Electronic Colloquium on Computational Complexity, Report No. 91 (2012)

One-counter verifiers for decidable languages*

Abuzer Yakaryılmaz
University of Latvia, Faculty of Computing, Raina bulv. 19, Rīga, LV-1586, Latvia
abuzer@lu.lv

July 16, 2012

Keywords: interactive proof systems, Arthur-Merlin games, decidable languages, counter automata, probabilistic and quantum computation

Abstract

Condon and Lipton (FOCS 1989) showed that the class of languages having a space-bounded interactive proof system (IPS) is a proper subset of decidable languages, where the verifier is a probabilistic Turing machine. In this paper, we show that if we use architecturally restricted verifiers instead of restricting the working memory, i.e. replacing the working tape(s) with a single counter, we can define some IPS's for each decidable language. Such verifiers are called two-way probabilistic one-counter automata (2pca's). Then, we show that by adding a fixed-size quantum memory to a 2pca, called a two-way one-counter automaton with quantum and classical states (2qcca), the protocol can be space efficient. As a further result, if the 2qcca can use a quantum counter instead of a classical one, then the protocol can even be public, also known as Arthur-Merlin games.

We also investigate the computational power of 2pca's and 2qcca's as language recognizers. We show that bounded-error 2pca's can be more powerful than their deterministic counterparts by giving a bounded-error simulation of their nondeterministic counterparts. Then, we present a new programming technique for bounded-error 2qcca's and show that they can recognize a language which seems not to be recognized by any bounded-error 2pca. We also obtain some interesting results for bounded-error 1-pebble quantum finite automata based on this new technique. Lastly, we prove a conjecture posed by Ravikumar (FSTTCS 1992) regarding 1-pebble probabilistic finite automata, i.e. they can recognize some nonstochastic languages with bounded error.

1 Introduction

The only known interactive proof systems (IPS) having a restricted verifier for decidable languages were given by Feige and Shamir [8] (and independently by Condon and Lipton [4]). Although the verifier is a one-way probabilistic finite automaton, the protocols require to communicate with two provers. Therefore, the question remains open for IPS with one prover. In fact, Condon and Lipton [4] showed that the class of languages having a space-bounded interactive proof system (IPS) is a proper subset of decidable languages, where the verifier is a probabilistic Turing machine (PTM). In this paper, we show that if we use architecturally restricted verifiers instead of restricting the working memory, i.e. replacing the working tape(s) with a single counter, we can define some IPS's for each decidable language.

*This work was partially supported by FP7 FET-Open project QCS.

ISSN 1433-8092

We present four new protocols for decidable languages. In the first protocol, the verifier is a two-way probabilistic one-counter automaton (2pca). By relaxing the requirement of having arbitrary small error bound, we obtain another protocol, in which the verifier does not need to move its input head to the left. In the third protocol, we show that if the verifier uses a fixed-size quantum register, called a two-way one-counter automaton with quantum and classical states (2qcca), then the protocol can also be space efficient. In all of these protocols, the verifiers hide some information from the prover. In the fourth protocol, we show that if we replace the classical counter of the verifier with a quantum counter, called a two-way quantum one-counter automaton (2qca), then the third protocol turns out to be public (Arthur-Merlin games), i.e. the prover always has a complete information about the verifier. The techniques behind these protocols are inspired from the previous weak-protocols, i.e. the nonmembers does not need to be rejected with high probability by the verifier, given for Turing recognizable languages by Condon and Lipton [4] and Yakaryılmaz [22].

We also examine 2pca's and 2qcca's as recognizers. We show that 2pca's form a bigger class than two-way deterministic one-counter automata (2dca's). We obtain this result by giving a bounded-error simulation of two-way nondeterministic one-counter automata. Then, we present a new programming technique for bounded-error 2qcca's and show that they can recognize a language which seems not to be recognized by any bounded-error 2pca. We also obtain some interesting results for bounded-error 1-pebble quantum finite automata based on this new technique. Moreover, we prove a conjecture posed by Ravikumar [19] regarding 1-pebble probabilistic finite automata, i.e. they can recognize some nonstochastic languages with bounded error.

To our knowledge, 2pca's have never been investigated before. The only related work that we know is [11], in which Hromkovic and Schnitger examined two-way probabilistic multi-counter machines that are restricted to polynomial time, but, they did not present any result related to 2pca's. 2qca's, on the other hand, are examined only by Yamasaki at. al. [27] as language recognizers, in which the authors presented some bounded-error 2qca algorithms for some languages. However, these languages are known to be recognized by 2dca's and bounded-error 2qcca's without a counter. Therefore, our results seem the first interesting results on 2qca's.

We provide the necessary background in Section 2. The four protocols for decidable languages are given in Section 3. The language recognition powers of probabilistic and quantum counter automata are investigated in Section 4. Lastly, the results on probabilistic and quantum finite automata with 1-pebble are given in Section 5.

2 Background

Throughout the paper, Σ not containing ¢ and \$ denotes the input alphabet and $\tilde{\Sigma} = \Sigma \cup \{¢, \$\}$. For a given string x, $|x|$ is the length of x and x_i is the i^{th} symbol of x, where $1 \leq i \leq |x|$. The string $¢x\$$ is represented by $\tilde{x}$. Moreover, $<>$ is the set of $\{\leftarrow, \downarrow, \rightarrow\}$, Θ is the set of $\{0, \pm\}$, and $\Diamond$ is the set of $\{-1, 0, 1\}$. $\mathbf{P}(\cdot)$ denotes all subsets of a given set.

Each model defined in the paper has a two-way infinite read-only input tape whose squares are indexed by integers. Any given input string, say $x \in \Sigma^*$, is placed on the tape as $\tilde{x}$ between the squares indexed by 1 and $|\tilde{x}|$. The tape has a single head, and it can stay in the same position ($\downarrow$) or move to one square to the left ($\leftarrow$) or to the right ($\rightarrow$) in one step. It must always be guaranteed that the input head never leaves $\tilde{w}$. Some models in the paper have also a counter, an infinite storage having two status, i.e. *zero* (0) or *nonzero* ($\pm$), and being updated by a value from $\Diamond$ in one step. We assume that the reader familar with the modes of language recognition with errors (see also Appendix A).

2.1 Classical models

A two-way deterministic one-counter automaton (2dca) is a two-way deterministic finite automaton with a counter. Formally, a 2dca $\mathcal{D}$ is a 6-tuple $\mathcal{D} = (S, \Sigma, \delta, s_1, s_a, s_r)$, where S is the set of states, $s_1 \in Q$ is the initial state, $s_a \in S$ and $s_r \in S$ $(s_a \neq s_r)$ are the accepting and rejecting states, respectively, and δ is the transition function governing the behaviour of $\mathcal{D}$ in each step, i.e. $\delta : S \times \tilde{\Sigma} \times \Theta \rightarrow S \times \Leftrightarrow \times \Diamond$. Specifically, $\delta(s, \sigma, \theta) \rightarrow (s', d_i, c)$ means that when $\mathcal{D}$ is in state $s \in S$, reads symbol $\sigma \in \tilde{\Sigma}$, and the status of its counter is $\theta \in \Theta$, then it updates its state to $s' \in S$, the position of the input head with respect to $d_i \in \Leftrightarrow$, and the value of the counter by $c \in \Diamond$.

At the beginning of the computation, $\mathcal{D}$ is in state s_1, the input head is placed on symbol ¢, and the value of the counter is set to zero. A configuration of $\mathcal{D}$ on a given input string is represented by a triple (s, i, v), where s is the state, i is the position of the input head, and v is the value of the counter. The computation is terminated and the input is accepted (rejected) by $\mathcal{D}$ when it enters to s_a (s_r). The class of languages recognized by 2dca's is denoted $\mathsf{2DCA}$.

We will use *the same terminology* also for the other models unless otherwise is specified. A two-way nondeterministic one-counter automaton (2nca), say $\mathcal{N}$, is a 2dca having capability of making nondeterministic choices in each step. The transition function of $\mathcal{N}$ is extended as follows: $\delta : S \times \tilde{\Sigma} \times \Theta \rightarrow \mathbf{P}(S \times \Leftrightarrow \times \Diamond)$. In other words, for each triple (s, σ, θ) (see above), there may be more than one transition. Thus, $\mathcal{N}$ can follow more than one path during the computation, and if any path ends with a decision of acceptance, then the input is accepted. The class of languages recognized by 2nca's is denoted $\mathsf{2NCA}$.

A two-way probabilistic one-counter automaton (2pca), say $\mathcal{P}$, is a 2dca having capability of making probabilistic choices in each step. In order to explicitly represent the probabilistic part the machine, each step is divided into two transitions. Formally, $\delta = (\delta_p, \delta_d)$. For each triple (s, σ, θ) (see above), there are some predefined outcomes, i.e. $\Delta_{(s,\sigma,\theta)} = \{1, \ldots, k_{(s,\sigma,\theta)}\}$. Each outcome is selected with some (rational) probability: $\delta_p(s, \sigma, \theta, \tau) = p_\tau \in \mathbb{Q}$, where $\tau \in \Delta$ and $\sum_{\tau \in \Delta} p_\tau = 1$. After observing the outcome (τ), the deterministic transition is implemented as follows: $\delta_d(s, \sigma, \theta) \xrightarrow{\tau} (s', d_i, c)$ (see above). Note that δ_d must be defined for each possible τ. The class of languages recognized by 2pca's with bounded error is denoted $\mathsf{2PCA}$. If we remove the counter, then we obtain a 2pfa (two-way probabilistic finite automaton). If the input head of a 2pca is not allowed to move to left, then we obtain a one-way probabilistic one-counter automaton (1pca).

A one-way deterministic two-counter automaton (1d2ca) is a one-way deterministic finite automaton with two counters. It was shown that any deterministic Turing machine (DTM) can be simulated by a 1d2ca[15]. We denote the class of decidable languages $\mathsf{DECIDABLE}$.

2.2 Quantum models

A two-way finite state automaton with quantum and classical states [1, 26] (2qcfa) is a 2pfa using a finite quantum register instead of a classical random generator. Note that the quantum register can keep some information by its (pure) quantum state as well as making probabilistic choices.[1]

Formally, a 2qcfa[2] $\mathcal{Q}$ is a 8 tuple $(S, Q, \Sigma, \delta, s_1, s_a, s_r, q_1)$, where, apart from a classical model, there are two different components: Q is the state set of quantum register and q_1 is its initial state. Similar to probabilistic models, $\delta = (\delta_q, \delta_d)$, where δ_q governing the quantum part. In each

[1]It was shown that 2qcfa's are more powerful than 2pfa's by Ambainis and Watrous [1]. Moreover, Yakaryılmaz and Say [24, 25] showed that they can also recognize many interesting languages.

[2]Although the formal definition of 2qcfa given here is a bit different than the ones given in [1, 26], all the models are equivalent.

step, firstly, δ_q determines a superoperator (see Figure 1 for the details) depending on the current classical state ($s \in S$) and scanning symbol ($\sigma \in \tilde{\Sigma}$), i.e. $\mathcal{E}_{s,\sigma}$, and then it is applied to the quantum register and one outcome, say τ, is observed. Secondly, the classical part of $\mathcal{Q}$ is updated depending on s, σ, and τ, which is formally represented as $\delta_c(s,\sigma) \xrightarrow{\tau} (s', d_i)$, where $s' \in S$ is the new classical state and $d_i \in <>$ is the update of the position of input tape. Note that δ_d must be defined for each possible τ.

The most general quantum operator is a superoperator, which generalizes stochastic and unitary operators and also includes measurement. Formally, a superoperator $\mathcal{E}$ is composed by a finite number of operation elements, $\mathcal{E} = \{E_1, \ldots, E_k\}$, satisfying that

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

where $k \in \mathbb{Z}^+$ and the indices are the measurement outcomes. When a superoperator, say $\mathcal{E}$, is applied to the quantum register in state $|\psi\rangle$, i.e. $\mathcal{E}(|\psi\rangle)$, we obtain the measurement outcome i with probability $p_i = \langle\widetilde{\psi_i}|\widetilde{\psi_i}\rangle$, where $|\widetilde{\psi_i}\rangle$, *the unconditional state vector*, is calculated as $|\widetilde{\psi_i}\rangle = E_i|\psi\rangle$ and $1 \leq i \leq k$. (Note that using unconditional state vector simplifies calculations in many cases.) 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, as a special operator, the quantum register can be initialized to a predefined quantum state. This initialize operator, which has only one outcome, is denoted $\acute{\mathcal{E}}$. In this paper, the entries of quantum operators are defined by rational numbers. Thus the probabilities of the outcomes are always rational numbers.

Figure 1: The details of superoperators [22]

A two-way one-counter automaton with quantum and classical states (2qcca) is a 2pca using a finite quantum register instead of a classical random generator. The formal definition of a 2qcca is exactly the same as a 2qcfa. In fact, a 2qcca is a 2qcfa with a classical counter. So, the transition functions of a 2qcfa (δ_q and δ_c) can be extended for a 2qcca with the following modifications:

- The superoperator is determined by also the status of the counter ($\theta \in \Theta$), i.e. $\mathcal{E}_{s,\sigma,\theta}$.
- The classical part of $\mathcal{Q}$ is updated depending on s, σ, θ, and τ, which is formally represented as $\delta_c(s,\sigma,\theta) \xrightarrow{\tau} (s', d_i, c)$, where $s' \in S$ is the new classical state, d_i is the update of the position of input tape, and $c \in \Diamond$ is update on the counter.

A generalization of 2qcca is a two-way quantum counter automaton (2qca) that uses a quantum counter instead of a classical one. (Note that this model is still not the most general one, but it is sufficiently general for our purpose.) We can see 2qca as the combination of a 2qcfa and a realtime quantum one-counter automaton (rt-qca) [21]: The 2qcfa part governs the computation, and access the counter through the rt-qca by feeding some input to it and also observing the outcomes. We will use this model in one of our results (Theorem 3), and our simple definition will also simplify the proof.

2.3 Interactive proof systems

In this part, we provide the necessary background, based on [7, 3], for the proof systems. An interactive proof system (IPS) consists of a prover (P) and a verifier (V). The verifier is a restricted/resource-bounded machine. The classical states of the verifier are partitioned into reading, communication, and halting (accepting or rejecting) states, and it has a special communication cell for communicating with the prover, where the capacity of the cell is finite.

The one-step transitions of the verifier can be described as follows. When in a reading state, the verifier implements its standard transition. When in a communication symbol, the verifiers writes a symbol on the communication cell with respect to the current state. Then, in response, the prover writes a symbol in the cell. Based on the state and the symbol written by prover, the verifier defines the next state of the verifier. Note that the communication is always classical even though the verifier can use some quantum memory.

The prover P is specified by a prover transition function, which determines the response of the prover to the verifier based on the input and the verifier's communication history until then. Note that this function does not need to be *computable*.

The prover-verifier pair (P, V) is an IPS for language $\mathtt{L}$ with error probability $\epsilon < \frac{1}{2}$ if (i) for all $x \in \mathtt{L}$, the probability that (P, V) accepts x is greater than $1 - \epsilon$, (ii) for all $x \notin \mathtt{L}$, and all provers P^*, the probability that (P^*, V) rejects x is greater than $1 - \epsilon$. These conditions are known as completeness and soundness, respectively.

An Arthur-Merlin (AM) proof system is a special case of IPS such that after each probabilistic or quantum operation, the outcome is automatically written on the communication cell, and so the prover can have complete information about the computation of the verifier.[3] We also refer them as public proof systems.

$\mathsf{IP(v)}$ represents the class of languages having an IPS with some error probability $\epsilon < \frac{1}{2}$, where the verifier is v-type. Moreover, $\mathsf{IP^*(v)}$ is a subset of $\mathsf{IP(v)}$ providing that each language in $\mathsf{IP^*(v)}$ has an IPS for any error bound. $\mathsf{AM(v)}$ and $\mathsf{AM^*(v)}$ are defined similarly.

3 Counter automata verifiers for decidable languages

In this section, we will present four different protocols for decidable languages. We begin with the classical verifiers.

Theorem 1. $\mathsf{IP^*(2pca)} = \mathsf{DECIDABLE}$.

Proof. The relation $\mathsf{IP(2pca)} \subseteq \mathsf{DECIDABLE}$ is trivial. We will give the proof for the other direction. The proof idea is inspired from the protocol given by Condon and Lipton [4].

Let $\mathtt{L}$ be a decidable language. Then there exists a 1d2ca $\mathcal{D}$, which halts on every input, recognizing $\mathtt{L}$ [16]. Any configuration of $\mathcal{D}$ on an input, say $x \in \Sigma^*$, can be represented by (s, i, u, v), where s is the state, i is the head position, and u and v are contents of the counters.

We will describe an IPS (P, V) for $\mathtt{L}$ by giving a simulation of $\mathcal{D}$ on the given input, say x, where V is a $2pca$. If V accesses the status of both counters in each step, then it can easily simulate $\mathcal{D}$ on x by tracing the state and the head position updates of $\mathcal{D}$. The prover can provide the contents of the counters for each step. But, the verifier should be careful about the cheating provers. For this purpose, V can use its counter. That is, in each step, the verifier can determine the changes on the counters, and so can compare the current value and the next value of a counter. Therefore, before starting the simulation, V equiprobably selects a counter of $\mathcal{D}$ to test the changes on it. Moreover, V should also compare the contents of the selected counter not only for $(2i-1)^{th}$ and $(2i)^{th}$ steps but also for $(2i)^{th}$ and $(2i+1)^{th}$ steps, where $i \geq 1$. Thus, V can start the comparisons from either the first step or the second step, which can also be decided equiprobably at the beginning of the simulation. Therefore, we can identify four comparison strategies, i.e. C_i^j (V selects the i^{th} counter of $\mathcal{D}$ and starts to compare from the step-j), where $1 \leq i, j \leq 2$. This is the base strategy of V. However, as described below, it is not sufficient to define a protocol for any error bound.

The simulation of $\mathcal{D}$ on x by (P, V) is executed in an infinite loop. In each round, a new simulation is started. V requests the contents of the counters for each step from the prover. Let w be the string obtained from the prover in a single round. The verifier expects w as

[3]Note that all the verifiers defined in the paper are allowed to use only rational number transitions.

$a^{u_1}b^{v_1}\#a^{u_2}b^{v_2}\#\cdots\#a^{u_t}b^{v_t}\#$ such that u_j (v_j) is the content of the first (the second) counter after j^{th} step, where $1 \leq j \leq t$ and $t \geq 1$. On the other hand, there are four disjoint cases for w as listed below:

- (C1) w is of the form $\boxed{(a^*b^*\#)^+}$,
- (C2) there is an a after b in w,
- (C3) w is infinite and of the form $\boxed{(a^*b^*\#)^+aaa\cdots}$ or $\boxed{(a^*b^*\#)^+a^*bbb\cdots}$, or
- (C4) w is infinite and of the form $\boxed{(a^*b^*\#)(a^*b^*\#)(a^*b^*\#)\cdots}$.

It is obvious that V can check C2 deterministically, and reject the input if there exists an a after b in w. In other words, such a round is certainly terminated with the decision of rejection. Therefore, in the remaining part, we assume that w satisfies one of the other cases.

If w is valid (correct), then V can exactly simulate $\mathcal{D}$ on x. Otherwise, the simulation may contain some defects, and so V may give a wrong decision. Moreover, V may also enter an infinite loop. Note that since P is honest and provides the valid w, we specifically focus on the strategies of cheating provers on the nonmembers: In each round, V should deal with infinite loops and should also guarantee that, for the nonmembers, the probability of accepting the input, which can only be given based on the simulation, is sufficiently smaller than the probability of rejecting the input due to detecting the defects on w. The aforementioned (base) strategy of V is quite strong, and so any invalid w is detected by at least one of C_i^j. But the prover can still mislead the verifier in the other choices. Thus, the defect can be detected with a probability at least $\frac{1}{4}$, and the verifier can follow an invalid w with a probability at most $\frac{3}{4}$. Therefore, when V is convinced to accept the input, it gives the decision of acceptance with probability $\frac{1}{k}$, and terminates the current round with the remaining probability $1-\frac{1}{k}$. So, the total accepting probability of an invalid computation ($\frac{3}{4k}$) can be sufficiently small compared to the rejecting probability due to the defect ($\frac{1}{4}$) by setting k to an appropriate value. However, there is still the problem of infinite loop. We can solve this problem by terminating the round with probability $\frac{1}{2}$ after obtaining a symbol w from the prover. Thus, any infinite loop can be terminated with probability 1. Although the probability of making decisions is dramatically decreased due to this new restart strategy, the ratio of accepting and rejecting probabilities for the nonmembers can still be preserved since any decision of acceptance can only be given after a defect.

Now, we can analyse the overall protocols. Let l be the length of the valid w. If $x \in \mathtt{L}$, then V accepts x with probability $\frac{1}{k2^l}$ in each round, and so it is accepted exactly. If $x \notin \mathtt{L}$, if there is no defect, then it is rejected with probability $\frac{1}{2^l}$ in a single round. If there is a defect, than it is detected by V after obtaining $(l_1)^{th}$ symbol of w, where $l_1 \leq l$. Moreover, the input can be accepted by V after obtaining $l_2 \geq l_1$ symbol of w. Then, the input is rejected with a probability at least $\frac{1}{42^{l_1}}$, and it is accepted with a probability at most $\frac{3}{4k2^{l_2}}$. Thus, the input is rejected with high probability depending on the value of k. Moreover, the protocol is always terminated with probability 1. □

In the protocol above, if we allow the infinite loops, we can still obtain an IPS for any decidable language by using the base strategy. Besides, it is sufficient to simulate $\mathcal{D}$ once. Thus, the verifier does not need to move its input head to the left.

Corollary 1. $\mathsf{IP(1pca)} = \mathsf{DECIDABLE}$.

Proof. The input is rejected with probability $\frac{3}{7}$ by the verifier at the beginning of the computation. Then the verifier follows its base strategy once. Therefore, the members are accepted with probability $\frac{4}{7}$ by the help of a honest prover, and the non-members are rejected with a probability at least $\frac{3}{7} + \frac{4}{7}\left(\frac{1}{4}\right) = \frac{4}{7}$. The error bound is $\frac{3}{7} < \frac{1}{2}$. □

We continue with the quantum verifiers. Recently, Yakaryılmaz [22] showed that for each Turing-recognizable language, say $\mathtt{L}$, there exists an AM proof systems with a 2qcfa verifier, say (P, V), such that each $x \in \mathtt{L}$ is accepted by V exactly and each $x \notin \mathtt{L}$ is accepted with a small probability. Such proof systems are also known as weak-IPS [7, 3].

By combining the protocol given in [22] with the first protocol given above (given in the proof of Theorem 1), we present two more protocols for decidable languages. A review of the protocol given in [22] is as follows. Let $\mathtt{L}$ be a decidable language, and $\mathcal{D}$ be a DTM, which halts on every input, recognizing $\mathtt{L}$. In this protocol (P, V) simulates the computation of $\mathcal{D}$ on a given input, say x. In an infinite loop, V requests the computation of $\mathcal{D}$ on x, as $w = c_1\$\$c_2\$\$c_3\$\$\cdots$, where $c_{i>0}$'s are some configurations of $\mathcal{D}$ on x and c_1 is the initial one. If $x \in \mathtt{L}$, P provides the valid w, and V accepts the input with some probability in each round, then it is accepted exactly. If $x \notin \mathtt{L}$, then the input is always rejected with a bigger probability than the accepting probability in a single round as long as the prover sends $\$\$$ symbols. If the prover does not send $\$\$$ after some point, the round is still terminated with probability 1, but probably with no decision. This is why the system is "weak". From a given configuration, the length of the next valid configuration can be easily determined, which can be differ at most one. So, if the verifier uses a classical counter, it can detect when the prover does not send $\$\$$ with some probability, i.e. instead of terminating the round with "no decision", the round is terminated with some nonzero "rejecting" probability. Similar to the first protocol given above, the verifier equiprobably decides to compare the lengths of which configurations, i.e. $(2i-1)^{th}$ and $(2i)^{th}$ configurations or $(2i)^{th}$ and $(2i+1)^{th}$ configurations, at the beginning of each round, where $i \geq 1$. Although the protocol given in [22] is a public one, the computations on the classical counter must be hidden from the prover in the new protocol. We can formalize this result as follows.

Theorem 2. $\mathsf{IP}^*(\mathsf{2qcca}) = \mathsf{DECIDABLE}$.

It is a well-known fact that the simulation of a DTM by a 1d2ca is space (and time) inefficient [14]. Therefore, we can say that for the same language, the latter protocol can be more space efficient than the former protocol for the members of the language since the latter protocol directly simulates a DTM. This can be seen as an advantage of using a few quantum states.

Corollary 2. *For any language recognized by a $s(n)$-space DTM, there exists an IPS with a 2qcca verifier such that the verifier uses $s(n)$-space on its counter for the members.*

Our fourth result is to make the third protocol (given for 2qcca) *public*. This can be achieved by replacing the classical counter with a quantum counter. The private part of the the latter protocol is to hide the probabilistic choice at the beginning of each round, based on which the verifier decides the lengths of which configurations will be compared. Since a quantum memory can be in superposition of more than one classical configuration, a 2qca can parallelly implement both choices in a public manner.

Theorem 3. $\mathsf{AM}^*(\mathsf{2qca}) = \mathsf{DECIDABLE}$.

Proof. The protocol is exactly the same as the third protocol except the counter operations. Therefore, we explain only this part. As mentioned in Section 2, we can see the verifier as the combination of a 2qcfa and a rt-qca. Remember that the verifier requests $w = c_1\$\$c_2\$\$c_3\$\$\cdots$ from the prover in each round. So, as long as getting w, the 2qcfa part can feed the following sequence $u = a^{|c_1|}\#i_1\#a^{|c_2|}\#i_2\#a^{|c_3|}\#i_3\#\cdots$ to rt-qca part, where i_j represents the expected change in the length of c_{j+1} based on c_j and $j > 0$. It is not hard to show that a rt-qca can check the equalities $|c_{j+1}| = |c_j| + i_j$ for each $j > 0$, and can also detect the case of $|c_{j+1}| > |c_j| + i_j$ with some probability. $\square$

4 Counter machines as recognizer

In this section, we examine the bounded-error computational powers of 2pca's and 2qcca's as language recognizers. We begin with a useful lemma and a lower bound to 2PCA.

Lemma 1. *Let $\mathcal{N} = (S, \Sigma, \delta, s_1, s_a, s_r)$ be a 2nca, x be an input, and $M = |S||\tilde{x}|$. If $s \in S$ is reachable from s_1 by $\mathcal{N}$ on x, then there is a path of length no more than M^2 from $(s_1, 1, 0)$ to (s, i, u) for some $1 \leq i \leq |\tilde{x}|$ and $u \leq M$ such that the value of counter never exceeds M.*

Proof. Let $path(s_1, s)$ be a path from $(s_1, 1, 0)$ to (s, i, u) for some $1 \leq i \leq |\tilde{x}|$ and $u \geq 0$. We can assume that there is no two configurations $(s', i', u_1 > 0)$ and $(s', i', u_2 > u_1)$ in $path(s_1, s)$ such that the latter one comes after the first one and the counter is never set to zero in between. Let T be the set of $S \times \{1, \ldots, |\tilde{x}|\}$. (Note that $|T| = M$.) Consider a subpath of $path(s_1, s)$, say $subpath(s_1, s)$, such that it starts with a configuration in which the value of the counter is zero and there is no further such a configuration in this subpath. Let c_j be the configuration in which the counter reaches the value $j > 0$ for the first time in $subpath(s_1, s)$. Each such c must have a different $t \in T$ value. Otherwise, our assumption would be violated. So, there can be at most $|T| = M$ such c's. That is, the counter value never exceeds M in $path(s_1, s)$, and so the length of the path can be at most M^2 by also assuming that there is no two identical configurations in the path. $\square$

Theorem 4. *Let* L *be a language recognized by a 2nca $\mathcal{N}$, then there exists a 2pca $\mathcal{P}$ recognizing* L *with one-sided bounded-error.*

Proof. Let x be an input string. We begin with constructing a 2pca, say $\mathcal{P}_1$ based on $\mathcal{N}$. Each nondeterministic transition of $\mathcal{N}$ is replaced with a probabilistic one: If l is the number of nondeterministic choices, then each probabilistic choice is made with probability $\frac{1}{l}$ in $\mathcal{P}_1$. Let k be the maximum number of nondeterministic choices in a single step of $\mathcal{N}$. Due to Lemma 1, we can say that, if $x \in \mathtt{L}$, then $\mathcal{P}_1$ accepts the input with a probability at least $\left(\frac{1}{k}\right)^{c|\tilde{x}|^2}$ for a suitable constant $c > 1$. As a further modification, $\mathcal{P}_1$ restarts the computation instead of rejecting the input. So, $\mathcal{P}_1$ can halt and accept the input with some probability if $x \in \mathtt{L}$, and $\mathcal{P}_1$ never halts if $x \notin \mathtt{L}$.

By using $\mathcal{P}_1$, we construct another 2pca, say $\mathcal{P}_2$. At the beginning of the computation, $\mathcal{P}_2$ simulates $\mathcal{P}_1$ with probability $\frac{3}{4}$, and executes a rejecting procedure with probability $\frac{1}{4}$, in which $\mathcal{P}_2$ rejects the input exactly with probability $\left(\frac{1}{k}\right)^{c|\tilde{x}|^2}$ and restarts the computation with the remaining probability. So, if $x \notin \mathtt{L}$, then the input is never accepted, and, if $x \in \mathtt{L}$, the accepting probability is always at least 3 times bigger than the rejecting probability.

If $\mathcal{N}$ always halts in each path, then $\mathcal{P}_2$ recognizes L with one-sided error bound $\frac{1}{4}$. But, if $\mathcal{N}$ does not halt in each branch, then $\mathcal{P}_2$ also enters an infinite loop in some paths. To handle this problem, we make another modification. Based on $\mathcal{P}_2$, we construct $\mathcal{P}$ as follows: In each step, $\mathcal{P}$ restarts the computation with probability $\frac{1}{2}$, and simulates $\mathcal{P}_2$ with probability $\frac{1}{2}$. Thus, each path of $\mathcal{P}_2$ can terminate with probability 1, and so L is recognized by $\mathcal{P}$ with one-sided error bound $\frac{1}{4}$. The error bound can be reduced to any desired value by using probability amplification techniques. $\square$

Remark that if a language is recognized by a 2nca, then it is recognized by a 2pca with one-sided unbounded error, vice versa. Therefore, in case of one-sided error, the language recognition power of 2pca's remain the same.

Corollary 3. *A language is recognized by a 2nca if and only if it is recognized by a 2pca with one-sided bounded-error.*

Moreover, due to the fact that $\mathsf{2DCA} \subsetneq \mathsf{2NCA}$ [2] and Theorem 4, we can say that bounded-error 2pca's are more powerful than 2dca's.

Corollary 4. $\mathsf{2DCA} \subsetneq \mathsf{2NCA} \subseteq \mathsf{2PCA}$.

Now, we turn our attention to the language recognition power of bounded-error 2qcca's. We begin with the definitions of two languages: $\mathtt{TWIN} = \{u\#u \mid u \in \{a,b\}^*\}$ and $\mathtt{EXIST\text{-}TWIN} = \{u\#v_1\#\cdots\#v_k \mid k \geq 1, u \in \{a,b\}^*, v_i \in \{a,b\}^* (1 \leq i \leq k), \text{and } \exists i \in \{1,\ldots,k\}(u = v_i)\}$.

Ďuriš and Galil [5, 6] showed that EXIST-TWIN cannot be recognized by any 2dca. Moreover, Chrobak stated [2] that EXIST-TWIN does not seem to be in $\mathsf{2NCA}$. We show that 2qcca's can recognize EXIST-TWIN for any error bound by using a new technique which calls a 2qcfa's as a black box. In the next section, we will also show that this programming technique can also be used by 2qcfa's having a pebble.

Theorem 5. EXIST-TWIN *can be recognized by a 2qcca $\mathcal{Q}$ with bounded error.*

Proof. Recently, Yakaryılmaz and Say [24, 25] showed that TWIN can be recognized by any 2qcfa for any negative one-sided error bound. Let $\mathcal{Q}_{\mathtt{TWIN}}$ be such a 2qcfa for error bound $\frac{1}{5}$. (We also refer the reader to Appendix B for the details of $\mathcal{Q}_{\mathtt{TWIN}}$.) We will use $\mathcal{Q}_{\mathtt{TWIN}}$ as a black box. As a special remark, $\mathcal{Q}_{\mathtt{TWIN}}$ reads the input from left to right in an infinite loop.

Let x be an input. We assume that x is of the form $u\#v_1\#\cdots\#v_k$ for some $k \geq 1$, where $u, v_i \in \{a,b\}^*$ $(1 \leq i \leq k)$. Otherwise, it is deterministically rejected. The idea behind the algorithm is that $\mathcal{Q}$ selects v_1, and then simulates $\mathcal{Q}_{\mathtt{TWIN}}$ by feeding $u\#v_1$ as the input. $\mathcal{Q}_{\mathtt{TWIN}}$ gives the decision of rejection only if $u \neq v_1$. So, whenever $\mathcal{Q}_{\mathtt{TWIN}}$ gives the decision of rejection, then $\mathcal{Q}$ continues by selecting v_2, and so on. We call each such selection, in which $\mathcal{Q}$ gives the decision of rejection, *completed.* If $x \notin \mathtt{EXIST\text{-}TWIN}$, then $\mathcal{Q}$ can obtain k completed-selection with some nonzero probability. Otherwise, this probability becomes zero since $\mathcal{Q}$ can obtain at most $(k-1)$ completed-selection. So, by accepting the input with some carefully tuned probability, we can obtain the desired machine. Note that $\mathcal{Q}$ always remembers its selection by using its counter. The pseudocode of the algorithm is given below.

```
FOR $i = 1$ TO $k$ ($v_i$ is selected)
    RUN $\mathcal{Q}_{\mathtt{TWIN}}$ on $x' = u\#v_i$
        IF $\mathcal{Q}_{\mathtt{TWIN}}$ $\underline{\text{accepts}}$ $x'$ THEN **TERMINATE** FOR-LOOP
        IF $\mathcal{Q}_{\mathtt{TWIN}}$ $\underline{\text{rejects}}$ $x'$ AND $i = k$ THEN **REJECT** the input
END FOR
**ACCEPT** $x$ with probability $\left(\frac{1}{5}\right)^k$
**RESTART** the algorithm
```

As can be seen from the pseudocode, the algorithm is actually executed in an infinite loop. We begin with analysing a single round of the algorithm. It is straightforward that if $x \in \mathtt{EXIST\text{-}TWIN}$, the input is accepted with probability $\left(\frac{1}{5}\right)^k$, and it is rejected with zero probability. If $x \notin \mathtt{EXIST\text{-}TWIN}$, $\mathcal{Q}_{\mathtt{TWIN}}$ halts with the decision of rejection with a probability at least $\frac{4}{5}$ in each iteration of the for-loop, and so the input is rejected with a probability at least $\left(\frac{4}{5}\right)^k$, and it is accepted with a probability no more than $\left(\frac{1}{5}\right)^k$. Therefore, the members are accepted exactly. Since the rejecting probability is at least 4^k times bigger than the accepting probability in a single round, the input is rejected with a probability at least $\frac{4}{5}$. The error bound can be easily reduced to any desired value. □

Note that since PTM's cannot recognize `TWIN` in sublogarithmic space [23], we cannot use the same idea for 2pca's. In fact, we believe that $\texttt{EXIST-TWIN} \notin \mathsf{2PCA}$.

Another interesting language is a unary one: $\texttt{USQUARE} = \{b^{n^2} \mid n \geq 1\}$. It is still not known whether `USQUARE` can be recognized by 2dca's [17, 18]. Since 2qcfa's can recognize $\texttt{SQUARE} = \{a^n b^{n^2} \mid n \geq 1\}$ with negative one-sided bounded error [25], we can also obtain the following result.

Theorem 6. `USQUARE` *can be recognized by a 2qcca with bounded error.*

Proof. Let $\mathcal{Q}_{\texttt{SQUARE}}$ be a 2qcfa recognizing `SQUARE` with error bound $\frac{1}{3}$. A 2qcca can iteratively $(i = 1, \ldots, |x|)$ splits the input, say x, as $b^i b^{|x|-i}$ by using its counter, and then can feed $x' = a^i b^{|x|}$ to $\mathcal{Q}_{\texttt{SQUARE}}$. The pseudocode of the algorithm is given below.

```
FOR $i = 1$ TO $|x|$
    RUN $\mathcal{Q}_{\texttt{SQUARE}}$ on $x' = a^i b^{|x|}$
        IF $\mathcal{Q}_{\texttt{SQUARE}}$ $\underline{\text{accepts}}$ $x'$ THEN TERMINATE FOR-LOOP
        IF $\mathcal{Q}_{\texttt{SQUARE}}$ $\underline{\text{rejects}}$ $x'$ AND $i = |x|$ THEN REJECT the input
END FOR
ACCEPT $x$ with probability $\left(\frac{1}{3}\right)^{2|x|}$
RESTART the algorithm
```

All the remaining details including the analysis of the algorithm is similar the algorithm given in the proof of Theorem 5. □

5 Pebble automata

A 1-pebble finite automaton has the capability of placing a pebble to at most one tape square, of sensing whether a tape square has a pebble or not, and removing the pebble from the marked tape square. The algorithms given for 2qcca's in the previous section can also be implemented by 1-pebble 2qcfa. In these algorithms, the counter is actually used to remember some positions on the input. A pebble can also be used in the same way by marking those positions. Therefore, we can conclude the following corollaries.

Corollary 5. `EXIST-TWIN` *and* `USQUARE` *can be recognized by some 1-pebble 2qcfa's with bounded error.*

By using the same idea, we can also show that $\texttt{SIAM-TWINS} = \{uu \mid u \in \{a,b\}^*\}$ can also be recognized by 1-pebble 2qcfa's. This is an interesting language since `SIAM-TWINS` cannot be recognized by any 1-pebble NTM using sublogarithmic space [12].

Theorem 7. `EXIST-TWIN` *can be recognized by a 1-pebble 2qcfa* $\mathcal{Q}$ *with bounded error.*

Proof. Let x be a given input. If x is not of the form ax_1ax_2 or bx_1bx_2, then it is rejected immediately, where $x_1, x_2 \in \{a,b\}^*$. Assume that $x = ax_1ax_2$. (The other case is the same). We will use the 2qcca algorithm given for `EXIST-TWIN` after some modifications. We give the the pseudocode of the algorithm.

```
BEGIN (OUTER-)LOOP
    TRY to MARK the next $a$ on $x$
        IF there is no such $a$, THEN **REJECT** $x$
        ELSE $x = ax_1a'x_2$ ($a'$ is the marked one)
    BEGIN (INNER-)LOOP
        RUN $\mathcal{Q}_{\texttt{TWIN}}$ on $x' = x_1\#x_2$
            IF $\mathcal{Q}_{\texttt{TWIN}}$ $\underline{\text{accepts}}$ $x'$ THEN **TERMINATE** (OUTER-)LOOP
            IF $\mathcal{Q}_{\texttt{TWIN}}$ $\overline{\underline{\text{rejects}}}$ $x'$ THEN **TERMINATE** (INNER-)LOOP
    END (INNER-)$\overline{\text{LOOP}}$
END (OUTER-)LOOP
**ACCEPT** $x$ with probability $\left(\frac{1}{5}\right)^{|x|}$
**RESTART** the algorithm
```

All the remaining details of the algorithm are similar to the previous ones. □

Ravikumar [20] showed that 1-pebble 2pfa's are more powerful than 2pfa's in the unbounded error case by giving an unbounded-error 1-pebble 2pfa for nonstochatic language $\texttt{CENTER} = \{ubv \mid u, v \in \{a,b\}^* \text{ and } |u| = |v|\}$. We show that the same separation for the bounded-error machines as conjectured by Ravikumar [19]. (Note that bounded-error 1-pebble 2qcfa's can recognize $\texttt{CENTER}$. However, we do not know such a 2qcfa for nonstochastic language $\texttt{SAY} = \{x \mid \exists x_1, x_2, y_1, y_2 \in \{a,b\}^*, x = x_1bx_2 = y_1by_2, |x_1| = |y_2|\}$ [10].)

Theorem 8. *1-pebble 2pfa's are more powerful than 2pfa's in the bounded error case.*

Proof. Lapinš [13] showed that language $\texttt{LAPINŠ} = \{a^m b^n c^p \mid m^4 > n^2 > p > 0\}$ is a nonstochastic, i.e. not recognized by any bounded-error 2pfa. It is not hard to show that $\texttt{LAPINŠ}$ can be recognized by a 1-pebble bounded-error 2pfa if there exists a bounded-error 1-pebble 2pfa for $\texttt{GREATER-SQUARE} = \{a^m b^n \mid m > n^2 > 0\}$. Therefore, it is sufficient to show that there exists a bounded-error 1-pebble 2pfa, say $\mathcal{P}$, for $\texttt{GREATER-SQUARE}$.

Let x be an input string of the form $a^m b^n$ $(m, n > 0)$. By using its pebble, $\mathcal{P}$ can feed $a^m b^{n^2}$, i.e. it can n-times read b^n, to any bounded-error 2pfa. Since language $\texttt{GREATER} = \{a^m b^n \mid m > n > 0\}$ can be recognized by 2pfa's with bounded error [9, 19], $\mathcal{P}$ can recognize $\texttt{GREATER-SQUARE}$ with bounded error. □

Acknowledgements. We would like to thank A. C. Cem Say not only for his useful comments on a draft of this paper but also for many helpful discussions on the subject matter of this paper. We would like to thank to Holger Petersen for kindly answering our questions and giving an idea used in the proof of Lemma 1.

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] Marek Chrobak. Nondeterminism is essential for two-way counter machines. In *Proceedings of the Mathematical Foundations of Computer Science 1984*, pages 240–244, 1984.

[3] Anne Condon. *Complexity Theory: Current Research*, chapter The complexity of space bounded interactive proof systems, pages 147–190. Cambridge University Press, 1993.

[4] Anne Condon and Richard J. Lipton. On the complexity of space bounded interactive proofs (extended abstract). In *FOCS'89: Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 462–467, 1989.

[5] Pavol Ďuriš and Zvi Galil. Fooling a two-way automaton or one pushdown store is better than one counter for two way machines (preliminary version). In *STOC'81: Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pages 177–188, 1981.

[6] Pavol Ďuriš and Zvi Galil. Fooling a two way automaton or one pushdown store is better than one counter for two way machines. *Theoretical Computer Science*, 21:39–53, 1982.

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

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

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

[10] Rūsiņš Freivalds, Abuzer Yakaryılmaz, and A. C. Cem Say. A new family of nonstochastic languages. *Information Processing Letters*, 110(10):410–413, 2010.

[11] Juraj Hromkovic and Georg Schnitger. On the power of randomized multicounter machines. *Theoretical Computer Science*, 330(1):135–144, 2005.

[12] Atsuyuki Inoue, Akira Ito, Katsushi Inoue, and Tokio Okazaki. Some properties of one-pebble Turing machines with sublogarithmic space. *Theoretical Computer Science*, 341(1-3):138–149, 2005.

[13] Jānis Lapiņš. On nonstochastic languages obtained as the union and intersection of stochastic languages. *Avtom. Vychisl. Tekh.*, (4):6–13, 1974. (Russian).

[14] Peter van Emde Boas. *Handbook of Theoretical Computer Science (vol. A)*, chapter Machine models and simulations, pages 1–66. 1990.

[15] Marvin Minsky. Recursive unsolvability of post's problem of "tag" and other topics in theory of Turing machines. *Annals of Mathematics*, 74(3):437–455, 1961.

[16] Marvin Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, 1967.

[17] Holger Petersen. Two-way one-counter automata accepting bounded languages. *SIGACT News*, 25(3):102–105, 1994.

[18] Holger Petersen. Private communication, June 2012.

[19] Bala Ravikumar. Some observations on 2-way probabilistic finite automata. In *FSTTCS'92: Proceedings of the 12th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 392–403, 1992.

[20] Bala Ravikumar. On some variations of two-way probabilistic finite automata models. *Theoretical Computer Science*, 376(1-2):127–136, 2007.

[21] A. C. Cem Say and Abuzer Yakaryılmaz. Quantum counter automata. *International Journal of Foundations of Computer Science*, (to appear).

[22] Abuzer Yakaryılmaz. Turing-equivalent automata using a fixed-size quantum memory. Technical Report arXiv:1205.5395, 2012.

[23] Abuzer Yakaryılmaz, Rūsiņš Freivalds, A. C. Cem Say, and Ruben Agadzanyan. Quantum computation with write-only memory. *Natural Computing*, 11(1):81–94, 2012.

[24] Abuzer Yakaryılmaz and A. C. Cem Say. Languages recognized by nondeterministic quantum finite automata. *Quantum Information and Computation*, 10(9&10):747–770, 2010.

[25] Abuzer Yakaryılmaz and A. C. Cem Say. Succinctness of two-way probabilistic and quantum finite automata. *Discrete Mathematics and Theoretical Computer Science*, 12(2):19–40, 2010.

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

[27] Tomohiro Yamasaki, Hirotada Kobayashi, and Hiroshi Imai. Quantum versus deterministic counter automata. *Theoretical Computer Science*, 334(1-3):275–297, 2005.

A The modes of language recognition with errors

A language L is said to be recognized by a machine $\mathcal{M}$ with error bound $\epsilon < \frac{1}{2}$, if $\mathcal{M}$ accepts each member of L with a probability at least $1 - \epsilon$, and $\mathcal{M}$ rejects each non-member of L with a probability at least $1 - \epsilon$. A language L is said to be recognized by a machine $\mathcal{M}$ with bounded error if it is recognized by $\mathcal{M}$ with an error bound.

A language L is said to be recognized by a machine $\mathcal{M}$ with (positive) one-sided error bound $\epsilon < 1$, if $\mathcal{M}$ accepts each member of L with a probability at least $1 - \epsilon$, and $\mathcal{M}$ rejects each non-member of L with probability 1. A language L is said to be recognized by a machine $\mathcal{M}$ with (positive) one-sided bounded error if it is recognized by $\mathcal{M}$ with a positive one-sided error bound.

A language L is said to be recognized by a machine $\mathcal{M}$ with negative one-sided error bound $\epsilon < 1$, if $\mathcal{M}$ accepts each member of L with probability 1, and $\mathcal{M}$ rejects each non-member of L with a probability at least $1 - \epsilon$. A language L is said to be recognized by a machine $\mathcal{M}$ with negative one-sided bounded error if it is recognized by $\mathcal{M}$ with a negative one-sided error bound.

A language L is said to be recognized by a machine $\mathcal{M}$ with unbounded-error if $\mathcal{M}$ accepts each member of L with a probability bigger than $\frac{1}{2}$, and $\mathcal{M}$ rejects each non-member of L with a probability at most $\frac{1}{2}$.

A language L is said to be recognized by a machine $\mathcal{M}$ with one-sided unbounded-error if $\mathcal{M}$ accepts each member of L with some nonzero probability, and $\mathcal{M}$ rejects each non-member of L with probability 1.

B A 2qcfa for TWIN language

In this section, we show that there exists a 2qcfa, say $\mathcal{Q}_{\texttt{TWIN}}$, recognizing TWIN with negative one-sided error bound $\frac{1}{5}$. Note that this error bound can be easily be reduced to any desired value by using probability amplification techniques.

Let $x \in \{a, b, \#\}^*$ be an input. If x does not contain exactly one #, then it is deterministically rejected. So, we assume that $x = u_1\#u_2$ in the following part, where $u_1, u_2 \in \{a, b\}^*$.

The quantum register of $\mathcal{Q}_{\texttt{TWIN}}$ has three states, i.e. q_1, q_2, and q_3. In an infinite loop, $\mathcal{Q}$ reads the input from left to right with a speed of one symbol per step. We call each iteration *a round*. During scanning the input, $\mathcal{Q}_{\texttt{TWIN}}$ is trying to encode u_1 and u_2 into the amplitudes of q_1 and q_2, respectively. Since the encoding techniques requires to reduce the amplitudes with a constant, the current round is terminated, and then a new round is initiated with some probability after reading each symbol. If $\mathcal{Q}$ succeeds to reach \$, then the amplitudes of q_1 and q_2 are subtracted, based on which the input is rejected, and the input is always accepted with a small probability. So, if $u_1 = u_2$, the input is only accepted in each round. Otherwise, the input is both accepted and rejected. By tuning the accepting probability sufficiently small than the minimum rejecting probability, we can obtain the desired machine. The technical details are given below.

We encode the strings in base-2 and use 0 for a's and 1 for b's. Since it can be easily understandable from the context, we will use string representation also for their encodings. The strings $0^{j_1}1u$ and $0^{j_2}1u$ can be different, but, their encodings are the same, where j_1 and j_2 are nonnegative integers. Therefore, we encode $1u_1$ and $1u_2$ instead of u_1 and u_2. Note that $u_1 = u_2$ if and only if $1u_1 = 1u_2$.

At the beginning of each round, the quantum register is set to

$$|\psi_0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Until reading \$ symbol, the round is terminated unless the outcome 1 is observed. In order to simplify the calculations, we trace the quantum part by unconditional state vectors.

After reading ¢, the following superoperator is applied:

$$\mathcal{E}_{¢} = \left\{ E_1 = \frac{1}{3}\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad E_2 = \frac{1}{3}\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{pmatrix}, \quad E_3 = \frac{1}{3}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix} \right\}.$$

Then, the (unconditional) state vector becomes

$$|\widetilde{\psi_1}\rangle = \frac{1}{3}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Thus, the first symbol of both $1u_1$ and $1u_2$ have been encoded.

Until reading symbol #, the remaining part of $1u_1$ is encoded into the amplitude of q_1 by using the following two superoperators. The first (second) one is applied after reading an a (a b).

$$\mathcal{E}_a = \left\{ E_1 = \frac{1}{3}\begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad E_2 = \frac{1}{3}\begin{pmatrix} 2 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad E_3 = \frac{1}{3}\begin{pmatrix} 0 & 2 & 2 \\ 0 & 2 & -2 \\ 0 & 0 & 0 \end{pmatrix} \right\}.$$

$$\mathcal{E}_b = \left\{ E_1 = \frac{1}{3}\begin{pmatrix} 2 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad E_2 = \frac{1}{3}\begin{pmatrix} -1 & 0 & 2 \\ 2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad E_3 = \frac{1}{3}\begin{pmatrix} 0 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & 0 & 0 \end{pmatrix} \right\}.$$

Then, the (unconditional) state vector becomes

$$|\widetilde{\psi_1}\rangle = \left(\frac{1}{3}\right)^{|¢u_1|}\begin{pmatrix} 1u_1 \\ 1 \\ 1 \end{pmatrix}.$$

After reading #, the following superoperator is applied:

$$\mathcal{E}_{\#} = \left\{ E_1 = \frac{1}{3}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad E_2 = \frac{1}{3}\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad E_3 = \frac{1}{3}\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} \right\}.$$

Then, the (unconditional) state vector becomes

$$|\widetilde{\psi_{|¢u_1\#|}}\rangle = \left(\frac{1}{3}\right)^{|¢u_1\#|}\begin{pmatrix} 1u_1 \\ 1 \\ 1 \end{pmatrix}.$$

Until reading symbol \$, the remaining part of $1u_2$ is encoded into the amplitude of q_2 by using the following two superoperators. The first (second) one is applied after reading an a (a b).

$$\mathcal{E}'_a = \left\{ E_1 = \frac{1}{3}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad E_2 = \frac{1}{3}\begin{pmatrix} 2 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 2 & 0 \end{pmatrix}, \quad E_3 = \frac{1}{3}\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right\}.$$

$$\mathcal{E}'_b = \left\{ E_1 = \frac{1}{3}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad E_2 = \frac{1}{3}\begin{pmatrix} 0 & 1 & -2 \\ 2 & 0 & 1 \\ -2 & 0 & 1 \end{pmatrix}, \quad E_3 = \frac{1}{3}\begin{pmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right\}.$$

Then, the (unconditional) state vector becomes

$$|\psi_{\widetilde{|¢u_1\#u_2|}}\rangle = \left(\frac{1}{3}\right)^{|¢u_1\#u_2|}\left(\begin{array}{c} 1u_1 \\ 1u_2 \\ 1 \end{array}\right).$$

After reading \$, the decision on the input is given by applying the following superoperatos. If outcome 1 is observed, then the input is rejected, if output 2 is observed, then the input is accepted, and a new round is initiated, otherwise.

$$\mathcal{E}_{\$} = \left\{ E_1 = \frac{1}{3}\left(\begin{array}{ccc} 2 & -2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right), E_2 = \frac{1}{3}\left(\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{array}\right), E_3 = \frac{1}{3}\left(\begin{array}{ccc} 2 & 2 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right), E_4 = \frac{1}{3}\left(\begin{array}{ccc} 0 & 0 & 2 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{array}\right) \right\}.$$

If output 1 is observed, then the (unconditional) state vector becomes

$$\left(\frac{1}{3}\right)^{|\tilde{x}|}\left(\begin{array}{c} 2(1u_1 - 1u_2) \\ 0 \\ 0 \end{array}\right).$$

That is, if $u_1 = u_2$, then the input is rejected with zero probability, and if $u_1 \neq u_2$, the input is rejected with probability

$$4\left(\frac{1}{3}\right)^{2|\tilde{x}|}(1u_1 - 1u_2),$$

which can be at least

$$4\left(\frac{1}{3}\right)^{2|\tilde{x}|}.$$

If output 2 is observed, then the (unconditional) state vector becomes

$$\left(\frac{1}{3}\right)^{|\tilde{x}|}\left(\begin{array}{c} 0 \\ 0 \\ 1 \end{array}\right).$$

That is, the input is always rejected with probability

$$\left(\frac{1}{3}\right)^{2|\tilde{x}|},$$

which is 4 times smaller than the minimum nonzero rejecting probability.

So, if $x \in \texttt{TWIN}$, then it is accepted with probability 1, and if $x \notin \texttt{TWIN}$, then it is accepted with a probability at most $\frac{1}{5}$, and rejected with a probability at least $\frac{4}{5}$.