Debates with Small Transparent Quantum Verifiers

Abuzer Yakaryılmaz[1,2,⋆], A.C. Cem Say[3], and H. Gökalp Demirci[4]

[1] University of Latvia, Faculty of Computing,
Raina bulv. 19, Rīga, LV-1586, Latvia
[2] National Laboratory for Scientific Computing,
Petrópolis, RJ, 25651-075, Brazil
[3] Boğaziçi University, Department of Computer Engineering,
Bebek 34342 İstanbul, Turkey
[4] University of Chicago, Department of Computer Science,
Chicago, IL 60637, USA
abuzer@lncc.br, say@boun.edu.tr, demirci@cs.uchicago.edu

Abstract. We study a model where two opposing provers debate over the membership status of a given string in a language, trying to convince a weak verifier whose coins are visible to all. We show that the incorporation of just two qubits to an otherwise classical constant-space verifier raises the class of debatable languages from at most NP to the collection of all Turing-decidable languages (recursive languages). When the verifier is further constrained to make the correct decision with probability 1, the corresponding class goes up from the regular languages up to at least E.

Keywords: quantum finite automata, quantum computing, probabilistic finite automata, Arthur-Merlin games, debate systems, zero-error.

1 Introduction

It is well known that the model of alternating computation is equivalent to a setup where two opposing debaters try to convince a resource-bounded deterministic verifier about whether a given input string is in the language under consideration or not [4]. Variants of this model where the verifier is probabilistic, and the communications between the debaters are restricted in several different ways, have been studied [6,11,9]. Quantum refereed games, where the messages exchanged between the debaters are quantum states, were examined by Gutoski and Watrous [13].

Most of the work cited above model the verifier as opaque, in the sense that the outcomes of its coin throws are not visible to the debaters, who have a correspondingly incomplete picture of its internal state during the debate. These models can therefore be classified as generalizations of private-coin interactive

⋆ Yakaryılmaz was partially supported by CAPES, ERC Advanced Grant MQC, FP7 FET project QALGO, and BÜVAK.

A.M. Shur and M.V. Volkov (Eds.): DLT 2014, LNCS 8633, pp. 327–338, 2014.

proof systems [12] to the competing multiple provers case. In this paper, we focus on models where all of the verifier's coins, as well as all communications, are publicly visible to all parties, making them generalizations of Arthur-Merlin games [3]. A recent result [17] established that a very small quantum component is sufficient to expand the computational power of classical proof systems of this kind considerably, by studying a setup where an otherwise classical constant-space verifier is augmented by a quantum register of just two qubits. We modify that protocol to show that the addition of a two-qubit quantum register to the classical finite state verifier raises the class of debatable languages from at most NP to that of all Turing-decidable languages. We also study the case where the verifier is required to take the correct decision with probability 1. We show that small quantum verifiers outperform their classical counterparts in this respect as well, exhibiting an increase from the class of regular languages to at least $\mathsf{E} = \mathsf{DTIME}(2^{\mathsf{O}(\mathsf{n})})$.[1]

The rest of this paper is structured as follows: Section 2 describes our model and reviews previous work. Our result on the computational power of the model with a two-qubit constant-space verifier in the two-sided bounded error case is presented in Section 3. Section 4 contains an examination of the more restricted zero-error case. Section 5 concludes the paper with some remarks on the possible usage of multihead automata as verifiers.

2 Preliminaries

Consider an interactive system consisting of three actors: two debaters named Player 1 (P1) and Player 0 (P0), respectively, and a Verifier (V). All actors have access to a common input string w. P1 tries to convince V that w is a member of the language L under consideration, whereas P0 wants to make V reject w as a non-member. The debaters communicate with each other through a communication cell which is seen by every actor. Each debater writes a symbol in the communication cell when its turn comes, and V executes a further step of its computation, taking this communication and the outcomes of its coin into account. The debate continues in this way until the computation of V is terminated as it reaches a decision. We assume that both debaters see the coin outcomes of V as they occur, and thereby have complete information about the state of the verifier at any point.

In such a setup, we say that language L *has a debate checkable by a machine* V *with error bound* $\epsilon \in [0, \frac{1}{2})$ if

- for each $w \in L$, P1 is able to make V accept w with probability at least $1-\epsilon$, no matter what P0 says in return,
- for each $w \notin L$, P0 is able to make V reject w with probability at least $1-\epsilon$, no matter what P1 says in return.

A language is said to be *debatable* if it has a debate checkable by some verifier.

[1] Note that E is a proper subset of $\mathsf{EXP} = \mathsf{DTIME}(2^{\mathsf{poly}(\mathsf{n})})$.

Note that the class of debatable languages is closed under complementation.

We focus on verifiers which are only allowed to operate under constant space bounds. When V is set to be a deterministic two-way finite automaton, the system described above is equivalent to an alternating two-way finite automaton, and the class of debatable languages coincides with the regular languages [15]. When one replaces V with a two-way probabilistic finite automaton, the class in question becomes one that should be denoted $\forall$BC-SPACE(1) in the terminology of [5], and is known to contain some nonregular languages [10], and to be contained in NP. We will show that the addition of a small amount of quantum memory to the probabilistic model increases the power hugely, all the way to the class of decidable languages.

The public-coin quantum verifier model that we will use is the two-way finite automaton with quantum and classical states (2QCFA) [2], in which the quantum and classical memories are nicely separated, allowing a precise quantification of the amount of "quantumness" required for our task. Such automata execute quantum and classical moves alternately at each step:

- First, a superoperator[2] (Figure 1), determined by the current classical state and the symbols being scanned on the input tape and in the communication cell, is applied to the quantum register (the quantum memory of the machine), with the outcome of the operator being automatically sent to the debaters. All entries of quantum operators are rational numbers, meaning that the probabilities of the outcomes are always rational,[3] and the debaters can easily keep track of the superposition in the quantum register.
- Then, the next classical state and tape head movement is determined by the current classical state and the observed outcome.

Execution halts when an outcome associated with "acceptance" or "rejection" is observed.

One obtains the definition of the quantum Arthur-Merlin (qAM) systems of [17] when one removes P0 from the picture described above. Our results on small quantum verifiers for debates are based on the following result:

Fact 1. *For any error bound $\epsilon > 0$, every Turing-recognizable language (recursively enumerable language) has an Arthur-Merlin system where the verifier uses just two quantum bits, (i.e. four quantum states,) members of the language are accepted with probability 1, nonmembers are accepted with a probability not greater than ϵ, and a dishonest P1 can cause the machine to run forever without reaching a decision.*

Proof. We outline the basic idea, referring the reader to [17] for a detailed exposition of this proof. Let T be the single-tape Turing machine recognizing the

[2] The usage of superoperators generalizes and simplifies the quantum transition setup of the 2QCFA's of [2]; see [20].

[3] The classical probabilistic finite automata, to which we compare our quantum model, can only flip fair coins. It is known that this is sufficient for two-way automata to realize any rational transition probability.

For a 2QCFA with j quantum states, each superoperator $\mathcal{E}$ is composed of a finite number of $j \times j$ matrices called *operation elements*, $\mathcal{E} = \{E_1, \ldots, E_k\}$, satisfying

$$\sum_{i=1}^{k} E_i^\dagger E_i = I, \tag{1}$$

where $k \in \mathbb{Z}^+$, and the indices are the measurement outcomes. When a superoperator $\mathcal{E}$ is applied to a quantum register in state $|\psi\rangle$, then we obtain the measurement outcome i with probability $p_i = \langle\widetilde{\psi_i}|\widetilde{\psi_i}\rangle$, where $|\widetilde{\psi_i}\rangle$ is calculated as $|\widetilde{\psi_i}\rangle = E_i|\psi\rangle$, and $1 \leq i \leq k$. If the outcome i is observed ($p_i > 0$), the new state of the system is obtained by normalizing $|\widetilde{\psi_i}\rangle$ which is $|\psi_i\rangle = \frac{|\widetilde{\psi_i}\rangle}{\sqrt{p_i}}$. Moreover, the quantum register can be set to a predefined quantum state by an initialize operator with a single outcome.

Fig. 1. Superoperators (adapted from [17])

language L under consideration. For any input string w, P1 (the only debater in this restricted scenario) is supposed to send the computation history (i.e. the sequence of configurations) of T on input w to the verifier V. Some of the possible outcomes of V's observations of its quantum register will be interpreted as "restart" commands to P1. At any point, V may interrupt P1 and ask it to restart sending the computation history from the beginning in this manner. (In fact, the verifier is highly likely to require a restart at each step.)

Whenever the verifier catches P1 lying (i.e. giving an incorrect configuration description), it rejects the input. If the verifier reads a computation history sent by P1 all the way to its completion by a halting configuration without detecting an incorrect configuration, it halts with a decision paralleling the one described in that history with a certain non-zero probability, and requests a restart with the remaining probability.

A classical public-coin finite automaton faced with this task would not be able to compare two consecutive configuration descriptions c_i and c_{i+1} (which may be very long).[4] A 2QCFA verifier handles this problem by encoding the substrings in question into the amplitudes of its quantum states.[5] Let $next(c)$ denote the description of the configuration that is the legitimate successor, according to the transition function of T, of configuration c, and let $e(x)$ denote an integer that encodes string x according to a technique to be described later. After the description c_{i+1} has been read, the amplitudes of the quantum states of V form the vector $\alpha\,(1 \;\; e(next(c_i)) \;\; e(c_{i+1}) \;\; e(next(c_{i+1})))^{\mathsf{T}}$, where α is a small rational number. (The amplitude of the first state is used as an auxiliary value during the encoding [17], as will also be seen in the next section.)

[4] The associated complexity class is known [7] to be included in P. When the verifier is allowed to hide its coins, its power increases [8].

[5] Actually, this encoding can also be performed by a classical probabilistic machine [16]. It is the subsequent subtraction that is impossible for classical automata.

When P1 concludes the presentation of c_{i+1}, V executes a move that has the effect of subtracting $\alpha e(next(c_i))$ from $\alpha e(c_{i+1})$, rejecting with a probability equal to the square of the difference, continuing with some little probability after placing the encoding of $next(c_{i+1})$ into the second state's amplitude and resetting the third and fourth amplitudes to zero for beginning the next encode-compare stage, and requesting a restart with the remaining probability. If c_{i+1}'s description is indeed equal to the valid successor of c_i, the subtraction mentioned above yields zero probability of rejection. Otherwise, the rejection probability arising from a transition error within a computation history is guaranteed to be a big multiple of the acceptance probability that may arise due to that spurious history ending with an accepting configuration.

If $w \in L$, P1 need only obey the protocol, sending the accepting computation history, restarting each time V tells it to do so. In each try, P1 has a small but nonzero probability of sending the full history without being interrupted, leading to a nonzero probability of halting with acceptance. Since V will detect no transition errors between configurations, the probability of rejection is zero.

If $w \notin L$, any attempt of P1 to trick V to accept w with high probability by sneaking a transition error to the history and ending it with an accepting configuration will be foiled, since the rejection probability associated with the defect in the history can be guaranteed to be as big a multiple of the final acceptance probability as one desires. There is, however, one annoyance that P1 can cause V in this case: If P1 sends an infinite-length "configuration description" at any point[6] during its presentation, V will never reach the point where it compares the two amplitudes it uses for encoding, and it will therefore fail to halt. □

3 Small Transparent Verifiers for All Decidable Languages

Our first result is a generalization of the proof of Fact 1 to the setup with two debaters described in the previous section.[7]

Theorem 1. *For every error bound $\epsilon > 0$, every Turing-decidable language has a debate checkable by a 2QCFA with four quantum states, and with error bounded by ϵ.*

Proof. We modify the verifier V described in the proof of Fact 1 in Section 2 to obtain a new verifier V_1 as follows: V_1 listens to both P0 and P1 in parallel. In the protocol imposed by V_1, both debaters are expected to behave exactly as P1 was supposed to behave in that earlier proof; transmitting the computation

[6] Except at the beginning, since V can check the first configuration itself by matching it with the input.

[7] In separate work, the techniques of [17] were used to define a model called q-alternation [18]. This model is distinct from debate checking in the same sense that the two equivalent definitions of classical nondeterminism (the "probabilistic machine with zero cut-point" and the "verifier-certificate" views) lead to quantum counterparts ([1] and [14], respectively) which are remarkably different from each other.

history of the single-tape Turing machine T for language L on input string w, interrupting and restarting transmissions whenever V_1 observes an outcome associated with the "restart" action in its quantum register.

The strategy of V_1 is based on the fact that the two debaters are bound to disagree at some point about the computation history of T on w. As long as the same description is coming in from both debaters, V_1 uses the same technique mentioned in the proof of Fact 1, to be described in more detail shortly, for encoding the successive configurations. At the first point within a history when a mismatch between the two debaters is detected, V_1 uses its register to flip a fair coin to choose to trace one or the other debater's transmission from that time. The chosen debater's description of what it purports to be the computation history is then checked exactly as in the earlier proof, and the other debater is ignored until a restart is issued by V_1 to both players during (or at the end of) that check. The truthful debater always obeys the protocol. In the case that the other debater's transmission is identical to that of the truthful one, V_1 parallels the decision of T depicted by both debaters.

If it sees the debater it is tracing violating the protocol, for instance, making a transition error, V_1 rules in favor of the other player. When it sees a debater announcing the end of a computation history, V_1 decides in that debater's favor with some probability, and demands a restart with the remaining probability. Like the program described in the proof of Fact 1, V_1 is constructed so that the probability of the decision caused by the detection of a transition error in a computation history is guaranteed to be much greater than the probability of the decision caused by mimicking the result described at the end of that history.

A full description of V_1 would involve the complete presentation of its classical transition function, as well as all the operation elements of every superoperator associated with every triple of state, input symbol, and debate symbol. We will give a higher-level description of the program and its execution at a level that will allow the interested reader to construct the full 2QCFA if she wishes to do so.

A segment of computation which begins with a (re)start, and ends with a halting or restarting configuration will be called a "round" [19]. In each such round, each debater is supposed to transmit a string of the form

$$c_1\$\$c_2\$\$\cdots c_{h-1}\$\$,$$

where c_1 is the description of the start configuration of T on w, each c_{i+1} is the legal successor of the corresponding c_i, and c_{h-1} is the last configuration in the computation history before the halting configuration. (V_1 will be able to understand whether the successor of c_{h-1} is an accepting or rejecting configuration by focusing on the symbols around the tape head in c_{h-1}.) We assume that each configuration description ends with the blank symbol #, and that the alphabet Γ used to write the configurations does not include the \$ symbol. Fix an ordering of the symbols in Γ, and let $e(\sigma)$ denote the position of any symbol $\sigma \in \Gamma$ in this ordering. Let m be an integer greater than the cardinality of Γ, we will fix its value later.

The state of the quantum register is set to $|\psi_{1,0}\rangle = (1\ 0\ 0\ 0)^{\mathsf{T}}$ at the beginning of each round.

Let l_i be the length of $c_1\$\$c_2\$\$\cdots c_i\$\$$ $(i > 0)$.

As it reads the string $w_1 = c_1\$\$$ from the debaters, V_1 both compares it with the input to catch a debater that may lie at this point, and also applies a superoperator corresponding to each symbol of w_1 to the register in order to encode $next(c_1)$ as a number in base m (times a factor that will be described later) into the amplitude of the second quantum state. One operation element of the superoperator $\mathcal{E}_{1,j}$ applied when reading the jth symbol, say, σ, of w_1 is

$$E_{1,j,1} = \frac{1}{d}\begin{pmatrix} 1 & 0 & 0 & 0 \\ e(\sigma) & m & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

where d is an integer which has the properties to be described now.[8] Since $\mathcal{E}_{1,j}$ would not obey the wellformedness criterion (Equation 1 in Figure 1) if its only operation element were $E_{1,j,1}$, we add as many 4×4 rational matrices as necessary as *auxiliary operation elements* of $\mathcal{E}_{1,j}$ to complement its single *main operation element* $E_{1,j,1}$ to ensure that Equation 1 is satisfied. Furthermore, we do this for all superoperators to be described in the rest of the program in such a way that each of their main operation elements can be written with the same factor $\frac{1}{d}$ in front, as we just did for $E_{1,j,1}$. This is the property that d must satisfy, and such a d can be found easily [17,20].

The observation outcome associated with all auxiliary operation elements will be interpreted as a "restart" command to the debaters. Some operation elements to be described below are associated with halting (acceptance or rejection). The outcomes of all remaining operation elements, including the $E_{1,j,1}$, are "continue" commands.

Depending on whether the length of T's configuration description increases as a result of its first move or not, we have the following cases:

- If $|next(c_1)| = |c_1|$, the main operation elements of $\mathcal{E}_{1,|c_1|}$ and $\mathcal{E}_{1,|c_1\$|}$ are

$$\frac{1}{d}\begin{pmatrix} 1 & 0 & 0 & 0 \\ e(\#) & m & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } \frac{1}{d}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

  respectively, since the encoding of $next(c_1)$ is finished by superoperator $\mathcal{E}_{1,|c_1|}$.
- If $|next(c_1)| = |c_1| + 1$, and the $|c_1|$th symbol of $next(c_1)$ is σ, the main operation elements of $\mathcal{E}_{1,|c_1|}$ and $\mathcal{E}_{1,|c_1\$|}$ are

$$\frac{1}{d}\begin{pmatrix} 1 & 0 & 0 & 0 \\ e(\sigma) & m & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ and } \frac{1}{d}\begin{pmatrix} 1 & 0 & 0 & 0 \\ e(\#) & m & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

[8] Note that the "names" we are using for the superoperators are based on their application position on the debater transmissions; this same superoperator would be applied again (but would have a different index in our exposition) if another σ comes up elsewhere in the transmission of c_1.

respectively, since the encoding of $next(c_1)$ is finished by superoperator $\mathcal{E}_{1,|c_1|+1}$.

The main operation element of $\mathcal{E}_{1,|c_1\$\$|}$ just multiplies the state vector by $\frac{1}{d}$.

As long as the debaters are in agreement, and a halting configuration has not been detected, each configuration description block $w_i = c_i\$\$$ $(i \geq 2)$ is processed in the following manner. The state vector is

$$|\widetilde{\psi_{i,0}}\rangle = \left(\frac{1}{d}\right)^{l_{i-1}} (1 \;\; e(next(c_{i-1})) \;\; 0 \;\; 0)^{\mathsf{T}}$$

at the beginning of the processing. The tasks are:

1. To encode c_i and $next(c_i)$ into the amplitudes of the third and fourth quantum states, respectively, during the processing of the substring $c_i\$$, and
2. To accept (resp. reject) the input if $next(c_i)$ is an accepting (resp. rejecting) configuration, or to prepare for the $(i+1)^{st}$ configuration description block if $next(c_i)$ is not a halting configuration, during the processing of the final \$ symbol.

The details of superoperators to encode c_i and $next(c_i)$ are similar to the ones given above. For each $j \in \{1, \ldots, |c_i| - 1\}$, the main operation element of $\mathcal{E}_{i,j}$ is

$$\frac{1}{d}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ e(\sigma) & 0 & m & 0 \\ e(\gamma) & 0 & 0 & m \end{pmatrix},$$

where σ and γ are the j'th symbols of c_i and $next(c_i)$, respectively. $\mathcal{E}_{i,|c_i|}$ and $\mathcal{E}_{i,|c_i\$|}$ handle the two cases where $e(next(c_i))$ may or may not be longer than $e(c_i)$, similarly to the superoperators seen for the processing of c_1 [17]. Thus, before applying $\mathcal{E}_{i,|c_i\$\$|}$, the state vector becomes

$$|\widetilde{\psi_{i,|c_i\$|}}\rangle = \left(\frac{1}{d}\right)^{l_i - 1} (1 \;\; e(next(c_{i-1}) \;\; e(c_i) \;\; e(next(c_i)))^{\mathsf{T}}. \tag{2}$$

Task (2) described above is to be realized by operator $\mathcal{E}_{i,|c_i\$\$|}$, which has one main operation element, as described in Figure 2.

After a disagreement between the debaters is noticed, the verifier picks a debater with probability $\frac{1}{2}$. (A fair coin can be implemented in this setup by the superoperator $\mathcal{E} = \left\{E_{h_1} = \frac{1}{2}I, E_{h_2} = \frac{1}{2}I, E_{t_1} = \frac{1}{2}I, E_{t_2} = \frac{1}{2}I\right\}$ with the outcomes for the first two operation elements interpreted as heads and the other ones as tails, for instance.) The processing of the transmission of the chosen debater is the same as the processing of the common stream, except for the last superoperator dealing with the final \$ symbol of each description block. That superoperator has two main operation elements. The first one realizes the first actual transition correctness check:

$$\frac{1}{d}\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

DESCRIPTION	OPERATOR
If $next(c_i)$ is a halting configuration, then this operator is applied with the action of acceptance or rejection, as indicated by $next(c_i)$, associated with the outcome. The input is thereby accepted or rejected with probability $p_1 = \left(\frac{1}{d}\right)^{2l_i}$. The round is terminated in this case.	$\frac{1}{d}\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$
If $next(c_i)$ is not a halting configuration, then this operator is applied. The state vector becomes $\widetilde{\lvert\psi_{i+1,0}\rangle} = \left(\frac{1}{d}\right)^{l_i}(1 \;\; e(next(c_i)) \;\; 0 \;\; 0)^{\mathsf{T}}$.	$\frac{1}{d}\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&0&0\\0&0&0&0\end{pmatrix}$

Fig. 2. Operation element for preparing for the next configuration in the debater stream

The associated action of this operation element is to reject the claim of this debater. Therefore, when talking to P0 (resp., P1), the input is accepted (resp., rejected) with probability $\left(\frac{1}{d}\right)^{2l_i}\left(e(next(c_{i-1})) - e(c_i)\right)^2$, which is zero if the check succeeds ($next(c_{i-1}) = c_i$), and is at least $p_2 = \left(\frac{1}{d}\right)^{2l_i} m^2$ if the check fails ($next(c_{i-1}) \neq c_i$). Since the last symbols of $next(c_{i-1})$ and c_i are identical, the value of $|e(next(c_{i-1})) - e(c_i)|$ can not be less than m in this case.

The second main operation element is the one already described in Figure 2, which either halts and decides, or readies the state vector for scanning the next configuration (with small probability) depending on whether that next configuration is a halting one or not.

Note that if the chosen debater is cheating and never sends any \$'s, then the communication with it terminates with probability 1 without any decision.

The overall acceptance probability of such a "program with restart" equals the ratio of the acceptance probability to the halting probability in a single round [19]. The probability that the truthful debater will be selected after the disagreement is $\frac{1}{2}$. If this happens, V_1 will reach a halting state with the correct decision with some small probability, and restart with the remaining probability. In case the other debater is selected, there are two different possibilities of deception. If that debater presents an infinite "configuration", V_1 will restart sooner or later. Otherwise, if a finite but spurious history with one or more incorrect transitions is presented, V_1 may make the wrong decision with some small probability p_1, but this is more than compensated by the much greater probability p_2 of its making the correct decision earlier on, when the transition error(s) in this history were detected. Overall, the error rate of ϵ of V_1 is bounded by $\frac{p_1}{p_1+p_2} = \frac{1}{m^2+1}$, and can be tuned down to any desired positive value by choosing m, the base of the encoding used, to be a sufficiently large integer. □

4 Debates with Zero Error

In classical computation, the benefits of using random bits come at the cost of incurring some nonzero probability of error; and "zero-error" probabilistic finite

automata can be shown trivially to be no more powerful than their deterministic counterparts. We will now show that randomness without some tolerance of error is not useful for classical finite-state verifiers of debates, and then prove that things change in the quantum case.

Theorem 2. *The computational power of a public-coin probabilistic debate checking system is reduced to the level of its deterministic counterpart when the verifier is not allowed to make any error in its final decision.*

Proof. Assume that a language L has a debate checkable by a probabilistic verifier V_p with zero error. We construct a deterministic verifier V_d with the same space and time bounds as V_p. V_d mimics V_p, except that whenever it needs to simulate a coin throw of V_p, it reads the corresponding bit from P0.

If a string $w \in L$, then P1 is able to convince V_p to accept w, no matter what P0, or the coins of V_p, can "say." Note that in such a case, P1 will be able to convince V_d to accept w, no matter what P0 can say. If $w \notin L$, then P0 is able to convince V_p to reject w, no matter what P1, or the coins of V_p, can say. In this case, P0 would of course be able to convince V_d to reject w, regardless of what P1 might say. □

As mentioned in Section 2, languages with debates checkable by deterministic finite state verifiers are regular, whereas probabilistic verifiers can handle some nonregular languages when some error is allowed. We will now see that our small quantum verifiers can do much more with zero error.

Theorem 3. *Every language in the class* E *has a debate checkable by a 2QCFA with four quantum states, and with zero error.*

Proof. Since $\mathsf{E} = \mathsf{ASPACE}(n)$ (the class of languages recognized by alternating Turing machines (ATMs) using linear space) [4], it is sufficient to show how to trace the execution of a linear-space alternating Turing machine (ATM). Let A be an ATM that decides a language L, using at most kn tape squares for its computation on any string of length n, for a positive integer k. Assume, without loss of generality, that A alternates between existential and universal states at each step, and that the start state is an existential state.

We construct a 2QCFA V_0 that checks debates on membership in L. V_0 is a variant of the verifier V_1 described in the proof of Theorem 1. In this version, the debaters play a game to produce a computation history of A on the input w of length n. The protocol dictates that P1 starts by announcing the first existential choice to be made. Both debaters then transmit the start configuration of A on w parallelly. P0 then announces the first universal choice as a response to the first move of P1, followed by both debaters transmitting the configuration that A would reach by executing the choice announced by P1 in the beginning. In general, the choice that determines configuration c_{i+1} is announced by the corresponding debater before the transmission of configuration c_i. As usual, the verifier may order the debaters to restart the whole thing at any step.

After using it to check that the first configuration description is accurate, V_0 starts moving its reading head on the input tape back and forth at the appropriate

speed to make sure that neither debater sends a configuration description longer than nk symbols in the rest of the transmission, deciding against any debater seen to violate this rule. As described for the verifiers in our earlier proofs, V_0 scans the parallel transmissions, encoding the last configuration descriptions it has seen, as well as their legal successors according to the choices that have already been announced by the debaters. If the debaters send the same complete history, V_0 halts and announces the result in that history. If the debaters disagree, V_0 flips a coin and picks one debater's transmission to trace, just like V_1. Unlike V_1, however, V_0 does not trace this debater until it sends a halting configuration. Instead, V_0 just performs the transition check between the previously sent configuration and the presently sent one,[9] and then issues a restart command. V_0 does not imitate any decision of A that it may see in the transmission of the chosen debater; the only way that V_0 can halt without any restarts after choosing a debater is by detecting a transition error, and deciding in favor of the other debater.

If both debaters obey the protocol, then P1 will always be able to demonstrate an accepting computation history of A on w if $w \in L$, and P0 will always be able to demonstrate a rejecting computation history of A on w if $w \notin L$. So let us examine the case where one debater is lying.

If V_0 chooses the truthful debater to trace, it will detect no error, and so will restart with certainty. If it chooses the other debater, it will detect a transition error and announce the correct decision with some probability, and restart with the remaining probability. There is no possibility that V_0 can make an error. □

5 Concluding Remarks

It is well known that finite automata with k classical input heads can use them as one can use logarithmic space; for instance, to count up to $O(n^k)$. One can therefore extend the argument of Theorem 3 to APSPACE (the class of languages recognized by ATMs using polynomial space), which equals EXPTIME [4], concluding that every language in the class EXPTIME has a zero-error (public-coin) debate checkable by a multiple-head 2QCFA with four quantum states

Debate systems with deterministic logarithmic-space (or equivalently, multi-head finite-state) verifiers which have the additional property that P0 can hide some of its messages to the verifier from P1 are known to correspond to the class EXPTIME. If one upgrades the verifier in this model to a probabilistic version, but demands that it should still make zero error, the computational power does not change, since zero-error probabilistic machines can be derandomized easily. We can therefore also state that every language in the class EXPTIME has such a "partial-information" debate checkable by a private-coin multiple-head two-way probabilistic finite automaton with zero error.

Acknowledgements. We thank the anonymous reviewers for their helpful comments.

[9] If the chosen debater attempts to send an exceedingly long configuration at this point, it will be caught by the control implemented by the input head.

References

1. Adleman, L.M., DeMarrais, J., Huang, M.D.A.: Quantum computability. SIAM Journal on Computing 26(5), 1524–1540 (1997)
2. Ambainis, A., Watrous, J.: Two–way finite automata with quantum and classical states. Theoretical Computer Science 287(1), 299–311 (2002)
3. Babai, L.: Trading group theory for randomness. In: STOC 1985, pp. 421–429 (1985)
4. Chandra, A.K., Kozen, D.C., Stockmeyer, L.J.: Alternation. Journal of the ACM 28(1), 114–133 (1981)
5. Condon, A.: Computational Models of Games. MIT Press (1989)
6. Condon, A., Feigenbaum, J., Lund, C., Shor, P.: Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions (extended abstract). In: STOC 1993, pp. 305–314. ACM (1993)
7. Condon, A., Ladner, R.E.: Probabilistic game automata. Journal of Computer and System Sciences 36(3), 452–489 (1988)
8. Condon, A., Lipton, R.J.: On the complexity of space bounded interactive proofs (extended abstract). In: FOCS 1989, pp. 462–467 (1989)
9. Demirci, H.G., Say, A.C.C., Yakaryılmaz, A.: The complexity of debate checking. Theory of Computing Systems (2014), doi:10.1007/s00224-014-9547-7
10. Dwork, C., Stockmeyer, L.: Finite state verifiers I: The power of interaction. Journal of the ACM 39(4), 800–828 (1992)
11. Feige, U., Kilian, J.: Making games short (extended abstract). In: STOC 1997, pp. 506–516. ACM (1997)
12. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
13. Gutoski, G., Watrous, J.: Toward a general theory of quantum games. In: STOC 2007, pp. 565–574 (2007)
14. Kitaev, A.Y., Shen, A., Vyalyi, M.N.: Classical and Quantum Computation. American Mathematical Society (2002)
15. Ladner, R.E., Lipton, R.J., Stockmeyer, L.J.: Alternating pushdown automata. In: FOCS 1978, pp. 92–106 (1978)
16. Rabin, M.O.: Probabilistic automata. Information and Control 6, 230–243 (1963)
17. Yakaryılmaz, A.: Public-qubits versus private-coins. Tech. rep. (2012), ECCC:TR12-130
18. Yakaryılmaz, A.: Quantum alternation. In: Bulatov, A.A., Shur, A.M. (eds.) CSR 2013. LNCS, vol. 7913, pp. 334–346. Springer, Heidelberg (2013)
19. Yakaryılmaz, A., Say, A.C.C.: Succinctness of two-way probabilistic and quantum finite automata. Discrete Mathematics and Theoretical Computer Science 12(2), 19–40 (2010)
20. Yakaryılmaz, A., Say, A.C.C.: Unbounded-error quantum computation with small space bounds. Information and Computation 279(6), 873–892 (2011)