to be silent, one cannot go beyond PSPACE. This is yet another example of the power of interaction in such systems.[12]

In Section 3.2.2, we will show that restricting these time-bounded verifiers to use space only in the order of the logarithm of their time bound does not decrease their power significantly:

Theorem 24. *For any $t(n) \geq n$,*

$$\mathsf{NSPACE}(t(n)) \subseteq \mathsf{CDEB}(\log t(n), t(n)^6, t(n)^2 \log t(n)) \subseteq \mathsf{PDEB}(\log t(n), t(n)^6, t(n)^2 \log t(n)) \subseteq \mathsf{DSPACE}(t(n)^6).$$

Corollary 3. $\mathsf{CDEB}(log, poly, \infty) = \mathsf{PDEB}(log, poly, \infty) = \mathsf{PSPACE}$,
$\mathsf{CDEB}(poly, exp, \infty) = \mathsf{PDEB}(poly, exp, \infty) = \mathsf{EXPSPACE}$.

Section 3.2.3 will show that, in the case of simultaneously polynomial-time logspace bounded verifiers, a logarithmic amount of randomness is sufficient to capture PSPACE by checking complete-information debates:

Theorem 25. $\mathsf{CDEB}(log, poly, log) = \mathsf{PDEB}(log, poly, log) = \mathsf{PSPACE}$.

Finally, we will define the quantified max word problem and give a result stating that no polynomial-time algorithm approximating the maximization version of the quantified max word problem within some polynomial factor exists, unless $\mathsf{P} = \mathsf{PSPACE}$, in Section 3.2.4.

3.2.1 Time Bounds

Theorem 23. *For any $t(n) \geq n$,*

$$\mathsf{NSPACE}(t(n)) \subseteq \mathsf{CDEB}(\infty, t(n)^2, \infty) \subseteq \mathsf{PDEB}(\infty, t(n)^2, \infty),$$

and

$$\mathsf{CDEB}(\infty, t(n), \infty) \subseteq \mathsf{PDEB}(\infty, t(n), \infty) \subseteq \mathsf{DSPACE}(t(n)).$$

Proof. The first containment in the first equation is from Theorem 2. The second containment in the first equation and the first containment in the second equation are straightforward. We give a proof for the second containment in the second equation.

Peterson and Reif's proof [34] of the fact that a partial-information debate can be simulated by a complete-information debate within the same time bound in the deterministic case can be modified to apply to probabilistic verifiers as well. They use the idea that a partial-information debate can also be constructed in two parts as a complete information debate if the verifier has the resources to put these parts together and treat them as one partial-information debate. P1 and P0 first carry out the "public part" of the debate, with the condition that the private messages of

[12] Notice the similar contrast between IP(log-space, poly-time)=PSPACE [9, 38] and $\mathsf{oneway\text{-}IP}$(log-space, poly-time)=$\mathsf{NP}$ [10].

P0 are censored (by P0 giving a special public symbol instead of each private message), and then, P1 shuts up, and P0 announces (again using only public symbols) what its private messages were supposed to be.

Using this idea, a $O(t(n))$-space deterministic TM can be shown to recognize an arbitrary language having partial-information debates checkable by a $t(n)$-time probabilistic verifier V. Basically, if Γ_1, Γ_0 and Δ are P1 and P0's public alphabets, and P0's private alphabet, respectively, the TM looks for a debate subtree of complete information debates constructed as described above on alphabets Γ_1 and $\Gamma_0 \cup \{\sharp\} \cup \Delta$, where $\sharp \notin \Gamma_1 \cup \Gamma_0 \cup \Delta$ is the blank symbol used to censor private messages of P0 in the "public part." It accepts if and only if such a debate subtree is found with the condition that every debate in it causes V to accept with high probability (i.e. for a majority of $t(n)$-length random strings), for the particular replacement of $\sharp$s with symbols from Δ associated with this subtree. □

Note that, similar to the situation with deterministic time-bounded verifiers (Theorems 2 and 8), letting the debate contain private messages does not yield a significant expansion in the class of languages having such debates checkable by probabilistic verifiers.

3.2.2 Simultaneous Time and Space Bounds

We now show that randomness does seem to help for complete-information debates. Recall that $\mathsf{CDEB}(s(n), \infty, 0) = \mathsf{DTIME}(2^{s(n)})$ (Theorem 1).

Theorem 24. *For any $t(n) \geq n$,*

$$\mathsf{NSPACE}(t(n)) \subseteq \mathsf{CDEB}(\log t(n), t(n)^6, t(n)^2 \log t(n)) \subseteq \\ \mathsf{PDEB}(\log t(n), t(n)^6, t(n)^2 \log t(n)) \subseteq \mathsf{DSPACE}(t(n)^6).$$

Proof. The last inclusion is due to Theorem 23, and we provide a proof for the first one. Let L be a language in $\mathsf{NSPACE}(t(n))$, and let M be an ATM that recognizes L in $ct(n)^2$ steps, for some constant c. We build a verifier V that halts within $O(t(n)^6)$ steps and uses $O(\log t(n))$ memory cells to check complete-information debates about whether M accepts the input string. Let ε be the desired error bound of V.

As we had in the protocol described in the proof of Lemma 10, P1 and P0 are supposed to provide the existential and universal choices of the simulated machine M in the debate. Since V does not have the resources to store a configuration of M, P0 is also expected to be giving a description of the configuration of M after each simulated step. The presented configuration descriptions should contain only tape content, tape head position and the state of M. V will keep track of M's input head position using its own input head. V also requires each such configuration description to be exactly $ct(n)^2$ symbols long, so P0 pads these messages with blanks when necessary. The players are supposed to restart the procedure after precisely $ct(n)^2$ configuration descriptions, which is sufficient to describe a complete computation of M on the input word, have been written in the debate.

Using $O(\log t(n))$ space, V can easily check that the number and the lengths of the configurations presented by P0 are legal. It is also easy for V to check if the first configuration message matches the initial configuration of M. However, checking whether the present configuration sent by P0 follows from the previous one is not something that V can do deterministically with this little memory. Instead, V randomly picks an integer k ($1 \leq k \leq ct(n)^2 - 2$) at the beginning of each simulation of M, and compares only the kth, $(k+1)$th and $(k+2)$th symbols of the ith configuration with the corresponding symbols of the $(i+1)$th configuration, for $1 \leq i < ct(n)^2 - 1$. If it sees a violation of the transition rules of M within this window, V detects P0's lie, and accepts the input. If V

fails to find any such error by P0 after reading $d = \lceil \ln \frac{1}{\varepsilon} \rceil ct(n)^2$ successive simulations of M in the debate, all of which are accompanied with computation paths ending with rejecting configurations, it rejects.

If the input string is not in L, there is a computational path of M which ends with a rejecting configuration no matter which existential choices are made by P1, and V rejects with probability 1 when this path is presented. Otherwise, P0 must sneak a transition error somewhere so that it can end up with a rejecting configuration. In any single simulation of M, V will fail to catch such a transition error with probability at most $\frac{ct(n)^2-1}{ct(n)^2}$. The probability that a member of L will be rejected by V is thus $\left(\frac{ct(n)^2-1}{ct(n)^2}\right)^d$, which can be shown to be not more than ε. V reads $\lceil \ln \frac{1}{\varepsilon} \rceil ct(n)^2$ simulations of M, each of which is $c^2 t(n)^4$ symbols long and requires $2c \log t(n)$ coin tosses; so it runs in $O(t(n)^6)$ time, and uses $O(t(n)^2 \log t(n))$ randomness. □

Thus, we have shown that allowing time-bounded verifiers checking complete-information debates to use randomness lets us constrain their space bounds logarithmically, without decreasing their power, at the expense of a small increase in the time bound, and that giving P0 further privacy does not add to the power under these bounds:

Corollary 3. $\mathsf{CDEB}(log, poly, \infty) = \mathsf{PDEB}(log, poly, \infty) = \mathsf{PSPACE}$,
$\mathsf{CDEB}(poly, exp, \infty) = \mathsf{PDEB}(poly, exp, \infty) = \mathsf{EXPSPACE}$.

Although we stated the corollary of Theorem 24 only for logspace polynomial-time and polynomial-space exponential-time verifiers, the theorem can be extended to apply for larger bounds (e.g. exponential-space double exponential-time). The next section shows that a special case, that of logspace polynomial-time verifiers, needs even fewer random bits to capture the same power.

3.2.3 Simultaneous Time, Space and Randomness Bounds

In this section, we prove that complete- and partial-information debates checkable by logspace polynomial-time verifiers exist for all languages in PSPACE even if the verifiers are restricted to use logaritmically many random bits. We modify the proof of Condon and Ladner [12] used to show that one-way interactive proof systems (see Section 2.2.3) with verifiers having such resource bounds can recognize all languages in NP.

Before moving to the main theorem, we review a method by Lipton [31] to compare multisets in an efficient way by computing their "fingerprints". The fingerprint of the multiset $\{x_1, x_2, ..., x_k\}$ is defined as $\prod_{i=1}^{k}(x_i + r) \bmod p$, where p is a prime and r is an integer in the range $[1, p-1]$. Note the following properties of fingerprints that allow computing them efficiently: The fingerprint of a set can be computed online (e.g. by only one pass over the set) and using space required to store p. Furthermore, the probability of the fingerprints of two different multisets being equal is very small if p and r are chosen randomly under certain conditions:

Lemma 12 ([31]). *The probability that two unequal sets $X = \{x_1, x_2, ..., x_k\}$ and $Y = \{y_1, y_2, ..., y_l\}$ get the same fingerprint is at most*

$$O\left(\frac{\log(b) + \log(m)}{bm} + \frac{1}{b^2 m}\right),$$

where all the elements of X and Y are b bit numbers and $m = \max(k, l)$, provided that the prime p is randomly selected among all the primes in the interval $\left[(bm)^2, 2(bm)^2\right]$ and r is selected randomly from the interval $[1, p-1]$.

Lemma 13. $\mathsf{PSPACE} \subseteq \mathsf{CDEB}(log, poly, log)$.

Proof. We will give a protocol to show that the quantified 3-SAT language has complete-information debates checkable by logspace polynomial-time verifiers using logarithmically many coins. Then, the result follows from the fact that every language in PSPACE is reducible to quantified 3-SAT in logspace.

First of all, notice that a naive approach to show quantified 3-SAT has such debates would be letting P1 and P0 give the truth assignments to existentially and universally quantified variables, respectively. However, checking such a debate may require the verifier to store the assignments, which are $O(n)$ bits long, in memory while checking if the formula is satisfied by these assignments. We give another debate protocol which needs only $O(\log n)$ space to verify.

Similar to the one the prover uses in [12], the protocol between P1 and P0 consists of two phases. In both phases, they will give truth value assignments to each appearence of a literal in the formula. P1 or P0 will be giving the message if the variable corresponding to the literal is quantified with an existential or a universal quantifier, respectively. In the first phase, assignments will be given in the order of the quantification of the variables. For example, given the formula

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \left[(x_2 \vee x_3 \vee \overline{x_1}) \wedge (x_1 \vee \overline{x_4} \vee \overline{x_2})(x_1 \vee x_4 \vee \overline{x_3}) \right],$$

the messages of P1 and P0 in the first phase might be like

$$\begin{array}{lllll} \text{P1}: & (\overline{x_1},1,F)(x_1,2,T)(x_1,3,T) & & (x_3,1,T)(\overline{x_3},3,F) & \\ \text{P0}: & & (x_2,1,F)(\overline{x_2},2,T) & & (\overline{x_4},2,T)(x_4,3,F), \end{array}$$

where a tuple (l, i, a) means that the assignment value $a \in \{T, F\}$ is given to literal l in the ith clause.

The second phase will consist of the same messages, but this time in order of the literals' appearances in the formula. For the example above, the second phase will be as follows.

$$\begin{array}{lllllll} \text{P1}: & & (x_3,1,T)(\overline{x_1},1,F)(x_1,2,T) & & (x_1,3,T) & & (\overline{x_3},3,F) \\ \text{P0}: & (x_2,1,F) & & (\overline{x_4},2,T)(\overline{x_2},2,T) & & (x_4,3,F) & \end{array}$$

The verifier does three checks. The first one is to check if all the appearances of the literals corresponding to a variable are given consistent assignments, meaning all the appearances of the variable have the same truth value and all the appearances of the negation of this variable have the negation of this truth value. Since the first phase gives all such assignments for a variable in succession, the verifier can do this check during the first phase deterministically using $O(\log n)$ space.

The second check is to see if the formula given in the input is satisfiable by these truth assignments. In the second phase, truth assignments of literals are given in order of their appearance in the formula, so this check can be completed deterministically while reading the debate part regarding the second phase.

Finally, the verifier needs to check if P1 and P0 are giving the same assignments to literals in the first and second phases. Since it cannot store all this information in $O(\log n)$ space, the verifier performs this check probabilistically by computing the fingerprints of the sets given in the first and second phases as they are being read, and then comparing them. The verifier compares the fingerprints of the sets given by P1 if the formula has turned out to be satisfable in the second check, and it compares the fingerprints of the sets given by P0 otherwise. If the fingerprints of the compared sets are not equal, the verifier declares the opponent as the winner. Otherwise, it accepts or rejects the input according to the result of the second check. As explained in Lemma 2 of [12], a verifier using logarithmically many random bits can select p and r by satisfying the conditions in Lemma 12 with probability $\frac{3}{4}$, and the fingerprints of different first and second phases will be the same with probability $O(\frac{1}{n})$, where n is the length of the formula. Since the truthful player is assumed to give exactly the same messages in both phases, if the other player gives truth value assignments that are not consistent with the first phase to mislead the verifier, it will be caught with high probability. $\square$

Thus, we have shown

Theorem 25. $\mathsf{CDEB}(log, poly, log) = \mathsf{PDEB}(log, poly, log) = \mathsf{PSPACE}$.

3.2.4 Nonapproximability of the Quantified Max Word Problem

Condon [10] has shown that languages recognized by one-way interactive proof systems with simultaneously polynomial-time and logarithmic-space bounded verifiers are polynomial time reducible to the max word problem for matrices, which is a variation of the well-known word problem for matrices. We will now use her technique to show that the maximization version of a related problem is difficult to approximate.

The *quantified max word problem for matrices* (QMW problem) is defined as follows. Given two finite sets $M_\exists$ and $M_\forall$ of $m \times m$ matrices, two m-length vectors v and b, a bound c, and quantifiers $Q_i \in \{\exists, \forall\}$, is it possible to satisfy the inequality $Q_1 M_1 Q_2 M_2 ... Q_k M_k [v M_1 M_2 ... M_k b^T > c]$, where the M_i variables will be selected from the members of $M_\forall$ if $Q_i = \forall$, and from the members of $M_\exists$ if $Q_i = \exists$, for $1 \leq i \leq k$?

Lemma 14. *Every language in* PSPACE *is polynomial-time reducible to the QMW problem.*

Proof. Let L be any language in PSPACE. Due to Corollary 3, there exists a logspace verifier V that checks complete-information debates for L with error probability ε, halting after reading $2t$ symbols of the debate for some polynomial t in the input length. Let Γ_1 and Γ_0 denote the public alphabets of the players as usual. Without loss of generality, assume that the reading states R of V are divided to two sets R_1 and R_0 such that $R_1 \cap R_0 = \emptyset$, $R_1 \cup R_0 = R$, and the messages of Pi in the debate will be read only on the states in R_i for $i \in \{0, 1\}$. We also assume the initial and halting states of V are in the set R_1. Since V uses logarithmic space, we can assume that it has $2m$ reading configurations, where m is a polynomial in the input length. Order these configurations so that the first m have their state components in R_1, whereas the ones from positions $m+1$ to $2m$ have their state components in R_0. Make sure that the initial configuration is at position 1.

On a specific input string w, we define $p(i, j, \sigma)$ as the probability of V eventually reaching reading configuration j (without visiting any other reading configurations in between) from reading configuration i, where it reads the symbol σ in the debate. Since the computation of V alternates between states in R_1 and R_0 (because of the way the debate is constructed and the assumptions made above), $p(i, j, \sigma) = 0$ for both $i, j \leq m$, and $i, j > m$. Furthermore, the value of any $p(i, j, \sigma)$ depends only on w, i, j, σ, and the transition function of V, and can be computed in polynomial time, using the procedure explained in detail in the proof of Theorem 2.1 in [10].[13] We define two sets $W_{P0} = \{W_{0,\sigma} \mid \sigma \in \Gamma_0\}$ and $W_{P1} = \{W_{1,\sigma} \mid \sigma \in \Gamma_1\}$, where each $W_{0,\sigma}$ is an $m \times m$ matrix containing $p(i+m, j, \sigma)$ as the jth entry of its ith row, and the $W_{1,\sigma}$s are matrices with $p(i, j+m, \sigma)$ as the jth entry of the ith row, for $1 \leq i, j \leq m$.

Let v and b be two vectors with m entries. v has a 1 in the first position, and 0 everywhere else. b has 1's in the positions corresponding to accepting configurations according to the ordering

[13] Note that what we call the "reading configurations" are named "communication configurations" in [10].

we defined above, and 0 everywhere else. Then $\exists M_1 \forall M_2 \exists M_3 ... \forall M_{2t} [v M_1 M_2 ... M_{2t} b^T > 1 - \varepsilon]$ constitutes an instance of the QMW problem, where M_is will be selected from sets $M_\forall = W_{P0}$ and $M_\exists = W_{P1}$.

We have seen to it that the jth entry of $v M_1 M_2 ... M_{2t}$ is the probability that V reaches the jth reading configuration after reading the complete-information debate corresponding to chosen matrices, and that $v M_1 M_2 ... M_{2t} b^T$ is greater than $1-\varepsilon$ if and only if the overall accepting probability of V in this case is sufficiently high. □

Lemma 14 reveals that the QMW problem is PSPACE-hard. The fact that any instance of the QMW problem can be solved using polynomial space by a simple exhaustive depth-first search on the finite game tree of the instance implies the following.

Corollary 4. *The quantified max word problem for matrices is* PSPACE*-complete.*

We now consider the maximization version of the QMW problem, MAX-QMW. Suppose that the matrices in the inequality of an instance Π of the QMW problem are chosen by two players named Player 1 and Player 0. In particular, Player 0 and Player 1 choose matrices quantified by $\forall$ and $\exists$, respectively, in the order of quantification. Their game returns the result of the matrix multiplication $v M_1 M_2 ... M_k b^T$. Let Ω_Π be the maximum number that Player 1 can guarantee to get as the product at the end, no matter what Player 0 does. MAX-QMW is the function from the domain consisting of instances of the quantified max-word problem to their Ω values.

We say that a function $g(x)$ can be approximated within factor $f(n) > 1$ if there is a polynomial-time algorithm which outputs a value in the interval $\left[\frac{g(x)}{f(|x|)}, g(x) f(|x|)\right]$ for any x in the domain of $g(x)$.

We now state an implication of Lemma 14. It is shown in essentially the same way with Theorem 3.1 of [10], which uses a reduction from one-way interactive proof systems to the max word problem that is similar to our Lemma 14.

Theorem 26. *The maximization version of the QMW problem cannot be approximated within factor n^c in polynomial time for any constant $c > 0$, unless* P = PSPACE.

Proof. Suppose that a polynomial-time algorithm A approximates the maximization version of the QMW problem within factor n^c for some constant c. We will show that if such A exists, every language in PSPACE has a polynomial-time algorithm.

It is well known that a language having a polynomial-time probabilistic algorithm also has a polynomial-time algorithm with error probability that is exponentially small in the input length [39]. The same reasoning easily applies for the case of polynomial-time logspace debate systems. Let L be an arbitrary language in PSPACE and V be the polynomial-time logspace verifier checking complete-information debates for L with error bound $\varepsilon = \frac{1}{2^{n^c}}$.

The polynomial-time reduction in Lemma 14 produces an instance of the QMW problem from V with the following property: The left-hand side of the inequality in this instance, which is the subject of the maximization version, can have value at most $\frac{1}{2^{n^c}}$ if the input is not in L, and will have value at least $1 - \frac{1}{2^{n^c}}$ otherwise. Since A approximates the maximization version of the QMW problem, given this instance of the QMW problem, it will output a value in the interval $\left[0, \frac{1}{2^{n^c}} n^c\right]$ for nonmembers of L, and a value in the interval $\left[(1 - \frac{1}{2^{n^c}}) \frac{1}{n^c}, 1\right]$ for members of L. These intervals do not intersect for $n > 1$, and we can decide the membership of any input using the result returned by A. Therefore, the polynomial-time reduction in Lemma 14, together with the approximation algorithm A, constitute a polynomial-time algorithm for an arbitrary language in PSPACE. □

4 Concluding Remarks

In this paper, we presented several characterizations of the classes of languages that have debates checkable by probabilistic verifiers under various combinations of resource bounds. These results lead to a better understanding of whether and how much randomness is useful for debate checking in several setups. For example, we have seen that increasing the amount of randomness available to the verifier enlarges the class of languages with complete-information debates, as demonstrated by the relations

$$\mathsf{CDEB}(cons, \infty, 0) = \mathsf{REG} \subsetneq \mathsf{CDEB}(cons, \infty, cons) = \mathsf{P},$$

and possibly by

$$\mathsf{CDEB}(log, poly, cons) = \mathsf{P} \subseteq \mathsf{CDEB}(log, poly, log) = \mathsf{PSPACE}.$$

In contrast, polynomial-time bounded verifiers checking partial-information debates gain no power from additional amounts of randomness and/or space over a wide range of bounds:

$$\mathsf{PDEB}(log, poly, 0) = \mathsf{PDEB}(poly, poly, poly) = \mathsf{PSPACE}$$

On the other hand, a possible increase in the power when both of these resources are increased simultaneously for exponential-time verifiers is hinted at by the relation

$$\mathsf{PDEB}(log, exp, cons) = \mathsf{EXPTIME} \subseteq \mathsf{EXPSPACE} = \mathsf{PDEB}(poly, exp, exp).$$

Intermediate classes such as $\mathsf{PDEB}(log, exp, exp)$ and $\mathsf{PDEB}(poly, exp, cons)$ would be interesting to characterize.

One of the motivations for our model was to distinguish the three agents (V, P1, and P0) in the debate checking scenario clearly from each other. Interpretations of the alternating TM variants, as well as some other models such as Condon's probabilistic game automata [8], sometimes merge the universal player and the verifier, and make it difficult to ask certain questions that are quite natural in the three-person model. One such question is whether anything changes if we make the coins of the verifier public to the provers. In the model of [8], V's coins are always visible to P0, and Condon shows that the class of languages with what we would call complete-information debates checkable by logspace verifiers whose coins are public to both P1 and P0 is contained in NP.[14] Our demonstration that $\mathsf{CDEB}(log, poly, poly) = \mathsf{PSPACE}$ (Corollary 3) therefore constitutes strong evidence that keeping the coins private increases the power of logspace debate checkers. On the other hand, the situation is different when the only bound imposed on the verifier is that of polynomial time. Since even a deterministic verifier is sufficient to check complete-information debates for languages in PSPACE (Theorem 2) in that case, we conclude, using Theorem 23, that revealing the coins of the verifier to provers does not decrease the power of polynomial-time debate checking as in the accept-reject model of [20] with perfect-information players.

Figure 2 provides an overview of the power of debate checking presented relative to well-known complexity classes. All classes with nonzero randomness bounds have been characterized in this paper.

Recently, Yakaryılmaz [41] examined the verification power of a two-way finite automaton model with access to a finite-size quantum register[15] in Arthur-Merlin games, and showed that such verifiers can be more powerful than their classical counterparts. It would be interesting to study debate systems with this kind of quantum verifiers.

[14] In [8], the class in question is denoted as $\forall\mathsf{BC\text{-}SPACE}(log)$.

[15] See [1] and [40] for the formal definitions.

$$\begin{tikzcd}[row sep=small]
\mathsf{EXPSPACE} & & = & \mathsf{CDEB}(poly, exp, exp) = \mathsf{PDEB}(poly, exp, exp) \\
\mathsf{EXPTIME} \arrow[u, phantom, sloped, "\subseteq"] & & = & \mathsf{PDEB}(cons, \infty, cons) = \mathsf{PDEB}(log, exp, cons) \subseteq \mathsf{PDEB}^{\forall\varepsilon}(cons, \infty, \infty) \\
& \mathsf{E} \arrow[ul, phantom, sloped, "\subseteq"] & = & \mathsf{PDEB}(cons, \infty, 0) \\
\mathsf{PSPACE} \arrow[uu, phantom, sloped, "\subseteq"] & & = & \mathsf{PDEB}(log, poly, log) = \mathsf{PDEB}(log, poly, poly) = \mathsf{PDEB}(\infty, poly, \infty) \\
\mathsf{PSPACE} \arrow[u, phantom, sloped, "="] & & = & \mathsf{CDEB}(log, poly, log) = \mathsf{CDEB}(log, poly, poly) = \mathsf{CDEB}(\infty, poly, \infty) \\
\mathsf{PSPACE} \arrow[u, phantom, sloped, "="] & & = & \mathsf{ZDEB}(cons, \infty, cons) = \mathsf{ZDEB}(log, exp, cons) \subseteq \mathsf{ZDEB}^{\forall\varepsilon}(cons, \infty, \infty) \\
& \mathsf{NSPACE}(n) \arrow[ul, phantom, sloped, "\subseteq"] & = & \mathsf{ZDEB}(cons, \infty, 0) \\
\Sigma_2\mathsf{TIME}(poly) \arrow[uu, phantom, sloped, "\subseteq"] & & = & \mathsf{ZDEB}(\infty, poly, 0) \\
\mathsf{NP} \arrow[u, phantom, sloped, "\subseteq"] & & & \\
\mathsf{P} \arrow[u, phantom, sloped, "\subseteq"] & & = & \mathsf{CDEB}(cons, \infty, cons) = \mathsf{CDEB}(log, poly, cons) \\
\mathsf{REG} \arrow[u, phantom, sloped, "\subset"] & & = & \mathsf{CDEB}(cons, \infty, 0)
\end{tikzcd}$$

Figure 2: The power of debate checking

References

[1] Andris Ambainis and John Watrous. Two–way finite automata with quantum and classical states. *Theoretical Computer Science*, 287(1):299–311, 2002.

[2] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, NY, USA, 2009.

[3] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[4] László Babai and Shlomo Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity class. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[5] Lszl Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[6] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, STOC '88, pages 113–131. ACM, 1988.

[7] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[8] Anne Condon. *Computational Models of Games.* The MIT Press, Cambridge, MA, USA, 1989.

[9] Anne Condon. Space-bounded probabilistic game automata. *Journal of the ACM*, 38(2):472–494, 1991.

[10] Anne Condon. The complexity of the max word problem and the power of one-way interactive proof systems. *Computational Complexity*, 3(3):292–305, 1993.

[11] Anne Condon, Joan Feigenbaum, Carsten Lund, and Peter Shor. Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions (extended abstract). In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, STOC '93, pages 305–314, New York, NY, USA, 1993. ACM.

[12] Anne Condon and Richard Ladner. Interactive proof systems with polynomially bounded strategies. *Journal of Computer and System Sciences*, 50(3):506–518, June 1995.

[13] Anne Condon and Richard J. Lipton. On the complexity of space bounded interactive proofs. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, pages 462–467. IEEE Computer Society, 1989.

[14] Cynthia Dwork and Larry Stockmeyer. Finite state verifiers I: The power of interaction. *Journal of the ACM*, 39(4):800–828, 1992.

[15] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.

[16] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete (preliminary version). In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, SFCS '91, pages 2–12. IEEE Computer Society, 1991.

[17] U. Feige, A. Shamir, and M. Tennenholtz. The noisy oracle problem. In *Advances in Cryptology - CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 284–296. Springer–Verlag, 1990.

[18] Uriel Feige and Joe Kilian. Making games short (extended abstract). In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, STOC '97, pages 506–516. ACM, 1997.

[19] Uriel Feige and Adi Shamir. Multi-oracle interactive protocols with space bounded verifiers. In *Structure in Complexity Theory Conference*, pages 158–164, 1989.

[20] Joan Feigenbaum, Daphne Koller, and Peter Shor. A game-theoretic classification of interactive complexity classes (extended abstract). In *Proceedings of the 10th Annual IEEE Conference on Computational Complexity*, pages 227–237, 1995.

[21] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.

[22] Rūsiņš Freivalds. Probabilistic two-way machines. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*, pages 33–45, 1981.

[23] Martin Furer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. In S. Micali, editor, *Advances in Computing Research 5: Randomness and Computation*, pages 429–442. JAI Press, 1989.

[24] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, New York, NY, USA, 2008.

[25] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 174–187, Washington, DC, USA, 1986. IEEE Computer Society.

[26] S Goldwasser and M Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, STOC '86, pages 59–68. ACM, 1986.

[27] Juris Hartmanis. On non-determinancy in simple computing devices. *Acta Informatica*, 1:336–344, 1972.

[28] K. N. King. Alternating multihead finite automata. *Theoretical Computer Science*, 61(2-3):149–174, 1988.

[29] Marcos Kiwi, Carsten Lund, Daniel Spielman, Alexander Russell, and Ravi Sundaram. Alternation in interaction. *Computational Complexity*, 9:202–246, 2000.

[30] Richard E. Ladner, Richard J. Lipton, and Larry J. Stockmeyer. Alternating pushdown automata. In *Proceedings of 19th Annual IEEE Symposium on Foundations of Computer Science*, pages 92–106. IEEE Computer Society Press, 1978.

[31] Richard J. Lipton. Efficient checking of computations. In *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '90, pages 207–215, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[32] Ioan I. Macarie. Multihead two-way probabilistic finite state automata. *Theory of Computing Systems*, 1997.

[33] Wolfgang J. Paul, Ernst J. Prauß, and Rudiger Reischuk. On alternation. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, SFCS '78, pages 113–122, Washington, DC, USA, 1978. IEEE Computer Society.

[34] Gary L. Peterson and John H. Reif. Multiple-person alternation. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, SFCS '79, pages 348–363. IEEE Computer Society, 1979.

[35] John H. Reif. Universal games of incomplete information. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, STOC '79, pages 288–308. ACM, 1979.

[36] John H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29(2):274 – 301, 1984.

[37] A. C. Cem Say and Abuzer Yakaryılmaz. Finite state verifiers with constant randomness. In *How the World Computes*, volume 7318 of *Lecture Notes in Computer Science*, pages 646–654. 2012.

[38] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[39] Michael Sipser. *Introduction to the Theory of Computation, 2nd edition.* Thomson Course Technology, Boston, MA, USA, 2006.

[40] Abuzer Yakaryılmaz and A. C. Cem Say. Unbounded-error quantum computation with small space bounds. *Information and Computation*, 279(6):873–892, 2011.

[41] Abuzer Yakaryılmaz. Public qubits versus private coins. In *The Proceedings of Workshop on Quantum and Classical Complexity*, pages 45–60. Univeristy of Latvia Press, 2013. ECCC:TR12-130.

[42] Abuzer Yakaryılmaz. Quantum alternation. In *8th International Computer Science Symposium in Russia, Proceedings*, volume 7913 of *LNCS*, pages 334–346. Springer, 2013.

[43] Abuzer Yakaryılmaz and A. C. Cem Say. Succinctness of two-way probabilistic and quantum finite automata. *Discrete Mathematics and Theoretical Computer Science*, 12(4):19–40, 2010.

ECCC ISSN 1433-8092
http://eccc.hpi-web.de