Quantum Random Access Stored-Program Machines

Qisheng Wang [*] Mingsheng Ying [†]

Abstract

Random access machines (RAMs) and random access stored-program machines (RASPs) are models of computing that are closer to the architecture of real-world computers than Turing machines (TMs). They are also convenient in complexity analysis of algorithms. The relationships between RAMs, RASPs and TMs are well-studied. However, clear relationships between their quantum counterparts are still missing in the literature. We fill in this gap by formally defining the models of quantum random access machines (QRAMs) and quantum random access stored-program machines (QRASPs) and clarifying the relationships between QRAMs, QRASPs and quantum Turing machines (QTMs). In particular, we show that $\mathbf{P} \subseteq \mathbf{EQRAMP} \subseteq \mathbf{EQP} \subseteq \mathbf{BQP} = \mathbf{BQRAMP}$, where **EQRAMP** and **BQRAMP** stand for the sets of problems that can be solved by polynomial-time QRAMs with certainty and bounded-error, respectively. At the heart of our proof, is a standardisation of QTM with an extended halting scheme, which is of independent interest.

Keywords: quantum computing, random access machine, stored-program machine, Turing machine.

[*]Qisheng Wang is with the Department of Computer Science and Technology, Tsinghua University, China (e-mail: `QishengWang1994@gmail.com`).

[†]Mingsheng Ying is with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China, and also with the Department of Computer Science and Technology, Tsinghua University, China (e-mail: `yingms@ios.ac.cn`).

Contents

1 Introduction

Models of Quantum Computing: Various traditional models of computing have been generalised to the quantum setting as the models of quantum computing (cf. [24]), including quantum Turing machines (QTMs) [10] and quantum circuits [11]. Several novel quantum computing models that have no classical counter-parts have also been proposed, e.g. measurement-based and one-way quantum computing [27, 28], adiabatic quantum computing [12]. Furthermore, the relationships between these models have been thoroughly studied [31, 4, 2].

Quantum Random Access Machines: Random access machines (RAMs)[1] and random access stored-program machines (RASPs) are another model of computing that is closer to the architecture of real-world computers than Turing machines (TMs). They are also convenient in complexity analysis of algorithms [8, 3].

The notion of quantum random access machine (QRAM) was first introduced in [17], as a basis of the studies of quantum programming. Essentially, it is a RAM in the traditional sense with the ability to perform a set of quantum operations on quantum registers, including: (1) state preparation, (2) certain unitary operations, and (3) quantum measurements.

Recently, several quantum computer architectures have been proposed based on the QRAM model with some practical quantum instruction sets, including IBM OpenQSAM [15], Rigetti's guil [29] and Delft's eQASM [13].

Contributions of This Paper: However, a clear relationship between QRAMs and QTMs is still missing in the literature. The aim of this paper is to fill in this gap. In [17], QRAMs were described only in an informal way, and to our best knowledge, QRASPs have not been introduced in the existing literature. For our purpose, we first formally define the models of QRAMs and QRASPs as appropriate generalisation of RAMs and RASPs [3].

It is worth mentioning that the formal model of QRASPs also provides us with a theoretical foundation of quantum programming (see [20] and Section 8.1 of [32] for a discussion of the significance of such a foundation). In particular, currently a quantum computer is essentially a quantum processor controlled by a classical computers, and quantum programs are stored in the classical computer (as classical data). In this sense, our QRASPs are an appropriate model of the existing quantum computers.

Based on the formal models of QRAMs and QRASPs, we then clarify the relationships between QTMs, QRAMs and QRASPs. Our main results are:

1. A $T(n)$-time QRAM (resp. QRASP) can be simulated by an $O(T(n))$-time QRASP (resp. QRAM).

2. A $T(n)$-time QRAM under the logarithmic (resp. constant) cost criterion can be simulated by an $\tilde{O}(T(n)^4)$-time (resp. $\tilde{O}(T(n)^8)$-time) QTM.

3. A $T(n)$-time QTM can be simulated within error $\varepsilon > 0$ by an $O(T(n)^2 \operatorname{polylog}(T(n), 1/\varepsilon))$-time QRAM (under both the logarithmic and constant cost criterions).

In comparison with the classical counterparts [8], $T(n)$-time RAMs under the logarithmic (resp. constant) criterion can be simulated by $O(T(n)^2)$-time (resp. $O(T(n)^3)$-time) TMs. Conversely, $T(n)$-time TMs can be simulated by $\tilde{O}(T(n))$-time RAMs.

The above results have some immediate corollaries on computational complexity. We define two complexity classes:

[1]In the quantum computation and information literature, QRAM is also used to refer Quantum Random Access Memory proposed by Giovannetti, Lloyd, and Maccone (*Phys. Rev. Lett.* 100(2018), 160501).

- **EQRAMP** stands for exact quantum random access machine polynomial-time, and
- **BQRAMP** stands for bounded-error quantum random access machine polynomial-time.

Then it holds that

$$\mathbf{P} \subseteq \mathbf{EQRAMP} \subseteq \mathbf{EQP} \subseteq \mathbf{BQP} = \mathbf{BQRAMP}. \tag{1}$$

Not much has been known in the literature about the relationship between **EQP** and other complexity classes. Here, an inclusion between **EQP** and **EQRAMP** is established. However, we still do not know which inclusion in (1) is proper.

Major Challenge: The main difficulty in comparing the computational power of QTMs, QRAMs and QRASPs comes from the difference between their halting schemes:

- There have been a bunch of discussions about the halting scheme of QTMs [23, 19, 26, 30, 21], where QTMs are required to *terminate exactly at a fixed time* (depending on the input) with certainty.
- On the other hand, it is reasonable to allow QRAMs and QRASPs to terminate at any time with an appropriate probability.

We resolve this issue by extending the model of QTMs [4] so that QTMs are allowed to terminate at any time with a non-zero probability. This extension of QTMs makes that QTMs match the halting scheme of QRAMs and QRASPs, and thus it is convenient to simulate QRAMs and QRASPs by QTMs. In order to deal with the extended halting scheme of QTMs, on the other hand, we introduce the notion of *standard* QTM as a stepping stone and give a constructive proof that every well-formed QTM within time $T(n)$ can be simulated by a *standard* QTM within time $O(T(n)\log^2 T(n))$, called a standardisation of QTM. Based on the notion of standard QTM, we are also able to give an alternative definition of complexity classes **EQP** and **BQP** in terms of our extended QTMs.

Organisation of the Paper: In Section 2, we recall from [4] the definition of QTMs and then introduce the notions of extended QTMs and standard QTMs. In Section 3, the formal definitions of QRAMs and QRASPs are given. Our main results are presented in Section 4.

The remaining sections are then devoted to provide all of the details. The computations of QRAMs and QRASPs are carefully described in Sections 5 and 6, respectively. The simulations of QRAMs and QRASPs with each other are described in Section 7, and the simulations of QRAMs and QTMs with each other are given in Section 8. The standardisation of QTM is given in Section 9.

2 QTMs

The purpose of this section is two-fold. For convenience of the reader, in the first two subsections, we review some basic notions of quantum Turing machines (QTMs). Our exposition is mainly based on [4]. In the last subsection, we define the notion of standard QTM and show that every well-formed QTM can be efficiently simulated by a standard QTM. This result will serve as a step stone in comparing the computational power of QTMs with that of QRAMs and QRASPs.

2.1 Single-Track QTMs

Let $T : \mathbb{N} \to \mathbb{N}$ be a mapping from natural numbers to themselves. We write $\mathbb{C}(T(n))$ for the set of all $T(n)$-time computable complex numbers, i.e. for every $x \in \mathbb{C}(T(n))$, there is a $T(n)$-time deterministic Turing machine M such that $|M(1^n) - x| < 2^{-n}$, where $M(1^n)$ denotes the output floating point complex number of M on input 1^n. Let $\tilde{\mathbb{C}}$ be the set of all polynomial-time computable complex numbers, i.e. $\tilde{\mathbb{C}} = \bigcup_{k=1}^{\infty} \mathbb{C}(n^k)$.

Definition 2.1. *A Quantum Turing Machine (QTM) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, q_f)$, where:*

1. *Q is a finite set of states;*
2. *Σ is a finite alphabet with blank symbol $\#$;*
3. *$\delta : Q \times \Sigma \times \Sigma \times Q \times \{L, R\} \to \tilde{\mathbb{C}}$ is the transition function;*
4. *$q_0 \in Q$ is the initial state; and*
5. *$q_f \in Q$ is the final state ($q_0 \neq q_f$).*

A configuration of the tape is described by a function $\mathcal{T} : \mathbb{Z} \to \Sigma$ such that $\mathcal{T}(m) = \#$ except for finitely many integers m. Thus, the symbol at position m on the tape is denoted $\mathcal{T}(m)$. We write $\Sigma^{\#} \subseteq \Sigma^{\mathbb{Z}}$ for the set of all possible tape configurations. Moreover, a (computational) configuration of M is a 3-tuple $c = (q, \mathcal{T}, \xi) \in Q \times \Sigma^{\#} \times \mathbb{Z}$, where q is the current state, $\mathcal{T}$ is the tape configuration and ξ is the head position. It represents a basis state $|c\rangle = |q\rangle_Q |\mathcal{T}\rangle_{\Sigma^{\#}} |\xi\rangle_{\mathbb{Z}} = |q, \mathcal{T}, \xi\rangle$ of the quantum machine. Therefore, the state Hilbert space of M is span$\{|c\rangle\}$, where c ranges over all configurations of M. The time evolution operator U of M is defined by δ as follows:

$$U |p, \mathcal{T}, \xi\rangle = \sum_{\sigma,q,d} \delta(p, \mathcal{T}(\xi), \sigma, q, d) \left|q, \mathcal{T}_{\xi}^{\sigma}, \xi + d\right\rangle,$$

where

$$\mathcal{T}_{\xi}^{\sigma}(m) = \begin{cases} \mathcal{T}(m) & m \neq \xi \\ \sigma & \text{otherwise.} \end{cases}$$

The relation $\delta(p, \tau, \sigma, q, d) = \alpha$ can be interpreted as follows: if the machine is in state p and the symbol on the tape head is τ, then with amplitude α the machine writes the symbol σ on the tape head, changes its state to q and moves to the direction d. It is reasonable to require that M is *well-formed* in the sense that its time evolution operator U is unitary, i.e. $U^\dagger U = UU^\dagger = I$. A deterministic Turing machine (DTM) can be regarded as a QTM with transition function $\delta : Q \times \Sigma \times \Sigma \times Q \times \{L, R\} \to \{0, 1\}$ such that for every $p \in Q$ and $\sigma \in \Sigma$, there is a unique triple $(\tau, q, d) \in \Sigma \times Q \times \{L, R\}$ with $\delta(p, \sigma, \tau, q, d) = 1$. Moreover, a reversible Turing machine (RTM) is a well-formed DTM.

The computation of M begins at time $t = 0$. The initial configuration is prepared to be $|c_0\rangle = |q_0, \mathcal{T}_0, 0\rangle$. At each step, M performs U on the current configuration $|c\rangle$ and makes a measurement to determine whether the configuration is in the final state q_f. The measurement result is "yes" with probability $p = \|P_F U |c\rangle\|^2$ and "no" with probability $1 - p$, where $P_F = |q_f\rangle_Q \langle q_f|$.

- If the result is "yes", then M halts with configuration $\frac{1}{\sqrt{p}} P_F U |c\rangle$;
- If the result is "no", then M continues running with configuration $\frac{1}{\sqrt{1-p}} P_F^{\perp} U |c\rangle$, where $P_F^{\perp} = I - P_F$.

The probability that M halts on $|c_0\rangle$ exactly at time t is $p(t) = \||c_t\rangle\|^2$, where $|c_t\rangle = P_F U (P_F^{\perp} U)^{t-1} |c_0\rangle$. M is said to halt on $|c_0\rangle$ within time T if $\sum_{t=1}^{T} p(t) = 1$; in this case, the configuration after M halts is a mixed state $\rho = \sum_{t=1}^{T} |c_t\rangle \langle c_t|$. Especially, M is said to halt on $|c_0\rangle$ exactly at time T, denoted $|c_0\rangle \xrightarrow[T]{M} |c_T\rangle$, if $p(T) = 1$.

Remark 2.1. *It should be pointed out that the above halting scheme of QTMs is different from the traditional ones (e.g. [4, 26, 25]). Traditional QTMs are supposed to be either in a final state or in a non-final state (see Definition 3.11 of [4], for example), and therefore always halt exactly at some time, which results in that the final configuration is always a pure quantum state. In our halting scheme, this restriction is removed and QTMs are allowed to be in a superposition of final and non-final states. Therefore, our halting scheme allows QTMs to halt at any time with a non-zero probability, and as a result, the final configuration is an ensemble of pure configurations, which is a mixed quantum state.*

2.2 Multi-Track QTMs

The alphabet Σ of a k-track QTM is regarded as the Cartesian product $\Sigma = \Sigma_1 \times \Sigma_2 \times \cdots \times \Sigma_k$. In particular, let $\#_i$ be the blank symbol in Σ_i for $1 \leq i \leq k$. For convenience, for every $x \in \Sigma^*$, we write x to indicate the tape $\mathcal{X}$:

$$\mathcal{X}(m) = \begin{cases} x(m) & 0 \leq m < |x|, \\ \# & \text{otherwise}, \end{cases}$$

where $x(m)$ is the m-th symbol of x. The joint of k tapes $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ is the tape of k tracks:

$$(\mathcal{X}_1; \mathcal{X}_2; \ldots; \mathcal{X}_k)(m) = (\mathcal{X}_1(m), \mathcal{X}_2(m), \ldots, \mathcal{X}_k(m))$$

for every $m \in \mathbb{Z}$. On input $x \in \{0,1\}^*$, we put x on the first track and leave the other tracks empty; that is, the initial tape $\mathcal{T}_x = x; \epsilon; \ldots; \epsilon$, where ϵ denotes the empty string. Let $T : \mathbb{N} \to \mathbb{N}$. A k-track QTM M is said to be within time $T(n)$, if for every $x \in \{0,1\}^*$, M halts on $|q_0, \mathcal{T}_x, 0\rangle$ within time $T(|x|)$, where $|x|$ is the length of x. Especially, M is said to be with exact time $T(n)$, if for every $x \in \{0,1\}^*$, M halts on $|q_0, \mathcal{T}_x, 0\rangle$ exactly at some time $\tau_x \leq T(|x|)$, where τ_x depends on x.

Let M be a QTM within time $T(n)$, and ρ_x the configuration after M halts on input x. The tape contents of ρ_x are obtained by performing a measurement on each position $-T(|x|) \leq m \leq T(|x|)$ (the head position ξ never goes beyond $|\xi| > T(|x|)$). We define function $\mathcal{T}_M : \{0,1\}^* \times \Sigma^{\#} \to [0,1]$ by letting for every $x \in \{0,1\}^*$, $\mathcal{T}_M(x, \mathcal{Z})$ be the probability that on input x, M halts with tape $\mathcal{Z}$ after measurement. Formally, $\mathcal{T}_M(x, \mathcal{Z}) = \operatorname{tr}\left(M_{\mathcal{Z}} \rho_x M_{\mathcal{Z}}^\dagger\right)$, where $M_{\mathcal{Z}} = |\mathcal{Z}\rangle_{\Sigma^\#} \langle \mathcal{Z}|$. We note that if $\mathcal{Z}(m) \neq \#$ for some position m with $|m| > T(|x|)$, then $\mathcal{T}_M(x, \mathcal{Z}) = 0$.

We design the output track to be the second track. Suppose $\mathcal{Z}$ is the tape after measurement. For $-T(|x|) \leq m \leq T(|x|)$, let $y_m \in \Sigma_2$ be the symbol of $\mathcal{Z}(m)$ in the second track. Then the output y is defined to be the concatenation of y_m's for $-T(|x|) \leq m \leq T(|x|)$ after ignoring blank symbols $\#_2$. We write $y = \operatorname{extract}(\mathcal{Z})$ if the contents of tape $\mathcal{Z}$ implies output y. We assume that y consists of only symbols 0 and 1, i.e. $y \in \{0,1\}^*$. In this way, QTM M defines a function $M : \{0,1\}^* \times \{0,1\}^* \to [0,1]$ so that M on input x outputs y with probability $M(x,y)$, i.e.

$$M(x,y) = \sum_{\operatorname{extract}(\mathcal{Z}) = y} \mathcal{T}_M(x, \mathcal{Z}).$$

In case $M(x,y) = 1$ for some $x, y \in \{0,1\}^*$, we may write $M(x) = y$, indicating that M on input x outputs y with certainty.

2.3 Standard QTMs

Note that the above definition of QTM is different from that in [4], where QTMs are prevented from reaching a superposition in which some configurations are in state q_f but others are not, and therefore intermediate measurements on the state of the QTM (i.e. to see whether the state is in q_f or not) will not modify the configurations during the computation. However, QTMs defined above are allowed to reach a superposition of the final state and the other states. For our purpose, it is natural to assume that each time the configuration is measured (and therefore changes), the configuration will collapse to one of the configurations either in q_f or not with probability according to the amplitudes. In this subsection, we establish a connection between these two kinds of QTMs. Let us first introduce several terminologies.

Definition 2.2 (Stationary QTMs). *A QTM M is said to be stationary, if it halts on every $x \in \{0,1\}^*$ and*

$$\mathrm{tr}(P_0 \rho_x) = 1,$$

where $P_0 = |0\rangle_{\mathbb{Z}} \langle 0|$, and ρ_x is the configuration after M halts on x.

Definition 2.3 (Normal Form QTMs). *A QTM M is said to be in normal form, if for every $\tau, \sigma \in \Sigma$, $q \in Q$ and $d \in \{L, R\}$,*

$$\delta(q_f, \tau, \sigma, q, d) = \begin{cases} 1 & (\sigma, q, d) = (\tau, q_0, R), \\ 0 & \text{otherwise.} \end{cases}$$

Definition 2.4 (Unidirectional QTMs). *A QTM M is said to be unidirectional, if for every $q \in Q$, there is a direction $d_q \in \{L, R\}$ such that*

$$\delta(p, \tau, \sigma, q, \bar{d}_q) = 0$$

for every $p \in Q$ and $\tau, \sigma \in \Sigma$, where $\bar{d}$ denotes the reverse direction of d.

Intuitively, a stationary QTM always halts with tape head at position 0, i.e. the starting position. In a normal form QTM, the transitions from q_f are technically specified for convenience, since every QTM always halts before any transition out of q_f. In a unidirectional QTM, any state can be entered from only one direction.

Let M be a normal form QTM and $|c_0\rangle = |q_0, \mathcal{T}_0, \xi_0\rangle$ its initial configuration. If the configuration $|c\rangle$ becomes $|c'\rangle$ after t steps, i.e. $U(P_F^{\perp} U)^{t-1} |c\rangle = |c'\rangle$, we write $|c\rangle \xrightarrow[t]{M} |c'\rangle$. If M halts on $|c_0\rangle$ exactly at time T with the final state $|c_f\rangle = |q_f, \mathcal{T}_f, \xi_f\rangle$, then we write $|\mathcal{T}_0, \xi_0\rangle \xrightarrow[T]{M} |\mathcal{T}_f, \xi_f\rangle$. If M is stationary (and thus) $\xi_0 = \xi_f = 0$, we simply write $|\mathcal{T}_0\rangle \xrightarrow[T]{M} |\mathcal{T}_f\rangle$. Moreover, if both $|\mathcal{T}_0\rangle$ and $|\mathcal{T}_f\rangle$ are in the computational basis, we often write $\mathcal{T}_0 \xrightarrow[T]{M} \mathcal{T}_f$.

Definition 2.5 (Standard QTM). *A QTM M is standard, if it is well-formed, normal form, stationary and unidirectional and there is a function $T : \mathbb{N} \to \mathbb{N}$ such that for every $x \in \{0,1\}^*$, M on input x halts exactly at time $T(|x|)$.*

For comparing different QTM models, we need the notion of time constructible function.

Definition 2.6 (Time Constructible Functions). *Let $T : \mathbb{N} \to \mathbb{N}$ and $T(n) \geq n$ for every $n \in \mathbb{N}$. $T(n)$ is said to be time constructible, if there is a standard QTM M with exact time $O(T(n))$ such that for every $x \in \{0,1\}^*$,*

$$x; \epsilon \xrightarrow[O(T(|x|))]{M} x; T(|x|). \tag{2}$$

Note that in (2), a natural number $n \in \mathbb{N}$ written on a tape or track indicates a binary string $a = a_0 a_1 \ldots a_{k-1} \in \{0,1\}^k$ such that k is the smallest positive integer that $2^k > n$ and $n = \sum_{i=0}^{k-1} 2^{k-i-1} a_i$.

Now we are able to show our first result that every well-formed QTM can be efficiently simulated by a standard QTM. For any QTM M, we write $\mathbb{C}(M)$ for the set of its transition coefficients of a QTM M; that is,

$$\mathbb{C}(M) = \{\delta(p, \sigma, \tau, q, d) : p, q \in Q, \sigma, \tau \in \Sigma, d \in \{L, R\}\}.$$

Theorem 2.1 (Standardisation)**.** *Let $T : \mathbb{N} \to \mathbb{N}$ be a function time constructible by QTM. For every well-formed and normal form QTM M within time $T(n)$, there is a standard QTM M' with exact time $O(T(n) \log^2 T(n))$ such that*

1. *$M(x, y) = M'(x, y)$ for every $x, y \in \{0,1\}^*$.*
2. *$\mathbb{C}(M') \subseteq \mathbb{C}(M) \cup \{0, 1\}$.*

The proof of Theorem 2.1 is given in Section 9.

3 QRAMs and QRASPs

It is a common sense in the existing literature that a practical quantum computer (of the first generation at least) consists of a classical computer with access to quantum registers, where the classical part performs classical computations and controls the evolution of quantum registers, and the quantum part can be initialised in certain states (e.g. basis state $|0\rangle$), perform elementary unitary operations (e.g. Hadamard, $\pi/8$ and CNOT gates), and be measured with the outcomes sent to the classical machine. The models of QRAMs and QRASPs are defined based on this intuition.

3.1 QRAMs

Formally, a quantum random access machine (QRAM) is a program P, i.e. a finite sequence of QRAM instructions operating on an infinite sequence of both classical and quantum registers. Each classical register holds an arbitrary integer (positive, negative, or zero), while each quantum register holds a qubit (in state $|0\rangle$, $|1\rangle$ or their superposition). The contents of the i-th ($i \geq 0$) classical (resp. quantum) register is denoted by X_i (resp. Q_i).

Associated with the machine is a cost function $l(n)$, which denotes the memory required to store, or the time required to load the number n. Two forms of $l(n)$ commonly used in studying classical RAMs are:

1. $l(n)$ is a constant, i.e. $l(n) = O(1)$; and
2. $l(n)$ is logarithmic, i.e. $l(n) = O(\log |n|)$, where $|n|$ is the absolute value of n.

Definition 3.1. *The instructions for QRAM and their execution times are given in Table 1.*

The QRAM instructions in Table 1 are divided into two types. The classical-type instructions are the same as those adopted in [8]. Here, i, j, k are any nonnegative integers, and m are integers between 0 and L (inclusive), where L is the length of the QRAM program and also denotes termination (see Section 5.1 for details about m). The effect of most of the instructions are obvious. For example, $X_i \leftarrow C$ causes X_i to hold C, while $X_i \leftarrow X_j \pm X_k$ causes X_i to hold the calculation result

Table 1: QRAM instructions

Type	Instruction	Execution time
Classical	$X_i \leftarrow C$, C any integer	1
Classical	$X_i \leftarrow X_j + X_k$	$l(X_j) + l(X_k)$
Classical	$X_i \leftarrow X_j - X_k$	$l(X_j) + l(X_k)$
Classical	$X_i \leftarrow X_{X_j}$	$l(X_j) + l(X_{X_j})$
Classical	$X_{X_i} \leftarrow X_j$	$l(X_i) + l(X_j)$
Classical	TRA m if $X_j > 0$	$l(X_j)$
Classical	READ X_i	$l(\text{input})$
Classical	WRITE X_i	$l(X_i)$
Quantum	CNOT$[Q_{X_i}, Q_{X_j}]$	$l(X_i) + l(X_j)$
Quantum	$H[Q_{X_i}]$	$l(X_i)$
Quantum	$T[Q_{X_i}]$	$l(X_i)$
Measurement	$X_i \leftarrow M[Q_{X_j}]$	$l(X_j)$

of $X_j \pm X_k$. The instruction TRA m if $X_j > 0$ causes the m-th instruction to be the next instruction to execute if $X_j > 0$. READ X_i causes X_i to hold the next input number on the input tape, while WRITE X_i causes X_i to be printed on the output tape. The indirect instruction $X_i \leftarrow X_{X_j}$ causes X_i to hold X_{X_j}, provided $X_j \geq 0$, while $X_{X_i} \leftarrow X_j$ causes X_{X_i} to hold X_j, provided $X_i \geq 0$. The indirect instructions allow a fixed program to access unbounded registers. It should be noted that classical registers are needed as classical address to indirectly access quantum registers.

The quantum-type instructions include quantum gates and measurements. For simplicity of presentation, we choose to use a minimal but universal set of quantum gates: CNOT (the Controlled-NOT gate), H (the Hadamard gate) and T (the $\pi/8$ gate). Indeed, any finite universal set of quantum gates is acceptable. The measurement instruction $X_i \leftarrow M[Q_{X_j}]$ is a bridge between classical and quantum registers, which causes X_i to hold the measurement result of Q_{X_j} in the computational basis, provided $X_j \geq 0$.

3.2 QRASPs

A quantum random access stored-program machine (QRASP) is a program P, i.e. a finite sequence of QRASP instructions operating on infinite sequences of both classical and quantum registers.

Definition 3.2. *The instructions for QRASP are given in Table 2.*

Note that the classical-type QRASP instructions are the same as RASP instructions defined in [8], and the quantum-type QRASP instructions are the same as in QRAMs.

Strictly speaking, a QRASP is a finite sequence of integers that are to be interpreted into QRASP instructions during the execution rather than an explicit program. The reason is that QRASP may modify itself during the execution and causes unpredictable interpreted QRASP instructions. Our machine has an accumulator (AC), which holds an arbitrary integer, an instruction counter (IC), and two infinite sequences of both classical and quantum registers. Each classical register X_i holds an arbitrary integer, while each quantum register Q_i holds a qubit. An instruction is stored in two or three consecutive classical registers depending on its operation code. The first classical register contains an operation code (shown in Table 2). In case that the operation code is beyond the range 1 to 11, the execution immediately terminates. The second (and the third if needed) classical register contains the parameter of the instruction. In fact, only the CNOT operation needs two parameters while other operations do not. It is noted that indirect addressing

Table 2: QRASP instructions

Type	Operation	Mnemonic	code	Description	Execution time
Classical	load constant	LOD, j	1	AC $\leftarrow j$; IC $\leftarrow$ IC $+ 2$	$l(\mathrm{IC}) + l(j)$
Classical	add	ADD, j	2	AC $\leftarrow$ AC $+ X_j$; IC $\leftarrow$ IC $+ 2$	$l(\mathrm{IC}) + l(j) + l(\mathrm{AC}) + l(X_j)$
Classical	subtract	SUB, j	3	AC $\leftarrow$ AC $- X_j$; IC $\leftarrow$ IC $+ 2$	$l(\mathrm{IC}) + l(j) + l(\mathrm{AC}) + l(X_j)$
Classical	store	STO, j	4	$X_j \leftarrow$ AC; IC $\leftarrow$ IC $+ 2$	$l(\mathrm{IC}) + l(j) + l(\mathrm{AC})$
Classical	branch on positive accumulator	BPA, j	5	if AC > 0 then IC $\leftarrow j$; otherwise IC $\leftarrow$ IC+2	$l(\mathrm{IC}) + l(j) + l(\mathrm{AC})$
Classical	read	RD, j	6	$X_j \leftarrow$ next input; IC $\leftarrow$ IC $+ 2$	$l(\mathrm{IC}) + l(j) + l(\mathrm{input})$
Classical	print	PRI, j	7	output X_j; IC $\leftarrow$ IC $+ 2$	$l(\mathrm{IC}) + l(j) + l(X_j)$
Quantum	CNOT	CNOT, j, k	8	CNOT$[Q_j, Q_k]$; IC $\leftarrow$ IC $+ 3$	$l(\mathrm{IC}) + l(j) + l(k)$
Quantum	H	H, j	9	$H[Q_j]$; IC $\leftarrow$ IC $+ 2$	$l(\mathrm{IC}) + l(j)$
Quantum	T	T, j	10	$T[Q_j]$; IC $\leftarrow$ IC $+ 2$	$l(\mathrm{IC}) + l(j)$
Measurement	measure	MEA, j	11	AC $\leftarrow M[Q_j]$; IC $\leftarrow$ IC $+ 2$	$l(\mathrm{IC}) + l(j)$
Termination	halt	HLT	-	stop	$l(\mathrm{IC}) + l(X_{\mathrm{IC}})$

is not allowed in QRASP, the programs need to modify themselves in order to access unbounded number of (both classical and quantum) registers.

4 Main Results

In this section, we state our main results, which clarify the relationships between QTMs, QRAMs and QRASPs.

The relationship between QRAMs and QRASPs is simple. We prove that QRAMs and QRASPs can simulate each other with constant slowdown. For a QRAM (or QRASP) P and $x, y \in \{0,1\}^*$, let $P(x, y)$ denote the probability that P on input x outputs y (see Section 5 and Section 6 for its formal definition).

Theorem 4.1. *Let $T : \mathbb{N} \to \mathbb{N}$ with $T(n) \geq n$.*

1. *For every $T(n)$-time QRAM P, there is a $O(T(n))$-time QRASP P' such that for every $x, y \in \{0,1\}^*$, $P(x, y) = P'(x, y)$.*
2. *For every $T(n)$-time QRASP P, there is a $O(T(n))$-time QRAM P' such that for every $x, y \in \{0,1\}^*$, $P(x, y) = P'(x, y)$.*

To further compare QRAMs (and thus QRASPs) with QTMs, we need a QRAM variant of time constructible functions.

Definition 4.1 (QRAM-Time Constructible Functions)**.** *Let $T : \mathbb{N} \to \mathbb{N}$ and $T(n) \geq n$ for every $n \in \mathbb{N}$. $T(n)$ is said to be QRAM-time constructible, if there is an $O(T(n))$-time QRAM P such that for every $x \in \{0,1\}^*$,*

$$P(x, T(|x|)) = 1,$$

where $T(|x|)$ denotes its binary form as in Definition 2.6.

The relationship between QRAMs and QTMs is then established in the following:

Theorem 4.2. 1. *Let $T : \mathbb{N} \to \mathbb{N}$. Suppose P is a $T(n)$-time QRAM. Then there is a well-formed and normal form QTM M within time $T'(n)$ such that*

   (a) $P(x, y) = M(x, y)$ for every $x, y \in \{0,1\}^*$.
   (b) $\mathbb{C}(M) = \{0, 1, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, \exp(i\pi/4)\}$.

   Moreover,

   (a) If $l(n)$ is logarithmic, then $T'(n) = O(T(n)^4)$.
   (b) If $l(n)$ is constant, then $T'(n) = O(T(n)^8)$.

2. *Let $T : \mathbb{N} \to \mathbb{N}$ be a QRAM-time constructible function, and $\lambda : \mathbb{N} \to \mathbb{N}$ with $\lambda(n) \geq n$. For every standard QTM M with exact time $T(n)$ and $\mathbb{C}(M) \subseteq \mathbb{C}(\lambda(n))$, there is a constant $c > 0$ such that for every $0 < \varepsilon < 1$, there is a $O(T(n)^2(\lambda(\log(T(n)/\varepsilon)))^c)$-time QRAM P such that $|M(x, y) - P(x, y)| < \varepsilon$ for every $x, y \in \{0,1\}^*$.*

It is well-known [6] that square roots are computable in polynomial time in n with precision n, and thus $\sqrt{2} \in \tilde{\mathbb{C}}$. So, it holds that $\mathbb{C}(M) \subseteq \tilde{\mathbb{C}}$ in the first part of the above theorem.

A combination of the above two theorems and Theorem 2.1 indicates that QTMs, QRAMs and QRASPs can simulate each other with polynomial slowdown.

The above results have some simple corollaries on quantum complexity classes. To present them, let us first recall the definitions of **EQP** and **BQP** from [4, 1].

Definition 4.2. *Let $L \subseteq \{0,1\}^*$. The language L is said to be in* ***EQP****, if there is a well-formed, normal form and stationary multi-track QTM M with exact time $T(n)$, satisfying:*

1. *$x \in L$ if and only if $M(x,1) = 1$;*
2. *$x \notin L$ if and only if $M(x,1) = 0$;*
3. *$T(n)$ is a polynomial in n.*

The language L is said to be in ***BQP****, if there is a well-formed, normal form, stationary, multi-track QTM M with exact time $T(n)$,*

1. *$x \in L$ if and only if $M(x,1) \geq \frac{2}{3}$;*
2. *$x \notin L$ if and only if $M(x,1) \leq \frac{1}{3}$;*
3. *$T(n)$ is a polynomial in n.*

The complexity classes **EQP'** and **BQP'** are defined by removing the stationary condition and allowing QTMs to be within time $T(n)$ in the above definition. Immediately from Theorem 2.1, we have:

Proposition 4.3. ***EQP*** $=$ ***EQP'*** *and* ***BQP*** $=$ ***BQP'****.*

The QTM in the first part of Theorem 4.2 is only guaranteed to be well-formed and normal form, but not to halt at an exact time. But Theorem 2.1 can be employed to strengthen it by a standard QTM with exact time $O(T(n)^4 \log^2 T(n))$ for logarithmic $l(n)$, and $O(T(n)^8 \log^2 T(n))$ for constant $l(n)$, provided $T(n)$ is time constructible.

Let **EQRAMP** and **BQRAMP** denote the classes of languages that are computable by exact and bounded-error quantum random access machine in polynomial time, respectively (see Section 5.3 for their formal definitions). Then by Theorem 4.2 and Proposition 4.3, immediately we have:

Theorem 4.4. ***P*** $\subseteq$ ***EQRAMP*** $\subseteq$ ***EQP*** *and* ***BQP*** $=$ ***BQRAMP****.*

Note that **BQP** and **BQRAMP** coincide, but it seems that **EQP** and **EQRAMP** do not. This is because a QRAM only has a finite number of quantum gate operations, while a QTM can have an infinite (but countable) number of quantum gate operations. It is almost impossible to simulate an infinite set of quantum gates by a finite set of quantum gates with no errors. On the other hand, in the definitions of **EQP** and **EQRAMP**, the probabilities are restricted to 0 and 1, which makes it unlikely that the two complexity classes coincide.

5 Computations of QRAMs

For this section on, we provide all of the details for proving our main results. In this section, we carefully describe the computations of QRAMs. It is presented in the forms of operational and denotational semantics of QRAMs in Subsections 5.1 and 5.2, respectively. QRAM computation and the notion of two complexity classes **EQRAMP** and **BQRAMP** are formally defined in Section 5.3.

Section 5.4 gives a useful method for shifting addresses, based on which we show that every QRAM can be simulated by an address-safe QRAM in the sense that it never accesses invalid addresses. Postponing measurements is a widely used technique in quantum computing. In Section 5.5, we introduce the notion of measurement-postponed QRAMs.

5.1 Operational semantics

Formally, a QRAM is represented by a sequence $P = P_0, P_1, P_2, \ldots, P_{L-1}$ of instructions with $L = |P|$ being the length of P, i.e. the number of instructions in this QRAM. In the execution of a QRAM, there is an instruction counter (IC) indicating which instruction to be executed. A configuration of the QRAM is a tuple $(\xi, \mu, |\psi\rangle, x, y)$, where:

1. $\xi \in \mathbb{N} \cup \{\downarrow\}$ denotes the current IC, with $\downarrow$ indicating the end of execution;
2. $\mu : \mathbb{N} \to \mathbb{Z}$ is the description of all contents of classical registers;
3. $|\psi\rangle \in \mathcal{H} = \bigotimes_{i=0}^{\infty} \mathcal{H}_i$ is the state of quantum registers, with $\mathcal{H}_i = \operatorname{span}\{|0\rangle_i, |1\rangle_i\}$;
4. $x \in \mathbb{Z}^\omega$ is a sequence of integers to read on the input tape;
5. $y \in \mathbb{Z}^*$ is a sequence of printed integers on the output tape.

We write $\mathcal{C} = (\mathbb{N} \cup \{\downarrow\}) \times \mathbb{Z}^{\mathbb{N}} \times \mathcal{H} \times \mathbb{Z}^\omega \times \mathbb{Z}^*$ for the set of all configurations. A configuration $c = (\xi, \mu, |\psi\rangle, x, y) \in \mathcal{C}$ is called a terminal configuration if $\xi = \downarrow$. Let $\mathcal{C}_f \subseteq \mathcal{C}$ denote the set of all terminal configurations.

The execution transition is a function $\to: \mathcal{C} \times \mathcal{C} \to [0,1] \times \mathbb{N}$. For two configurations c and c', $\to (c, c') = (p, T)$ means that configuration c is changed to configuration c' in time T with probability p after executing one instruction. For readability, we write

$$c \xrightarrow[T]{p} c',$$

wherein p can be ignored if $p = 1$. Then the operational semantics of QRAMs is defined by the following transitional rules:

1. If ξ is out of range $[0, L)$,
$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[1]{} (\downarrow, \mu, |\psi\rangle, x, y).$$
2. If P_ξ has the form $X_i \leftarrow C$,
$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[1]{} (\xi + 1, \mu_i^C, |\psi\rangle, x, y),$$
where
$$\mu_a^b = \begin{cases} \mu(j) & j \neq a, \\ b & j = a. \end{cases}$$
3. If P_ξ has the form $X_i \leftarrow X_j + X_k$,
$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(j))+l(\mu(k))]{} (\xi + 1, \mu_i^{\mu(j)+\mu(k)}, |\psi\rangle, x, y).$$
4. If P_ξ has the form $X_i \leftarrow X_j - X_k$,
$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(j))+l(\mu(k))]{} (\xi + 1, \mu_i^{\mu(j)-\mu(k)}, |\psi\rangle, x, y).$$

5. If P_ξ has the form $X_i \leftarrow X_{X_j}$, whenever $\mu(j) \geq 0$, then

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(j))+l(\mu(\mu(j)))]{} (\xi+1, \mu_i^{\mu(\mu(j))}, |\psi\rangle, x, y);$$

otherwise,

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(j))]{} (\downarrow, \mu, |\psi\rangle, x, y).$$

6. If P_ξ has the form $X_{X_i} \leftarrow X_j$, whenever $\mu(i) \geq 0$, then

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(i))+l(\mu(j))]{} (\xi+1, \mu_{\mu(i)}^{\mu(j)}, |\psi\rangle, x, y);$$

otherwise,

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(i))]{} (\downarrow, \mu, |\psi\rangle, x, y).$$

7. If P_ξ has the form TRA m if $X_j > 0$, whenever $\mu(j) > 0$, then

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(j))]{} (m, \mu, |\psi\rangle, x, y);$$

otherwise,

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(j))]{} (\xi+1, \mu, |\psi\rangle, x, y).$$

8. If P_ξ has the form READ X_i, let a be the first integer in x, then

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(a)]{} (\xi+1, \mu_i^a, |\psi\rangle, x', y),$$

where x' denotes the string obtained by deleting the first integer x.

9. If P_ξ has the form WRITE X_i,

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(i))]{} (\xi+1, \mu, |\psi\rangle, x, y'),$$

where y' denotes the string obtained by appending an integer $\mu(i)$ to y.

10. If P_ξ has the form CNOT$[Q_{X_i}, Q_{X_j}]$, whenever $\mu(i) \geq 0$ and $\mu(j) \geq 0$, then

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(i))+l(\mu(j))]{} (\xi+1, \mu, \mathrm{CNOT}_{\mu(i),\mu(j)} |\psi\rangle, x, y),$$

where for $a_0, a_1, a_2, \dots \in \{0,1\}$, $\mathrm{CNOT}_{i,j} |a_0, a_1, a_2, \dots\rangle = |b_0, b_1, b_2, \dots\rangle$ with

$$b_k = \begin{cases} a_k & \text{if } k \neq j, \\ a_i \oplus a_j & \text{otherwise,} \end{cases}$$

and $\oplus$ denotes modulo-2 addition; otherwise,

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(i))+l(\mu(j))]{} (\downarrow, \mu, |\psi\rangle, x, y).$$

11. If P_ξ has the form $A[Q_{X_i}]$ with $A = H$ or T, whenever $\mu(i) \geq 0$, then

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(i))]{} (\xi + 1, \mu, A_{\mu(i)}|\psi\rangle, x, y),$$

where

$$A_i \bigotimes_{j=0}^{\infty} |a_j\rangle = \bigotimes_{j=0}^{i-1} |a_j\rangle \otimes A|a_i\rangle \otimes \bigotimes_{j=i+1}^{\infty} |a_j\rangle$$

for $a_0, a_1, a_2, \dots \in \{0, 1\}$; otherwise,

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(i))]{} (\downarrow, \mu, |\psi\rangle, x, y).$$

12. If P_ξ has the form $X_i \leftarrow M[Q_{X_j}]$, whenever $\mu(j) \geq 0$, then

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(j))]{\|P_j|\psi\rangle\|^2} (\xi + 1, \mu_i^0, \frac{P_j|\psi\rangle}{\|P_j|\psi\rangle\|^2}, x, y),$$

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(j))]{1-\|P_j|\psi\rangle\|^2} (\xi + 1, \mu_i^1, \frac{(I - P_j)|\psi\rangle}{1 - \|P_j|\psi\rangle\|^2}, x, y),$$

where

$$P_j = \bigotimes_{i=0}^{j-1} I_i \otimes |0\rangle_j \langle 0| \otimes \bigotimes_{i=j+1}^{\infty} I_i;$$

otherwise,

$$(\xi, \mu, |\psi\rangle, x, y) \xrightarrow[l(\mu(i))]{} (\downarrow, \mu, |\psi\rangle, x, y).$$

5.2 Denotational semantics

An execution path $\pi = c_0, c_1, \dots, c_n$ is a non-empty sequence of configurations, i.e. $\pi \in \mathcal{C}^+$. We define the length of π as $|\pi| = n$. A path is called terminal, if c_n is a terminal configuration. For readability, an execution path π is usually written as

$$\pi : c_0 \xrightarrow[T_1]{p_1} c_1 \xrightarrow[T_2]{p_2} c_2 \xrightarrow[T_3]{p_3} \dots \xrightarrow[T_{n-1}]{p_{n-1}} c_{n-1} \xrightarrow[T_n]{p_n} c_n,$$

where for every $1 \leq i \leq n$, $c_{i-1} \xrightarrow[T_i]{p_i} c_i$ is a transition defined in the above subsection. We may simply write:

$$\pi : c_0 \xrightarrow[T]{p}{}^n c_n,$$

where

$$T = \sum_{i=1}^{n} T_i, \qquad p = \prod_{i=1}^{n} p_i.$$

It should be noted that there could be multiple paths of the form $c_0 \xrightarrow[T]{p}{}^n c_n$ with different pairs of p and T.

For simplicity of presentation, let us introduce several abbreviations:

- $\pi.p_1$ (resp. $\pi.T_1$) denotes the transition probability (resp. time) in the first step of π.

- $\pi.p$ (resp. $\pi.T$) denotes the transition probability (resp. time) of π.
- $\pi.c_{|\pi|}.\xi = \downarrow$ means that the last configuration of π is a terminal configuration; that is, $\pi.c_{|\pi|} \in \mathcal{C}_f$, where $|\pi|$ is the length of π.

Moreover, for a QRAM P, we use the following notations:

- $\mathcal{P}$ denotes the set of all execution paths of P.
- $\mathcal{P}(c)$ denotes the set of all execution paths starting from c (with positive probabilities), i.e.
$$\mathcal{P}(c) = \{\pi \in \mathcal{P} : \pi.c_0 = c \text{ and } \pi.p > 0\}.$$
- $\mathcal{P}_f(c)$ denotes the set of all terminal execution paths starting from c, i.e.
$$\mathcal{P}_f(c) = \{\pi \in \mathcal{P}(c) : \pi.c_{|\pi|} \in \mathcal{C}_f\}.$$
- $\mathcal{P}^{=n}(c)$ denotes the set of all execution paths of length n, starting from c, i.e.
$$\mathcal{P}^{=n}(c) = \{\pi \in \mathcal{P}(c) : |\pi| = n\}.$$
- $\mathcal{P}_f^{=n}(c)$ denotes the set of all terminal execution paths of length n, starting from c, i.e.
$$\mathcal{P}_f^{=n}(c) = \{\pi \in \mathcal{P}^{=n}(c) : \pi.c_n \in \mathcal{C}_f\}.$$
- $\mathcal{P}^n(c)$ denotes the set of all execution paths starting from c within n steps, i.e.
$$\mathcal{P}^n(c) = \bigcup_{m=0}^{n} \mathcal{P}_f^{=m}(c).$$

Definition 5.1. *1. The n-step semantics function $[\![P]\!]^n : \mathcal{C} \to (\mathcal{C} \to [0,1])$ is defined by*
$$[\![P]\!]^n(c) = \sum_{\pi \in \mathcal{P}^n(c)} \pi.p \cdot \mathbb{I}_{\pi.c_{|\pi|}},$$
where
$$\mathbb{I}_c(c') = \begin{cases} 1 & c = c', \\ 0 & \textit{otherwise}. \end{cases}$$

2. The semantic function $[\![P]\!] : \mathcal{C} \to (\mathcal{C} \to [0,1])$ is defined by
$$[\![P]\!](c) = \lim_{n\to\infty} [\![P]\!]^n(c).$$

It should be noted that $[\![P]\!](c)$ may not exist.

Definition 5.2. *The worst case running time $\tau_P : \mathcal{C} \to \mathbb{N} \cup \{\infty\}$ is defined by*
$$\tau_P(c) = \sup\{\pi.T : \pi \in \mathcal{P}(c)\}.$$

The following lemma is straightforward:

Lemma 5.1. *For any $c \in \mathcal{C}$, we have:*

1. *if $\tau_P(c) < \infty$, then $[\![P]\!](c) = [\![P]\!]^{\tau_P(c)}(c)$;*
2. *if $\tau_P(c) < \infty$ and $[\![P]\!](c)(c') > 0$ for some $c' \in \mathcal{C}$, then $c' \in \mathcal{C}_f$.*

5.3 QRAM computations

We are interested in the time required for a QRAM to recognize a language on a finite alphabet $\Sigma = \{\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_{m-1}\}$. An input string $x = \sigma_{i_0}\sigma_{i_1}\dots\sigma_{i_{n-1}}$ is represented in the machine as the sequence of integers $i_0, i_1, \dots, i_{n-1}, (-1)^\omega$, where the infinite occurrences of -1 at the end of the sequence indicate the end of the string. This input convention guarantees that whenever an instruction of the form READ X_i is executed, X_i always obtain -1 if the string has been read up. We use $in : \Sigma^* \to \mathbb{Z}^\omega$ to denote this conversion from an input string x to the contents on input tape $in(x)$.

After the execution of the QRAM, a finite sequence y of integers is obtained on the output tape. In order to extract the output string on Σ from the contents on the output tape, we define $out : \mathbb{Z} \to \Sigma$ by

$$out(n) = \begin{cases} \sigma_n & 0 \le n < m-1, \\ \sigma_{m-1} & \text{otherwise.} \end{cases}$$

This function can be extended to $out : \mathbb{Z}^* \to \Sigma^*$ by concatenation of each single conversion.

Example 1. *Consider the simplest alphabet $\Sigma = \{0, 1\}$. The input string $x = 0101$ is converted to $in(x) = 0, 1, 0, 1, -1, -1, -1, \dots$. An output string $out(y) = 0111$ is extracted from the contents $y = 0, 1, 2, 3$ on the output tape.*

With the above input/output conventions, we can now describe the notion of QRAM computation. Before the computation, IC is set to 0 initially, with all classical registers being zero and all quantum registers being $|0\rangle$; that is, $\mu_0 = 0$ and

$$|\psi_0\rangle = \bigotimes_{i=0}^{\infty} |0\rangle_i \,.$$

Suppose the input string is x, then the initial configuration is $c_x = (0, \mu_0, |\psi_0\rangle, in(x), \epsilon)$. The distribution $\mathcal{D} : \Sigma^* \to (\mathcal{C} \to [0,1])$ of terminal configurations is:

$$\mathcal{D}(x) = [\![P]\!](c_x).$$

The computational result $P : \Sigma^* \times \Sigma^* \to [0,1]$ of QRAM P is defined by

$$P(x, y) = \sum_{c \in \mathcal{C}_f : out(c.y) = y} \mathcal{D}(x)(c),$$

and the worst case running time $T_P : \Sigma^* \to \mathbb{N} \cup \{\infty\}$ is defined by

$$T_P(x) = \tau_P(c_x).$$

Definition 5.3. *Let $T : \mathbb{N} \to \mathbb{N}$. P is said to be a $T(n)$-time QRAM, if for every $x \in \Sigma^*$, $T_P(x) \le T(|x|)$.*

In particular, P is said to be a polynomial-time QRAM, if it is a $p(n)$-time QRAM for some polynomial p. Furthermore, two complexity classes are defined as follows:

- **EQRAMP** stands for Exact Quantum Random Access Machine Polynomial-time. More precisely, a language $L \subseteq \{0,1\}^*$ is said to be in **EQRAMP**, if there is a polynomial-time QRAM P such that for every $x \in \{0,1\}^*$,

1. $x \in L \iff P(x, 1) = 1$,
2. $x \notin L \iff P(x, 1) = 0$.

- **BQRAMP** stands for Bounded-error Quantum Random Access Machine Polynomial-time. More precisely, a language $L \subseteq \{0, 1\}^*$ is said to be in **BQRAMP**, if there is a polynomial-time QRAM P such that for every $x \in \{0, 1\}^*$,

  1. $x \in L \iff P(x, 1) \geq \frac{2}{3}$,
  2. $x \notin L \iff P(x, 1) \leq \frac{1}{3}$.

5.4 Address Shifting and Address-safe QRAMs

In order to describe algorithms more conveniently, we introduce the technique of address shifting, which enables us to flexibly deal with free variables.

Lemma 5.2 (Address Shifting)**.** *Let $T : \mathbb{N} \to \mathbb{N}$. For every $T(n)$-time QRAM P and every integer $k > 0$, there is an $O(T(n))$-time QRAM P' such that*

1. *$P(x, y) = P'(x, y)$ for every $x, y \in \Sigma^*$.*
2. *P' never accesses to the classical registers $X_1, X_2, \ldots, X_k$.*

With the help of Lemma 5.2, we show that address-safety can be enforced for QRAMs.

Lemma 5.3. *Let $T : \mathbb{N} \to \mathbb{N}$. For every $T(n)$-time QRAM P, there is a $O(T(n))$-time QRAM P' such that*

1. *$P(x, y) = P'(x, y)$ for every $x, y \in \Sigma^*$.*
2. *P' never accesses to an invalid address in the execution. We call such a QRAM P' address-safe.*

The proofs of Lemma 5.2 and Lemma 5.3 are given in Appendix A.

5.5 Measurement-postponed QRAMs

In this subsection, we generalise the technique of postponing measurements, which has been widely used in quantum computing, to QRAMs.

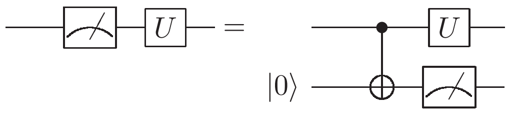
Definition 5.4. *A QRAM is said to be measurement-postponed, if no further operations are performed on the quantum registers once they are measured.*

Theorem 5.4. *Let $T : \mathbb{N} \to \mathbb{N}$. For every $T(n)$-time QRAM P, there is a $O(T(n))$-time QRAM P' such that*

1. *$P(x, y) = P'(x, y)$ for every $x, y \in \Sigma^*$.*
2. *P' is measurement-postponed.*

Proof. Suppose P consists of L instructions $P_0, P_1, \ldots, P_{L-1}$. By Lemma 5.3, we can assume that P is address-safe. In order to postpone measurements, we recall the technique shown in Figure 1. Inspired by this, we use a special variable *mea* to count how many measurements are performed. We split quantum registers into two disjoint parts, one of which is of even addresses and the

Figure 1: Quantum circuits for postponing measurements



other is of odd addresses. The quantum registers of even addresses are used for quantum gate operations while the rest (i.e. those of odd addresses) are used only for measurements. For a better understanding, we first give an intuition behind our construction. We use two functions $f(x) = 2x$ and $g(x) = 2x + 1$. For a quantum gate (CNOT, Hadamard and $\pi/8$ gates), say CNOT$[Q_a, Q_b]$, we can perform CNOT$[Q_{f(a)}, Q_{f(b)}]$. For a measurement, say $M[Q_a]$, let *mea* be the current number of measurements that have been performed, then we can perform CNOT$[Q_{f(a)}, Q_{g(mea)}]$ and then measure $Q_{g(mea)}$, i.e. $M[Q_{g(mea)}]$.

In order to precisely describe how to postpone measurements, we first list the lengths needed for all instructions in Table 3. For $0 \le l < L$, we write *length*(l) for the length needed for postponing measurements according to Table 3. To label the instructions in P', we define:

$$\mathit{label}(l) = \sum_{i=0}^{l-1} \mathit{length}(i)$$

for $0 \le l \le L$. Especially, the length of P' is defined to be $L' = \mathit{label}(L)$.

Table 3: Lengths of QRAM instructions for postponing measurements

Type	Instruction	length
Classical	$X_i \leftarrow C$, C any integer	1
Classical	$X_i \leftarrow X_j + X_k$	1
Classical	$X_i \leftarrow X_j - X_k$	1
Classical	$X_i \leftarrow X_{X_j}$	1
Classical	$X_{X_i} \leftarrow X_j$	1
Classical	TRA m if $X_j > 0$	1
Classical	READ X_i	1
Classical	WRITE X_i	1
Quantum	CNOT$[Q_{X_i}, Q_{X_j}]$	7
Quantum	$H[Q_{X_i}]$	4
Quantum	$T[Q_{X_i}]$	4
Measurement	$X_i \leftarrow M[Q_{X_j}]$	12

Now we are ready to describe our construction of P'. For every $0 \le l < L$, we convert P_l to one or more instructions in P'. Note that we need three extra variables mea, a and b.

Case 1. If P_l is CNOT$[Q_{X_i}, Q_{X_j}]$, the instructions for P' are as follows.

```
$label(l)$ :$a \leftarrow 0$
        $a \leftarrow a + X_i$
        $a \leftarrow a + a$
        $b \leftarrow 0$
        $b \leftarrow b + X_j$
        $b \leftarrow b + b$
        CNOT$[Q_a, Q_b]$
```

Case 2. If P_l is $H[Q_{X_i}]$ (resp. $T[X_{X_i}]$), the instructions for P' are as follows.

```
$label(l)$ :$a \leftarrow 0$
        $a \leftarrow a + X_i$
        $a \leftarrow a + a$
        $H[Q_a]$(resp. $T[Q_a]$)
```

Case 3. If P_l is $X_i \leftarrow M[Q_{X_j}]$, the instructions for P' are as follows.

```
$label(l)$ :$a \leftarrow 1$
        $mea \leftarrow mea + a$
        $b \leftarrow 0$
        $b \leftarrow b + mea$
        $b \leftarrow b + b$
        $a \leftarrow 1$
        $b \leftarrow b + a$
        $a \leftarrow 0$
        $a \leftarrow a + X_j$
        $a \leftarrow a + a$
        CNOT$[Q_a, Q_b]$
        $X_i \leftarrow M[Q_b]$
```

Case 4. If P_l is jumping, i.e. TRA m if $X_j > 0$, we use a single modified instruction

$$label(l) : \text{TRA } m' \text{ if } X_{j+\delta} > 0$$

with $m' = label(m)$.

Case 5. For other cases, use the same instructions as in P.

It can be seen that the constructed QRAM P' simulates QRAM P with a constant factor slowdown. Since mea always increases, no quantum register will be measured more than once. To the end, according to Lemma 5.2, the variables mea, a and b are involved by shifting the address to the right by $\delta = 4$. □

Remark 5.1. *Since the CNOT gate is considered to be more expensive than a single-qubit gate in the current generation of quantum hardware (for more details about the cost of quantum gates, see [18]), one may be concerned about how many CNOT gates are used in the measurement-postponed simulation. In the proof of Theorem 5.4, we see that every time a measurement is performed in the simulated QRAM, an additional qubit is introduced and an extra CNOT gate is performed in the simulation. Therefore, if the simulated QRAM performs measurements for m times, the measurement-postponed simulation will need m additional qubits and will require m extra CNOT gates.*

6 Computations of QRASPs

In this section, we define the computations of QRASPs in terms of operational and denotational semantics, in parallel with what we did for QRAMs in the last section.

6.1 Operational semantics

A configuration of a QRASP P is a tuple $(\xi, \zeta, \mu, |\psi\rangle, x, y)$, where:

1. $\xi \in \mathbb{N} \cup \{\downarrow\}$ denotes the current IC, with $\downarrow$ indicating the end of execution.
2. $\zeta \in \mathbb{Z}$ denotes the current AC.
3. $\mu : \mathbb{N} \to \mathbb{Z}$ is the description of all contents of classical registers;
4. $|\psi\rangle \in \mathcal{H} = \bigotimes_{i=0}^{\infty} \mathcal{H}_i$ is the state of all quantum registers, with $\mathcal{H}_i = \text{span}\{|0\rangle_i, |1\rangle_i\}$;
5. $x \in \mathbb{Z}^\omega$ is a sequence of integers to read on the input tape;
6. $y \in \mathbb{Z}^*$ is a sequence of printed integers on the output tape.

We write $\mathcal{C} = (\mathbb{N} \cup \{\downarrow\}) \times \mathbb{Z} \times \mathbb{Z}^{\mathbb{N}} \times \mathcal{H} \times \mathbb{Z}^\omega \times \mathbb{Z}^*$ for the set of all configurations. A configuration $c = (\xi, \zeta, \mu, |\psi\rangle, x, y) \in \mathcal{C}$ is a terminal configuration if $\xi = \downarrow$. Let $\mathcal{C}_f \subseteq \mathcal{C}$ denote the set of all terminal configurations.

Similar to the case of QRAMs, the execution transition of a QRASP P is a function $\rightarrow: \mathcal{C} \times \mathcal{C} \to [0,1] \times \mathbb{N}$ defined by the following rules:

1. When $\mu(\xi)$ is beyond $[1, 11]$,
$$(\xi, \zeta, \mu, |\psi\rangle, x, y) \xrightarrow[l(\xi)+l(\mu(\xi))]{} (\downarrow, \zeta, \mu, |\psi\rangle, x, y).$$
2. When $\mu(\xi) = 1$,
$$(\xi, \zeta, \mu, |\psi\rangle, x, y) \xrightarrow[l(\xi)+l(\mu(\xi+1))]{} (\xi + 2, \mu(\xi + 1), \mu, |\psi\rangle, x, y).$$
3. When $\mu(\xi) = 2$, if $\mu(\xi + 1) \geq 0$, then
$$(\xi, \zeta, \mu, |\psi\rangle, x, y) \xrightarrow[l(\xi)+l(\mu(\xi+1))+l(\zeta)+l(\mu(\mu(\xi+1)))]{} (\xi + 2, \zeta + \mu(\mu(\xi + 1)), \mu, |\psi\rangle, x, y);$$
otherwise,
$$(\xi, \zeta, \mu, |\psi\rangle, x, y) \xrightarrow[l(\xi)+l(\mu(\xi+1))]{} (\downarrow, \zeta, \mu, |\psi\rangle, x, y).$$

4. When $\mu(\xi)=3$, if $\mu(\xi+1)\geq 0$, then

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))+l(\zeta)+l(\mu(\mu(\xi+1)))]{}(\xi+2,\zeta-\mu(\mu(\xi+1)),\mu,|\psi\rangle,x,y);$$

otherwise,

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))]{}(\downarrow,\zeta,\mu,|\psi\rangle,x,y).$$

5. When $\mu(\xi)=4$, if $\mu(\xi+1)\geq 0$, then

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))+l(\zeta)]{}(\xi+2,\zeta,\mu^{\zeta}_{\mu(\xi+1)},|\psi\rangle,x,y);$$

otherwise,

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))]{}(\downarrow,\zeta,\mu,|\psi\rangle,x,y).$$

6. When $\mu(\xi)=5$, if $\zeta>0$, then:
   (a) if $\mu(\xi+1)\geq 0$,

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))+l(\zeta)]{}(\mu(\xi+1),\zeta,\mu,|\psi\rangle,x,y),$$

   (b) if $\mu(\xi+1)<0$,

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))+l(\zeta)]{}(\downarrow,\zeta,\mu,|\psi\rangle,x,y);$$

   otherwise,

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\zeta)]{}(\xi+2,\zeta,\mu,|\psi\rangle,x,y).$$

7. When $\mu(\xi)=6$, if $\mu(\xi+1)\geq 0$, then let a be the first integer in x, and

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))+l(a)]{}(\xi+2,\zeta,\mu^{a}_{\mu(\xi+1)},|\psi\rangle,x',y),$$

   where x' denotes the string obtained by deleting the first integer from x; otherwise,

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))]{}(\downarrow,\zeta,\mu,|\psi\rangle,x,y).$$

8. When $\mu(\xi)=7$, if $\mu(\xi+1)\geq 0$, then

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))+l(\mu(\mu(\xi+1)))]{}(\xi+2,\zeta,\mu,|\psi\rangle,x,y'),$$

   where y' denotes the string obtained by appending $\mu(\mu(\xi+1))$ to y; otherwise,

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))]{}(\downarrow,\zeta,\mu,|\psi\rangle,x,y).$$

9. When $\mu(\xi)=8$, if $\mu(\xi+1)\geq 0$ and $\mu(\xi+2)\geq 0$, then

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))+l(\mu(\xi+2))]{}(\xi+3,\zeta,\mu,\mathrm{CNOT}_{\mu(\xi+1),\mu(\xi+2)}|\psi\rangle,x,y);$$

   otherwise,

$$(\xi,\zeta,\mu,|\psi\rangle,x,y)\xrightarrow[l(\xi)+l(\mu(\xi+1))]{}(\downarrow,\zeta,\mu,|\psi\rangle,x,y).$$

10. When $\mu(\xi) = 9$, if $\mu(\xi+1) \geq 0$, then

$$(\xi, \zeta, \mu, |\psi\rangle, x, y) \xrightarrow[l(\xi)+l(\mu(\xi+1))]{} (\xi+2, \zeta, \mu, H_{\mu(\xi+1)} |\psi\rangle, x, y);$$

otherwise,

$$(\xi, \zeta, \mu, |\psi\rangle, x, y) \xrightarrow[l(\xi)+l(\mu(\xi+1))]{} (\downarrow, \zeta, \mu, |\psi\rangle, x, y).$$

11. When $\mu(\xi) = 10$, if $\mu(\xi+1) \geq 0$, then

$$(\xi, \zeta, \mu, |\psi\rangle, x, y) \xrightarrow[l(\xi)+l(\mu(\xi+1))]{} (\xi+2, \zeta, \mu, T_{\mu(\xi+1)} |\psi\rangle, x, y);$$

otherwise,

$$(\xi, \zeta, \mu, |\psi\rangle, x, y) \xrightarrow[l(\xi)+l(\mu(\xi+1))]{} (\downarrow, \zeta, \mu, |\psi\rangle, x, y).$$

12. When $\mu(\xi) = 11$, if $\mu(\xi+1) \geq 0$, then

$$(\xi, \zeta, \mu, |\psi\rangle, x, y) \xrightarrow[l(\xi)+l(\mu(\xi+1))]{\|P_{\mu(\xi+1)}|\psi\rangle\|^2} (\xi+2, 0, \mu, \frac{P_{\mu(\xi+1)} |\psi\rangle}{\|P_{\mu(\xi+1)} |\psi\rangle\|^2}, x, y),$$

$$(\xi, \zeta, \mu, |\psi\rangle, x, y) \xrightarrow[l(\xi)+l(\mu(\xi+1))]{1-\|P_{\mu(\xi+1)}|\psi\rangle\|^2} (\xi+2, 1, \mu, \frac{(I - P_{\mu(\xi+1)}) |\psi\rangle}{1 - \|P_{\mu(\xi+1)} |\psi\rangle\|^2}, x, y);$$

otherwise,

$$(\xi, \zeta, \mu, |\psi\rangle, x, y) \xrightarrow[l(\xi)+l(\mu(\xi+1))]{} (\downarrow, \zeta, \mu, |\psi\rangle, x, y).$$

6.2 Denotational semantics

The notions of execution path, semantic function and worst case running time for a QRASP can be defined in the same way as those for a QRAM in Subsection 5.2. Moreover, it is easy to show that Lemma 5.1 holds for QRASPs too.

6.3 QRASP computations

A QRASP is a sequence $P = P_0, P_1, \ldots, P_{L-1}$ of integers with $L = |P|$ the length of P, which is initially stored in the classical registers. More precisely, the initial contents of classical registers are described by

$$\mu_0(\xi) = \begin{cases} P_\xi & 0 \leq \xi < L, \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, initially IC and AC are set to 0 and all quantum registers are $|0\rangle$, i.e.

$$|\psi_0\rangle = \bigotimes_{i=0}^{\infty} |0\rangle_i \,.$$

Similar to the case of QRAMs, the computation of a QRASP P is defined on finite strings over a finite alphabet Σ. Suppose the input string is $x \in \Sigma^*$, then the computation starts from the configuration

$$c_x = (0, 0, \mu_0, |\psi_0\rangle, in(x), \epsilon).$$

The computational result $P : \Sigma^* \times \Sigma^* \to [0,1]$ is then defined by

$$P(x,y) = \sum_{c \in \mathcal{C}_f : out(c.y) = y} [\![P]\!](c_x)(c),$$

and the worst case running time $T_P : \Sigma^* \to \mathbb{N} \cup \{\infty\}$ is defined by

$$T_P(x) = \tau_P(c_x).$$

Furthermore, let $T : \mathbb{N} \to \mathbb{N}$. Then P is said to be a $T(n)$-time QRASP, if for every $x \in \Sigma^*$, $T_P(x) \leq T(|x|)$.

7 Comparison of QRAMs and QRASPs

With the precise definitions of QRAMs and QRASPs given in the previous sections, we are now ready to prove Theorem 4.1. Subsection 7.1 shows how QRAMs can simulate QRASPs, and Subsection 7.2 shows how QRASPs can simulate QRAMs.

7.1 QRAMs simulate QRASPs

7.1.1 Simulation construction

Let QRASP P be given as a sequence $P_0, P_1, P_2, \ldots, P_{L-1}$ of integers. The idea of the simulation is to hardcode P into the classical registers of a QRAM P', and then simulate the execution of P. The details of the simulation are presented in Algorithm 1. For readability, we only present P' as pseudo-codes. (The translation from the pseudo-codes to QRAM instructions can be done in a familiar way, and the details are provided in Appendix B for completeness).

7.1.2 Correctness proof

The remaining part of this subsection is devoted to prove correctness of Algorithm 1; that is, for any QRASP P, the QRAM P' constructed by Algorithm 1 can simulate P with a suitable time complexity.

Let $(\xi, \zeta, \mu, |\psi\rangle, x, y)$ be a configuration of P and $(\xi', \mu', |\psi'\rangle, x', y')$ a configuration of P'. We use $\mu'(var)$ to denote the value of variable var stored in P' according to μ'.

Definition 7.1. *We say that a QRAM configuration $(\xi', \mu', |\psi'\rangle, x', y')$ agrees with a QRASP configuration $(\xi, \zeta, \mu, |\psi\rangle, x, y)$, written $(\xi', \mu', |\psi'\rangle, x', y') \models (\xi, \zeta, \mu, |\psi\rangle, x, y)$, if*

1. *$\mu'(AC) = \zeta$, ($\mu'(flag) = 1$ or $\xi' = \downarrow$), $\mu'(memory[j]) = \mu(j)$ for every $j \in \mathbb{N}$, $|\psi'\rangle = |\psi\rangle$, $x' = x$ and $y' = y$ in the case $\xi = \downarrow$; or*
2. *$\mu'(AC) = \zeta$, $\mu'(IC) = \xi$, $\mu'(memory[j]) = \mu(j)$ for every $j \in \mathbb{N}$, $|\psi'\rangle = |\psi\rangle$, $x' = x$ and $y' = y$ in the case $\xi \in \mathbb{N}$.*

Lemma 7.1. *For every $c' \in \mathcal{C}'$, there is a unique $c \in \mathcal{C}$ such that $c' \models c$.*

Proof. We only need to observe: (1) if $c' \models c_1$ and $c' \models c_2$, then $c_1 = c_2$; and (2) for every c', there is a c such that $c' \models c$. □

Let $\mathcal{C}$ and $\mathcal{C}'$ be the set of configurations of P and P', respectively, and let $c_0 \in \mathcal{C}$ and $c_0' \in \mathcal{C}'$ be their initial configurations. We write $\mathcal{C}'_{\mathrm{L5}} = \{c' \in \mathcal{C}' : c'.\xi = 5\}$ for the set of configurations of P' that reaches Line 5 in Algorithm 1 (Here, we use the line number to indicate the current IC).

Algorithm 1 QRAM pseudo-code for simulating QRASP.

Input: The input of the QRASP to be simulated.

Output: The intended output of the QRASP to be simulated.

```
integer array $memory$;
integer $\mathrm{IC}, \mathrm{AC}, flag, op, j, k$;
$memory[0] \leftarrow P_0; memory[1] \leftarrow P_1; \dots memory[L-1] \leftarrow P_{L-1}$;
while $flag = 0$ do
    $op \leftarrow memory[\mathrm{IC}]$;
    if $op = 1$ then
        $j \leftarrow memory[\mathrm{IC}+1]$; $\mathrm{AC} \leftarrow j$; $\mathrm{IC} \leftarrow \mathrm{IC}+2$;
    else if $op = 2$ then
        $j \leftarrow memory[\mathrm{IC}+1]$; $\mathrm{AC} \leftarrow \mathrm{AC} + memory[j]$; $\mathrm{IC} \leftarrow \mathrm{IC}+2$;
    else if $op = 3$ then
        $j \leftarrow memory[\mathrm{IC}+1]$; $\mathrm{AC} \leftarrow \mathrm{AC} - memory[j]$; $\mathrm{IC} \leftarrow \mathrm{IC}+2$;
    else if $op = 4$ then
        $j \leftarrow memory[\mathrm{IC}+1]$; $memory[j] \leftarrow \mathrm{AC}$; $\mathrm{IC} \leftarrow \mathrm{IC}+2$;
    else if $op = 5$ then
        if $\mathrm{AC} > 0$ then
            $j \leftarrow memory[\mathrm{IC}+1]$; $\mathrm{IC} \leftarrow j$;
        else
            $\mathrm{IC} \leftarrow \mathrm{IC}+2$;
        end if
    else if $op = 6$ then
        $j \leftarrow memory[\mathrm{IC}+1]$; READ $memory[j]$; $\mathrm{IC} \leftarrow \mathrm{IC}+2$;
    else if $op = 7$ then
        $j \leftarrow memory[\mathrm{IC}+1]$; WRITE $memory[j]$; $\mathrm{IC} \leftarrow \mathrm{IC}+2$;
    else if $op = 8$ then
        $j \leftarrow memory[\mathrm{IC}+1]$; $k \leftarrow memory[\mathrm{IC}+2]$; $\mathrm{CNOT}[Q_j, Q_k]$; $\mathrm{IC} \leftarrow \mathrm{IC}+3$;
    else if $op = 9$ then
        $j \leftarrow memory[\mathrm{IC}+1]$; $H[Q_j]$; $\mathrm{IC} \leftarrow \mathrm{IC}+2$;
    else if $op = 10$ then
        $j \leftarrow memory[\mathrm{IC}+1]$; $T[Q_j]$; $\mathrm{IC} \leftarrow \mathrm{IC}+2$;
    else if $op = 11$ then
        $j \leftarrow memory[\mathrm{IC}+1]$; $\mathrm{AC} \leftarrow M[Q_j]$; $\mathrm{IC} \leftarrow \mathrm{IC}+2$;
    else
        $flag \leftarrow 1$;
    end if
end while
```

Lemma 7.2. *Let $c' \in \mathcal{C}'_{L5}$ and $c, d \in \mathcal{C}$. If $c' \models c$, and $c \xrightarrow[T]{p} d$, then there is a $d' \in \mathcal{C}'_{L5}$ such that $d' \models d$ and $c' \xrightarrow[\Theta(T)]{p} d'$.*

Proof. Direct from the operational semantics. □

We use $\mathcal{P}$ and $\mathcal{P}'$ to denote the sets of all possible execution paths of P and P', respectively. Let $\pi' \in \mathcal{P}'_f(c'_0)$ be a path of length $|\pi'| = k$:

$$\pi' : c'_0 \xrightarrow[T'_1]{p'_1} c'_1 \xrightarrow[T'_2]{p'_2} \dots \xrightarrow[T'_{k-1}]{p'_{k-1}} c'_{k-1} \xrightarrow[T'_k]{p'_k} c'_k,$$

and let $0 < i_0 < i_1 < \dots < i_{m-1} < k$ be all indices that $c'_{(j)} = c'_{i_j} \in \mathcal{C}'_{\mathrm{L5}}$ for $0 \le j < m$. Then it can be written as

$$\pi' : c'_0 \xrightarrow[T'_{(0)}]{p'_{(0)}}{}^{*} c'_{(0)} \xrightarrow[T'_{(1)}]{p'_{(1)}}{}^{*} c'_{(1)} \xrightarrow[T'_{(2)}]{p'_{(2)}}{}^{*} \dots \xrightarrow[T'_{(m-1)}]{p'_{(m-1)}}{}^{*} c'_{(m-1)} \xrightarrow[T'_{(m)}]{p'_{(m)}}{}^{*} c'_{(m)} = c'_k,$$

where

$$p'_{(j)} = \prod_{l=i_{j-1}+1}^{i_j} p'_l,$$

and

$$T'_{(j)} = \sum_{l=i_{j-1}+1}^{i_j} T'_l$$

for $0 \le j \le m$, and $i_{-1} = 0, i_m = k$. We define $\|\pi'\| = m$.

Lemma 7.3. *$c'_{(0)} \models c_0$ and $c'_0 \xrightarrow[O(1)]{1}{}^{*} c'_{(0)}$.*

Proof. Obvious. □

Definition 7.2. *Let $\pi' \in \mathcal{P}'_f(c'_{(0)})$ and $\pi \in \mathcal{P}_f(c_0)$. Then we say that π' agrees with π, denoted $\pi' \models \pi$, if*

1. *$\|\pi'\| = |\pi|$.*
2. *$c'_{(j)} \models c_j$ for $0 \le j \le \|\pi'\|$.*
3. *$p'_{(j)} = p_j$ and $T'_{(j)} = \Theta(T_j)$ for $1 \le j \le \|\pi'\|$.*

Lemma 7.4. *For every $\pi' \in \mathcal{P}'_f(c'_{(0)})$, there is a unique $\pi \in \mathcal{P}_f(c_0)$ such that $\pi' \models \pi$.*

Proof. Obvious. □

Lemma 7.4 implies that P' is time bounded by $O(T(n))$. Let $T' : \mathbb{N} \to \mathbb{N}$ be the worst case running time of P'.

Lemma 7.5. *For every $\pi \in \mathcal{P}_f(c_0)$, there is a unique $\pi' \in \mathcal{P}'_f(c'_{(0)})$ such that $\pi' \models \pi$. We use $h : \mathcal{P}(c_0) \to \mathcal{P}'(c'_{(0)})$ to denote this bijection.*

Proof. (**Existence**) Directly by Lemma 7.2.

(**Uniqueness**) For every $\pi \in \mathcal{P}(c_0)$, we choose an arbitrary $\pi' \in \mathcal{P}'(c'_{(0)})$ such that $\pi' \models \pi$ and write $h(\pi) = \pi'$. We note that

$$1 = \sum_{\pi \in \mathcal{P}(c_0)} \pi.p = \sum_{\pi \in \mathcal{P}(c_0)} h(\pi).p \leq \sum_{\pi' \in \mathcal{P}'(c'_{(0)})} \pi'.p = 1,$$

the uniqueness of $h(\pi)$ follows immediately. □

Finally, we are ready to show that P' actually simulates P. Let $x \in \Sigma^*$ be the input string and the initial configuration of P be $c_0 = (0, 0, \mu_0, |\psi_0\rangle, in(x), \epsilon)$. Since P is a $T(n)$-time QRASP, $|\pi| \leq T(|x|)$ is finite for every $\pi \in \mathcal{P}(c_0)$. Now that each transition leads to at most two branches, $|\mathcal{P}(c_0)| \leq 2^{T(|x|)}$ must be finite too. Thus, for every $y \in \Sigma^*$, we have:

$$\begin{aligned}
P(x,y) &= \sum_{c \in \mathcal{C}_f : out(c.y) = y} [\![P]\!](c_0)(c) \\
&= \sum_{c \in \mathcal{C}_f : out(c.y) = y} [\![P]\!]^{T(|x|)}(c_0)(c) \\
&= \sum_{\pi \in \mathcal{P}^{T(|x|)}(c_0) : out(\pi.c_{|\pi|}.y) = y} \pi.p \\
&= \sum_{\pi \in \mathcal{P}^{T(|x|)}(c_0) : out(h(\pi).c_{|f(\pi)|}.y) = y} h(\pi).p \\
&= \sum_{\pi' \in \mathcal{P}'^{T'(|x|)}(c'_{(0)}) : out(\pi'.c_{|\pi'|}.y) = y} \pi'.p \\
&= \sum_{\pi' \in \mathcal{P}'^{T'(|x|)}(c'_0) : out(\pi'.c_{|\pi'|}.y) = y} \pi'.p \\
&= \sum_{c' \in \mathcal{C}'_f : out(c'.y) = y} [\![P']\!]^{T'(|x|)}(c'_0)(c) \\
&= \sum_{c' \in \mathcal{C}'_f : out(c'.y) = y} [\![P']\!](c'_0)(c) \\
&= P'(x,y).
\end{aligned}$$

7.2 QRASPs simulate QRAMs

7.2.1 Simulation construction

Let QRAM P be a sequence $P_0, P_1, \ldots, P_{L-1}$ of QRAM instructions. By Lemma 5.3, we may assume that P is address-safe without any loss of generality. The QRASP P' that simulates QRAM P is defined as follows. Let δ be an integer greater than the length of P', i.e. $\delta > |P'|$. It will be shown later that $\delta = 20L$ (a finite number) is enough. Define the simulating length *simulate*(P_i) of P_i being the length of QRASP code intended to simulate the QRAM instruction P_i. The intended value for *simulate*(P_i) are shown in Table 4 according to the instruction type of P_i. In order to deal with the jump instruction "TRA m if $X_j > 0$", *label*(m) is needed, which is defined to be the

Table 4: Simulating length of QRAM instructions by QRASP

Type	Instruction	Simulating length
Classical	$X_i \leftarrow C$, C any integer	4
Classical	$X_i \leftarrow X_j + X_k$	8
Classical	$X_i \leftarrow X_j - X_k$	8
Classical	$X_i \leftarrow X_{X_j}$	12
Classical	$X_{X_i} \leftarrow X_j$	12
Classical	TRA m if $X_j > 0$	6
Classical	READ X_i	2
Classical	WRITE X_i	2
Quantum	$\mathrm{CNOT}[Q_{X_i}, Q_{X_j}]$	15
Quantum	$H[Q_{X_i}]$	8
Quantum	$T[Q_{X_i}]$	8
Measurement	$X_i \leftarrow M[Q_{X_j}]$	10

jump address in QRASP corresponding to the jump address m in QRAM. More precisely,

$$\mathit{label}(m) = \sum_{i=0}^{m-1} \mathit{simulate}(P_i).$$

The length of our QRASP P' is designed to be $L' = |P'| = \mathit{label}(L)$. For every $0 \le i < L$, the instruction P_i is interpreted into QRASP instructions as $\mathit{simulate}(P_i)$ integers starting from $\mathit{label}(i)$. In other words, the QRASP instructions $P'_{\mathit{label}(i)}, P'_{\mathit{label}(i)+1}, \ldots, P'_{\mathit{label}(i+1)-1}$ are corresponding to QRAM instruction P_i.

Now for $0 \le l < L$, we present the QRASP code for simulating P_l. Here we only display those for quantum instructions (The simulations of classical instructions are standard and thus omitted here; they are provided in Appendix C for completeness). For readability, the QRASP codes are written by means of QRASP mnemonics.

1. P_l is of the form $\mathrm{CNOT}[Q_{X_i}, Q_{X_j}]$. The QRASP code is

$$\begin{aligned}
\mathit{label}(l) :& \mathrm{LOD}, \delta \\
& \mathrm{ADD}, i + \delta \\
& \mathrm{STO}, a + 1 \\
& \mathrm{LOD}, \delta \\
& \mathrm{ADD}, j + \delta \\
& \mathrm{STO}, a + 2 \\
a :& \mathrm{CNOT}, 0, 0
\end{aligned}$$

Note that $a = \mathit{label}(l) + 12$.

2. P_l is of the form $A[Q_{X_i}]$ with $A = H$ or T. The QRASP code is

$$\begin{aligned} \mathit{label}(l) :&\mathrm{LOD}, \delta \\ &\mathrm{ADD}, i+\delta \\ &\mathrm{STO}, a+1 \\ a :&\mathrm{A}, 0 \end{aligned}$$

   Note that $a = \mathit{label}(l) + 6$.

3. P_l is of the form $X_i \leftarrow M[Q_{X_j}]$. The QRASP code is

$$\begin{aligned} \mathit{label}(l) :&\mathrm{LOD}, \delta \\ &\mathrm{ADD}, j+\delta \\ &\mathrm{STO}, a+1 \\ a :&\mathrm{MEA}, 0 \\ &\mathrm{STO}, i+\delta \end{aligned}$$

   Note that $a = \mathit{label}(l) + 6$.

7.2.2 Correctness proof

The proof is similar to that given in Subsection 7.1.2. We put it in Appendix D for completeness.

8 Comparison of QRAMs and QTMs

In this section, we prove Theorem 4.2. Subsection 8.1 shows how QTMs can simulate QRAMs, and Subsection 8.2 describes how QRAMs can simulate QTMs.

8.1 QTMs simulate QRAMs

Our simulation of QRAMs by QTMs is carried out in two steps. First, we introduce the notion of Turing machines with a quantum device (TM^{Q}s) and prove in Subsection 8.1.1 that every QRAM can be simulated by a measurement-postponed TM^{Q}. The main technique here is based on the idea of simulating RAMs by TMs given in [8]. Then we show in Subsection 8.1.2 that a measurement-postponed TM^{Q} can be simulated by a well-formed and normal form QTM. The main technique here is based on the idea of simulating TMs by RTMs given in [4].

8.1.1 TMs with quantum devices simulate QRAMs

Let us first define the notion of TM with a quantum device.

Definition 8.1. *A TM with a quantum device (TM^Q) is a* 8*-tuple* $M = (Q, Q_s, Q_t, \Sigma, \delta, \lambda, q_s, q_f)$, *where:*

1. *Q is a finite set of states;*
2. *$Q_s \subseteq Q$ and $Q_t \subseteq Q$ are two disjoint sets of states as interactor for the quantum device;*

3. *$\delta : (Q \setminus Q_s) \times \Sigma \to \Sigma \times (Q \setminus Q_t) \times \{L, R\}$ is the transition function;*

4. *$\lambda : (Q_s \times \Sigma^{\#} \times \mathcal{H}) \times (Q_t \times \mathcal{H}) \to [0, 1]$ is the transition function for quantum device, where $\mathcal{H} = \bigotimes_{i=0}^{\infty} \mathcal{H}_i$, and $\mathcal{H}_i = \operatorname{span}\{|0\rangle_i, |1\rangle_i\}$. It is required that for every $p \in Q_s, \mathcal{T} \in \Sigma^{\#}, |\psi\rangle \in \mathcal{H}$,*
$$\sum_{q \in Q_t, |\phi\rangle \in \mathcal{H}} \lambda((p, \mathcal{T}, |\psi\rangle), (q, |\phi\rangle)) = 1;$$

5. *$q_s \in Q \setminus Q_s \setminus Q_t$ is the initial state;*

6. *$q_f \in Q \setminus Q_s \setminus Q_t$ is the final state.*

A configuration of TM^{Q} is a tuple $c = (q, \mathcal{T}, \xi, |\psi\rangle) \in Q \times \Sigma^{\#} \times \mathbb{Z} \times \mathcal{H}$. Let $\mathcal{C} = Q \times \Sigma^{\#} \times \mathbb{Z} \times \mathcal{H}$ be the set of configurations. The one step execution transition of TM^{Q} is a function $\to: \mathcal{C} \times \mathcal{C} \to [0, 1]$ defined by the following rules: let $c = (p, \mathcal{T}, \xi, |\psi\rangle)$,

1. if $p = q_f$, then the execution terminates.

2. if $p \in Q \setminus Q_s \setminus \{q_f\}$ and $\delta(p, \mathcal{T}(\xi)) = (q, \sigma, d)$, then after one step, the configuration will become $c' = (q, \mathcal{T}_\xi^\sigma, \xi + d, |\psi\rangle)$, i.e.
$$(p, \mathcal{T}, \xi, |\psi\rangle) \xrightarrow{1} (q, \mathcal{T}_\xi^\sigma, \xi + d, |\psi\rangle),$$

3. if $p \in Q_s$ and $\lambda((p, \mathcal{T}, |\psi\rangle), (q, |\phi\rangle)) = a$, then after one step, the configuration will become $c' = (q, \mathcal{T}, \xi, |\phi\rangle)$ with probability a, i.e.
$$(p, \mathcal{T}, \xi, |\psi\rangle) \xrightarrow{a} (q, \mathcal{T}, \xi, |\phi\rangle).$$

An execution path is then a non-empty sequence of configurations associated with probabilities:
$$\pi : c_0 \xrightarrow{a_1} c_1 \xrightarrow{a_2} c_2 \dots c_{n-1} \xrightarrow{a_n} c_n.$$
The length of π is $|\pi| = n$, and the probability of path π is $a = \prod_{i=1}^n a_i$. In this case, we can simply denote $\pi : c_0 \xrightarrow{a}{}^{n} c_n$. Let $T : \mathbb{N} \to \mathbb{N}$ and $|\psi_0\rangle = \bigotimes_{i=0}^{\infty} |0\rangle_i$. A TM^{Q} is called $T(n)$-time, if for every $x \in \{0, 1\}^*$, and every execution path $(q_0, \mathcal{T}_x, 0, |\psi_0\rangle) \xrightarrow{a}{}^{t} c$, if $a > 0$, then $t \le T(|x|)$.

Now we can explain the basic idea of our simulation of a QRAM by a TM^{Q}. The TM^{Q} used to simulate a QRAM needs the following (a finite number of) tracks:

1. "input": This track initially contains the input.

2. "output": This track contains the output after the machine halts.

3. "creg": This track contains contents of classical registers. The format is designed to be $\#\mathrm{L}a_1\mathrm{M}b_1\mathrm{R}\mathrm{L}a_2\mathrm{M}b_2\mathrm{R}\dots\mathrm{L}a_r\mathrm{M}b_r\mathrm{R}\#$, indicating that the content of classical register a_i is b_i for $1 \le i \le s$. In particular, if the number x is not found among $a_1, a_2, \dots, a_r$, then the content of classical register is 0.

4. "qcnt" — the quantum register counter: This track contains a single non-negative number indicating the number of used quantum registers.

5. "qreg": This track contains a correspondence list from virtual addresses to physical addresses. Similar to track "creg", the format is designed to be #Lu_1Mv_1RLu_2Mv_2R…Lu_sMv_sR#, indicating that the virtual address u_i corresponds to physical address v_i for $1 \leq i \leq s$. To avoid too large addresses of quantum registers, the machine re-numbers every used address of quantum registers (virtual address) to a small number (physical address). Each time a quantum register a is accessed in the QRAM, the machine checks whether the virtual address a is collected in "qreg". If so, convert a to its corresponding physical address; and if not, increment the quantum register counter and assign a with the value of the current quantum register counter ("qcnt") as its physical address (and add this assignment to the correspondence list).

6. "qdev" — a track for interactions to the quantum device: This track is used for quantum operation calls to quantum device. In our case, there are four kinds of quantum operations, i.e. CNOT, Hadamard and $\pi/8$ gates and measurements. The format of this track will be defined shortly.

7. "qret" — a track for calling back after quantum operations: This track contains a single symbol denoting the returning state, i.e. the next state it should be, after the quantum operations. This track is used as a system stack in the computers.

8. "work$i$" for $i \geq 1$ — work tracks: The contents of these tracks are ignorant and they will be cleaned to an empty track after use as a hub.

The output of TM$^{\mathrm{Q}}$ is defined to be the contents in track "output". Moreover, TM$^{\mathrm{Q}}$ M defines a function $M : \{0,1\}^* \times \{0,1\}^* \to [0,1]$ with $M(x,y)$ being the probability that M on input x outputs y. In our model, there are only four kinds of quantum operations: CNOT, Hadamard and $\pi/8$ gates, and measurements. So, we use $q_s^C, q_s^H, q_s^T, q_s^M$ to denote their initial states and $q_t^C, q_t^H, q_t^T, q_{t0}^M, q_{t1}^M$ to denote their terminating states, and put $Q_s = \{q_s^C, q_s^H, q_s^T, q_s^M\}$ and $Q_t = \{q_t^C, q_t^H, q_t^T, q_{t0}^M, q_{t1}^M\}$. Furthermore, the format of track "qdev" is defined as follows:

1. For a CNOT gate, the machine reads the contents s of track "qdev". The string $s \in \{0,1,2\}^*$ is assumed to consist of one 1, one 2 and some 0s. Let 1 and 2 be the a-th and the b-th elements of s (0-indexed), respectively, and let $|\psi\rangle \in \mathcal{H}$ be the quantum state before the application of the gate. When the gate is applied, the state of TM$^{\mathrm{Q}}$ is changed from q_s^C to q_t^C with the quantum state becoming $\mathrm{CNOT}[a,b]\,|\psi\rangle$; that is, the a-th qubit acts as the control qubit and the b-th qubit as the target qubit (0-indexed).

2. For a Hadamard (resp. $\pi/8$) gate, the machine reads the contents s of track "qdev". The string $s \in \{0,1\}^*$ is assumed to consist of one 1 and several 0s. Let 1 be the a-th element of s (0-indexed), and let $|\psi\rangle \in \mathcal{H}$ be the quantum state before application of the gate. When the gate operation is applied, the state of TM$^{\mathrm{Q}}$ is changed from q_s^H (res. q_s^T) to q_t^H (resp. q_t^T); that is, the gate is performed on the a-th qubit (0-indexed) with the quantum state becoming $H[a]\,|\psi\rangle$ (res. $T[a]\,|\psi\rangle$).

3. For a measurement, the machine reads the contents s of track "qdev". The string $s \in \{0,1\}^*$ is assumed to consist of one 1 and several 0s. Let 1 be the a-th element of s (0-indexed), and let $|\psi\rangle \in \mathcal{H}$ be the quantum state before the measurement. When the measurement is performed, the state of TM$^{\mathrm{Q}}$ is changed from q_s^M to q_{t0}^M such that the quantum state becomes $|\phi_0\rangle$ with probability p_0, and to q_{t1}^M such that the quantum state becomes $|\phi_1\rangle$ with probability

p_1, where:

$$p_0 = \|M_0[a]\,|\psi\rangle\|^2, \quad |\phi_0\rangle = \frac{M_0[a]\,|\psi\rangle}{\|M_0[a]\,|\psi\rangle\|}, \quad p_1 = \|M_1[a]\,|\psi\rangle\|^2, \quad |\phi_1\rangle = \frac{M_1[a]\,|\psi\rangle}{\|M_1[a]\,|\psi\rangle\|}$$

and $M_0[a] = |0\rangle_a \langle 0|$, $M_1[a] = I - M_0[a]$.

Now let $P = P_0, P_1, \ldots, P_{L-1}$ be a QRAM to be simulated by a TM^{Q}. By Lemma 5.3 and Lemma 5.4, we can assume that P is address-safe and measurement-postponed without any loss of generality. For every $0 \le l < L$, we use a bunch of states $(p_l, 0), (p_l, 1), \ldots, (p_l, k_l)$ to simulate instruction P_l, where for every l, k_l is an appropriate integer. The state $(p_l, 0)$ indicates the beginning of the simulation of P_l. During the simulation of P_l, the intermediate states $(p_l, 1), \ldots, (p_l, k_l)$ may be visited. In particular, $(p_L, 0)$ indicates the termination of the simulation, and is going to become q_f, which indicates the termination of the execution of TM^{Q}.

Before constructing TM^{Q}, we first show how integers are stored in our machine. For every integer $n \in \mathbb{Z}$, we use $bin(n) \in \{0,1\}^*$ to denote its binary form. The first symbol of $bin(n)$ is 0 if $n \ge 0$ and 1 otherwise, which is followed by a binary representation of $|n|$. Conversely, we use $dec(x)$ to denote the decimal value of the binary string x if it is valid. We also need some TMs that perform arithmetic and other basic operations:

- M_{inc} — a TM for increment by one: for every $a \in \mathbb{Z}$,

  $$M_{\mathrm{inc}}(bin(a)) = bin(a+1).$$

  The time of M_{inc} is $O(\log|a|)$.

- M_{add} — a TM for addition: for every $a, b \in \mathbb{Z}$,

  $$M_{\mathrm{add}}(bin(a); bin(b); \epsilon) = M_{\mathrm{add}}(bin(a); bin(b); bin(a+b)).$$

  The time of M_{add} is $O(\log|a| + \log|b|)$.

- M_{sub} — a TM for subtraction. Formally, for every $a, b \in \mathbb{Z}$,

  $$M_{\mathrm{sub}}(bin(a); bin(b); \epsilon) = M_{\mathrm{add}}(bin(a); bin(b); bin(a-b)).$$

  The time of M_{sub} is $O(\log|a| + \log|b|)$.

- M_{gtz} — a TM for checking positivity: for every $a \in \mathbb{Z}$,

  $$M_{\mathrm{gtz}}(bin(a)) = \begin{cases} 1 & a > 0, \\ 0 & \text{otherwise.} \end{cases}$$

  The time of M_{gtz} is $O(\log|a|)$.

- M_{clean} — a TM that cleans a track: for every $x \in \{0,1\}^*$, $M_{\mathrm{clean}}(x) = \epsilon$. The time of M_{clean} on input x is $O(|x|)$.

- M_{read} — a TM that reads a symbol from a track: for every $x = \sigma_0\sigma_1\ldots\sigma_{k-1} \in \{0,1\}^*$,

  $$M_{\mathrm{read}}(x; \epsilon) = \begin{cases} \sigma_1 \ldots \sigma_{k-1}; bin(0) & k \ge 1 \text{ and } \sigma_0 = 0, \\ \sigma_1 \ldots \sigma_{k-1}; bin(1) & k \ge 1 \text{ and } \sigma_0 = 1, \\ \epsilon; bin(-1) & k = 0. \end{cases}$$

  The time of M_{read} is $O(|x|)$.

- $M_{\text{write}(a)}$ — a TM that writes a specific (pre-determined) content a to the end of a track: for every $x \in \{0,1\}^*$, $M_{\text{write}(a)}(x) = xa$. The time of $M_{\text{write}(a)}$ is $O(|x|)$.

- M_{append} — a TM that appends the contents in the second track to the end of the first track: for every $x, y \in \{0,1\}^*$, $M_{\text{append}}(x;y) = xy;y$. The time of M_{append} is $O(|x|\,|y|)$.

- M_{fetch} — a TM that fetches the contents of registers: suppose the contents in the first track is $z = \text{L}a_1\text{M}b_1\text{R}\text{L}a_2\text{M}b_2\text{R}\ldots\text{L}a_r\text{M}b_r\text{R}$. For every $x \in \{0,1\}^*$,

$$M_{\text{fetch}}(z;x;\epsilon) = \begin{cases} z;x;b_i & a_i = x, \\ z;x;bin(0) & \text{otherwise.} \end{cases}$$

  The time of M_{fetch} is $O(|z|\,(|x|+|y|))$, where y is the contents in the third track after execution.

- M_{update} — a TM that updates the contents of registers: suppose the contents in the first track is $z = \text{L}a_1\text{M}b_1\text{R}\text{L}a_2\text{M}b_2\text{R}\ldots\text{L}a_r\text{M}b_r\text{R}$. For every $x \in \{0,1\}^*$,

$$M_{\text{update}}(z;x;y) = \begin{cases} \text{L}a_1\text{M}b_1\text{R}\ldots\text{L}a_i\text{M}y\text{R}\ldots\text{L}a_r\text{M}b_r\text{R};x;y & a_i = x, \\ z\text{L}x\text{M}y\text{R};x & \text{otherwise.} \end{cases}$$

  The time of M_{update} is $O(|z|\,(|x|+|y|))$.

- M_{qget} — a TM that gets the physical address of a virtual address: suppose the contents in the first track is $z = \text{L}a_1\text{M}b_1\text{R}\text{L}a_2\text{M}b_2\text{R}\ldots\text{L}a_r\text{M}b_r\text{R}$. For every $c \in \mathbb{N}$ and $x \in \{0,1\}^+$,

$$M_{\text{qget}}(z;bin(c);x;\epsilon) = \begin{cases} z;bin(c);x;b_i & x = a_i, \\ z\text{L}x\text{M}bin(c+1)\text{R};bin(c+1);x;bin(c+1) & \text{otherwise.} \end{cases}$$

  The time of M_{qget} is $O(|z|\,(|x|+|y|+\log|c|))$, where y is the contents in the fourth track after execution.

- M_{untary} — a TM that converts a non-negative integer c to a string 0^c1: for every $c \in \mathbb{N}$,

$$M_{\text{untary}}(bin(c)) = 0^c1.$$

  The time of M_{untary} is $O(c\log c)$. This TM is used to produce contents in track "qdev" for q_s^H, q_s^T and q_s^M calls.

- M_{pair} — a TM that converts a two non-negative integers a and b ($a \neq b$) to a string $s_{ab} \in \{0,1,2\}^*$ of length $|s_{ab}| = \max\{a,b\}+1$ that

$$s_{ab}(c) = \begin{cases} 1 & c = a, \\ 2 & c = b, \\ 0 & \text{otherwise,} \end{cases}$$

  where $s(c)$ denotes the c-th symbol of s (0-indexed). Formally, for every $a, b \in \mathbb{N}$ with $a \neq b$, then

$$M_{\text{pair}}(bin(a);bin(b);\epsilon) = bin(a);bin(b);s_{ab}.$$

  The time of M_{pair} is $O((a+b)(\log a + \log b))$. This TM is used to produce contents in track "qdev" for q_s^C calls.

Now we are ready to construct the TM^{Q}. We assume that all of the TMs introduced above are stationary. Before simulation, we should initialize the "qcnt" track to be decimal value of zero, i.e.

$$\begin{aligned} q_0 :& M_{\mathrm{write}(bin(0))}[\mathrm{qcnt}] \\ q_1 :& \text{transition to } (p_0, 0) \end{aligned}$$

For every $0 \leq l < L$, if P_l is a classical instruction, it can be simulated in a standard way, and the details are omitted here but provided in Appendix E. The following are the simulation of quantum instruction P_l:

1. If P_l has the form $\mathrm{CNOT}[Q_{X_i}, Q_{X_j}]$, then we use:

$$\begin{aligned} (p_l, 0) :& M_{\mathrm{write}(i)}[\mathrm{work1}] \\ (p_l, 1) :& M_{\mathrm{fetch}}[\mathrm{creg}, \mathrm{work1}, \mathrm{work2}] \\ (p_l, 2) :& M_{\mathrm{qget}}[\mathrm{qreg}, \mathrm{qcnt}, \mathrm{work2}, \mathrm{work3}] \\ (p_l, 3) :& M_{\mathrm{write}(j)}[\mathrm{work4}] \\ (p_l, 4) :& M_{\mathrm{fetch}}[\mathrm{creg}, \mathrm{work4}, \mathrm{work5}] \\ (p_l, 5) :& M_{\mathrm{qget}}[\mathrm{qreg}, \mathrm{qcnt}, \mathrm{work5}, \mathrm{work6}] \\ (p_l, 6) :& M_{\mathrm{pair}}[\mathrm{work3}, \mathrm{work6}, \mathrm{qdev}] \\ (p_l, 7) :& M_{\mathrm{write}((p_l, 9))}[\mathrm{qret}] \\ (p_l, 8) :& \text{transition to } q_s^C \\ q_t^C, & (p_l, 9)_{\mathrm{qret}} \rightarrow \#_{\mathrm{qret}}, (p_l, 9), L \\ (p_l, 9), & \#_{\mathrm{qret}} \rightarrow \#_{\mathrm{qret}}, (p_l, 10), R \\ (p_l, 10) :& M_{\mathrm{clean}}[\mathrm{work1}] \\ (p_l, 11) :& M_{\mathrm{clean}}[\mathrm{work2}] \\ (p_l, 12) :& M_{\mathrm{clean}}[\mathrm{work3}] \\ (p_l, 13) :& M_{\mathrm{clean}}[\mathrm{work4}] \\ (p_l, 14) :& M_{\mathrm{clean}}[\mathrm{work5}] \\ (p_l, 15) :& M_{\mathrm{clean}}[\mathrm{work6}] \\ (p_l, 16) :& M_{\mathrm{clean}}[\mathrm{qdev}] \\ (p_l, 17) :& \text{transition to } (p_{l+1}, 0) \end{aligned}$$

2. If P_l has the form $A[Q_{X_i}]$ with $A = H$ or T, then we use:

$$\begin{aligned}
(p_l, 0) :& M_{\text{write}(i)}[\text{work1}] \\
(p_l, 1) :& M_{\text{fetch}}[\text{creg}, \text{work1}, \text{work2}] \\
(p_l, 2) :& M_{\text{qget}}[\text{qreg}, \text{qcnt}, \text{work2}, \text{qdev}] \\
(p_l, 3) :& M_{\text{untary}}[\text{qdev}] \\
(p_l, 4) :& M_{\text{write}((p_l,6))}[\text{qret}] \\
(p_l, 5) :& \text{transition to } q_s^A \\
q_t^A, & (p_l, 6)_{\text{qret}} \to \#_{\text{qret}}, (p_l, 6), L \\
(p_l, 6), & \#_{\text{qret}} \to \#_{\text{qret}}, (p_l, 7), R \\
(p_l, 7) :& M_{\text{clean}}[\text{work1}] \\
(p_l, 8) :& M_{\text{clean}}[\text{work2}] \\
(p_l, 9) :& M_{\text{clean}}[\text{qdev}] \\
(p_l, 10) :& \text{transition to } (p_{l+1}, 0)
\end{aligned}$$

3. If P_l has the form $X_i \leftarrow M[Q_{X_j}]$, then we use:

$$\begin{aligned}
(p_l, 0) :& M_{\text{write}(j)}[\text{work1}] \\
(p_l, 1) :& M_{\text{fetch}}[\text{creg}, \text{work1}, \text{work2}] \\
(p_l, 2) :& M_{\text{qget}}[\text{qreg}, \text{qcnt}, \text{work2}, \text{qdev}] \\
(p_l, 3) :& M_{\text{untary}}[\text{qdev}] \\
(p_l, 4) :& M_{\text{write}((p_l,6))}[\text{qret}] \\
(p_l, 5) :& \text{transition to } q_s^M \\
q_{t0}^M, & (p_l, 6)_{\text{qret}} \to \#_{\text{qret}}, (p_l, 6), L \\
q_{t1}^M, & (p_l, 6)_{\text{qret}} \to \#_{\text{qret}}, (p_l, 9), L \\
(p_l, 6), & \#_{\text{qret}} \to \#_{\text{qret}}, (p_l, 7), R \\
(p_l, 7) :& M_{\text{write}(bin(0))}[\text{work3}] \\
(p_l, 8) :& \text{transition to } (p_l, 11) \\
(p_l, 9), & \#_{\text{qret}} \to \#_{\text{qret}}, (p_l, 10), R \\
(p_l, 10) :& M_{\text{write}(bin(1))}[\text{work3}] \\
(p_l, 11) :& M_{\text{write}(i)}[\text{work4}] \\
(p_l, 12) :& M_{\text{update}}[\text{creg}, \text{work4}, \text{work3}] \\
(p_l, 13) :& M_{\text{clean}}[\text{work1}] \\
(p_l, 14) :& M_{\text{clean}}[\text{work2}] \\
(p_l, 15) :& M_{\text{clean}}[\text{work3}] \\
(p_l, 16) :& M_{\text{clean}}[\text{work4}] \\
(p_l, 17) :& M_{\text{clean}}[\text{qdev}] \\
(p_l, 18) :& \text{transition to } (p_{l+1}, 0)
\end{aligned}$$

To conclude this subsection, we prove the correctness of our simulation. Let $\mathcal{M}^{\mathrm{Q}}$ denote the TM^{Q} constructed according to the above description.

Lemma 8.1. *For every $x, y \in \{0,1\}^*$, $P(x,y) = \mathcal{M}^{Q}(x,y)$.*

Proof. Clear by the construction. □

Lemma 8.2. *Suppose P is a $T(n)$-time QRAM. Then:*

1. *If $l(n)$ is logarithmic, then the number of non-empty symbols in every track of $\mathcal{M}^{Q}$ is $O(T(n))$.*
2. *If $l(n)$ is constant, then the number of non-empty symbols in every track of $\mathcal{M}^{Q}$ is $O(T(n)^2)$.*

Proof. We need to only focus on the tracks "creg" and "qreg".

Case 1. $l(n)$ is logarithmic: For each executed classical-type QRAM instruction, at most one classical register is altered. Let t_i denote the execution time of the i-th executed instruction. After the execution of this instruction, the number of non-empty symbols in track "creg" increases at most $O(t_i)$. Therefore, the number of non-empty symbols in track "creg" is

$$\sum_i O(t_i) = O(T(n)).$$

For each executed quantum-type QRAM instruction, at most one virtual address is assigned a physical address. Similarly, after the i-th executed instruction, the number of non-empty symbols in track "qreg" increases at most $O(t_i)$. Therefore, the number of non-empty symbols in track "qreg" is also $O(T(n))$.

Case 2. $l(n)$ is constant: The analysis is similar. The only difference is that after each executed instruction, the number of non-empty symbols in track "creg" and "qreg" increase at most $O(T(n)t_i)$. This is because after executing $T(n)$ QRAM instructions, the largest possible address could be $2^{T(n)}$ (by repeatedly executing the instruction $X_i \leftarrow X_j + X_k$ with $i = j = k$), which is of length $T(n)$ in its binary representation. Therefore, the number of non-empty symbols in track "qreg" is $O(T(n)^2)$. □

Lemma 8.3. *Suppose P is a $T(n)$-time QRAM. Then,*

1. *If $l(n)$ is logarithmic, then $\mathcal{M}^{Q}$ is a $O(T(n)^2)$-time TM^{Q}.*
2. *If $l(n)$ is constant, then $\mathcal{M}^{Q}$ is a $O(T(n)^4)$-time TM^{Q}.*

Proof. **Case 1**. $l(n)$ is logarithmic: Let t_i denote the execution time of the i-th executed instruction, then the lengths of addresses accessed are bounded by $O(t_i)$. $\mathcal{M}^{\mathrm{Q}}$ simulates this instruction with $O(t_i T(n))$ time, which comes from the usage of those basic TMs, e.g. M_{append}, M_{fetch}, M_{update}, M_{qget}, M_{untary} and M_{pair}. Therefore, $\mathcal{M}^{\mathrm{Q}}$ is $O(T(n)^2)$-time.

Case 2. $l(n)$ is constant: The lengths of addresses accessed are bounded by $O(T(n))$ for each executed instruction. Hence, $\mathcal{M}^{\mathrm{Q}}$ simulates each instruction with $O(T(n)^3)$ time. Consequently, $\mathcal{M}^{\mathrm{Q}}$ is $O(T(n)^4)$-time. □

8.1.2 QTMs simulate TM$^{\mathrm{Q}}$s

Our strategy of simulating $\mathcal{M}^{\mathrm{Q}}$ - the TM$^{\mathrm{Q}}$ constructed above - by a QTM is presented in the following lemma and its proof.

Lemma 8.4. *There is a well-formed, normal form, stationary and unidirectional QTM M within time $O(T(n)^2)$ such that $M(x,y) = \mathcal{M}^{\mathrm{Q}}(x,y)$ for every $x, y \in \{0,1\}^*$.*

Proof. Let $\mathcal{M}^{\mathrm{Q}} = (Q, Q_s, Q_t, \Sigma, \delta, \lambda, q_0, q_f)$. We recall that $\mathcal{M}^{\mathrm{Q}}$ is measurement-postponed and stationary. Our basic idea is to simulate $\mathcal{M}^{\mathrm{Q}}$ by maintaining a history to make the simulation reversible. The technique we use here is partly borrowed from [4].

The TM M used to simulate $\mathcal{M}^{\mathrm{Q}}$ has five tracks:

- The first track, with alphabet $\Sigma_1 = \Sigma$, is used to simulate the tape of $\mathcal{M}^{\mathrm{Q}}$.
- The second track, with track $\Sigma_2 = \{\#, @\}$, is used to store a @ indicating the position of tape head of $\mathcal{M}^{\mathrm{Q}}$.
- The third track, with alphabet $\Sigma_3 = \{\#, \$\} \cup ((Q \setminus Q_s) \times \Sigma)$ is used to write down a list of the transitions taken by $\mathcal{M}^{\mathrm{Q}}$, starting with the end marker \$.
- The fourth track is a "quantum" track and will be defined later.
- The fifth track is an "extra" quantum track for measurements.

Now we can elaborate the construction of M. We use $\forall_i$ to denote any symbol in Σ_i and $\forall_3'$ to denote any symbol in $\Sigma_3 \setminus \{\#, \$\}$. Since the fourth and the fifth tracks are not used in classical operations, we write only the first three tracks in the transitions unless the fourth or the fifth track are needed. The first stage of simulation needs the state set

$$Q_1 = Q \cup ((Q \setminus Q_t) \times (Q \setminus Q_s) \times \Sigma \times \{1,2,3,4\}) \cup ((Q \setminus Q_t) \times \{5,6,7\}) \cup \{q_a, q_b\}.$$

The initial state is q_a and the final state is q_f. The transitions are defined as follows:

1. At the beginning, we write an end marker @ on the second track and end marker \$ on the third track, and then come back to the initial position with state q_0. Include the instructions:

$$\begin{array}{llcllll}
q_a, & (\forall_1, \#_2, \#_3) & \to & (\forall_1, @, \#), & q_b, & R; & 1 \\
q_b, & (\forall_1, \#_2, \#_3) & \to & (\forall_1, \#, \$), & q_c, & R; & 1 \\
q_c, & (\forall_1, \forall_2, \forall_3) & \to & (\forall_1, \forall_2, \forall_3), & q_d, & L; & 1 \\
q_d, & (\forall_1, \forall_2, \forall_3) & \to & (\forall_1, \forall_2, \forall_3), & q_e, & L; & 1 \\
q_e, & (\forall_1, \forall_2, \forall_3) & \to & (\forall_1, \forall_2, \forall_3), & q_g, & L; & 1 \\
q_g, & (\forall_1, \forall_2, \forall_3) & \to & (\forall_1, \forall_2, \forall_3), & q_0, & R; & 1
\end{array}$$

2. For $p \in Q \setminus Q_s, \sigma \in \Sigma$ with $p \neq q_f$ and with transition $\delta(p, \sigma) = (\tau, q, d)$ in M, we make transitions to go from p to q updating the first track, i.e. the simulated tape of M, and adding (p, σ) to the end of the history. Include the instructions:

$$\begin{array}{llcllll}
p, & (\sigma, @, \forall_3) & \to & (\tau, \#_2, \forall_3), & (q, p, \sigma, 1), & d; & 1 \\
(q, p, \sigma, 1), & (\forall_1, \#_2, \$) & \to & (\forall_1, @, \$), & (q, p, \sigma, 3), & R; & 1 \\
(q, p, \sigma, 1), & (\forall_1, \#_2, \forall_3') & \to & (\forall_1, @, \forall_3'), & (q, p, \sigma, 3), & R; & 1 \\
(q, p, \sigma, 1), & (\forall_1, \#_2, \#_3) & \to & (\forall_1, @, \#_3), & (q, p, \sigma, 2), & R; & 1 \\
(q, p, \sigma, 2), & (\forall_1, \#_2, \#_3) & \to & (\forall_1, \#_2, \#_3), & (q, p, \sigma, 2), & R; & 1 \\
(q, p, \sigma, 2), & (\forall_1, \#_2, \$) & \to & (\forall_1, \#_2, \$), & (q, p, \sigma, 3), & R; & 1 \\
(q, p, \sigma, 3), & (\forall_1, \#_2, \forall_3') & \to & (\forall_1, \#_2, \forall_3'), & (q, p, \sigma, 3), & R; & 1 \\
(q, p, \sigma, 3), & (\forall_1, \#_2, \#_3) & \to & (\forall_1, \#_2, (p, \sigma)), & (q, 4), & R; & 1
\end{array}$$

Whenever $(q, 4)$ is reached, the tape head is on the first blank after the end of the history (on the third track). Now we move the tape head back to the position of tape head of M by including the instructions:

$$\begin{array}{cccccc}
(q,4), & (\forall_1, \#_2, \#_3) & \to & (\forall_1, \#_2, \#_3), & (q,5), & L; \quad 1 \\
(q,5), & (\forall_1, \#_2, \forall_3') & \to & (\forall_1, \#_2, \forall_3'), & (q,5), & L; \quad 1 \\
(q,5), & (\forall_1, \#_2, \$) & \to & (\forall_1, \#_2, \$), & (q,6), & L; \quad 1 \\
(q,5), & (\forall_1, @, \forall_3') & \to & (\forall_1, @, \forall_3'), & (q,7), & L; \quad 1 \\
(q,5), & (\forall_1, @, \$) & \to & (\forall_1, @, \$), & (q,7), & L; \quad 1 \\
(q,6), & (\forall_1, \#_2, \#_3) & \to & (\forall_1, \#_2, \#_3), & (q,6), & L; \quad 1 \\
(q,6), & (\forall_1, @, \#_3) & \to & (\forall_1, @, \#_3), & (q,7), & L; \quad 1 \\
(q,7), & (\forall_1, \forall_2, \forall_3) & \to & (\forall_1, \forall_2, \forall_3), & q, & R; \quad 1
\end{array}$$

3. For $p \in Q_s$, we need the "quantum" track (namely, the fourth track) to simulate quantum operations, but the second and third tracks are useless now. Thus in the following discussion, we only write the first track and the "quantum" track in transitions. Let $\Sigma_q = \{0, 1\}$ denote the alphabet of the "quantum" track, whose contents are initially set to 0 (in other words, 0 is used to be the empty symbol in the "quantum" track, for readability). The four quantum-type operations are implemented as follows.

   Case 1. $p = q_s^H$. Include the instructions:

$$\begin{array}{cccccc}
q_s^H, & (\forall_1, \forall_q) & \to & (\forall_1, \forall_q), & q_1^H, & L; \quad 1 \\
q_1^H, & (\#_1, \forall_q) & \to & (\#_1, \forall_q), & q_2^H, & R; \quad 1 \\
q_2^H, & (0, \forall_q) & \to & (0, \forall_q), & q_2^H, & R; \quad 1 \\
q_2^H, & (1, 0) & \to & (1, 0), & q_2^H, & R; \quad 1/\sqrt{2} \\
q_2^H, & (1, 0) & \to & (1, 1), & q_2^H, & R; \quad 1/\sqrt{2} \\
q_2^H, & (1, 1) & \to & (1, 0), & q_2^H, & R; \quad 1/\sqrt{2} \\
q_2^H, & (1, 1) & \to & (1, 1), & q_2^H, & R; \quad -1/\sqrt{2} \\
q_2^H, & (\#_1, 0) & \to & (\#_1, 0), & q_3^H, & L; \quad 1 \\
q_3^H, & (0, \forall_q) & \to & (0, \forall_q), & q_3^H, & L; \quad 1 \\
q_3^H, & (1, \forall_q) & \to & (1, \forall_q), & q_3^H, & L; \quad 1 \\
q_3^H, & (\#_1, 0) & \to & (\#_1, 0), & q_t^H, & R; \quad 1
\end{array}$$

   Case 2. $p = q_s^T$. Include the instructions:

$$\begin{array}{cccccc}
q_s^T, & (\forall_1, \forall_q) & \to & (\forall_1, \forall_q), & q_1^T, & L; \quad 1 \\
q_1^T, & (\#_1, \forall_q) & \to & (\#_1, \forall_q), & q_2^T, & R; \quad 1 \\
q_2^T, & (0, \forall_q) & \to & (0, \forall_q), & q_2^T, & R; \quad 1 \\
q_2^T, & (1, 0) & \to & (1, 0), & q_2^T, & R; \quad 1 \\
q_2^T, & (1, 1) & \to & (1, 1), & q_2^T, & R; \quad \exp(i\pi/4) \\
q_2^T, & (\#_1, 0) & \to & (\#_1, 0), & q_3^T, & L; \quad 1 \\
q_3^T, & (0, \forall_q) & \to & (0, \forall_q), & q_3^T, & L; \quad 1 \\
q_3^T, & (1, \forall_q) & \to & (1, \forall_q), & q_3^T, & L; \quad 1 \\
q_3^T, & (\#_1, 0) & \to & (\#_1, 0), & q_t^T, & R; \quad 1
\end{array}$$

Case 3. $p = q_s^C$. Include the instructions:

q_s^C,	$(\forall_1, \forall_q)$	$\rightarrow$	$(\forall_1, \forall_q)$,	q_1^C,	L;	1
q_1^C,	$(\#_1, \forall_q)$	$\rightarrow$	$(\#_1, \forall_q)$,	$(q_2^C, 0)$,	R;	1
$(q_2^C, 0)$,	$(0, \forall_q)$	$\rightarrow$	$(0, \forall_q)$,	$(q_2^C, 0)$,	R;	1
$(q_2^C, 0)$,	$(1, 0)$	$\rightarrow$	$(1, 0)$,	$(q_2^C, 0)$,	R;	1
$(q_2^C, 0)$,	$(1, 1)$	$\rightarrow$	$(1, 1)$,	$(q_2^C, 1)$,	R;	1
$(q_2^C, 0)$,	$(2, \forall_q)$	$\rightarrow$	$(2, \forall_q)$,	$(q_2^C, 0)$,	R;	1
$(q_2^C, 0)$,	$(\#_1, \forall_q)$	$\rightarrow$	$(\#_1, \forall_q)$,	$(q_3^C, 0)$,	L;	1
$(q_2^C, 1)$,	$(0, \forall_q)$	$\rightarrow$	$(0, \forall_q)$,	$(q_2^C, 1)$,	R;	1
$(q_2^C, 1)$,	$(1, 0)$	$\rightarrow$	$(1, 0)$,	$(q_2^C, 1)$,	R;	1
$(q_2^C, 1)$,	$(1, 1)$	$\rightarrow$	$(1, 1)$,	$(q_2^C, 0)$,	R;	1
$(q_2^C, 1)$,	$(2, \forall_q)$	$\rightarrow$	$(2, \forall_q)$,	$(q_2^C, 1)$,	R;	1
$(q_2^C, 1)$,	$(\#_1, \forall_q)$	$\rightarrow$	$(\#_1, \forall_q)$,	$(q_3^C, 1)$,	L;	1
$(q_3^C, 0)$,	$(0, \forall_q)$	$\rightarrow$	$(0, \forall_q)$,	$(q_3^C, 0)$,	L;	1
$(q_3^C, 0)$,	$(1, \forall_q)$	$\rightarrow$	$(1, \forall_q)$,	$(q_3^C, 0)$,	L;	1
$(q_3^C, 0)$,	$(2, 0)$	$\rightarrow$	$(2, 0)$,	$(q_3^C, 0)$,	L;	1
$(q_3^C, 0)$,	$(2, 1)$	$\rightarrow$	$(2, 1)$,	$(q_3^C, 0)$,	L;	1
$(q_3^C, 0)$,	$(\#_1, \forall_q)$	$\rightarrow$	$(\#_1, \forall_q)$,	$(q_4^C, 0)$,	R;	1
$(q_3^C, 1)$,	$(0, \forall_q)$	$\rightarrow$	$(0, \forall_q)$,	$(q_3^C, 1)$,	L;	1
$(q_3^C, 1)$,	$(1, \forall_q)$	$\rightarrow$	$(1, \forall_q)$,	$(q_3^C, 1)$,	L;	1
$(q_3^C, 1)$,	$(2, 0)$	$\rightarrow$	$(2, 1)$,	$(q_3^C, 1)$,	L;	1
$(q_3^C, 1)$,	$(2, 1)$	$\rightarrow$	$(2, 0)$,	$(q_3^C, 1)$,	L;	1
$(q_3^C, 1)$,	$(\#_1, \forall_q)$	$\rightarrow$	$(\#_1, \forall_q)$,	$(q_4^C, 1)$,	R;	1
$(q_4^C, 0)$,	$(0, \forall_q)$	$\rightarrow$	$(0, \forall_q)$,	$(q_4^C, 0)$,	R;	1
$(q_4^C, 0)$,	$(1, 0)$	$\rightarrow$	$(1, 0)$,	$(q_4^C, 0)$,	R;	1
$(q_4^C, 0)$,	$(1, 1)$	$\rightarrow$	$(1, 1)$,	$(q_4^C, 1)$,	R;	1
$(q_4^C, 0)$,	$(2, \forall_q)$	$\rightarrow$	$(2, \forall_q)$,	$(q_4^C, 0)$,	R;	1
$(q_4^C, 0)$,	$(\#_1, \forall_q)$	$\rightarrow$	$(\#_1, \forall_q)$,	q_5^C,	L;	1
$(q_4^C, 1)$,	$(0, \forall_q)$	$\rightarrow$	$(0, \forall_q)$,	$(q_4^C, 1)$,	R;	1
$(q_4^C, 1)$,	$(1, 0)$	$\rightarrow$	$(1, 0)$,	$(q_4^C, 1)$,	R;	1
$(q_4^C, 1)$,	$(1, 1)$	$\rightarrow$	$(1, 1)$,	$(q_4^C, 0)$,	R;	1
$(q_4^C, 1)$,	$(2, \forall_q)$	$\rightarrow$	$(2, \forall_q)$,	$(q_4^C, 1)$,	R;	1
q_5^C,	$(0, \forall_q)$	$\rightarrow$	$(0, \forall_q)$,	q_5^C,	L;	1
q_5^C,	$(1, \forall_q)$	$\rightarrow$	$(1, \forall_q)$,	q_5^C,	L;	1
q_5^C,	$(2, \forall_q)$	$\rightarrow$	$(2, \forall_q)$,	q_5^C,	L;	1
q_5^C,	$(\#_1, \forall_q)$	$\rightarrow$	$(\#_1, \forall_q)$,	q_t^C,	R;	1

Case 4. $p = q_s^M$. In order to make the two possible measurement outcomes distinguishable in the successive configurations, we need an extra track with alphabet $\Sigma_e = \{\#, 0, 1\}$. Include

the instructions:

$$\begin{array}{llcllll}
q_s^M, & (\forall_1, \forall_q, \forall_e) & \to & (\forall_1, \forall_q, \forall_e), & q_1^M, & L; & 1 \\
q_1^M, & (\#_1, \forall_q, \forall_e) & \to & (\#_1, \forall_q, \forall_e), & q_2^M, & R; & 1 \\
q_2^M, & (0, \forall_q, \forall_e) & \to & (0, \forall_q, \forall_e), & q_2^M, & R; & 1 \\
q_2^M, & (1, 0, \#_e) & \to & (1, 0, 0), & (q_3^M, 0), & R; & 1 \\
q_2^M, & (1, 1, \#_e) & \to & (1, 1, 1), & (q_3^M, 1), & R; & 1 \\
(q_3^M, x), & (0, \forall_q, \forall_e) & \to & (0, \forall_q, \forall_e), & (q_3^M, x), & R; & 1 \\
(q_3^M, x), & (\#_1, \forall_q, \forall_e) & \to & (\#_1, \forall_q, \forall_e), & (q_4^M, x), & L; & 1 \\
(q_4^M, x), & (0, \forall_q, \forall_e) & \to & (0, \forall_q, \forall_e), & (q_4^M, x), & L; & 1 \\
(q_4^M, x), & (1, \forall_q, \forall_e) & \to & (1, \forall_q, \forall_e), & (q_4^M, x), & L; & 1 \\
(q_4^M, x), & (\#_1, \forall_q, \forall_e) & \to & (\#_1, \forall_q, \forall_e), & q_{tx}^M, & R; & 1
\end{array}$$

In the above construction, the "extra" track is used to describe the measurement results at each position in the "quantum" track, which guarantees that no interference happens between the branches with different measurement results. This construction heavily depends on the condition that the simulated TM^Q is measurement-postponed because each position of the extra track is allowed to be altered during the execution only once.

4. Finally, to make M in normal form, we add the transition:

$$q_f, \quad (\forall_1, \forall_2, \forall_3) \quad \to \quad (\forall_1, \forall_2, \forall_3), \quad q_a, \quad R; \quad 1$$

It can be easily verified that the QTM M constructed above is well-formed, normal form, stationary, unidirectional, and within time $O(T(n)^2)$. □

8.2 QRAMs simulate QTMs

Now we turn to consider how to simulate QTMs by QRAMs. Our simulation strategy is divided into the following three steps:

1. Simulate a QTM by a family of quantum circuits with the technique developed in [31] and [25] — Subsection 8.2.1.
2. Use the Solovay-Kitaev algorithm [9] to decompose the gates used in these quantum circuits into basic gates H, T and CNOT, within bounded errors — Subsection 8.2.2.
3. Translate the family of quantum circuits with the basic gates into a QRAM — Subsection 8.2.3.

8.2.1 Quantum Circuit Families simulate QTMs

We write $[n] = \{1, 2, \ldots, n\}$ and let $\mathcal{G}$ be a finite set of gates with their qubits indexed from 1 to n. A k-qubit quantum gate ($1 \leq k \leq n$) can be written as $G = U[q_1, ..., q_k]$, indicating a $2^k \times 2^k$-unitary operator U on qubits $q_1, \ldots, q_k \in [n]$, where $q_1, \ldots, q_k$ are pairwise distinct.

Definition 8.2. *An n-qubit a-input b-output quantum circuit C over $\mathcal{G}$ is a 4-tuple $C = (\mathcal{U}, A, B, f)$, where:*

1. *$\mathcal{U}$ is a finite sequence of gates from $\mathcal{G}$;*
2. *$A \subseteq [n]$ with $|A| = a$ is the set of input qubits;*

3. *$B \subseteq [n]$ with $|B| = b$ is the set of output qubits;*

4. *$f : [n] \setminus A \to \{0,1\}$ is the initial setting for non-input qubits.*

Now we describe the computation of circuit C. Suppose $A = \{i_1, i_2, \ldots, i_a\}$ and $\mathcal{U} = G_1 G_2 ... G_t$. For every $x = x_1 x_2 \ldots x_a \in \{0,1\}^a$, the input state is set to $|\psi_x\rangle = |u_1\rangle |u_2\rangle \ldots |u_n\rangle$, where

$$u_k = \begin{cases} x_l & i_l = k, \\ f(k) & \text{otherwise.} \end{cases}$$

The final state of $\mathcal{U}$ on input x is $|\phi_x\rangle = G_t \ldots G_2 G_1 |\psi_x\rangle$. Then we measure it in the computational basis of the output qubits q_i ($i \in B$), and C outputs $y = y_1 y_2 \ldots y_b \in \{0,1\}^b$ with probability $\|M_y |\phi_x\rangle\|^2$, where measurement operator $M_y = \sum_{v \in S_y} |v\rangle \langle v|$ and

$$S_y = \{v \in \{0,1\}^n : v_{j_l} = y_l \text{ for every } l \in [n]\}.$$

In this way, C defines a function $C : \{0,1\}^a \times \{0,1\}^b \to [0,1]$ that

$$C(x,y) = \|M_y |\phi_x\rangle\|^2,$$

meaning that C on input x outputs y with probability $C(x,y)$.

Lemma 8.5. *Let $C_1 = (\mathcal{U}_1, A, B, f)$ and $C_2 = (\mathcal{U}_2, A, B, f)$, and let $0 < \epsilon < 1$. If $\|\mathcal{U}_1 - \mathcal{U}_2\|_2 < \varepsilon$, then for every $x \in \{0,1\}^{|A|}$ and $y \in \{0,1\}^{|B|}$:*

$$|C_1(x,y) - C_2(x,y)| < 3\varepsilon.$$

Proof. We can write $\mathcal{U}_1 = \mathcal{U}_2 + J$ with $\|J\|_2 < \varepsilon$. Then:

$$\begin{aligned} |C_1(x,y) - C_2(x,y)| &= \left| \|M_y \mathcal{U}_1 |\psi_x\rangle\|^2 - \|M_y \mathcal{U}_2 |\psi_x\rangle\|^2 \right| \\ &= \left| \langle \psi_x | \mathcal{U}_1^\dagger M_y \mathcal{U}_1 |\psi_x\rangle - \langle \psi_x | \mathcal{U}_2^\dagger M_y \mathcal{U}_2 |\psi_x\rangle \right| \\ &\le \|\mathcal{U}_1^\dagger M_y \mathcal{U}_1 - \mathcal{U}_2^\dagger M_y \mathcal{U}_2\|_2 \\ &\le \|(\mathcal{U}_2^\dagger + J^\dagger) M_y (\mathcal{U}_2 + J) - \mathcal{U}_2^\dagger M_y \mathcal{U}_2\|_2 \\ &= \|J^\dagger M_y \mathcal{U}_2 + \mathcal{U}_2^\dagger M_y J + J^\dagger M_y J\|_2 \\ &\le \|J\|_2 + \|J\|_2 + \|J\|_2^2 \\ &< 2\varepsilon + \varepsilon^2 < 3\varepsilon. \end{aligned}$$

□

Suppose the unitary operators appearing in $\mathcal{G}$ are $U_1, U_2, \ldots, U_m$, and for $1 \le i \le m$, U_i is a c_i-qubit unitary operator, i.e. a $2^{c_i} \times 2^{c_i}$ unitary matrix. Then the description of circuit C is a sequence of integers of the form

$$\begin{array}{llll} g_1, & q_{1,1}, & \ldots, & q_{1,c_{g_1}}, \\ g_2, & q_{2,1}, & \ldots, & q_{2,c_{g_2}}, \\ \ldots, & & & \\ g_t, & q_{t,1}, & \ldots, & q_{t,c_{g_t}}, \\ -1, & & & \\ i_1, & i_2, & \ldots, & i_a, \\ -1, & & & \\ j_1, & j_2, & \ldots, & j_b, \\ -1, & & & \\ f_1, & f_2, & \ldots, & f_n, \\ -1. & & & \end{array}$$

The sequence consists of four parts with -1 as the separator.

1. The first part describes $\mathcal{U} = G_1 G_2 \dots G_t$, where $G_i = U_{g_i}[q_{i,1}, \dots, q_{i,c_{g_i}}]$ for $1 \le i \le t$.
2. The second part describes $A = \{i_1, i_2, \dots, i_a\}$.
3. The third part describes $B = \{j_1, j_2, \dots, j_b\}$.
4. The fourth part describes f such that $f(k) = f_k$ for every $k \notin A$.

The arguments t, a, b and n are obtained by counting the integers in their corresponding parts.

Definition 8.3. *Let M be a QTM and $\{C_n\}_{n=0}^{\infty}$ a family of quantum circuits, where C_n is an n-input $b(n)$-output quantum circuit for every $n \in \mathbb{N}$ and $b(n) = (2t_n + 1)\lceil \log_2 |\Sigma| \rceil$. We say that $\{C_n\}_{n=0}^{\infty}$ simulates M if for every $x \in \{0,1\}^*$ and $y \in \{0,1\}^*$:*

$$M(x, y) = \sum_{\mathit{extract}(\mathit{tape}(z)) = y} C_{|x|}(x, z),$$

where tape(z) denotes the tape that z represents. More precisely, if we write $z = z_1 z_2 \dots z_{2t+1}$ with $z_i \in \{0,1\}^{\lceil \log_2 |\Sigma| \rceil}$ for every $1 \le i \le 2t+1$ and regard z_i as an integer of binary form z_i, then

$$\mathit{tape}(z)(m) = \begin{cases} \mathit{out}(z_{m+t+1}) & -t \le m \le t, \\ \# & \text{otherwise}, \end{cases}$$

where $\Sigma = \{\sigma_0, \sigma_1, \dots, \sigma_{|\Sigma|-1}\}$.

It was proven in [31, 25] that each QTM can be efficiently simulated by a family of quantum circuits. One of the main results in [31, 25] can be restated in a way convenient for our purpose as the following:

Theorem 8.6. *Let $T : \mathbb{N} \to \mathbb{N}$ with $T(n) \ge n$ that is time-constructible (for example, by a RAM). For every standard QTM M with exact time $T(n)$, one can find:*

- *three unitary matrices U_1, U_2, U_3 with their elements in $\mathbb{C}(M) \cup \{0, 1\}$, each of which acts on at most 6ℓ qubits, where $\ell = 2 + \lceil \log_2(|Q| + 1) \rceil + \lceil \log_2 |\Sigma| \rceil$, and*
- *a classical algorithm $\mathbb{A}$ with time complexity $O(T(n)^2 l(T(n)))$ (considered as a RAM with cost function $l(n)$ being constant or logarithmic),*

such that

1. *for every $n \in \mathbb{N}$, on input 1^n, $\mathbb{A}$ outputs the description of a $k(n)$-qubit n-input $b(n)$-output quantum circuit C_n of size $O(T(n)^2)$, only using unitary matrices U_1, U_2, U_3, where:*

$$k(n) = (2T(n) + 4)\ell, \quad b(n) = \lceil \log_2 |\Sigma| \rceil \, (2T(n) + 1);$$

2. *$\{C_n\}_{n=0}^{\infty}$ simulates M.*

Theorem 8.6 also holds for QRAMs instead of RAMs, if $T(n)$ is assumed to be QRAM-time constructible.

8.2.2 The Solovay-Kitaev Algorithm

Our next step is to decompose the unitary matrices U_1, U_2, U_3 given in Theorem 8.6 into the basic gates H, T and CNOT. Let us first briefly review the Solovay-Kitaev algorithm from [9].

Definition 8.4. *A set $\mathcal{W}$ of $d \times d$ matrices is called universal for $SU(d)$ if:*

1. *$\mathcal{W} \subseteq SU(d)$, i.e. for every $U \in \mathcal{W}$, $U^\dagger U = UU^\dagger = I$ and $|U| = 1$.*
2. *For every $U \in \mathcal{W}$, we also have $U^\dagger \in \mathcal{W}$.*
3. *for every $U \in SU(d)$ and $\varepsilon > 0$, there is a sequence $U_1, U_2, \ldots, U_m \in \mathcal{W}$ such that*

$$\|U - U_m \ldots U_2 U_1\|_2 < \varepsilon.$$

Theorem 8.7 (Solovay-Kitaev Theorem [9])**.** *Let $\mathcal{W} = \{U_1, U_2, \ldots, U_k\}$ be universal for $SU(d)$. Then there is a classical algorithm with time complexity $O(\log^c(1/\varepsilon))$ (of which the constant factor depends on d) for some constant $c > 0$ that on input $\varepsilon > 0$ and $U \in SU(d)$, outputs a sequence $i_1, i_2, \ldots, i_m \in [k]$ such that*

1. *$\|U - U_{i_m} \ldots U_{i_2} U_{i_1}\|_2 < \varepsilon$.*
2. *$m = O(\log^c(1/\varepsilon))$.*

More explicitly, U is represented by a $d \times d$ unitary matrix each of whose elements is described as a floating point number within a high enough precision whose length is bounded by $O(\log^c(1/\varepsilon))$.

Now we can use the Solovay-Kitaev algorithm to find good approximations of U_1, U_2, U_3 in Theorem 8.6 by the basic gates. Since U_1, U_2, U_3 are unitary operators on $d = 6\ell$ qubits, we choose the set of basic gates:

$$\mathcal{G} = \{\text{CNOT}[a,b], H[a], T[a] : 1 \le a, b \le d, a \ne b\},$$

where CNOT$[a,b]$ denotes a CNOT gate with the a-th qubit as its control qubit and the b-th qubit as its target qubit, and $H[a]$ and $T[a]$ denote Hadamard and $\pi/8$ gates on the a-th qubit. For every $\varepsilon > 0$, since the matrix elements of U_1, U_2, U_3 are in $\mathbb{C}(\lambda(n))$ (with $\lambda(n) \ge n$ polynomial), we can compute each element within a high enough precision in $O(\lambda(\log(1/\varepsilon))^c)$ time. Therefore, using the algorithm stated in Theorem 8.7, we can decompose U_1, U_2, U_3 into the basic gates H, T and CNOT within precision ε in $O(\lambda(\log(1/\varepsilon))^c)$ time.

8.2.3 QRAMs simulate QTMs

Now we can finish the construction of the QRAM P that given $\varepsilon > 0$, simulates a standard QTM M in the sense that

$$|P(x,y) - M(x,y)| < \varepsilon.$$

Suppose M is a standard QTM with exact time $T(n)$ and $T(n)$ is QRAM-time constructible. Since a RAM can be seen as a special QRAM, we can turn the algorithm given in Theorem 8.6 to a $O(T(n)^2 l(T(n)))$-time QRAM P_1 that on input 1^n, outputs a description of quantum circuit C_n. By the Solovay-Kitaev algorithm, there is a $O(\lambda(\log(1/\epsilon))^c)$-time QRAM P_2 that on input $\epsilon > 0$ and $U \in SU(d)$ with matrix elements in $\mathbb{C}(\lambda(n))$, outputs a sequence of basic gates $G_1, G_2, \ldots, G_m$ of length $m = O(\log^c(1/\epsilon))$ such that

$$\|U - G_m \ldots G_2 G_1\|_2 < \epsilon.$$

Then the QRAM can be described as follows:

Step 0. We hardcode the three quantum gates U_1, U_2, U_3 in Theorem 8.6 into our QRAM P for the later use.

Step 1. Read the input string $x \in \{0,1\}^*$ and count the length of x, i.e. $n = |x|$.

Step 2. Apply P_1 on input 1^n and obtain a description of quantum circuit C_n. According to Theorem 8.6, there are $t = O(T(n)^2)$ gates in C_n, and each of them is an application of the unitary operator U_1, U_2 or U_3.

Step 3. For each gate G_i in C_n, apply P_2 to get an approximation $\tilde{G}_i$ of G_i such that $\|G_i - \tilde{G}_i\| < \epsilon$, where $\epsilon = \varepsilon/3t$. This takes time $O(\lambda(\log(1/\epsilon))^c)$. Note that the size of $\tilde{G}_i$ is $O(\log^c(1/\epsilon))$. By replacing each G_i in C_n by $\tilde{G}_i$, we obtain a circuit $\tilde{C}_n$ consisting of only the basic gates H, T and CNOT. By Lemma 8.5, we have:

$$\left| C_n(x,y) - \tilde{C}_n(x,y) \right| < \varepsilon.$$

Step 4. Simulate $\tilde{C}_n$ with quantum-type QRAM instructions.

Note that the size of $\tilde{C}_n$ is $O(t \log^c(1/\epsilon))$. Therefore, QRAM P has running time

$$O(t \operatorname{poly}(\lambda(\log(1/\epsilon)))) = O(T(n)^2 \operatorname{poly}(\lambda(\log(T(n)/\varepsilon)))).$$

9 Standardisation of QTMs

The aim of this section is to prove the Standardisation Theorem for QTMs (Theorem 2.1).

9.1 Properties of TMs and QTMs

We first present several lemmas about reversible TMs and QTMs needed in our proof of Theorem 2.1. Some of them are generalised from [4], and some are new.

9.1.1 Several Lemmas for TMs

Definition 9.1. *A deterministic (classical) TM is said to be oblivious if its running time and head position at each time step depend only on the length of input. That is, there is a function $T : \mathbb{N} \to \mathbb{N}$ and a function $pos : \mathbb{N} \times \mathbb{N} \to \mathbb{Z}$ such that on input $x \in \{0,1\}^*$, the running time is $T(|x|)$ and the head position at time t is $pos(|x|, t)$.*

We note that a stationary, normal form, oblivious reversible TM is a standard QTM. We need the following lemmas, and their proofs are given in Appendix F.

Lemma 9.1 (Incrementing)**.** *There is a stationary oblivious reversible TM M that on input $x \in \{0,1\}^+$, produces $x^+ = (x+1) \bmod 2^{|x|}$ in $O(|x|^2)$ time, where $x+1$ denotes the arithmetic addition of x and 1, and $|x|$ denotes the length of x in binary. In other words,*

$$x \xrightarrow[T]{M} x^+,$$

where $T = O(|x|^2)$ and depends only on $|x|$.

Lemma 9.2 (Equality Checking)**.** *There is a stationary oblivious reversible TM M that can check whether the contents in the first and second tracks are equal and puts the outcome in the third track. Formally, for $x, y \in \{0,1\}^+$ with $|x| = |y|$,*

$$x; y; 0 \xrightarrow[T]{M} \begin{cases} x; y; 1 & x = y \\ x; y; 0 & x \neq y \end{cases},$$

where $T = O(|x|^2)$ and depends only on $|x|$.

Lemma 9.3. *There are two stationary reversible TMs M_{shl} and M_{shr} such that for every $x \in \{0,1\}^+$,*

$$x \xrightarrow[T]{M_{\text{shl}}} \operatorname{shl} x \text{ and } x \xrightarrow[T]{M_{\text{shr}}} \operatorname{shr} x,$$

where $T = O(|x|^2)$ and depends only on $|x|$. Here, we write $\operatorname{shl} : \Sigma^{\#} \to \Sigma^{\#}$ for "shift left" and $\operatorname{shr} : \Sigma^{\#} \to \Sigma^{\#}$ for "shift right"; that is, $(\operatorname{shl} \mathcal{T})(m) = \mathcal{T}(m+1)$ and $(\operatorname{shr} \mathcal{T})(m) = \mathcal{T}(m-1)$ for every $\mathcal{T} \in \Sigma^{\#}$.

9.1.2 Several Lemmas for QTMs

Lemma 9.4 (Dovetailing Lemma, Lemma 4.9 of [4])**.** *For any two well-formed, normal form and stationary QTMs M_1 and M_2, there is a well-formed, normal form and stationary QTM M such that*

$$|\mathcal{T}_0\rangle \xrightarrow[T_1]{M_1} |\mathcal{T}_1\rangle \xrightarrow[T_2]{M_2} |\mathcal{T}_2\rangle \Longrightarrow |\mathcal{T}_0\rangle \xrightarrow[T_1+T_2]{M} |\mathcal{T}_2\rangle .$$

Lemma 9.5 (Unidirection Lemma, Lemma 5.5 of [4])**.** *For every QTM $M = (Q, \Sigma, \delta, q_0, q_f)$ with time evolution operator U, there is a QTM $M' = (Q', \Sigma, \delta', q_0, q_f)$ with $Q \subseteq Q'$ and time evolution operator U' such that for every $q \in Q \setminus \{q_f\}, \mathcal{T} \in \Sigma^{\#}$ and $\xi \in \mathbb{Z}$, we have*

$$U |q, \mathcal{T}, \xi\rangle = U'(P_F^{\perp} U')^4 |q, \mathcal{T}, \xi\rangle ,$$

where $P_F^{\perp} = I - P_F$ and $P_F = |q_f\rangle_Q \langle q_f|$. Moreover, if M is well-formed, then so is M'.

Intuitively, the above lemma shows that any (well-formed) QTM can be converted to a (well-formed) unidirectional QTM with slowdown by a factor of 5.

9.2 Proof of Theorem 2.1

Now we are ready to prove Theorem 2.1. The proof is split into the following five steps:

Step 1. Let $\ell = |T(|x|)|$ be the length of $T(|x|)$ in binary. By the definition of $T(n)$, there is a standard QTM M_1 such that

$$x; \epsilon; \epsilon \xrightarrow[O(T(|x|))]{M_1} x; T(|x|); 0^{\ell}.$$

It can be easily obtained by binding all non-blank symbols in the second track with a 0 symbol in the third track.

Step 2. After Step 1, suppose the state is q_1 (and the head position is 0), we construct a standard QTM M_2 that adds a single symbol 0 into the fourth track by the following transitions:

$$\begin{array}{llcll} q_1, & (\forall_1, \forall_2, \forall_3, \#_4) & \to & (\forall_1, \forall_2, \forall_3, 0), & q_2, \quad L \\ q_2, & (\forall_1, \forall_2, \forall_3, \forall_4) & \to & (\forall_1, \forall_2, \forall_3, \forall_4), & q_0, \quad R \end{array}$$

Then we have:

$$x; T(|x|); 0^{\ell}; \epsilon \xrightarrow[2]{M_2} x; T(|x|); 0^{\ell}; 0.$$

Now the preparation is completed, and we will start from the configuration $\left|q_0, x; T(|x|); 0^{\ell}; 0, 0\right\rangle$.

Step 3. Let M be a QTM to be standardised. By Lemma 9.5, we may assume that M is unidirectional. Let d_q be the direction of q, and let M_{shl} and M_{shr} be the reversible TMs constructed in Lemma 9.3. We construct M_3 as follows. For every $p \in Q \setminus \{q_f\}$, $q \in Q$ and $\tau, \sigma \in \Sigma$ with $\delta(p, \tau, \sigma, q, d_q) \neq 0$,

Case 1. $d_q = R$. M_3 should include these instructions:

$$\begin{array}{llcllll}
p, & (\tau, \forall_2, \forall_3, \forall_4) & \to & (\sigma, \forall_2, \forall_3, \forall_4), & (q,1), & R; & \delta(p, \tau, \sigma, q, R) \\
(q,1), & (\forall_1, \forall_2, \forall_3, \forall_4) & \to & (\forall_1, \forall_2, \forall_3, \forall_4), & (q,2), & L; & 1 \\
(q,2), & (\forall_1, \forall_2, \forall_3, \forall_4) & \to & (\forall_1, \forall_2, \forall_3, \forall_4), & (q,3), & L; & 1 \\
(q,3), & (\forall_1, \#_2, \#_3, \forall_4) & \to & (\forall_1, \#_2, \#_3, \forall_4), & (q,4), & R; & 1 \\
(q,4) & & \to & & (q,5): & & M_{\mathrm{shr}}[2,3,4] \\
(q,5), & (\forall_1, \#_2, \#_3, \forall_4) & \to & (\forall_1, \#_2, \#_3, \forall_4), & (q,6), & R; & 1
\end{array}$$

$(q, 4)$ and $(q, 5)$ are regarded as the initial state and final state of M_{shr}, respectively, that shifts the second, third and fourth tracks right by a cell.

Case 2. $d_q = L$. M_3 should include these instructions:

$$\begin{array}{llcllll}
p, & (\tau, \forall_2, \forall_3, \forall_4) & \to & (\sigma, \forall_2, \forall_3, \forall_4), & (q,1), & L; & \delta(p, \tau, \sigma, q, L) \\
(q,1), & (\forall_1, \forall_2, \forall_3, \forall_4) & \to & (\forall_1, \forall_2, \forall_3, \forall_4), & (q,2), & R; & 1 \\
(q,2) & & \to & & (q,3): & & M_{\mathrm{shl}}[2,3,4] \\
(q,3), & (\forall_1, \forall_2, \forall_3, \forall_4) & \to & (\forall_1, \forall_2, \forall_3, \forall_4), & (q,4), & L; & 1 \\
(q,4), & (\forall_1, \forall_2, \forall_3, \forall_4) & \to & (\forall_1, \forall_2, \forall_3, \forall_4), & (q,5), & L; & 1 \\
(q,5), & (\forall_1, \#_2, \#_3, \forall_4) & \to & (\forall_1, \#_2, \#_3, \forall_4), & (q,6), & R; & 1
\end{array}$$

$(q, 2)$ and $(q, 3)$ are regarded as the initial state and final state of M_{shl}, respectively, that shifts the second, third and fourth tracks left by a cell.

Now both cases are in state $(q, 6)$. Let M_{inc} and M_{eq} be the RTMs constructed in Lemma 9.1 and Lemma 9.2, respectively. We include the instructions:

$$\begin{array}{lcll}
(q,6) & \to & (q,7): & M_{\mathrm{inc}}[3] \\
(q,7) & \to & (q,8): & M_{\mathrm{eq}}[2,3,4]
\end{array}$$

The procedure from $(q, 6)$ and $(q, 7)$ performs incrementing on the third track according to M_{inc}. The procedure from $(q, 7)$ to $(q, 8)$ performs equality checking on the second and third tracks and puts the result on the fourth track according to M_{eq}. To the end of the simulation at this step, we include these instructions:

$$\begin{array}{llcllll}
(q,8), & (\forall_1, \forall_2, \forall_3, \forall_4) & \to & (\forall_1, \forall_2, \forall_3, \forall_4), & (q,9), & L; & 1 \\
(q,9), & (\forall_1, \#_2, \#_3, \#_4) & \to & (\forall_1, \#_2, \#_3, \#_4), & q, & R; & 1
\end{array}$$

Note that for every $p \in Q \setminus \{q_f\}$, $\mathcal{T} \in \Sigma^{\#}$ and $\xi \in \mathbb{Z}$,

$$|p, \mathcal{T}, \xi\rangle \xrightarrow[1]{M} \sum_{\sigma,q} \delta(p, \mathcal{T}(\xi), \sigma, q, d_q) \left|q, \mathcal{T}_\xi^\sigma, \xi + d_q\right\rangle .$$

We conclude that for every $T, t \in \mathbb{Z}$ with $0 \le t < T - 1$,

$$\left|p, \mathcal{T}; \mathrm{shr}^{\xi}\left(T; t; 0\right), \xi\right\rangle \xrightarrow[\Delta(\ell)]{M_3} \sum_{\sigma,q} \delta(p, \mathcal{T}(\xi), \sigma, q, d_q) \left|q, \mathcal{T}_\xi^\sigma; \mathrm{shr}^{\xi+d_q}\left(T; t+1; 0\right), \xi + d_q\right\rangle .$$

For the case $t = T - 1$,

$$\left|p, \mathcal{T}; \mathrm{shr}^{\xi}\left(T; T-1; 0\right), \xi\right\rangle \xrightarrow[\Delta(\ell)]{M_3} \sum_{\sigma,q} \delta(p, \mathcal{T}(\xi), \sigma, q, d_q) \left|q, \mathcal{T}_\xi^\sigma; \mathrm{shr}^{\xi+d_q}\left(T; T; 1\right), \xi + d_q\right\rangle ,$$

where $\Delta(\ell) = T_{\text{sh}}(\ell) + T_{\text{inc}}(\ell) + T_{\text{eq}}(\ell) + 7 = O(\log^2 T)$. Here, $\text{shr}^k \mathcal{T}$ denotes the tape that shifts $\mathcal{T}$ right by k steps, i.e. $(\text{shr}^k \mathcal{T})(m) = \mathcal{T}(m-k)$.

Step 4. QTM M_4 is constructed as follows. We introduce a special state $q_a \notin Q$ and set $q'_f \notin Q$ to be the final state of M'. Moreover, we need a fifth track and mark @ on the fifth track to distinguish the usual simulation and the extending procedure. For the final state q_f, we include the instructions:

$$\begin{array}{cccccc}
q_f, & (\forall_1, \forall_2, \forall_3, \forall_4, \#_5) & \to & (\forall_1, \forall_2, \forall_3, \forall_4, @), & (q_f, 1), & L;\ \ 1 \\
(q_f, 1), & (\forall_1, \#_2, \#_3, \#_4, \#_5) & \to & (\forall_1, \#_2, \#_3, \#_4, \#_5), & q_a, & R;\ \ 1
\end{array}$$

It takes two steps to transfer state q_f to state q_a with a marker @ on the fifth track, i.e.

$$\left| q_f, \mathcal{T}; \text{shr}^{\xi}\left(T; t; z\right); \epsilon, \xi \right\rangle \xrightarrow[2]{M_4} \left| q_a, \mathcal{T}; \text{shr}^{\xi}\left(T; t; z; @\right), \xi \right\rangle$$

for $0 \le t \le T$ and $z \in \{0, 1\}$.

Now include the instructions of q_a as follows:

$$\begin{array}{cccccc}
q_a, & (\forall_1, \forall_2, \forall_3, 1, @) & \to & (\forall_1, \forall_2, \forall_3, 1, @), & q'_f, & R;\ \ 1 \\
q_a, & (\forall_1, \forall_2, \forall_3, 0, @) & \to & (\forall_1, \forall_2, \forall_3, 0, @), & (q_a, 1), & R;\ \ 1 \\
(q_a, 1), & (\forall_1, \forall_2, \forall_3, \forall_4, \#_5) & \to & (\forall_1, \forall_2, \forall_3, \forall_4, @), & (q_a, 2), & L;\ \ 1 \\
(q_a, 2), & (\forall_1, \forall_2, \forall_3, \forall_4, \forall_5) & \to & (\forall_1, \forall_2, \forall_3, \forall_4, \forall_5), & (q_a, 3), & L;\ \ 1 \\
(q_a, 3), & (\forall_1, \#_2, \#_3, \#_4, \forall_5) & \to & (\forall_1, \#_2, \#_3, \#_4, \forall_5), & (q_a, 4), & R;\ \ 1 \\
(q_a, 4) & & \to & & (q_a, 5) : & M_{\text{shr}}[2, 3, 4] \\
(q_a, 5), & (\forall_1, \#_2, \#_3, \#_4, \forall_5) & \to & (\forall_1, \#_2, \#_3, \#_4, \forall_5), & (q_a, 6), & R;\ \ 1 \\
(q_a, 6) & & \to & & (q_a, 7) : & M_{\text{inc}}[3] \\
(q_a, 7) & & \to & & (q_a, 8) : & M_{\text{eq}}[2, 3, 4] \\
(q_a, 8), & (\forall_1, \forall_2, \forall_3, \forall_4, \forall_5) & \to & (\forall_1, \forall_2, \forall_3, \forall_4, \forall_5), & (q_a, 9), & L;\ \ 1 \\
(q_a, 9), & (\forall_1, \forall_2, \forall_3, \forall_4, @) & \to & (\forall_1, \forall_2, \forall_3, \forall_4, @), & q_a, & R;\ \ 1
\end{array}$$

We conclude that for $\eta \le \xi$ and $0 \le t < T - 1$,

$$\left| q_a, \mathcal{T}; \text{shr}^{\xi}\left(T; t; 0\right); \text{shr}^{\eta} @^{\xi - \eta + 1}, \xi \right\rangle \xrightarrow[\Delta(\ell)]{M_4} \left| q_a, \mathcal{T}; \text{shr}^{\xi+1}\left(T; t+1; 0\right); \text{shr}^{\eta} @^{\xi - \eta + 2}, \xi + 1 \right\rangle.$$

For the case $t = T - 1$, we have

$$\left| q_a, \mathcal{T}; \text{shr}^{\xi}\left(T; T-1; 0\right); \text{shr}^{\eta} @^{\xi - \eta + 1}, \xi \right\rangle \xrightarrow[\Delta(\ell)]{M_4} \left| q_a, \mathcal{T}; \text{shr}^{\xi+1}\left(T; T; 1\right); \text{shr}^{\eta} @^{\xi - \eta + 2}, \xi + 1 \right\rangle.$$

And the case $t = T$,

$$\left| q_a, \mathcal{T}; \text{shr}^{\xi}\left(T; T; 1\right); \text{shr}^{\eta} @^{\xi - \eta + 1}, \xi \right\rangle \xrightarrow[1]{M_4} \left| q'_f, \mathcal{T}; \text{shr}^{\xi}\left(T; T; 1\right); \text{shr}^{\eta} @^{\xi - \eta + 1}, \xi + 1 \right\rangle.$$

By Lemma 9.4, dovetailing the four QTMs M_1, M_2, M_3 and M_4 will obtain a well-formed, normal form and unidirectional but not stationary QTM M'. Since the contents of fifth track allows distinguishing the result obtained at any time the state $|q_f\rangle$ is measured during the execution by the number of @ in the fifth track, it can be verified that M' satisfies the condition claimed in the theorem statement (except for that M' is not stationary).

Step 5. This step fills meaningless instructions, which will not modify the contents of the first track, in order to make M' stationary. We need three time stamps T_1, T_2 and T_3 with $T < T_1 <$

$T_2 < T_3 = O(T)$. In the construction of Step 4, when a single symbol 1 is found in the fourth track, we still move right (and print @ on the fifth track) until the time accumulator (i.e. the content of the third track) reaches T_1 instead of stopping at state q'_f. We may set T_1 sufficiently large, e.g. $T_1 = 4T$, in order to guarantee that the head position at time T_1 is positive. Let ξ_1 be the head position at time T_1. It should be noted that in order to obtain the current head position, we can use a new track to maintain the current head position in the simulation of each step by M_{inc} and M_{dec} constructed in Lemma 9.1.

Our strategy is to move right until the time accumulator reaches T_2 and then move left until the time accumulator reaches T_3 such that the head position at time T_3 is 0. Note that the head position will be $\xi_1 + T_2 - T_1$ at time T_2. In order to make the head position at time T_3 to be 0, the condition $T_3 - T_2 = \xi_1 + T_2 - T_1$ should hold, i.e. $T_2 = (T_1 + T_3 - \xi_1)/2$, which allows to compute T_2 when the time accumulator reaches T_1 (and then ξ_1 is known). In order to make the time evolution unitary, we need three more tracks to print symbol @ for T_1, T_2 and T_3 (similar to Step 4).

We can set appropriate values for T_1 and T_3 to achieve these, for example $T_1 = 4T$ and $T_3 = 10T$. To see that the constructed QTM is stationary, we give an intuitive explanation here. Let τ_x be the running time of M on input x and ξ_x be the head position of M at time τ_x, which is also the head position of M' when the state q_a is met for the first time. We have $|\xi_x| \leq \tau_x \leq T = T(|x|)$. After that, our QTM M' has four procedures:

Procedure 0. In the simulation of M for time stamp ranged from τ_x to T, the head position keeps going right. A symbol @ is printed on each position between ξ_x and ξ_0 of the fifth track, where $\xi_0 - \xi_x = T - \tau_x$. We call the fifth track the 0th buffer track.

Procedure 1. After Procedure 0, the head position keeps going right. Another (empty) track, called the 1st buffer track, is used to print a symbol @ on each position between ξ_0 and ξ_1, where $\xi_1 - \xi_0 = T_1 - T$.

Procedure 2. After Procedure 1, the head position keeps going right. Another (empty) track, called the 2nd buffer track, is used to print a symbol @ on each position between ξ_1 and ξ_2, where $\xi_2 - \xi_1 = T_2 - T_1$ and $T_2 = (T_1 + T_3 - \xi_1)/2$.

Procedure 3. After Procedure 2, the head position keeps going left. Another (empty) track, called the 3rd buffer track, is used to print a symbol @ on each position between ξ_3 and ξ_2, where $\xi_2 - \xi_3 = T_3 - T_2$.

Note that T_1 and ξ_1 always have the same parity, and $T_3 = 10T$ is even, we conclude that T_2 is always integer. Instead, $T_2 \geq T_3/2 = 5T$ and $T_2 \leq (T_1 + T_3)/2 = 7T$. Therefore, it holds that

$$T < T_1 = 4T < 5T \leq T_2 \leq 7T < T_3 = 10T.$$

On the other hand, we have:

$$\xi_3 = \xi_2 + T_2 - T_3 = T_2 - T_1 + \xi_1 + T_2 - T_3 = 0,$$

which implies QTM M' is stationary. In the simulation of each step of the four procedures, printing each symbol @ (except the first printed symbol at each procedure) takes exactly $\Delta(\ell)$ steps (see the construction in Step 4). Therefore, M' halts exactly at time $T' = O(T_3\Delta(\ell)) = O(T \log^2 T)$, and M' is a standard QTM that simulates M.

10 Conclusion

In this paper, we formally define the notions of QRAM and QRASP. The relationships between the computational powers of QRAMs, QRASPs and QTMs are established by overcoming the difficulty of mismatch between the halting scheme of QTMs and that of QRAMs and QRASPs through a technique for standardisation of QTMs. These results further help us to clarify the relationships between complexity classes **P**, **EQRAMP**, **EQP**, **BQRAMP** and **BQP**.

The models of QRAMs and QRASPs defined in this paper can be further extended in several dimensions:

- The addressing adopted in our QRAM model is classical in the sense that an address indicating which quantum register (qubit) to perform a quantum gate is obtained from a classical register. This perfectly match the current architecture design of quantum computer, like a co-processor used together with a classical computer. But one can also conceive a fully quantum computer in the future which utilises only quantum registers but no classical registers. Such a machine should allow to access simultaneously the data of several different registers via a superposition of addresses. Indeed, such a notion of quantum addressing was already introduced in the quantum random access memory model [14], and a possible quantum optical implementation is also proposed there. A model of QRAMs with quantum addressing is certainly an interesting topic for future research.

- In QRASPs considered in this paper, a program is stored in classical registers, and thus treated as classical data rather than quantum data. For a QRASP modelling a fully quantum computer, however, a program will be encoded as quantum data. Consequently, the quantum programming paradigm of superposition of programs proposed in [32] can be realised in such a generalised QRASP model.

- Several new parallel quantum algorithms or parallel implementation of existing quantum algorithms have been developed, e.g. [5, 7, 22]. On the other hand, a parallel quantum programming language was defined in [33, 34]. This motivates us to extend our QRAM and QRASP models to parallel quantum random access machines (PQRAMs), as a quantum generalisation of PRAMs [16].

References

[1] L. M. Adleman, J. Demarrais and M. A. Huang. Quantum Computability. *SIAM Journal of Computation*, 26(5): 1524-1540. 1997.

[2] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd and O. Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Journal on Computing*, 37(1): 166-194. 2007.

[3] A. V. Aho, J. E. Hopcroft and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. 1974.

[4] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5): 1411-1473. 1997.

[5] S. Bravyi, D. Gosset and R. König. Quantum advantage with shallow circuits. *Science*, 362: 308-311. 2018.

[6] R. P. Brent. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. *Analytic Computational Complexity*, pp. 151-176. 1975.

[7] R. Cleve and J. Watrous, Fast parallel circuits for the quantum Fourier transform, In: *Proceedings of the 41st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2000, pp. 526-536.

[8] S. A. Cook and R. A. Reckhow. Time bounded random access machine. *Journal of Computer and System Sciences*, 7: 354-375. 1973.

[9] C. M. Dawson and M. A. Nielsen. The Solovay-Kitaev algorithm. *Quantum Information and Computation*, 6(1):81-95. 2006.

[10] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 400(1818): 97-117. 1985.

[11] D. Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 425(1868): 73-90. 1989.

[12] E. Farhi, J. Goldstone, S. Gutmann and M. Sipser. Quantum computation by adiabatic evolution. *Arxiv*: quant-ph/0001106. 2000.

[13] X. Fu et al. eQASM: an executable quantum instruction set architecture. In: *2019 IEEE International Symposium on High Performance Computer Architecture*, pp. 224-237. 2019.

[14] V. Giovannetti, S. Lloyd and L. Maccone. Quantum random access memory. *Physical Review Letters*, 100(16): 160501. 2008.

[15] IBM Q Team. IBM Quantum Experience. https://www.ibm.com/quantum-computing/, 2018.

[16] R. Karp and V. Ramachandran, *A Survey of Parallel Algorithms for Shared-Memory Machines*, University of California, Berkeley, Department of EECS, Tech. Rep. UCB/CSD-88-408, 1988.

[17] E. Knill. Conventions for quantum pseudocode. *LANL report*: LAUR-96-2724. 1996.

[18] S. Lee, S. J. Lee, T. Kim, J. S. Lee and J. Biamonte. The cost of quantum gate primitives. *Journal of Multiple-Valued Logic and Soft Computing*, 12(5-6): 561-573. 2006.

[19] N. Linden and S. Popescu. The halting problem for quantum computers. In: arXiv:quant-ph/9806054. 1998.

[20] J. Miszczak, Models of quantum computation and quantum programming languages, *Bulletin of the Polish Academy of Sciences: Technical Sciences*, 59(3), 305-324, 2011.

[21] T. Miyadera and M. Ohya. On halting process of quantum Turing machine. *Open Systems and Information Dynamics*, 12(3): 261-265. 2005.

[22] C. Moore and M. Nilsson, Parallel quantum computation and quantum codes, *SIAM Journal on Computing* 31(2001) 799-815.

[23] J. M. Myers. Can a universal quantum computer be fully quantum. *Physical Review Letters*, 78(9): 1823-1824. 1997.

[24] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. 2002.

[25] H. Nishimura and M. Ozawa. Computational complexity of uniform quantum circuit families and quantum Turing machines. *Theoretical Computer Science*, 276(1-2): 147-181. 2002.

[26] M. Ozawa. Quantum nondemolition monitoring of universal quantum computers. *Physical Review Letters*, 80(3): 631-634. 1998.

[27] R. Raussendorf and H. J. Briegel. A one-way quantum computer. *Physical Review Letters*, 86(22): 5188. 2001.

[28] R. Raussendorf, D. E. Browne and H. J. Briegel. Measurement-based quantum computation on cluster states. *Physical Review A*, 68(2): 022312. 2003.

[29] S. S. Robert, J. C. Michael and W. J. Zeng. A practical quantum instruction set architecture. *Arxiv*:1608.03355.

[30] Y. Shi. Remarks on universal quantum computer. *Physics Letters A*, 293: 277-282. 2002.

[31] A. C. Yao. Quantum circuit complexity. *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pp. 352-361. 1993.

[32] M. S. Ying, *Foundations of Quantum Programming*, Morgan Kaufmann, 2016.

[33] M. S. Ying, L. Zhou and Y. J. Li. Reasoning about parallel quantum programs. *Arxiv*:1810.11334. 2018.

[34] M. S. Ying, L. Zhou, Y. J. Li and Y. Feng. A proof system for disjoint parallel quantum programs. *Theoretical Computer Science*, 897:164-184. 2022.

A Proofs of Address Shifting and Address-Safe QRAMs

Proof of Lemma 5.2

Suppose P consists of L instructions $P_0, P_1, \ldots, P_{L-1}$. Let $\delta = k+1$. In the following, we shift the address to the right by δ with the help of X_0. The modified instructions for address shifting are listed in Table 5. Most of them are directly obtained except indirect addressing and jumping.

Table 5: Modified QRAM instructions for address shifting by δ

Type	Instruction	Modified instruction
Classical	$X_i \leftarrow C$, C any integer	$X_{i+\delta} \leftarrow C$
Classical	$X_i \leftarrow X_j + X_k$	$X_{i+\delta} \leftarrow X_{j+\delta} + X_{k+\delta}$
Classical	$X_i \leftarrow X_j - X_k$	$X_{i+\delta} \leftarrow X_{j+\delta} - X_{k+\delta}$
Classical	$X_i \leftarrow X_{X_j}$	$X_{i+\delta} \leftarrow X_{X_{j+\delta}+\delta}$
Classical	$X_{X_i} \leftarrow X_j$	$X_{X_{i+\delta}+\delta} \leftarrow X_{j+\delta}$
Classical	TRA m if $X_j > 0$	TRA m' if $X_{j+\delta} > 0$
Classical	READ X_i	READ $X_{i+\delta}$
Classical	WRITE X_i	WRITE $X_{i+\delta}$
Quantum	$\mathrm{CNOT}[Q_{X_i}, Q_{X_j}]$	$\mathrm{CNOT}[Q_{X_{i+\delta}}, Q_{X_{j+\delta}}]$
Quantum	$H[Q_{X_i}]$	$H[Q_{X_{i+\delta}}]$
Quantum	$T[Q_{X_i}]$	$T[Q_{X_{i+\delta}}]$
Measurement	$X_i \leftarrow M[Q_{X_j}]$	$X_{i+\delta} \leftarrow M[Q_{X_{j+\delta}}]$

In order to precisely describe how to make this shifting, we first list the lengths needed for all instructions in Table 6. For $0 \le l < L$, we write *length*(l) to denote the length needed for address shifting according to Table 6. In order to label the instructions in P', we define:

$$label(l) = \sum_{i=0}^{l-1} length(i)$$

for $0 \leq l \leq L$. Especially, the length of P' is defined to be $L' = label(L)$.

Table 6: Lengths of QRAM instructions for address shifting

Type	Instruction	length
Classical	$X_i \leftarrow C$, C any integer	1
Classical	$X_i \leftarrow X_j + X_k$	1
Classical	$X_i \leftarrow X_j - X_k$	1
Classical	$X_i \leftarrow X_{X_j}$	6
Classical	$X_{X_i} \leftarrow X_j$	6
Classical	TRA m if $X_j > 0$	1
Classical	READ X_i	1
Classical	WRITE X_i	1
Quantum	CNOT$[Q_{X_i}, Q_{X_j}]$	1
Quantum	$H[Q_{X_i}]$	1
Quantum	$T[Q_{X_i}]$	1
Measurement	$X_i \leftarrow M[Q_{X_j}]$	1

Now we are ready to describe how to construct P'. For every $0 \leq l < L$, we convert P_l to one or more instructions in P'.

Case 1. If P_l is indirect addressing, the two modified instructions for $X_i \leftarrow X_{X_j}$ and $X_{X_i} \leftarrow X_j$ are indeed problematic in Table 5. To resolve this issue, we consider $X_i \leftarrow X_{X_j}$ for example and use the following instructions with the help of X_0:

$$\begin{aligned} label(l):& X_0 \leftarrow 0 \\ & X_0 \leftarrow X_0 - X_{j+\delta} \\ & \text{TRA } L' \text{ if } X_0 > 0 \\ & X_0 \leftarrow \delta \\ & X_0 \leftarrow X_0 + X_{j+\delta} \\ & X_{i+\delta} \leftarrow X_{X_0} \end{aligned}$$

Similarly, the instructions for $X_{X_i} \leftarrow X_j$ are as follows:

$$\begin{aligned} label(l):& X_0 \leftarrow 0 \\ & X_0 \leftarrow X_0 - X_{i+\delta} \\ & \text{TRA } L' \text{ if } X_0 > 0 \\ & X_0 \leftarrow \delta \\ & X_0 \leftarrow X_0 + X_{i+\delta} \\ & X_{X_0} \leftarrow X_{j+\delta} \end{aligned}$$

Case 2. If P_l is jumping, i.e. TRA m if $X_j > 0$, we use a single modified instruction:

$$label(l): \text{TRA } m' \text{ if } X_{j+\delta} > 0$$

with $m' = \mathit{label}(m)$.

Case 3. For other cases, use the instructions according to Table 6.

It can be seen that the constructed QRAM P' can simulate QRAM P through shifting the address to the right by $\delta = k+1$, with $X_1, X_2, \dots, X_k$ untouched. Instead, the slowdown is a constant factor, which depends on k.

Proof of Lemma 5.3

The construction of P' is straightforward. Note that the only way to access to an invalid address is indirect addressing. Thus, we could avoid accessing to an invalid address by checking whether the indirect address is valid beforehand. For example, if an instruction P_l is of the form $X_i \leftarrow X_{X_j}$, then it can be replaced in P' by introducing an independent variable *tmp* with the code

```
$tmp \leftarrow 0$
$tmp \leftarrow tmp - X_j$
TRA $L'$ if $tmp > 0$
$X_i \leftarrow X_{X_j}$
```

where L' denotes the length of P'. Instead, the value of m in the jumping instruction should be changed to an appropriate value m' similar to Lemma 5.2.

According to Lemma 5.2, the variable *tmp* is involved by shifting the address to the right by $\delta = 2$.

B QRAM instructions for simulating QRASPs

To obtain a program P' consisting of only QRAM instructions from the pseudo-code given in Algorithm 1, we have to:

1. transfer the **if** and **while** statements to QRAM instructions; and
2. replace every variable by a classical register with an explicit index.

We will carefully describe how to transform the pseudo-code given in Algorithm 1 into QRAM P' in Appendices B.1-B.4.

B.1 Equality Checking

Given two classical registers a and b, it is a basic operation to check whether their contents are equal: $a = b$? We observe that $a \neq b$ if and only if $|a - b| > 0$. Algorithm 2 provides a simple method to compare a and b using three extra disposable registers, with the result $\mathit{res} = |a - b|$.

To simplify the QRAM code, we use $\mathit{res} \leftarrow |a - b|$ to indicate the code in Algorithm 2 in the following discussions.

B.2 Encoding the if and while statements by QRAM instructions

We interpret **if** and **while** statements by QRAM instructions in the general case separately.

For **if** statement, e.g. Algorithm 3, we provide a QRAM interpretation in Algorithm 4.

For **while** statement, e.g. Algorithm 5, we provide a QRAM interpretation in Algorithm 6.

Algorithm 2 QRAM code for checking whether $a = b$.

```
Input: $a$ and $b$.
Output: $res > 0$ if $a \neq b$ and 0 otherwise.
$tmp0 \leftarrow a$; $tmp1 \leftarrow b$;
$tmp0 \leftarrow tmp0 - tmp1$;
TRA 6 if $tmp0 > 0$;
$tmp1 \leftarrow 0$;
$tmp0 \leftarrow tmp1 - tmp0$;
$res \leftarrow tmp0$;
```

Algorithm 3 Example code for if $a = b$.

```
if $a = b$ then
    label0: statements;
else
    label1: statements;
end if
label2: statements;
```

Algorithm 4 QRAM code for if $a = b$.

```
$res \leftarrow |a - b|$;
TRA label1 if $res > 0$;
label0: statements;
$res \leftarrow 1$;
TRA label2 if $res > 0$;
label1: statements;
label2: statements;
```

Algorithm 5 Example code for while $a = b$.

```
while $a = b$ do
    label1: statements;
end while
label2: statements;
```

Algorithm 6 QRAM code for while $a = b$.

```
label0: $res \leftarrow |a - b|$;
TRA label2 if $res > 0$;
label1: statements;
$res \leftarrow 1$;
TRA label0 if $res > 0$;
label2: statements;
```

B.3 Replacing every variable by a classical register with an explicit index

In the previous interpretation, there are only three extra classical registers used, namely *tmp0*, *tmp1* and *res*. We assign each of the nine classical registers *tmp0*, *tmp1*, *res*, IC, AC, *flag*, *op*, j, k from the 0-th to the 8-th classical registers, respectively. Let $\delta = 9$ indicate the offset. Then the array *memory* is assigned to begin at the δ-th classical register. More precisely, *memory*$[j]$ is assigned to the $(\delta + j)$-th classical register.

B.4 Assertion for valid addressing

In the QRAM construction, accessing to *memory*$[j]$ is dangerous, because j can be negative but the address it is assigned to, i.e. $(\delta + j)$, could still be valid. Therefore, an assertion is needed before each access to *memory*$[j]$. Algorithm 7 provides a possible solution. We use a QRAM instruction trick here: before accessing to $X_{j+\delta}$, we try to access to X_j (in QRAM address) but ignore the addressing results. This works because if $j < 0$, X_j will be invalid, then the QRAM terminates as we want; if $j \geq 0$, it goes as if nothing happened (we have accessed to X_j without modifying anything).

Algorithm 7 QRAM code for accessing *memory*$[j]$.

```
1: $tmp1 \leftarrow X_j$;
2: $tmp0 \leftarrow \delta$
3: $j \leftarrow j + tmp0$;
4: $res \leftarrow X_j$;
```

C QRASP instructions for simulating QRAMs

1. P_l is of the form $X_i \leftarrow C$. The QRASP code is

$$\begin{aligned} label(l) :& \mathrm{LOD}, C \\ & \mathrm{STO}, i + \delta \end{aligned}$$

2. P_l is of the form $X_i \leftarrow X_j + X_k$. The QRASP code is

$$\begin{aligned} label(l) :& \mathrm{LOD}, 0 \\ & \mathrm{ADD}, j + \delta \\ & \mathrm{ADD}, k + \delta \\ & \mathrm{STO}, i + \delta \end{aligned}$$

3. P_l is of the form $X_i \leftarrow X_j - X_k$. The QRASP code is

$$\begin{aligned} label(l) :& \mathrm{LOD}, 0 \\ & \mathrm{ADD}, j + \delta \\ & \mathrm{SUB}, k + \delta \\ & \mathrm{STO}, i + \delta \end{aligned}$$

4. P_l is of the form $X_i \leftarrow X_{X_j}$. The QRASP code is

$$\begin{aligned}
\mathit{label}(l) :&\text{LOD}, \delta \\
&\text{ADD}, j+\delta \\
&\text{STO}, a+1 \\
&\text{LOD}, 0 \\
a :&\text{ADD}, 0 \\
&\text{STO}, i+\delta
\end{aligned}$$

It is noted that $a = \mathit{label}(l) + 8$.

5. P_l is of the form $X_{X_i} \leftarrow X_j$. The QRASP code is

$$\begin{aligned}
\mathit{label}(l) :&\text{LOD}, \delta \\
&\text{ADD}, i+\delta \\
&\text{STO}, a+1 \\
&\text{LOD}, 0 \\
&\text{ADD}, j+\delta \\
a :&\text{STO}, 0
\end{aligned}$$

It is noted that $a = \mathit{label}(l) + 10$.

6. P_l is of the form TRA m if $X_j > 0$. The QRASP code is

$$\begin{aligned}
\mathit{label}(l) :&\text{LOD}, 0 \\
&\text{ADD}, j+\delta \\
&\text{BPA}, \mathit{label}(m)
\end{aligned}$$

7. P_l is of the form READ X_i. The QRASP code is

$$\mathit{label}(l) :\text{RD}, i+\delta$$

8. P_l is of the form WRITE X_i. The QRASP code is

$$\mathit{label}(l) :\text{PRI}, i+\delta$$

D Correctness of simulating QRAMs by QRASPs

We note that $L' = |P'| = \mathit{label}(L) = \mathit{label}(|P|) \le 15L < 20L = \delta$.

Definition D.1. *We say that a QRASP configuration $c' = (\xi', \zeta', \mu', |\psi'\rangle, x', y')$ agrees with a QRAM configuration $c = (\xi, \mu, |\psi\rangle, x, y)$, denoted $c' \models c$, if*

1. *$\xi' = \downarrow$, $\mu'(i+\delta) = \mu(i)$ for every $i \in \mathbb{N}$, $|\psi'\rangle = |\psi\rangle$, $x' = x$ and $y' = y$ in the case $\xi = \downarrow$; or*

2. *$\xi' = label(\xi)$, $\mu'(i+\delta) = \mu(i)$ for every $i \in \mathbb{N}$, $|\psi'\rangle = |\psi\rangle$, $x' = x$ and $y' = y$ in the case $\xi \in \mathbb{N}$.*

Lemma D.1. *For every c', there is a unique c such that $c' \models c$.*

Let $\mathcal{C}$ and $\mathcal{C}'$ be the set of configurations of P and P', respectively, and $c_0 \in \mathcal{C}$ and $c'_0 \in \mathcal{C}'$ be their initial configurations. Define $\mathcal{C}'_{\mathcal{L}} \subseteq \mathcal{C}'$ being the set of configurations, whose ICs are in

$$\mathcal{L} = \{label(0), label(1), \ldots, label(L)\}.$$

Lemma D.2. *Let $c' \in \mathcal{C}'_{\mathcal{L}}$ and $c, d \in \mathcal{C}$. If $c' \models c$, and $c \xrightarrow[T]{p} d$, then there is a $d' \in \mathcal{C}'_{\mathcal{L}}$ such that $d' \models d$ and $c' \xrightarrow[\Theta(T)]{p} d'$.*

Proof. Direct from the operational semantics. □

We write $\mathcal{P}$ and $\mathcal{P}'$ for the sets of all possible execution paths of P and P', respectively. Let $\pi' \in \mathcal{P}'_f(c'_0)$ be a path of length $|\pi'| = k$:

$$\pi' : c'_0 \xrightarrow[T'_1]{p'_1} c'_1 \xrightarrow[T'_2]{p'_2} \ldots \xrightarrow[T'_{k-1}]{p'_{k-1}} c'_{k-1} \xrightarrow[T'_k]{p'_k} c'_k.$$

Let $0 = i_0 < i_1 < \cdots < i_{m-1} < i_m = k$ be all indices such that $c'_{(j)} = c'_{i_j} \in \mathcal{C}'_{\mathcal{L}}$ for $0 \le j \le m$. Then the execution path π' can be written as

$$\pi' : c'_0 = c'_{(0)} \xrightarrow[T'_{(1)}]{p'_{(1)}}{}^{*} c'_{(1)} \xrightarrow[T'_{(2)}]{p'_{(2)}}{}^{*} \ldots \xrightarrow[T'_{(m-1)}]{p'_{(m-1)}}{}^{*} c'_{(m-1)} \xrightarrow[T'_{(m)}]{p'_{(m)}}{}^{*} c'_{(m)} = c'_k,$$

where

$$p'_{(j)} = \prod_{l=i_{j-1}+1}^{i_j} p'_l,$$

and

$$T'_{(j)} = \sum_{l=i_{j-1}+1}^{i_j} T'_l$$

for $1 \le j \le m$. We write $\|\pi'\| = m$.

Lemma D.3. $c'_{(0)} \models c_0$.

Definition D.2. *Let $\pi' \in \mathcal{P}'_f(c'_{(0)})$ and $\pi \in \mathcal{P}_f(c_0)$. Then we say that π' agrees with π, denoted $\pi' \models \pi$, if*

1. *$\|\pi'\| = |\pi|$.*

2. *$c'_{(j)} \models c_j$ for $0 \le j \le \|\pi'\|$.*

3. *$p'_{(j)} = p_j$ and $T'_{(j)} = \Theta(T_j)$ for $1 \le j \le \|\pi'\|$.*

Lemma D.4. *For every $\pi' \in \mathcal{P}'_f(c'_{(0)})$, there is a unique $\pi \in \mathcal{P}_f(c_0)$ such that $\pi' \models \pi$.*

It follows immediately from Lemma D.4 that P' is time bounded by $O(T(n))$. Let $T' : \mathbb{N} \to \mathbb{N}$ be the worst case running time of P'.

Lemma D.5. *For every $\pi \in \mathcal{P}_f(c_0)$, there is a unique $\pi' \in \mathcal{P}'_f(c'_{(0)})$ such that $\pi' \models \pi$. We use $h : \mathcal{P}(c_0) \to \mathcal{P}'(c'_{(0)})$ to denote this bijection.*

Proof. (**Existence**) Direct from Lemma D.2.

(**Uniqueness**) For every $\pi \in \mathcal{P}(c_0)$, we choose an arbitrary $\pi' \in \mathcal{P}'(c'_{(0)})$ such that $\pi' \models \pi$ and define $h(\pi) = \pi'$. Then

$$1 = \sum_{\pi \in \mathcal{P}(c_0)} \pi.p = \sum_{\pi \in \mathcal{P}(c_0)} h(\pi).p \leq \sum_{\pi' \in \mathcal{P}'(c'_{(0)})} \pi'.p = 1.$$

The uniqueness of $h(\pi)$ follows. ☐

Now let $x \in \Sigma^*$ be the input string and the initial configuration of P is $c_0 = (0, \mu_0, |\psi_0\rangle, in(x), \epsilon)$. Since P is a $T(n)$-time QRAM, $|\pi| \leq T(|x|)$ is finite for every $\pi \in \mathcal{P}(c_0)$, and $|\mathcal{P}(c_0)| \leq 2^{T(|x|)}$ is also finite because each transition leads to at most two branches. So for every $y \in \Sigma^*$, we have:

$$\begin{aligned}
P(x,y) &= \sum_{c \in \mathcal{C}_f : out(c.y) = y} [\![P]\!](c_0)(c) \\
&= \sum_{c \in \mathcal{C}_f : out(c.y) = y} [\![P]\!]^{T(|x|)}(c_0)(c) \\
&= \sum_{\pi \in \mathcal{P}^{T(|x|)}(c_0) : out(\pi.c_{|\pi|}.y) = y} \pi.p \\
&= \sum_{\pi \in \mathcal{P}^{T(|x|)}(c_0) : out(h(\pi).c_{|f(\pi)|}.y) = y} h(\pi).p \\
&= \sum_{\pi' \in \mathcal{P}'^{T'(|x|)}(c'_{(0)}) : out(\pi'.c_{|\pi'|}.y) = y} \pi'.p \\
&= \sum_{\pi' \in \mathcal{P}'^{T'(|x|)}(c'_0) : out(\pi'.c_{|\pi'|}.y) = y} \pi'.p \\
&= \sum_{c' \in \mathcal{C}'_f : out(c'.y) = y} [\![P']\!]^{T'(|x|)}(c'_0)(c) \\
&= \sum_{c' \in \mathcal{C}'_f : out(c'.y) = y} [\![P']\!](c'_0)(c) \\
&= P'(x,y).
\end{aligned}$$

E $\mathrm{TM}^{\mathbf{Q}}$ instructions for simulating QRAMs

1. If P_l has the form $X_i \leftarrow C$, the following shows several steps to achieve the simulation with two work tracks work1 and work2.

$$\begin{aligned}
(p_l, 0) &: M_{\text{write}(i)}[\text{work1}] \\
(p_l, 1) &: M_{\text{write}(C)}[\text{work2}] \\
(p_l, 2) &: M_{\text{update}}[\text{creg}, \text{work1}, \text{work2}] \\
(p_l, 3) &: M_{\text{clean}}[\text{work1}] \\
(p_l, 4) &: M_{\text{clean}}[\text{work2}] \\
(p_l, 5) &: \text{transition to } (p_{l+1}, 0)
\end{aligned}$$

2. If P_l has the form $X_i \leftarrow X_j + X_k$,

$$\begin{aligned}
(p_l, 0) &: M_{\text{write}(j)}[\text{work1}] \\
(p_l, 1) &: M_{\text{fetch}}[\text{creg}, \text{work1}, \text{work2}] \\
(p_l, 2) &: M_{\text{write}(k)}[\text{work3}] \\
(p_l, 3) &: M_{\text{fetch}}[\text{creg}, \text{work3}, \text{work4}] \\
(p_l, 4) &: M_{\text{add}}[\text{work2}, \text{work4}, \text{work5}] \\
(p_l, 5) &: M_{\text{write}(i)}[\text{work6}] \\
(p_l, 6) &: M_{\text{update}}[\text{creg}, \text{work6}, \text{work5}] \\
(p_l, 7) &: M_{\text{clean}}[\text{work1}] \\
(p_l, 8) &: M_{\text{clean}}[\text{work2}] \\
(p_l, 9) &: M_{\text{clean}}[\text{work3}] \\
(p_l, 10) &: M_{\text{clean}}[\text{work4}] \\
(p_l, 11) &: M_{\text{clean}}[\text{work5}] \\
(p_l, 12) &: M_{\text{clean}}[\text{work6}] \\
(p_l, 13) &: \text{transition to } (p_{l+1}, 0)
\end{aligned}$$

3. If P_l has the form $X_i \leftarrow X_j - X_k$,

$$\begin{aligned}
(p_l,0) :& M_{\text{write}(j)}[\text{work1}]\\
(p_l,1) :& M_{\text{fetch}}[\text{creg}, \text{work1}, \text{work2}]\\
(p_l,2) :& M_{\text{write}(k)}[\text{work3}]\\
(p_l,3) :& M_{\text{fetch}}[\text{creg}, \text{work3}, \text{work4}]\\
(p_l,4) :& M_{\text{sub}}[\text{work2}, \text{work4}, \text{work5}]\\
(p_l,5) :& M_{\text{write}(i)}[\text{work6}]\\
(p_l,6) :& M_{\text{update}}[\text{creg}, \text{work6}, \text{work5}]\\
(p_l,7) :& M_{\text{clean}}[\text{work1}]\\
(p_l,8) :& M_{\text{clean}}[\text{work2}]\\
(p_l,9) :& M_{\text{clean}}[\text{work3}]\\
(p_l,10) :& M_{\text{clean}}[\text{work4}]\\
(p_l,11) :& M_{\text{clean}}[\text{work5}]\\
(p_l,12) :& M_{\text{clean}}[\text{work6}]\\
(p_l,13) :& \text{transition to } (p_{l+1},0)
\end{aligned}$$

4. If P_l has the form $X_i \leftarrow X_{X_j}$,

$$\begin{aligned}
(p_l,0) :& M_{\text{write}(j)}[\text{work1}]\\
(p_l,1) :& M_{\text{fetch}}[\text{creg}, \text{work1}, \text{work2}]\\
(p_l,2) :& M_{\text{fetch}}[\text{creg}, \text{work2}, \text{work3}]\\
(p_l,3) :& M_{\text{write}(i)}[\text{work4}]\\
(p_l,4) :& M_{\text{update}}[\text{creg}, \text{work4}, \text{work3}]\\
(p_l,5) :& M_{\text{clean}}[\text{work1}]\\
(p_l,6) :& M_{\text{clean}}[\text{work2}]\\
(p_l,7) :& M_{\text{clean}}[\text{work3}]\\
(p_l,8) :& M_{\text{clean}}[\text{work4}]\\
(p_l,9) :& \text{transition to } (p_{l+1},0)
\end{aligned}$$

5. If P_l has the form $X_{X_i} \leftarrow X_j$,

$$\begin{aligned}
(p_l, 0) :& M_{\text{write}(j)}[\text{work1}] \\
(p_l, 1) :& M_{\text{fetch}}[\text{creg}, \text{work1}, \text{work2}] \\
(p_l, 0) :& M_{\text{write}(i)}[\text{work3}] \\
(p_l, 1) :& M_{\text{fetch}}[\text{creg}, \text{work3}, \text{work4}] \\
(p_l, 4) :& M_{\text{update}}[\text{creg}, \text{work4}, \text{work2}] \\
(p_l, 5) :& M_{\text{clean}}[\text{work1}] \\
(p_l, 6) :& M_{\text{clean}}[\text{work2}] \\
(p_l, 7) :& M_{\text{clean}}[\text{work3}] \\
(p_l, 8) :& M_{\text{clean}}[\text{work4}] \\
(p_l, 9) :& \text{transition to } (p_{l+1}, 0)
\end{aligned}$$

6. If P_l has the form TRA m if $X_j > 0$,

$$\begin{aligned}
(p_l, 0) :& M_{\text{write}(j)}[\text{work1}] \\
(p_l, 1) :& M_{\text{fetch}}[\text{creg}, \text{work1}, \text{work2}] \\
(p_l, 2) :& M_{\text{gtz}}[\text{work2}] \\
(p_l, 3) :& M_{\text{clean}}[\text{work1}] \\
(p_l, 4),\ & 0_{\text{work2}} \rightarrow \#_{\text{work2}}, (p_l, 5), L \\
(p_l, 4),\ & 1_{\text{work2}} \rightarrow \#_{\text{work2}}, (p_l, 6), L \\
(p_l, 5),\ & \#_{\text{work2}} \rightarrow \#_{\text{work2}}, (p_{l+1}, 0), R \\
(p_l, 6),\ & \#_{\text{work2}} \rightarrow \#_{\text{work2}}, (p_m, 0), R
\end{aligned}$$

7. If P_l has the form READ X_i,

$$\begin{aligned}
(p_l, 0) :& M_{\text{read}}[\text{input}, \text{work1}] \\
(p_l, 1) :& M_{\text{write}(i)}[\text{work2}] \\
(p_l, 2) :& M_{\text{update}}[\text{creg}, \text{work2}, \text{work1}] \\
(p_l, 3) :& M_{\text{clean}}[\text{work1}] \\
(p_l, 4) :& M_{\text{clean}}[\text{work2}] \\
(p_l, 5) :& \text{transition to } (p_{l+1}, 0)
\end{aligned}$$

8. If P_l has the form WRITE X_i,

$$\begin{aligned}
(p_l, 0) :& M_{\text{write}(i)}[\text{work1}] \\
(p_l, 1) :& M_{\text{fetch}}[\text{creg}, \text{work1}, \text{work2}] \\
(p_l, 2) :& M_{\text{gtz}}[\text{work2}] \\
(p_l, 3) :& M_{\text{append}}[\text{output}, \text{work2}] \\
(p_l, 4) :& M_{\text{clean}}[\text{work1}] \\
(p_l, 5) :& M_{\text{clean}}[\text{work2}] \\
(p_l, 6) :& \text{transition to } (p_{l+1}, 0)
\end{aligned}$$

F Details of Section 9.1

F.1 Several Lemmas for QTMs

In order to prove our lemmas, we need some lemmas mainly from [4].

Lemma F.1 (Lemma B.7 of [4]). *There is a stationary oblivious reversible TM M such that*

$$x; \epsilon \xrightarrow[T]{M} x; x \text{ and } x; x \xrightarrow[T]{M} x; \epsilon,$$

where $T = 2|x| + 4$.

Lemma F.2 (Lemma B.6 of [4]). *There is a stationary oblivious reversible TM M such that*

$$x; y \xrightarrow[T]{M} y; x,$$

where $T = 2\max\{|x|, |y|\} + 4$.

Lemma F.3 (Reversal Lemma, Lemma 4.12 of [4]). *For every well-formed and stationary QTM M, there is a well-formed and stationary QTM M' such that*

$$|\mathcal{T}\rangle \xrightarrow[T]{M} |\mathcal{T}'\rangle \Longrightarrow |\mathcal{T}'\rangle \xrightarrow[T+2]{M'} |\mathcal{T}\rangle .$$

F.2 Proof of lemmas in Section 9.1

The following Lemmas F.4, F.5 and F.6 are essentially Theorem B.8 and Theorem B.9 in [4], but here they are slightly strengthened for our purpose.

Lemma F.4. *For every stationary deterministic TM M, there is a stationary reversible TM M' such that*

$$\mathcal{T} \xrightarrow[T]{M} \mathcal{T}' \Longrightarrow \mathcal{T}; \epsilon; \epsilon \xrightarrow[T']{M'} \mathcal{T}'; @; \mathcal{T}^h,$$

where:

1. *$\mathcal{T}^h = \#\$(p_0, \sigma_0) \dots (p_{T-1}, \sigma_{T-1})$ encodes the history, and p_t and σ_t denote the state and the symbol at the head position at time t in the execution of M, respectively;*
2. *$T' = O(T^2)$.*

Moreover, if M is oblivious, then so is M'.

Proof. **Step 1**. At the very beginning, we construct a RTM M_1 that writes an end marker @ on the second track and end marker \$ on the third track, and then comes back to the initial position with state q_0, by including these instructions:

$$\begin{array}{lcccl}
q_a, & (\forall_1, \#_2, \#_3) & \to & (\forall_1, @, \#), & q_b, & R \\
q_b, & (\forall_1, \#_2, \#_3) & \to & (\forall_1, \#, \$), & q_c, & R \\
q_c, & (\forall_1, \forall_2, \forall_3) & \to & (\forall_1, \forall_2, \forall_3), & q_d, & L \\
q_d, & (\forall_1, \forall_2, \forall_3) & \to & (\forall_1, \forall_2, \forall_3), & q_e, & L \\
q_e, & (\forall_1, \forall_2, \forall_3) & \to & (\forall_1, \forall_2, \forall_3), & q_g, & L \\
q_g, & (\forall_1, \forall_2, \forall_3) & \to & (\forall_1, \forall_2, \forall_3), & q_0, & R
\end{array}$$

It is easy to see that

$$|q_a, \mathcal{T}; \epsilon; \epsilon, 0\rangle \xrightarrow[T_1]{M_1} |q_0, \mathcal{T}; @; \#\$, 0\rangle,$$

where $T_1 = 6$.

Step 2. We construct RTM M_2 as follows. For $p \in Q \setminus \{q_f\}$ and $\sigma \in \Sigma$ with transition $\delta(p, \sigma) = (\tau, q, d)$ in M, we make transitions to go from p to q updating the first track, i.e. the simulated tape of M, and adding (p, σ) to the end of the history. Include these instructions:

$$\begin{array}{cccccc}
p, & (\sigma, @, \forall_3) & \to & (\tau, \#_2, \forall_3), & (q, p, \sigma, 1), & d \\
(q, p, \sigma, 1), & (\forall_1, \#_2, \$) & \to & (\forall_1, @, \$), & (q, p, \sigma, 3), & R \\
(q, p, \sigma, 1), & (\forall_1, \#_2, \forall_3') & \to & (\forall_1, @, \forall_3'), & (q, p, \sigma, 3), & R \\
(q, p, \sigma, 1), & (\forall_1, \#_2, \#_3) & \to & (\forall_1, @, \#_3), & (q, p, \sigma, 2), & R \\
(q, p, \sigma, 2), & (\forall_1, \#_2, \#_3) & \to & (\forall_1, \#_2, \#_3), & (q, p, \sigma, 2), & R \\
(q, p, \sigma, 2), & (\forall_1, \#_2, \$) & \to & (\forall_1, \#_2, \$), & (q, p, \sigma, 3), & R \\
(q, p, \sigma, 3), & (\forall_1, \#_2, \forall_3') & \to & (\forall_1, \#_2, \forall_3'), & (q, p, \sigma, 3), & R \\
(q, p, \sigma, 3), & (\forall_1, \#_2, \#_3) & \to & (\forall_1, \#_2, (p, \sigma)), & (q, 4), & R
\end{array}$$

When $(q, 4)$ is reached, the tape head is on the first blank after the end of the history (on the third track). Now we move the tape head back to the position of tape head of M by including these instructions:

$$\begin{array}{cccccc}
(q, 4), & (\forall_1, \#_2, \#_3) & \to & (\forall_1, \#_2, \#_3), & (q, 5), & L \\
(q, 5), & (\forall_1, \#_2, \forall_3') & \to & (\forall_1, \#_2, \forall_3'), & (q, 5), & L \\
(q, 5), & (\forall_1, \#_2, \$) & \to & (\forall_1, \#_2, \$), & (q, 6), & L \\
(q, 5), & (\forall_1, @, \forall_3') & \to & (\forall_1, @, \forall_3'), & (q, 7), & L \\
(q, 5), & (\forall_1, @, \$) & \to & (\forall_1, @, \$), & (q, 7), & L \\
(q, 6), & (\forall_1, \#_2, \#_3) & \to & (\forall_1, \#_2, \#_3), & (q, 6), & L \\
(q, 6), & (\forall_1, @, \#_3) & \to & (\forall_1, @, \#_3), & (q, 7), & L \\
(q, 7), & (\forall_1, \forall_2, \forall_3) & \to & (\forall_1, \forall_2, \forall_3), & q, & R
\end{array}$$

It is easy to see that if $|q_0, \mathcal{T}, 0\rangle \xrightarrow[T]{M} |q_f, \mathcal{T}', 0\rangle$, then

$$|q_0, \mathcal{T}; @; \#\$, 0\rangle \xrightarrow[T_2]{M_2} \left|q_f, \mathcal{T}'; @; \mathcal{T}^h, 0\right\rangle,$$

where

$$T_2 = \sum_{t=0}^{T-1} \left(2\left[(t+3) - pos(\mathcal{T}, t)\right] + \begin{cases} 5 & d(\mathcal{T}, t) = L \\ 1 & d(\mathcal{T}, t) = R \end{cases} \right) = O(T^2),$$

$pos(\mathcal{T},t)$ and $d(\mathcal{T},t)$ denote the head position and the chosen direction at time t in the execution of $\mathcal{M}$ starting from tape content $\mathcal{T}$, respectively. We note that $d(\mathcal{T},t) = pos(\mathcal{T},t+1) - pos(\mathcal{T},t)$.

Conclusion. The RTM M' is obtained by dovetailing the two RTMs M_1 and M_2 by Lemma 9.4, which immediately yields

$$\mathcal{T} \xrightarrow[T]{M} \mathcal{T}' \Longrightarrow \mathcal{T}; \epsilon; \epsilon \xrightarrow[T']{M'} \mathcal{T}'; @; \mathcal{T}^h,$$

where $T' = T_1 + T_2 = O(T^2)$. Moreover, if M is oblivious, for every input $x \in \{0,1\}^*$, i.e. the initial tape content is $\mathcal{T}_x$, the running time and the head position of M can be denoted by $T = T(|x|)$ and $pos(\mathcal{T},t) = pos(|x|,t)$, respectively. It can be seen that the constructed M' is also oblivious by noticing that

1. During the simulation for time t of M, the head position starts at $pos(|x|,t)$ and goes right to $t+3$ and back. The head position of M' during the whole execution of this part of simulation only depends on $pos(|x|,t)$.

2. T_2 depends only on $pos(|x|,t)$ because $pos(\mathcal{T},t) = pos(|x|,t)$ and $d(\mathcal{T},t) = d(|x|,t) = pos(|x|,t+1) - pos(|x|,t)$. Therefore, the running time $T' = T_1 + T_2$ of M' depends only on $|x|$.

□

Lemma F.5. *Let M be a stationary deterministic TM such that for every input $x \in \{0,1\}^*$,*

$$x \xrightarrow[T]{M} M(x).$$

There is a stationary reversible TM M' such that

$$x; \epsilon \xrightarrow[T']{M'} x; M(x) \text{ and } x; M(x) \xrightarrow[T']{M'} x; \epsilon,$$

where $T' = O(T^2)$. Moreover, if M is oblivious and the length of $M(x)$ only depends on $|x|$, then M' is oblivious.

Proof. Let M_h be the constructed RTM corresponding to M in Lemma F.4 and M_h^{-1} be its reversal by Lemma F.3, and M_c be the constructed RTM in Lemma F.1. Then M' is constructed by dovetailing $M_h[1,2,3]$, $M_c[1,4]$ and $M_h^{-1}[1,2,3]$ by Lemma 9.4. We could verify that:

$$\begin{aligned} x; \epsilon; \epsilon; \epsilon &\xrightarrow[T_h]{M_h[1,2,3]} M(x); @; \mathcal{T}^h; \epsilon \\ &\xrightarrow[T_c]{M_c[1,4]} M(x); @; \mathcal{T}^h; M(x) \\ &\xrightarrow[T_h+2]{M_h^{-1}[1,2,3]} x; \epsilon; \epsilon; M(x) \end{aligned}$$

and

$$\begin{aligned} x; \epsilon; \epsilon; M(x) &\xrightarrow[T_h]{M_h[1,2,3]} M(x); @; \mathcal{T}^h; M(x) \\ &\xrightarrow[T_c]{M_c[1,4]} M(x); @; \mathcal{T}^h; \epsilon \\ &\xrightarrow[T_h+2]{M_h^{-1}[1,2,3]} x; \epsilon; \epsilon; \epsilon \end{aligned}$$

with running time $T' = T_h + T_c + T_h + 2 = O(T^2)$. $\square$

Lemma F.6. *Let M_1, M_2 be two stationary deterministic TMs such that for every $x \in \{0,1\}^*$,*

$$x \xrightarrow[T_1]{M_1} M_1(x) \text{ and } M_1(x) \xrightarrow[T_2]{M_2} x.$$

There are two stationary reversible TMs N_1 and N_2 such that

$$x \xrightarrow[T']{N_1} M_1(x) \text{ and } M_1(x) \xrightarrow[T']{N_2} x,$$

where $T' = O(T_1^2 + T_2^2)$. Moreover, if M_1 and M_2 are oblivious and the length of $M_1(x)$ only depends on $|x|$, then N_1 and N_2 are oblivious.

Proof. Let M_1' and M_2' be the RTMs constructed by Lemma F.5 and M_{swap} be the RTM in Lemma F.2. N_1 is constructed by dovetailing $M_1'[1,2]$, $M_{swap}[1,2]$ and $M_2'[1,2]$. Verify that

$$x; \epsilon \xrightarrow[T_1']{M_1'[1,2]} x; M_1(x) \xrightarrow[T_{swap}]{M_{swap}[1,2]} M_1(x); x \xrightarrow[T_2']{M_2'[1,2]} M_1(x); \epsilon,$$

where $T_1' = O(T_1^2), T_2' = O(T_2^2)$ and $T_{swap} = O(|x| + |M(x)|)$. N_2 is constructed by dovetailing $M_2'[1,2]$, $M_{swap}[1,2]$ and $M_1'[1,2]$. Verify that

$$M_1(x); \epsilon \xrightarrow[T_2']{M_2'[1,2]} M_1(x); x \xrightarrow[T_{swap}]{M_{swap}[1,2]} x; M_1(x) \xrightarrow[T_1']{M_1'[1,2]} x; \epsilon,$$

$\square$

Now we are ready to prove Lemma 9.1 and Lemma 9.2.

Proof of Lemma 9.1

The proof is immediately shown by giving two oblivious DTMs using Lemma F.6.

Below is an oblivious TM M_+:

$$\begin{array}{rccrrc}
q_0, & \forall & \to & \forall, & q_1, & L \\
q_1, & \# & \to & \#, & q_2, & R \\
q_2, & x & \to & x, & q_2, & R \\
q_2, & \# & \to & \#, & (q_3,1), & L \\
(q_3,0), & 0 & \to & 0, & (q_3,0), & L \\
(q_3,0), & 1 & \to & 1, & (q_3,0), & L \\
(q_3,0), & \# & \to & \#, & q_f, & R \\
(q_3,1), & 0 & \to & 1, & (q_3,0), & L \\
(q_3,1), & 1 & \to & 0, & (q_3,1), & L \\
(q_3,1), & \# & \to & \#, & q_f, & R
\end{array}$$

It can be verified that M_+ increments x by 1 and has running time $2|x| + 4 = O(|x|)$.

Below is an oblivious TM M_-:

$q_0,$	$\forall$	$\rightarrow$	$\forall,$	$q_1,$	L
$q_1,$	$\#$	$\rightarrow$	$\#,$	$q_2,$	R
$q_2,$	x	$\rightarrow$	$x,$	$q_2,$	R
$q_2,$	$\#$	$\rightarrow$	$\#,$	$(q_3,1),$	L
$(q_3,0),$	0	$\rightarrow$	$0,$	$(q_3,0),$	L
$(q_3,0),$	1	$\rightarrow$	$1,$	$(q_3,0),$	L
$(q_3,0),$	$\#$	$\rightarrow$	$\#,$	$q_f,$	R
$(q_3,1),$	0	$\rightarrow$	$1,$	$(q_3,1),$	L
$(q_3,1),$	1	$\rightarrow$	$0,$	$(q_3,0),$	L
$(q_3,1),$	$\#$	$\rightarrow$	$\#,$	$q_f,$	R

It can be verified that M_- decrements x by 1 and has running time $2|x|+4 = O(|x|)$.

Proof of Lemma 9.2

The proof is immediately shown by giving an oblivious DTM using Lemma F.6. It is noted that the given DTM itself is the reversal of it.

The an oblivious TM $M_=$ is as below:

$q_0,$	$(\forall_1,\forall_2,\forall_3)$	$\rightarrow$	$(\forall_1,\forall_2,\forall_3),$	$q_1,$	L
$q_1,$	$(\forall_1,\forall_2,\forall_3)$	$\rightarrow$	$(\forall_1,\forall_2,\forall_3),$	$q_2,$	R
$q_2,$	$(x_1,x_2,\forall_3)$	$\rightarrow$	$(x_1,x_2,\forall_3),$	$q_2,$	R
$q_2,$	$(\#_1,\#_2,\#_3)$	$\rightarrow$	$(\#_1,\#_2,\#_3),$	$(q_3,0),$	L
$(q_3,0),$	$(x,x,\forall_3)$	$\rightarrow$	$(x,x,\forall_3),$	$(q_3,0),$	L
$(q_3,0),$	$(0,1,\forall_3)$	$\rightarrow$	$(0,1,\forall_3),$	$(q_3,1),$	L
$(q_3,0),$	$(1,0,\forall_3)$	$\rightarrow$	$(1,0,\forall_3),$	$(q_3,1),$	L
$(q_3,0),$	$(\#_1,\#_2,\#_3)$	$\rightarrow$	$(\#_1,\#_2,\#_3),$	$(q_4,0),$	R
$(q_3,1),$	$(x_1,x_2,\forall_3)$	$\rightarrow$	$(x_1,x_2,\forall_3),$	$(q_3,1),$	L
$(q_3,1),$	$(\#_1,\#_2,\#_3)$	$\rightarrow$	$(\#_1,\#_2,\#_3),$	$(q_4,1),$	R
$(q_4,0),$	$(x_1,x_2,0)$	$\rightarrow$	$(x_1,x_2,0),$	$q_5,$	L
$(q_4,0),$	$(x_1,x_2,1)$	$\rightarrow$	$(x_1,x_2,1),$	$q_5,$	L
$(q_4,1),$	$(x_1,x_2,0)$	$\rightarrow$	$(x_1,x_2,1),$	$q_5,$	L
$(q_4,1),$	$(x_1,x_2,1)$	$\rightarrow$	$(x_1,x_2,0),$	$q_5,$	L
$q_5,$	$(\forall_1,\forall_2,\forall_3)$	$\rightarrow$	$(\forall_1,\forall_2,\forall_3),$	$q_f,$	R

It can be verified that $M_=$ checks whether x and y are equal and has running time $2|x|+6 = O(|x|)$. Moreover, we have that $M_=(M_=(x;y;z)) = x;y;z$.

Finally, we give a tape shifting lemma in order to prove Lemma 9.3.

Lemma F.7 (Tape Shifting)**.** *There is a stationary reversible TM M that copies the first track to the second track if the content of the first track is shifted left or right by one step. Formally, for every $x \in \{0,1\}^+$,*

$$\operatorname{shl} x; \epsilon \xrightarrow[T]{M} \operatorname{shl} x; \operatorname{shl} x \text{ and } \operatorname{shr} x; \epsilon \xrightarrow[T]{M} \operatorname{shl} x; \operatorname{shr} x,$$

where $T = 2|x| + 8$.

Proof. Below is the construction. We use σ to denote any symbol other than #, d to denote both directions L and R and $\bar{d}$ to denote the reverse direction of d.

$q_0,$	$(\sigma, \#)$	$\rightarrow$	$(\sigma, \#),$	$(q_L, 1),$	L
$q_0,$	$(\#, \#)$	$\rightarrow$	$(\#, \#),$	$(q_R, 1),$	R
$(q_d, 1),$	$(\sigma, \#)$	$\rightarrow$	$(\sigma, \#),$	$(q_d, 2),$	L
$(q_d, 2),$	$(\#, \#)$	$\rightarrow$	$(\#, \#),$	$(q_d, 3),$	R
$(q_d, 3),$	$(\sigma, \#)$	$\rightarrow$	$(\sigma, \sigma),$	$(q_d, 3),$	R
$(q_d, 3),$	$(\#, \#)$	$\rightarrow$	$(\#, \#),$	$(q_d, 4),$	L
$(q_d, 4),$	(σ, σ)	$\rightarrow$	$(\sigma, \sigma),$	$(q_d, 4),$	L
$(q_d, 4),$	$(\#, \#)$	$\rightarrow$	$(\#, \#),$	$(q_d, 5),$	R
$(q_d, 5),$	(σ, σ)	$\rightarrow$	$(\sigma, \sigma),$	$(q_d, 6),$	$\bar{d}$
$(q_L, 6),$	(σ, σ)	$\rightarrow$	$(\sigma, \sigma),$	$q_7,$	L
$(q_R, 6),$	$(\#, \#)$	$\rightarrow$	$(\#, \#),$	$q_7,$	L
$q_7,$	$(\forall, \forall)$	$\rightarrow$	$(\forall, \forall),$	$q_f,$	R

□

Now we are ready to prove Lemma 9.3.

Proof of Lemma 9.3

Below is an oblivious TM M_r:

$q_0,$	$\forall$	$\rightarrow$	$\forall,$	$q_1,$	L
$q_1,$	$\#$	$\rightarrow$	$\#,$	$q_2,$	R
$q_2,$	x	$\rightarrow$	$x,$	$q_2,$	R
$q_2,$	$\#$	$\rightarrow$	$\#,$	$q_3,$	L
$q_3,$	x	$\rightarrow$	$x,$	$(q_4, x),$	R
$q_3,$	$\#$	$\rightarrow$	$\#,$	$q_6,$	R
$(q_4, x),$	$\forall$	$\rightarrow$	$x,$	$q_5,$	L
$q_5,$	$\forall$	$\rightarrow$	$\forall,$	$q_3,$	L
$q_6,$	$\forall$	$\rightarrow$	$\#,$	$q_7,$	L
$q_7,$	$\#$	$\rightarrow$	$\#,$	$q_f,$	R

$\forall$ denotes any symbol in Σ while x denotes any symbol in Σ other than #. It can be verified that M_r shifts the tape right by a cell and has running time $4\,|x| + 6 = O(|x|)$.

Below is an oblivious TM M_l:

$q_0,$	$\forall$	$\rightarrow$	$\forall,$	$q_1,$	L
$q_1,$	$\#$	$\rightarrow$	$\#,$	$q_2,$	R
$q_2,$	x	$\rightarrow$	$x,$	$(q_3, x),$	L
$q_2,$	$\#$	$\rightarrow$	$\#,$	$q_5,$	L
(q_3, x)	$\forall$	$\rightarrow$	$x,$	$q_4,$	R
q_4	$\forall$	$\rightarrow$	$\forall,$	$q_2,$	R
$q_5,$	$\forall$	$\rightarrow$	$\#,$	$q_6,$	L
$q_6,$	x	$\rightarrow$	$x,$	$q_6,$	L
$q_6,$	$\#$	$\rightarrow$	$\#,$	$q_7,$	R
$q_7,$	$\forall$	$\rightarrow$	$\forall,$	$q_f,$	R

It can be verified that M_r shifts the tape left by a cell and has running time $4\,|x| + 6 = O(|x|)$.

Let M_l' and M_r' be the RTMs constructed by Lemma F.4 corresponding to M_l and M_r, respectively. Let M_c be the RTM in Lemma F.7. It is noted that for every $x \in \{0,1\}^+$,

$$x; \epsilon; \epsilon; \epsilon \xrightarrow[T_l]{M_l'[1,2,3]} \operatorname{shl} x; @; \mathcal{T}^{hl}; \epsilon \xrightarrow[T_c]{M_c[1,4]} \operatorname{shl} x; @; \mathcal{T}^{hl}; \operatorname{shl} x$$

and

$$x; \epsilon; \epsilon; \epsilon \xrightarrow[T_r]{M_r'[1,2,3]} \operatorname{shr} x; @; \mathcal{T}^{hr}; \epsilon \xrightarrow[T_c]{M_c[1,4]} \operatorname{shr} x; @; \mathcal{T}^{hr}; \operatorname{shr} x.$$

Moreover, it can be verified that $T_l = T_r = O(|x|^2)$.

We note that

$$\operatorname{shl} x; @; \mathcal{T}^{hl}; \operatorname{shl} x \xrightarrow[T_l+2]{M_l'^{-1}[1,2,3]} x; \epsilon; \epsilon; \operatorname{shl} x,$$

$$\operatorname{shl} x; @; \mathcal{T}^{hl}; \operatorname{shl} x \xrightarrow[T_l+2]{M_l'^{-1}[4,2,3]} \operatorname{shl} x; \epsilon; \epsilon; x,$$

$$\operatorname{shr} x; @; \mathcal{T}^{hr}; \operatorname{shr} x \xrightarrow[T_r+2]{M_r'^{-1}[1,2,3]} x; \epsilon; \epsilon; \operatorname{shr} x,$$

$$\operatorname{shr} x; @; \mathcal{T}^{hr}; \operatorname{shr} x \xrightarrow[T_r+2]{M_l'^{-1}[4,2,3]} \operatorname{shr} x; \epsilon; \epsilon; x.$$

According to these four cases, we can obtain four RTMs as follows:

$$x; \epsilon \xrightarrow[T_l^1]{M_l^1} \operatorname{shl} x; x,$$

$$x; \epsilon \xrightarrow[T_l^2]{M_l^2} x; \operatorname{shl} x,$$

$$x; \epsilon \xrightarrow[T_r^1]{M_r^1} \operatorname{shr} x; x,$$

$$x; \epsilon \xrightarrow[T_r^2]{M_r^2} x; \operatorname{shr} x$$

with $T_l^1 = T_l^2 = T_r^1 = T_r^2 = O(|x|^2)$.

We use M_L and M_R to denote the RTM that moves the tape head left and right, respectively, without modifying anything. Formally,

$$|\mathcal{T}, \xi\rangle \xrightarrow[3]{M_L} |\mathcal{T}, \xi - 1\rangle$$

and

$$|\mathcal{T}, \xi\rangle \xrightarrow[3]{M_R} |\mathcal{T}, \xi + 1\rangle.$$

The running time is 3 because M_L and M_R should be in normal form, and we achieve this by making M_L go left, left and right and making M_R go right, left, right, both of which need three steps. The construction of M_L and M_R is trivial. Now we are able to build two RTMs that just shift left or right the whole tape. Note that

$$|x; \epsilon, 0\rangle \xrightarrow[T_l^1]{M_l^1} |\operatorname{shl} x; x, 0\rangle \xrightarrow[3]{M_L} |\operatorname{shl} x; x, -1\rangle \xrightarrow[T_r^2]{M_r^2} |\operatorname{shl} x; \epsilon, -1\rangle \xrightarrow[3]{M_R} |\operatorname{shl} x; \epsilon, 0\rangle,$$

and

$$|x; \epsilon, 0\rangle \xrightarrow[T_r^1]{M_r^1} |\operatorname{shr} x; x, 0\rangle \xrightarrow[3]{M_R} |\operatorname{shr} x; x, 1\rangle \xrightarrow[T_l^2]{M_l^2} |\operatorname{shr} x; \epsilon, 1\rangle \xrightarrow[3]{M_L} |\operatorname{shr} x; \epsilon, 0\rangle.$$