Quantum Logarithmic Space and Post-Selection

François Le Gall ✉
Graduate School of Mathematics, Nagoya University, Japan

Harumichi Nishimura ✉ ⓘ
Graduate School of Informatics, Nagoya University, Japan

Abuzer Yakaryılmaz[1] ✉ 🏠 ⓘ
Center for Quantum Computer Science, University of Latvia, Rīga, Latvia
QWorld Association, Tallinn, Estonia

───── **Abstract** ─────

Post-selection, the power of discarding all runs of a computation in which an undesirable event occurs, is an influential concept introduced to the field of quantum complexity theory by Aaronson (Proceedings of the Royal Society A, 2005). In the present paper, we initiate the study of post-selection for space-bounded quantum complexity classes. Our main result shows the identity $\mathsf{PostBQL} = \mathsf{PL}$, i.e., the class of problems that can be solved by a bounded-error (polynomial-time) logarithmic-space quantum algorithm with post-selection ($\mathsf{PostBQL}$) is equal to the class of problems that can be solved by unbounded-error logarithmic-space classical algorithms (PL). This result gives a space-bounded version of the well-known result $\mathsf{PostBQP} = \mathsf{PP}$ proved by Aaronson for polynomial-time quantum computation. As a by-product, we also show that PL coincides with the class of problems that can be solved by bounded-error logarithmic-space quantum algorithms that have no time bound.

1 Introduction

Post-selection. Post-selection is the power of discarding all runs of a computation in which an undesirable event occurs. This concept was introduced to the field of quantum complexity theory by Aaronson [1]. While unrealistic, post-selection turned out to be an extremely useful tool to obtain new and simpler proofs of major results about classical computation, and also prove new results about quantum complexity classes. The most celebrated result is arguably the identity $\mathsf{PostBQP} = \mathsf{PP}$ proved by Aaronson [1], which shows that the class of

---

[1] Part of this research was done while Yakaryılmaz was visiting Kyoto University in November 2016 and March 2017.

problems that can be solved by a bounded-error polynomial-time quantum algorithm with post-selection (PostBQP) is equal to the class of problems that can be solved by unbounded-error polynomial-time classical algorithms (PP), and thus makes possible to bridge quantum complexity classes and classical complexity classes.

Space-bounded quantum complexity classes.   The study of space-bounded quantum Turing machines was initiated by Watrous [16]. Watrous showed in particular that any quantum Turing machine running in space s can be simulated by an unbounded-error probabilistic Turing machine running in space $O(s)$. This result implies the identity PQL = PL, where PQL denotes the class of problems that can be solved by unbounded-error logarithmic-space quantum Turing machines, and PL denotes the class of problems that can be solved by unbounded-error logarithmic-space classical Turing machines. The main open question of the field is whether bounded-error quantum Turing machines can be simulated space-efficiently by *bounded-error* classical Turing machines.

A major step towards establishing the superiority of space-bounded quantum Turing machines over space-bounded classical (bounded-error) Turing machines has been the construction by Ta-Shma [14] of logarithmic-space quantum algorithms for inverting well-conditioned matrices (it is unknown how to perform the same task classically in logarithmic space). While Ta-Shma's quantum algorithm used intermediate measurements, a version of this quantum algorithm without measurement was later constructed by Fefferman and Lin [6] (see also [5] for a related result on space-efficient error reduction for unitary quantum computation). Very recent works [7, 8] have further showed that many other problems from linear algebra involving well-conditioned matrices can be solved as well in logarithmic space by quantum algorithms, and additionally showed that intermediate measurements can be removed from any space-bounded quantum computation.

Our results.   In view of the impact of the concept of post-selection to quantum complexity theory and in view of the surge of recent activities on space-bounded quantum complexity classes, a natural question is investigating the power of post-selection for space-bounded quantum complexity classes. To our knowledge, this question has not been investigated so far in the literature (while the notion of post-selection was previously studied in quantum automata theory [21]). In this paper, we tackle this question and obtain the following result (here PostBQL denotes the class of problems that can be solved by a bounded-error polynomial-time logarithmic-space quantum Turing machine that uses post-selection – see Section 2 for a formal definition):

▶ **Theorem 1** (Main Theorem). PostBQL = PL.

This result thus gives a space-bounded version of the result PostBQP = PP mentioned above for polynomial-time complexity classes. This enables us to bridge quantum complexity classes and classical complexity classes for space-bounded computation as well, and thus suggests that post-selection may become a useful tool to analyze space-bounded (quantum and classical) computation as well. Actually, as a by-product of our main result, we also obtain the fact that PL coincides with the class of problems that can be solved by bounded-error logarithmic-space quantum algorithms that has no time bound (namely, the bounded-error logarithmic-space quantum algorithms are as computationally powerful as the unbounded-error ones under no time restriction).

We additionally present several results about logarithmic-space quantum computation with post-selection in Section 4.

Overview of our techniques. As for the result $\mathsf{PostBQP} = \mathsf{PP}$ proved by Aaronson [1], the nontrivial part of the proof of our main theorem is the simulation of a probabilistic machine by a post-selecting quantum simulation machine. The simulation technique given in [1] requires a polynomial amount of qubits, and thus cannot be used in our setting since we are limited to a logarithmic amount of qubits. Therefore, we propose a different simulation, which is composed of three parts. First, we show how to simulate the computation of a logarithmic-space probabilistic Turing machine by a logarithmic-width probabilistic circuit K (Section 3.1). Note that the computation process of K is represented by a mixture $\sum_j p_j C_j$, which means that the configuration is in C_j with probability p_j. (It can be written as $\sum_j p_j |C_j\rangle\langle C_j|$ when the mixed state formalism [11] is used.) Here, we can assume that there are unique accepting and rejecting configurations C_a and C_r. Thus, the final mixture of K can be represented in the form of $pC_a + (1-p)C_r$, where $p > 1/2$ if the input is a yes-instance, and $p < 1/2$ if it is a no-instance. Second, we give a simulation of the probabilistic circuit K by a logarithmic-space quantum Turing machine M with post-selection (Section 3.2). Note that this simulation is done in a *coherent* manner. Namely, if the mixture of K at some step is $\sum_j p_j C_j$, the quantum state of M at the corresponding simulation step should be the normalized state of $\sum_j p_j |C_j\rangle$. Thus, M produces the normalized state of $|\psi\rangle = p|C_a\rangle + (1-p)|C_r\rangle$ as the final outcome. In fact, we use the power of post-selection for this simulation, and the final outcome can be obtained after post-selection with an exponentially small probability. Then, the third part is fairly similar to the approach used in [1]: using polynomial number of states constructed from the same number of copies of $|\psi\rangle$, we use repetition and post-selection to increase the success probability of the simulation (Section 3.3).

2 Preliminaries

2.1 Space-bounded probabilistic Turing machines

A classical space-bounded Turing machine has an input tape and a work tape. Both tapes are infinite and their cells are indexed by integers, each of which contains the blank symbol (#) unless it is overwritten with a different symbol. The input tape has a read-only head and the work tape has a read/write head. Each head can access a single cell in each time step and, after each transition, it can stay on the same cell, move one cell to the right, or move one cell to the left.

The input alphabet is denoted Σ and the work tape alphabet is denoted Γ, none of which contains the blank symbol. Moreover, $\tilde{\Sigma} = \Sigma \cup \{\#\}$ and $\tilde{\Gamma} = \Gamma \cup \{\#\}$. For a given string x, $|x|$ represents the length of x.

Formally, a (space-bounded) probabilistic Turing machine (PTM) M is a 7-tuple

$$M = (S, \Sigma, \Gamma, \delta, s_i, s_a, s_r),$$

where S is the set of states, $s_i \in S$ 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 described below.

At the beginning of the computation, the given input, say $x \in \Sigma^*$, is placed on the input tape between the first cell and the $|x|$-th cell, the input tape head and the work tape head are placed on the cells indexed by 0s, and the state is set to s_i. In each step, M evolves with respect to the transition function and the computation is terminated after entering s_a or s_r. In the former (latter) case, the decision of "acceptance" ("rejection") is made. It must be guaranteed that the input tape head never visits the cells indexed by -1 and $|x| + 2$. The

formal definition of δ is as follows:

$$\delta : S \times \tilde{\Sigma} \times \tilde{\Gamma} \times S \times \tilde{\Gamma} \times \{-1, 0, 1\} \times \{-1, 0, 1\} \to \left\{0, \frac{1}{2}, 1\right\}.$$

Suppose that M is in $s \in S$ and reads $\sigma \in \tilde{\Sigma}$ and $\gamma \in \tilde{\Gamma}$ on the input and work tapes, respectively. Then, in one step, the new state is set to $s' \in S$, the symbol $\gamma' \in \tilde{\Gamma}$ is written on the cell under the work tape head, and the positions of the input and work tape heads are respectively updated with respect to $d_i \in \{-1, 0, 1\}$ and $d_w \in \{-1, 0, 1\}$, with probability

$$\delta(s, \sigma, \gamma, s', \gamma', d_i, d_w),$$

where the input (work) tape head moves one cell to the left if $d_i = -1$ ($d_w = -1$) and one cell to the right if $d_i = 1$ ($d_w = 1$). Remark that any transition with zero probability is never implemented. To be a well-formed PTM, for each triple (s, σ, γ),

$$\sum_{s' \in S, \gamma' \in \tilde{\Gamma}, d_i \in \{-1,0,1\}, d_w \in \{-1,0,1\}} \delta(s, \sigma, \gamma, s', \gamma', d_i, d_w) = 1.$$

For a given input $x \in \Sigma^*$, M can follow more than one computation path. A computation path either halts with a decision or runs forever. A halting path is called accepting (rejecting) if the decision of "acceptance" ("rejection") is made on this path. The accepting (rejecting) probability of M on x is the cumulative sum over all accepting (rejecting) paths.

A language L is said to be recognized by PTM M with unbounded error if and only if any $x \in L$ is accepted by M with probability more than $1/2$ and any $x \notin L$ is accepted with probability less than $1/2$. A language L is said to be recognized by PTM M with error bound $\varepsilon < 1/2$ if and only if any $x \in L$ is accepted by M with probability at least $1 - \varepsilon$ and any $x \notin L$ is rejected with probability at least $1 - \varepsilon$. When $\varepsilon > 0$ is a constant (independent of the input), it is said that L is recognized by M with bounded error. As a special case, if all non-members of L are accepted with probability 0, then it is called one-sided bounded-error. A PTM making only deterministic transitions (i.e., such that the range of the transition function is $\{0, 1\}$) is a deterministic Turing machine (DTM).

The range of the transition function can also be defined as $[0, 1] \cap \mathbb{Q}$, and thus the PTM, called rational valued PTM, can make more than one transition with rational valued probabilities in each step. Remark that all results presented in this paper are also followed for rational valued PTMs. A nondeterministic Turing machine (NTM) can be defined as a rational valued PTM and a language is said to be recognized by a NTM if and only if for any member there is at least one accepting path and for any non-member there is no accepting path (or equivalently any member is accepted with nonzero probability and any non-member is accepted with zero probability).

A language is recognized by a machine in (expected) time $t(n)$ and space $s(n)$ if the machine, on a given input x, runs no more than (expected) $t(|x|)$ time steps and visits no more than $s(|x|)$ different cells on its work tape with non-zero probability.

The class PL (L and NL) is the set of languages recognized by unbounded-error PTMs (DTMs and NTMs) in logarithmic space (with no time restriction). It is shown that each of these classes coincides with the subclass such that the running time of the corresponding machines is polynomially bounded (note that the proof is nontrivial for PL [10]).

The class BPL (RL) is the set of languages recognized by bounded-error PTMs (one-sided bounded-error PTMs) *in polynomial time* and logarithmic space. On contrary to the above three classes PL, L, NL, it is unknown that these two classes are the same as their corresponding classes such that the underlying machines have no time restriction, which we denote by BPL(∞) (RL(∞)).

Any language L is in $\mathsf{C_=L}$ [2] if and only if there exists a polynomial-time logarithmic-space PTM M such that any $x \in L$ is accepted by M with probability $\frac{1}{2}$ and any $x \notin L$ is accepted by M with probability other than $\frac{1}{2}$.

2.2   Turing machines with post-selection

A postselecting PTM (PostPTM) has the ability to discard some predetermined outcomes and then makes its decision with the rest of the outcomes, which is guaranteed to happen with non-zero probability (see [1, 21]). Formally, a PostPTM is a modified PTM with three halting states. A PTM has the accepting state s_a and the rejecting state s_r as the halting states. A PostPTM has an additional halting state s_n called the non-postselecting halting state. In this paper, we require that a PostPTM must halt its computation absolutely, i.e., there is no infinite loop.

For a given input x, let $p_{acc,M}(x)$ ($p_{rej,M}(x)$ and $p_{npost,M}(x)$) be the probability of PostPTM M ending in s_a (s_r and s_n). Since M halts absolutely, we know that

$$p_{acc,M}(x) + p_{rej,M}(x) + p_{npost,M}(x) = 1.$$

Due to post-selection, we discard the probability $p_{npost,M}(x)$ and then make a normalization on $p_{acc,M}(x)$ and $p_{rej,M}(x)$ for the final decision. Thus, the input x is accepted (rejected) by M with probability

$$\tilde{p}_{acc,M}(x) := \frac{p_{acc,M}(x)}{p_{acc,M}(x) + p_{rej,M}(x)} \quad \left(\tilde{p}_{rej,M}(x) := \frac{p_{rej,M}(x)}{p_{acc,M}(x) + p_{rej,M}(x)} \right).$$

The postselecting counterparts of BPL and RL are $\mathsf{PostBPL}$ and PostRL, respectively. (For instance, L is in $\mathsf{PostBPL}$ if and only if there are a polynomial-time logarithmic-space PostPTM M and a constant $\varepsilon < 1/2$ such that $\tilde{p}_{acc,M}(x)$ is at least $1 - \varepsilon$ when x is in L, and $\tilde{p}_{rej,M}(x)$ is at least $1 - \varepsilon$ when x is not in L). Let $\mathsf{PostEPL}$ denote the class of languages recognized with no error (or exactly) by polynomial-time logarithmic-space PostPTMs (i.e., L is in $\mathsf{PostEPL}$ if and only if there is a polynomial-time logarithmic-space PostPTM M such that $p_{acc,M}(x) > 0$ and $p_{rej,M}(x) = 0$ when x is in L, and $p_{acc,M}(x) = 0$ and $p_{rej,M}(x) > 0$ when x is not in L).

2.3   Space-bounded quantum Turing machines and complexity classes

The initial quantum Turing machine (QTM) models (e.g., [4, 3, 16]) were defined fully quantum. While quantum circuits have been used more widely in literature, QTMs are still the main computational models when investigating space bounded complexity classes. However, their definitions have been modified since 90s (e.g., [17, 15, 14, 6]). The main modifications are that the computation is governed classically and the quantum part can be seen like a quantum circuit. This paper follows these modifications. To be more precise, our QTM is a PTM augmented with a quantum tape. Here, the quantum tape is designed like a quantum circuit, i.e., it contains a qubit (or qudit) in each tape cell and it can have more than one tape head so that a quantum gate can be applied to a few qubits at the same time.

We remark that the result given in this paper can also be obtained by the other space-bounded QTMs defined in literature [18, 17, 20, 15, 14], where algebraic numbers are used as transition values. The main advantage of the aforementioned modifications in QTMs is to simplify the proofs and the descriptions of quantum algorithms.

Formally, a (space-bounded) QTM M is a 9-tuple

$$M = (S, \Sigma, \Gamma, \delta_q, \delta_c, s_i, s_a, s_r, \Omega),$$

where, different from the PTMs, the transition function is composed by two functions δ_q and δ_c that are responsible for the transitions on quantum and classical parts, respectively, and Ω is the set of contents of a classical register storing quantum measurement outcomes. (Similarly to the PTMs, S is the set of internal states, Σ is the input alphabet, Γ is the work tape alphabet, and s_i, s_a, and s_r are respectively the initial state, the accepting state, and the rejecting state.) As the physical structure, M additionally has a quantum tape with l heads, and the classical register storing a value in $\Omega = \{1, \ldots, m\}$, where $l, m > 0$ are constants (independent of the input given to M). The quantum tape heads are numbered from 1 to l. For simplicity, we assume that the quantum tape contains only qubits (with states $|0\rangle$ and $|1\rangle$) in its cells. Each cell is set to $|0\rangle$ at the beginning of the computation. For a given input $x \in \Sigma^*$, the classical part is initialized as described for PTMs. The l tape heads on the quantum tape are placed on the qubits numbered from 0 to $l - 1$.

The overall computation of M is governed classically. Each transition of M has two phases, quantum and classical, which alternate. We define the transition functions δ_q and δ_c different from the transition functions of PTMs. Suppose that M is in $s \in S$ and reads $\sigma \in \tilde{\Sigma}$ and $\gamma \in \tilde{\Gamma}$, respectively. For each triple (s, σ, γ), $\delta_q(s, \sigma, \gamma)$ can be the identity operator, a projective measurement (in the computational basis), or a unitary operator. If it is the identity operator, the quantum phase is skipped by setting the value of the classical register to 1 (in Ω). If the quantum operator is unitary, then the corresponding unitary operator is applied to the qubits under the heads on the quantum tape, and the value in the classical register is set to 1 (in Ω). If it is a measurement operator, then the corresponding projective measurement is done on the qubits under the heads on the quantum tape, and the measurement outcome, represented by an integer between 1 and $m' \leq m$ (in Ω), is written in the classical register, where m' is the total number of all possible measurement outcomes of the measurement operator.

After the quantum phase, the classical phase is implemented. For each quadruple $(s, \sigma, \gamma, \omega)$, δ_c returns the new state, the symbol written on the work tape, and updates of all heads, where $\omega \in \Omega$.

The termination of the computation of M is the same as the PTMs, i.e., done by entering the accepting state s_a or the rejecting state s_r. One time step corresponds to a single transition. We add the number of qubits visited with non-zero probability during the computation (as well as the number of cells visited on the classical work tape) to the space usage.

Remark that any QTM using superoperators can be simulated by a QTM using unitary operators and measurements with negligible memory and time overheads, i.e., by using extra quantum and classical states, any superoperator can be implemented by unitary operators and measurements in constant steps (e.g. [11, 13]).

Since the computation of the QTM defined above is controlled classically, a postselecting QTM (PostQTM) can be defined similar to PostPTMs: the PostQTM has an additional classical halting state s_n, and any computation that ends in s_n is discarded when calculating the overall accepting and rejecting probability on the given input.

The quantum counterparts of BPL (BPL(∞)), RL, PL, NL, PostBPL, PostRL, and PostEPL are BQL (BQL(∞)), RQL, PQL, NQL[2], PostBQL, PostRQL, and PostEQL, respectively, where QTMs use algebraic numbers as transition amplitudes.

The following relations on logarithmic space quantum and classical complexity classes

---

[2] Note that NQL is the quantum counterpart of NL based on the criterion by the accepting probabilities of the underlying machine, not the certificate-based counterpart (QMAL).

are already known [9, 16, 17, 7]:

$$\mathsf{L} \subseteq \mathsf{NL} = \mathsf{coNL} \subseteq \mathsf{coC_{=}L} = \mathsf{NQL} \subseteq \mathsf{PL} = \mathsf{PQL}.$$

$$\mathsf{L} \subseteq \mathsf{BPL} \subseteq \mathsf{BQL} \subseteq \mathsf{PL} = \mathsf{PQL}.$$

3 Main Result

In this section, our main theorem ($\mathsf{PostBQL} = \mathsf{PL}$) is proved. We start with the easy inclusion.

▶ **Theorem 2.** $\mathsf{PostBQL} \subseteq \mathsf{PQL} = \mathsf{PL}$

Proof. Any polynomial-time logarithmic-space PostQTM M can be easily converted to a polynomial-time logarithmic-space QTM M' such that M' enters the accepting and rejecting states with equal probability when M enters the non-postselecting halting state. Thus the balance between accepting and rejecting probabilities is preserved, and the language recognized by M with bounded-error is recognized by M' with unbounded error.   ◀

In the rest of this section, we give the proof of the following inclusion.

▶ **Theorem 3.** $\mathsf{PL} \subseteq \mathsf{PostBQL}$.

As described in Section 1, the proof of Theorem 3 consists of three parts, each of which will be given in the next three subsections. We start by giving an overview of the first part. Let L be a language in PL. Then there exists a PTM M recognizing L with unbounded error such that M on input x halts in $|x|^k$ steps by using at most $d \log(|x|)$ space for some fixed positive integers d and k.

Without loss of generality, we can assume that M always splits into two paths in every step, the work tape alphabet of M has only two symbols 0 and 1, and M halts only when the work tape contains only blanks and both tape heads are placed on the 0-th cells, i.e., there exist a single accepting and a single rejecting configurations. Let m be the number of internal states.

We fix x as the given input with length $|x| = n$. Any configuration of M is represented by a 4-tuple of binary strings

$$(s, h_{\mathrm{in}}, w, h_{\mathrm{wk}}),$$

where s is the internal state, h_{in} is the position of the input head, w is the content of the work tape, and h_{wk} is the position of the work tape. (We also assume that w is always a binary string, which does not contain any blank symbol.) The set of all configurations is denoted by C^x, i.e., $C^x = \{C_1, \ldots, C_N\}$ for some N polynomial in n. The length of any configuration is

$$l = \lceil \log m \rceil + \lceil \log n \rceil + \lceil d \log n \rceil + \lceil \log(d \log n) \rceil \in O(\log n).$$

Based on C^x, we define a stochastic matrix P_x, called *the configuration matrix*, whose columns and rows are indexed by configurations and its (j, i)-th entry represents the probability going from C_i to C_j. Then, the whole computation of M on x can be traced by an N-dimensional column vector, called *configuration vector*:

$$v_{l+1} = P_x v_l,$$

where $1 \leq l \leq n^k$ and v_l represents the probability distribution of the configurations after the l-th step. Here, v_0 is the initial configuration vector having a single nonzero entry, that is

1, corresponding to the initial configuration, and v_{n^k} is the final configuration vector having at most two nonzero entries that keep the overall accepting and rejecting probabilities:

$$v_{n^k} = P_x^{n^k} v_0.$$

Since the computation is split into two paths in each step with equal probability, the overall accepting (A) and rejecting probabilities (R) are respectively of the forms

$$\frac{A'}{2^{n^k}} \text{ and } \frac{R'}{2^{n^k}},$$

where $0 \le A', R' \le 2^{n^k}$, $A' + R' = 2^{n^k}$, and $A' \ne 2^{n^k-1}$.

We present a simulation of the above matrix-vector multiplication in logarithmic space. It is clear that keeping all entries of a single configuration vector separately requires polynomial space in n. On the other hand, a single configuration can be kept in logarithmic space. Therefore, we keep a mixture of configurations as a single summation for any time step. In other words, we can keep v_i as

$$v_i[1]C_1 + v_i[2]C_2 + \cdots + v_i[n^k]C_{n^k},$$

where each coefficient $v_i[j]$ represents the probability of being in the corresponding configuration C_j. The transition from v_i to v_{i+1} can be obtained in a single step by applying P_x. However, in our simulation, we can do this in n^k sub-steps. The idea is as follows: In the j-th sub-step, we check whether our mixture has C_j or not. If it exists, then C_j is evolved to C_j' and C_j'' that are the configurations obtained from C_j in a single step when the outcome of the coin is respectively heads or tails. In this way, from the mixture corresponding to v_i, we obtain the next mixture:

$$v_{i+1}[1]C_1 + v_{i+1}[2]C_2 + \cdots + v_{i+1}[n^k]C_{n^k}.$$

Then, the final mixture is

$$AC_a + RC_r,$$

where C_a and C_r are the accepting and rejecting configurations, respectively.

We present the details of this simulation in the following subsection.

3.1   Probabilistic circuit

In this subsection, it is shown that we can construct, in deterministic logarithmic space, a logarithmic-width and polynomial-depth probabilistic circuit $K_{M,x}$ that simulates M on x.

Note that a logarithmic-space DTM can easily output each element of C^x. Moreover, for any $C_j \in C^x$, it can also easily output two possible next configurations C_j' and C_j'' such that M switches from C_j to C_j' if the result of the coin flip is heads and it switches from C_j to C_j'' if the result of the coin flip is tails.

A logarithmic-space DTM D described below can output the desired probabilistic circuit $K_{M,x}$ with width $l+3$ where (i) the first bit is named as *the random bit* that is used for coin flip, (ii) the second and third bits are named as *the block control bit* and *the configuration control bit* that are used to control the transition between the configurations in each time step, and (iii) the rest of the bits hold a configuration of M on x.

The circuit $K_{M,x}$ consists of n^k *blocks*, and D outputs the n^k blocks. Each block corresponds to a single time step of M on x:

$$block_1, block_2, \ldots, block_{n^k},$$

where each block is identical, i.e., each block implements the transition matrix P_x operating on configurations. Remark that, after $block_i$, we have the mixture representing v_i.

Before each block, the random bit is set to 0 or 1 with equal probability and the block control bit is set to 1. As long as the block control bit is 1, the configurations are checked one by one in the block. Once it is set to 0, the remaining configurations are skipped.

Any block is composed by N *parts* where each part corresponds to a single configuration:

$$part_1, part_2, \ldots, part_N.$$

Here, $part_j$ implements the transitions from C_j in a single step. In $part_j$, we do the following items:

1. If *the block control bit* is 0, then SKIP the remaining items. Otherwise, CONTINUE.
2. SET *the configuration control bit* to 1 (here we assume that M is in C_j).
3. It checks whether M is in C_j.
   - If M is not in C_j, SET *the configuration control bit* to 0 and SKIP the remaining items. (Remark that the block control bit is still 1 in this case, and thus the next configuration C_{j+1} will be checked in $part_{j+1}$.)
   - Otherwise (i.e., if M indeed is in C_j), CONTINUE.
4. SWITCH from C_j to C'_j if *the random bit* is 0 and SWITCH from C_j to C''_j if *the random bit* is 1.
5. SET *the block control bit* to 0.

After all n^k blocks, D outputs the last block called *decision block*. In the last block, it is checked whether the last configuration is C_a or C_r. If it is C_a (resp. C_r), then the first bit of the decision block is set to 1 (resp. 0).

For the above operations, we can use some gates operating on no more than four bits that are the first three bits and one bit from the rest in each time. With l sequential gates, we can determine whether we are in C_j or not. Similarly, with l sequential gates, we can implement the transition from C_j to C'_j and, with another l sequential gates, we can implement the transition from C_j to C''_j. Here, using l sequential gates allows us to keep the size of any gate no more than 4 bits as shown in Fig. 1 (where $H_1, \ldots, H_l$ denote the l sequential gates).
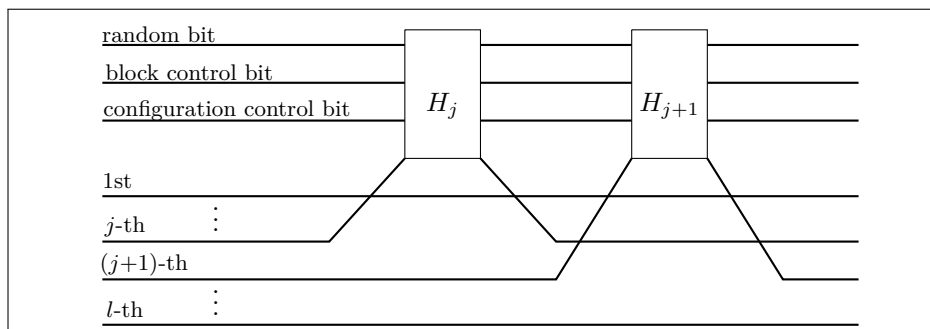
Figure 1 Two sequential gates operating on the first three bits and one bit from the rest.

When physically implementing the above circuit $K_{M,x}$, before each block, the circuit will be in a single configuration, and during executing the block, only the part corresponding to this configuration will be active, and thus the circuit will switch to one of the two possible next configurations. After *the decision block*, we will observe the first bit as 1 and 0 with probabilities A and R, respectively.

Remark that the set of all possible gates which can be used in the above circuit is finite and independent of the input x. The only probabilistic gate is a single bit operator implementing a fair coin toss. The rest of gates are deterministic and basically they are controlled operators with maximum dimension of 16.

Before continuing with the quantum part, we make further simplifications on $K_{M,x}$. As 2-bit AND and OR gates[3] and 1-bit NOT gate form a universal gate set for classical circuits, each deterministic gate (operating at most 4 bits) can be replaced by some finite numbers of NOT, AND, OR, and some 1-bit resetting gates with help of a few extra auxiliary bits used for intermediate calculations, which are appended to the bottom part of the circuit. Let $G = \{G_0, G_1, \ldots, G_t\}$ be the new set of our gates, where G_0 implements the fair coin by outputting the values 0 and 1 with equal probability, and the values are used by the deterministic gates whenever it is needed.

We denote the simplified circuit as $K'_{M,x}$ or shortly as K'. Let l' be the width of K' (note that $l' = l + O(1) = O(\log n)$). Thus, we have K' such that the probability of observing 1 (resp. 0) on the first bit is A (resp. R).

3.2   QTM part

In this subsection, we give a logarithmic-space postselecting QTM that simulates the computation of K' in a *coherent* manner, as described in Section 1.

A logarithmic-space (postselecting) QTM can trace the computation of K' on its quantum tape by help of its classical part. Since the circuit K' is deterministic logarithmic-space constructible, the classical part of the QTM helps to create the parts of K' on the quantum tape whenever it is needed. Moreover, any mixture of the configurations in K' is kept in a pure state of l' qubits (described below).

The QTM uses $l' + 2$ active qubits on the quantum tape for tracing K' on the input. The last two qubits are auxiliary, and the first l' qubits are used to keep the probabilistic state of K'. We consider the quantum tape as a logarithmic-width quantum circuit simulating K'.

For each gate of K', say G_j, we apply a unitary gate (operator) operating on at most 4 qubits, say U_j. Therefore, we use 4 tape heads on the quantum tape.

During the simulation, the first l' qubits are always kept in a superposition and after each unitary operator the last qubit or the last two qubits are always measured. If the outcome is 0 or 00, then the computation continues. Otherwise, the computation is terminated in the non-postselecting state.

In the probabilistic circuit K', G_0 is applied on the first qubit. For each G_0, we apply

$$U_0 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

on the first and the last qubits, measure the last qubit, and continue if $|0\rangle$ is observed:

$$U_0|00\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle \xrightarrow{\text{post-selection}} \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle.$$

$$U_0|10\rangle = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle \xrightarrow{\text{post-selection}} \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle.$$

---

[3] We assume that these gates are represented by 4×4 matrices.

Thus, coin-flipping operator can be easily implemented.

For the other operators (including the ones given below), we use the techniques given in [12]. For any G_j ($1 \leq j \leq t$), we apply unitary operator U_j acting on four qubits. Before applying U_j, the quantum part is in

$$\sum_{a,b\in\{0,1\}} \alpha_{a,b}|ab00\rangle,$$

since the last two qubits are measured before and any outcome other than $|00\rangle$ is discarded by entering the non-postselecting state. Thus, only $4 \times 4 = 16$ entries of U_j affects the above quantum state. We construct U_j step by step as follows. These 16 entries are set to the corresponding values from G_j. Thus, the probabilistic state, which is kept in the pure state, can be traced exactly up to some normalization factor.

Without loss of generality, we assume that (by reordering the quantum states) these 16 values are placed in the top left corner. Then, U_j is of the form

$$\frac{1}{e}\left(\begin{array}{c|c|c|c} G_j & G'_j & G''_j & 0 \\ \hline * & * & * & * \end{array}\right),$$

where e is the normalization factor and all G_j, G'_j, and G''_j are 4×4 matrices.

The entries of G'_j are set in order to make the first four rows pairwise orthogonal:

$$G'_j = \left(\begin{array}{cccc} 1 & 0 & 0 & 0 \\ \gamma_{1,2} & 1 & 0 & 0 \\ \gamma_{1,3} & \gamma_{2,3} & 1 & 0 \\ \gamma_{1,4} & \gamma_{2,4} & \gamma_{3,4} & 0 \end{array}\right),$$

where the values are set column by column. The values of $\gamma_{1,2}$, $\gamma_{1,3}$, and $\gamma_{1,4}$ are set to the appropriate values such that the first row becomes orthogonal to the second, the third, and the fourth ones, respectively. Similarly, we set the values of the second and third columns. Since G_j is composed by integers, G'_j is also composed by integers.

The entries of G''_j are set in order to make the first four rows with equal length, say e, which is a square of an integer:

$$G''_j = \left(\begin{array}{cccc} \gamma_1 & 0 & 0 & 0 \\ 0 & \gamma_2 & 0 & 0 \\ 0 & 0 & \gamma_3 & 0 \\ 0 & 0 & 0 & \gamma_4 \end{array}\right),$$

where diagonal entries are picked as the square roots of some integers. Remark that the entries of G''_j does not change the pair-wise orthogonality of the first four rows. Moreover, at this point, the first four rows become pair-wise orthonormal (due to normalization factor e). One can easily fill up the rest of the matrix with some arbitrary algebraic numbers in order to have a complete unitary matrix.

Since the set of G depends on the transitions of the PTM M, each U_j can be kept in the description of the QTM.

By using the above quantum operators, we can simulate K' with exponentially small probability. Only note that, due to normalization factors, the computation is terminated in the non-postselecting state with some probabilities after applying each unitary gate.

At the end of the simulation of K', we separate the first qubit from the rest of qubits, each of which is set to $|0\rangle$. Then, we have this unnormalized quantum state in the first qubit:

$$(1-A)|0\rangle + A|1\rangle = \left(\begin{array}{c} 1-A \\ A \end{array}\right).$$

The operator $\begin{pmatrix} 1/2 & 3/2 \\ 1/2 & -1/2 \end{pmatrix}$ maps the above quantum state to

$$|\tilde{u}\rangle = \begin{pmatrix} \frac{1}{2} + A \\ \frac{1}{2} - A \end{pmatrix}.$$

Since this operator can be also implemented with post-selection by using an extra qubit, the new unnormalized quantum state is set to $|\tilde{u}\rangle$.

If $A = 0$, then the quantum state $|u\rangle$, that is the normalized version of $|\tilde{u}\rangle$, is identical to $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. If $A < \frac{1}{2}$, then the quantum state $|u\rangle$ lies between $|+\rangle$ and $|0\rangle$, and thus it is closer to $|+\rangle$ compared to $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. If $A > \frac{1}{2}$, then $|u\rangle$ lies between $|0\rangle$ and $|-\rangle$, and thus it is closer to $|-\rangle$ compared to $|+\rangle$.

After making a measurement in $\{|+\rangle, |-\rangle\}$ basis, we can easily distinguish the cases whether A is close to 0 or A is close to 1 with bounded error. In the case of when A is close to $\frac{1}{2}$, the probability of observing these basis states can be very close to each other. In Section 3.3, we use a modified version of the trick used by Aaronson [1] to increase the success probability. Actually, we will need to use the above QTM $O(n^k)$ times sequentially in logarithmic space.

3.3   Executing a series of QTMs

Let p be our integer parameter from the set $\{0, 1, \ldots, n^k\}$. For each p, we consider a QTM $M[p]$ as follows. First, we execute the above QTM in Section 3.2, and then transform $|\tilde{u}\rangle$ to

$$|\tilde{u}_p\rangle = \begin{pmatrix} \frac{1}{2} + A \\ 2^{n^k-p}\left(\frac{1}{2} - A\right) \end{pmatrix}$$

in $(n^k - p)$ iterations. In each iteration, we combine the first qubit with another qubit in state $|0\rangle$, apply the quantum operator

$$\frac{1}{2}\begin{pmatrix} 1 & \sqrt{3} & 0 & 0 \\ \sqrt{3} & -1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix},$$

and then the second qubit is measured. If the measurement outcome is $|0\rangle$, then the computation continues. Otherwise, the computation is terminated by entering the non-postselecting state. (By induction, we can easily see that $|\tilde{u}\rangle \xrightarrow{n^k-p \text{ steps}} |\tilde{u}_p\rangle$.) Note that for each p, the QTM $M[p]$ can be done in logarithmic space as the QTM described in Section 3.2 is done in $O(\log n)$ space, and the counter for the iteration for creating $|\tilde{u}\rangle$ needs $O(\log n)$ space as well.

By substituting $A = \frac{A'}{2^{n^k}}$, the quantum state $|\tilde{u}_p\rangle$ can be rewritten as

$$|\tilde{u}_p\rangle = \begin{pmatrix} \frac{1}{2} + \frac{A'}{2^{n^k}} \\ 2^{n^k-p}\left(\frac{1}{2} - \frac{A'}{2^{n^k}}\right) \end{pmatrix} = \begin{pmatrix} \frac{1}{2} + \frac{A'}{2^{n^k}} \\ 2^{n^k-p}\left(\frac{2^{n^k}-2A'}{2^{n^k+1}}\right) \end{pmatrix} = \begin{pmatrix} \frac{1}{2} + \frac{A'}{2^{n^k}} \\ \frac{2^{n^k}-2A'}{2^{p+1}} \end{pmatrix}.$$

It is easy to see that

- when $A < \frac{1}{2}$ $\left(A' < \frac{2^{n^k}}{2} \Rightarrow 2A' < 2^{n^k}\right)$, the normalized state $|u_p\rangle$ of $|\tilde{u}_p\rangle$ lies in the first quadrant, and thus it is closer to $|+\rangle$, and
- when $A > \frac{1}{2}$ $\left(A' > \frac{2^{n^k}}{2} \Rightarrow 2A' > 2^{n^k}\right)$, $|u_p\rangle$ lies in the fourth quadrant, and thus it is closer to $|-\rangle$.
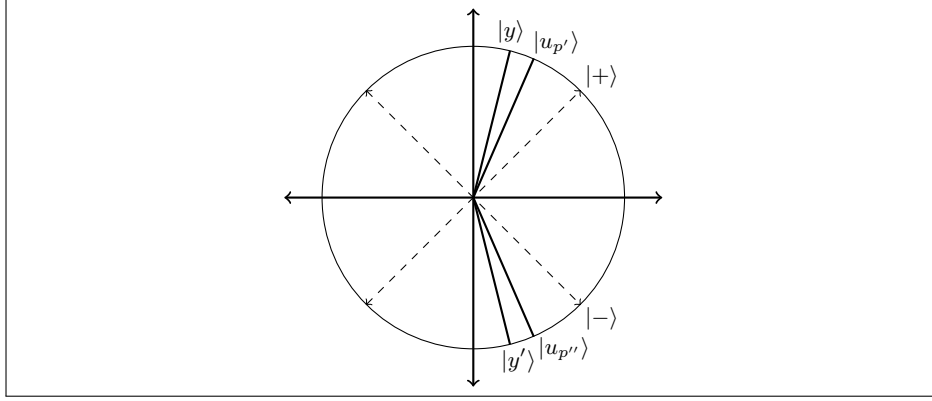
Figure 2 The visualization of $|y\rangle$ and $|u_{p'}\rangle$ when $A < \frac{1}{2}$, and $|y'\rangle$ and $|u_{p''}\rangle$ when $A > \frac{1}{2}$.

Case $A < \frac{1}{2}$: As $A \leq \frac{1}{2} - \frac{1}{2^{n^k}}$ (recall that A is the accepting probability of the PTM M on input x that halts in n^k steps),

$$\frac{2^{n^k} - 2A'}{2^{p+1}} \geq \frac{2}{2^{p+1}}.$$

Thus, there exists a value of p, say p', such that

$$\frac{2^{n^k} - 2A'}{2^{p'+1}} \in [1, 2].$$

Then, since $\frac{1}{2} \leq \frac{1}{2} + \frac{A'}{2^{n^k}}$ and $2 \geq \frac{2^{n^k} - 2A'}{2^{p'+1}}$, the quantum state $|y\rangle = \frac{2}{\sqrt{17}} \begin{pmatrix} 1/2 \\ 2 \end{pmatrix}$ lies between $|1\rangle$ and $|u_{p'}\rangle$, and since $\frac{1}{2} \leq \frac{1}{2} + \frac{A'}{2^{n^k}} < 1$ and $2 \geq \frac{2^{n^k} - 2A'}{2^{p'+1}} \geq 1$, $|u_{p'}\rangle$ lies between $|y\rangle$ and $|+\rangle$ (see Fig. 2). Thus, the probability of observing $|+\rangle$ after measuring $|u_{p'}\rangle$ in $\{|+\rangle, |-\rangle\}$ basis is always greater than

$$\frac{25}{34} > \frac{7}{10}$$

since $|\langle y|+\rangle|^2 = \frac{25}{34}$.

Case $A > \frac{1}{2}$: The case is similar to the previous case. There exists a value of p, say p'', such that

$$\frac{2^{n^k} - 2A'}{2^{p''+1}} \in [-2, -1].$$

Then, the quantum state $|y'\rangle = \frac{2}{\sqrt{17}} \begin{pmatrix} 1/2 \\ -2 \end{pmatrix}$ lies between $-|1\rangle$ and $|u_{p''}\rangle$ and $|u_{p''}\rangle$ lies between $|y'\rangle$ and $|-\rangle$ (see Fig. 2). Thus the probability of observing $|u_{p''}\rangle$ when measuring in $\{|+\rangle, |-\rangle\}$ basis is always greater than

$$\frac{25}{34} > \frac{7}{10}.$$

Now the overall quantum algorithm is as follows:

1. Prepare counter C to 0. For each $p \in \{0, 1, \ldots, n^k - 1\}$, the following steps are implemented.
   a. We execute the above QTM $M[p]$, and make the measurement at the end in $\{|+\rangle, |-\rangle\}$ basis. (Note that the execution can be discarded by entering the non-postselecting state in the procedure of Section 3.2.)
   b. If the measurement result corresponds to $|+\rangle$, then we reset the quantum register to all $|0\rangle$ (note that this is possible using the classical control since all the non-$|0\rangle$ qubits are induced only by post-selection, and thus we know what states they are in), and add $+1$ to C.
   c. If the measurement result corresponds to $|-\rangle$, then we reset the quantum register to all $|0\rangle$, and add -1 to C.
2. If $C = n^k$ (namely, we observe $|+\rangle$ in all executions), then the input is rejected.
3. If $C = -n^k$ (namely, we observe $|-\rangle$ in all executions), then the input is accepted.
4. Otherwise (namely, if we observe the outcomes $|+\rangle$ and $|-\rangle$ at least once in some executions), the computation is terminated in the non-postselecting state.

Note that the overall quantum algorithm is implemented in logarithmic space since the counter is clearly implemented in $O(\log n)$ space, and $M[p]$ is also implemented in $O(\log n)$ space, and each iteration of step 1 is done by the reuse of the classical and quantum registers.

The analysis of the algorithm is as follows:

- When $A < \frac{1}{2}$, the probability of observing $|+\rangle$ is always greater than $|-\rangle$ in each execution and at least once it is $\frac{7}{3}$ times more. Thus, if $x \notin L$, the probability of observing all $|+\rangle$'s is at least $\frac{7}{3}$ times more than the probability of observing all $|-\rangle$'s after all executions.
- When $A > \frac{1}{2}$, the probability of observing $|-\rangle$ is always greater than $|+\rangle$ in each execution and at least once it is $\frac{7}{3}$ times more. Thus, if $x \in L$, the probability of observing all $|-\rangle$'s is at least $\frac{7}{3}$ times more than the probability of observing all $|+\rangle$'s after all executions.

Therefore, after normalizing the final accepting and rejecting postselecting probabilities, it follows that L is recognized by a polynomial-time logarithmic-space postselecting QTM with error bound $\frac{3}{10}$. This completes the proof of Theorem 3. (The error bound can easily be decreased by using the standard probability amplification techniques.)

3.4   Additional result

Additionally, we can show that PostBQL is contained in the class of languages recognized by logarithmic space bounded-error QTMs that halt in *expected exponential time.*

▶ **Theorem 4.** PostBPL $\subseteq$ BPL(exp) *and* PostBQL $\subseteq$ BQL(exp)*, where* BPL(exp) *(*BQL(exp)*) is the class of languages recognized by logarithmic space bounded-error PTMs (QTMs) that halt in expected exponential time.*

Proof. Let M be a polynomial-time logarithmic-space PostPTM. By restarting the whole computation from the beginning instead of entering the non-postselecting state, we can obtain a logarithmic-space exponential-time PTM M' from M, i.e., (i) the restarting mechanism does not require any extra space, and, (ii) since M produces no less than exponentially small halting probability in polynomial time, M' halts with probability 1 in exponential expected time. Both machines recognize the same language with the same error bound since the restarting and postselecting mechanism can be used interchangeably [19, 21], i.e., the accepting and rejecting probabilities by M and M' are the same on every input. Thus, we can conclude that PostBPL $\subseteq$ BPL(exp). In the same way, we can obtain that PostBQL $\subseteq$ BQL(exp). ◀

As $\mathsf{BQL(exp)} \subseteq \mathsf{BQL(\infty)} \subseteq \mathsf{PQL}$ by definition and Watrous showed $\mathsf{PQL} = \mathsf{PL}$ [17], our main result ($\mathsf{PL} = \mathsf{PostBQL}$) leads to the following equivalence among $\mathsf{BQL(exp)}$, $\mathsf{BQL(\infty)}$ and PL.

▶ **Corollary 5.** $\mathsf{PL} = \mathsf{PQL} = \mathsf{PostBQL} = \mathsf{BQL(exp)} = \mathsf{BQL(\infty)}$.

We leave open whether $\mathsf{BPL(exp)}$ is contained in $\mathsf{PostBPL}$.

4   Related Results

In this section, we provide several results on logarithmic-space complexity classes with post-selection. The first result is a characterization of NL by logarithmic-space complexity classes.

▶ **Theorem 6.** $\mathsf{NL} = \mathsf{PostEPL} = \mathsf{PostRL}$.

Proof. We start with the first equality $\mathsf{NL} = \mathsf{PostEPL}$. Let $L \in \mathsf{NL}$. Since $\mathsf{NL} = \mathsf{coNL}$ [9], $\overline{L}$ is also in NL. Then, there exist polynomial-time logarithmic-space NTMs N_1 and N_2 recognizing L and $\overline{L}$. Based on N_1 and N_2, we can construct a polynomial-time logarithmic-space PostPTM M such that M executes N_1 and N_2 with equal probability on the given input. Then, M accepts the input if N_1 accepts and rejects the input if N_2 accepts. Any other outcome is discarded by M. Therefore, (i) any $x \in L$ is accepted with nonzero probability and rejected with zero probability by M, and, (ii) any $x \in \overline{L}$ is accepted with zero probability and rejected with nonzero probability by M. Thus, L is recognized by M with no error, and thus $L \in \mathsf{PostEPL}$.

Let $L \in \mathsf{PostEPL}$. Then, there exists a polynomial-time logarithmic-space PostPTM M recognizing L with no error. Based on M, we can construct a polynomial-time logarithmic-space NTM N such that N executes M on the given input and switches to the rejecting state if M ends in the non-postselecting halting state. Thus, N accepts all and only strings in L. Therefore, $L \in \mathsf{NL}$.

Now we are done with equality $\mathsf{NL} = \mathsf{PostEPL}$. It is trivial that $\mathsf{PostEPL} \subseteq \mathsf{PostRL}$. To complete the proof, it is enough to show that $\mathsf{PostRL} \subseteq \mathsf{NL}$. If a language is recognized by a polynomial-time logarithmic-space PostPTM M with one-sided bounded-error, then it is also recognized by a polynomial-time logarithmic-space NTM M' where M' is modified from M such that if M enters the non-postselelecting state, then M' enters the rejecting state.   ◀

By using the same argument, we can also obtain the following result on quantum class $\mathsf{PostEQL}$ (note that the first equality comes from $\mathsf{NQL} = \mathsf{coC_{=}L}$ [16, 7]).

▶ **Theorem 7.** $\mathsf{C_{=}L} \cap \mathsf{coC_{=}L} = \mathsf{NQL} \cap \mathsf{coNQL} = \mathsf{PostEQL}$.

As will be seen below, the relation between $\mathsf{PostEQL}$ and $\mathsf{PostRQL}$ seems different from the relation between their classical counterparts since $\mathsf{C_{=}L}$ and $\mathsf{coC_{=}L}$ may be different classes. Remark that it is also open whether NL is a proper subset of $\mathsf{C_{=}L} \cap \mathsf{coC_{=}L}$ or not.

By using the quantum simulation given in Section 3, we can obtain the following result.

▶ **Theorem 8.** $\mathsf{coC_{=}L} = \mathsf{PostRQL}$.

Proof. It is easy to see that $\mathsf{PostRQL} \subseteq \mathsf{NQL}$. Let L be a language in $\mathsf{PostRQL}$ and M be a polynomial-time logarithmic-space PostQTM recognizing L with one-sided bounded-error. By changing the transitions to the non-postselecting state of M to the rejecting state, we can obtain a polynomial-time logarithmic-space NQTM recognizing L, and thus $\mathsf{PostRQL} \subseteq \mathsf{NQL}$. Since $\mathsf{NQL} = \mathsf{coC_{=}L}$ [16, 7], we obtain $\mathsf{PostRQL} \subseteq \mathsf{coC_{=}L}$.

Now we prove the other direction. Let L be in $\mathsf{coC_=L}$. Then there exists a polynomial-time logarithmic-space PTM M' that accepts any non-member of L with probability $\frac{1}{2}$ and any member with probability different from $\frac{1}{2}$. Let x be a given input with length n.

We use the simulation given in Section 3. We make the same assumptions on the PTM M' except that M' accepts some string with probability $\frac{1}{2}$ and M' never accepts any string with probability in the following interval

$$\left(\frac{1}{2} - \frac{1}{2^{n^k}}, \frac{1}{2} + \frac{1}{2^{n^k}} \right)$$

for some fixed integer k. This condition is trivial if the running time never exceeds n^k, i.e., the total number of probabilistic branches never exceeds 2^{n^k}.

Then, we construct a polynomial-time logarithmic-space PostQTM as described in Section 3 with the following unnormalized final quantum state:

$$\begin{pmatrix} 2A - 1 \\ 2^{-n^k} \end{pmatrix},$$

where A is the accepting probability of M'. We measure this qubit and accept (reject) the input, if we observe $|0\rangle$ ($|1\rangle$). All the other outcomes are discarded by entering the non-postselecting state.

It is clear that for any non-member of L, A is always equal to $\frac{1}{2}$, and thus the QTM accepts the input with zero probability and rejects the input with some non-zero probability. Therefore, any non-member of L is rejected with probability 1.

On the other hand, for any member, the amplitude of $|0\rangle$ is at least twice of the amplitude of $|1\rangle$, and thus the accepting probability is at least four times more than the rejecting probability. Thus, any member is accepted with probability at least $\frac{4}{5}$. The success probability can be increased by using the standard probability amplification techniques.   ◀

References

1 Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A*, 461(2063):3473–3482, 2005.

2 Eric Allender and Mitsunori Ogihara. Relationships among PL, #L, and the determinant. *RAIRO Theoretical Informatics and Applications*, 30(1):1–21, 1996.

3 Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.

4 David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A*, 400:97–117, 1985.

5 Bill Fefferman, Hirotada Kobayashi, Cedric Yen-Yu Lin, Tomoyuki Morimae, and Harumichi Nishimura. Space-efficient error reduction for unitary quantum computations. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming*, volume 55 of *LIPIcs*, pages 14:1–14:14, 2016.

6 Bill Fefferman and Cedric Yen-Yu Lin. A complete characterization of unitary quantum space. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference*, volume 94 of *LIPIcs*, pages 4:1–4:21, 2018.

7 Bill Fefferman and Zachary Remscrim. Eliminating intermediate measurements in space-bounded quantum computation. In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing, to appear*, 2021. Also available at arXiv:2006.03530.

8 Uma Girish, Ran Raz, and Wei Zhan. Quantum logspace algorithm for powering matrices with bounded nom. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming, to appear*, 2021. Also available at arXiv:2006.04880.

9   Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, 1988.

10   Hermann Jung. On probabilistic time and space. In *Automata, Languages and Programming*, volume 194 of *LNCS*, pages 310–317. Springer, 1985.

11   Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

12   A. C. Cem Say and Abuzer Yakaryılmaz. Computation with narrow CTCs. In *Unconventional Computation*, volume 6714 of *LNCS*, pages 201–211. Springer, 2011.

13   A. C. Cem Say and Abuzer Yakaryılmaz. Quantum finite automata: A modern introduction. In *Computing with New Resources*, volume 8808 of *LNCS*, pages 208–222. Springer, 2014.

14   Amnon Ta-Shma. Inverting well conditioned matrices in quantum logspace. In *44th ACM Annual Symposium on Theory of Computing*, pages 881–890. ACM, 2013.

15   Dieter van Melkebeek and Thomas Watson. Time-space efficient simulations of quantum computations. *Theory of Computing*, 8(1):1–51, 2012.

16   John Watrous. Space-bounded quantum complexity. *Journal of Computer and System Sciences*, 59(2):281–326, 1999.

17   John Watrous. On the complexity of simulating space-bounded quantum computations. *Computational Complexity*, 12(1-2):48–84, 2003.

18   John Watrous. Quantum computational complexity. In *Encyclopedia of Complexity and System Science*. Springer, 2009. Also available at arXiv:0804.3401.

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

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

21   Abuzer Yakaryılmaz and A. C. Cem Say. Proving the power of postselection. *Fundamenta Informaticae*, 123(1):107–134, 2013.