The Black-Box Simulation Barrier Persists in a Fully Quantum World

Nai-Hui Chia[1], Kai-Min Chung[2], Xiao Liang[3], and Jiahui Liu[4]

[1] Rice University, USA
nc67@rice.edu
[2] Academia Sinica, Taiwan
kmchung@iis.sinica.edu.tw
[3] The Chinese University of Hong Kong, Hong Kong
xiaoliang@cuhk.edu.hk
[4] Massachusetts Institute of Technology and Fujitsu Research, USA
jiahuiliu@csail.mit.edu

Abstract. Zero-Knowledge (ZK) protocols have been a subject of intensive study due to their fundamental importance and versatility in modern cryptography. However, the inherently different nature of quantum information significantly alters the landscape, necessitating a re-examination of ZK designs.

A crucial aspect of ZK protocols is their round complexity, intricately linked to *simulation*, which forms the foundation of their formal definition and security proofs. In the *post-quantum* setting, where honest parties and their communication channels are all classical but the adversaries could be quantum, Chia, Chung, Liu, and Yamakawa [FOCS'21] demonstrated the non-existence of constant-round *black-box-simulatable* ZK arguments (BBZK) for **NP** unless **NP** $\subseteq$ **BQP**. However, this problem remains widely open in the full-fledged quantum future that will eventually arrive, where all parties (including the honest ones) and their communication are naturally quantum.

Indeed, this problem is of interest to the broader theory of quantum computing. It has been an important theme to investigate how quantum power fundamentally alters traditional computational tasks, such as the *unconditional* security of Quantum Key Distribution and the incorporation of Oblivious Transfers in MiniQCrypt. Moreover, quantum communication has led to round compression for commitments and interactive arguments. Along this line, the above problem is of great significance in understanding whether quantum computing could also change the nature of ZK protocols in some fundamentally manner.

We resolved this problem by proving that only languages in **BQP** admit constant-round *fully-quantum* BBZK. This result holds significant implications. Firstly, it illuminates the nature of quantum zero-knowledge and provides valuable insights for designing future protocols in the quantum realm. Secondly, it relates ZK round complexity with the intriguing problem of **BQP** vs **QMA**, which is out of the reach of previous analogue impossibility results in the classical or post-quantum setting. Lastly, it justifies the need for the *non-black-box* simulation techniques or the relaxed security notions employed in existing constant-round fully-quantum BBZK protocols.

Keywords: Quantum Zero-Knowledge · Black-Box · Impossibility

Table of Contents

1 Introduction

Zero-Knowledge (ZK) protocols, pioneered by Goldwasser, Micali, and Rackoff [GMR85], enable a prover to demonstrate the truth of a statement (e.g., one in an **NP** language) without disclosing additional information (e.g., the **NP** witness) beyond the statement's truthfulness. Since their inception, ZK protocols have garnered considerable research interest and emerged as a cornerstone of cryptography. Apart from serving as versatile primitives, ZK protocols are integral to *secure multi-party computation* (MPC), another cornerstone of modern cryptography. Furthermore, exploration of ZK protocols has unveiled intriguing aspects of computational complexity theory, as evidenced by various studies [IY88, SV97a, SV97b, BGG+90, BMO90, GSY19, GIK+23] etc. Thus, investigating the nature of ZK protocols is of significant importance.

Round Complexity and Black-Box Simulation. A crucial aspect of ZK protocol research is understanding their round complexity, which is intricately linked to the concept of *simulation*, upon which the formal definition of ZK protocols (and consequently their security proofs) relies.

The formal definition of ZK mandates the presence of a *simulator* $\mathcal{S}$ that interacts with a malicious verifier $\mathcal{V}^*$ and convinces $\mathcal{V}^*$ that it is communicating with the honest prover. Importantly, $\mathcal{S}$ possesses no secret input akin to that of the honest prover. Thus, the success of $\mathcal{S}$ in convincing $\mathcal{V}^*$ encapsulates the intuitive essence of ZK protocols — $\mathcal{V}^*$ gains no extra information (other than the statement's truthfulness) from the interaction with the honest prover.

Typically, the simulator $\mathcal{S}$ interacts with $\mathcal{V}^*$ in a *black-box* manner, meaning that $\mathcal{S}$ solely leverages the Input/Output behavior of $\mathcal{V}^*$ and treats it as an oracle. This black-box approach is arguably the most natural choice for simulation, often being simpler and more modular compared to non-black-box methods. Indeed, black-box simulation is widely employed in most positive results for ZK (and MPC) protocols, including notable works such as [GMW86, FS90, GK96a, BCY91, BJY97], among others (exceptions are discussed in Sec. 1.2).

Strong impossibility results have been established for ZK arguments[5] employing black-box simulation (dubbed BBZK henceforth). Goldreich and Krawczyk [GK96b] demonstrated that no three-round BBZK protocols or public-coin constant-round BBZK protocols for **NP** exist unless **NP** $\subseteq$ **BPP**. Barak and Lindell [BL02] further established that constant-round BBZK protocols with *strict polynomial-time* simulation are non-existent unless **NP** $\subseteq$ **BPP**. Here, strict polynomial-time simulation indicates that the simulator always runs in a fixed polynomial time, unlike an *expected polynomial-time* simulator whose runtime is polynomial time only in expectation. Notably, as per [BL02], all existing constant-round BBZK protocols for **NP** rely on expected polynomial-time simulation.

These impossibility results significantly advance our comprehension of zero-knowledge and offer valuable guidance for positive results (i.e., constructions).

BBZK in the Quantum Era. All the above results are in the classical setting. However, it is known that quantum information behaves in a fundamentally different manner. This poses an intriguing question: *What can we say about the round complexity of ZK protocols in the quantum setting?* Toward answering this question, two models need consideration — the *post-quantum* model and the *fully quantum* model.

Post-Quantum ZK. This is the model where honest parties and their communication channels are all classical, but the adversary could be a quantum machine. This model is particularly interesting

[5] This refers to ZK protocols with *computational* soundness guarantee.

in the near future, where adversaries may gain early access to quantum computing capabilities while honest parties are not required to catch up to remain protected against them.

Exciting progress has been made in this model. The recent breakthrough by Bitansky and Shmueli [BS20] presents a constant-round post-quantum ZK argument for **NP** using *non-black-box simulation*. On the other hand, Chia, Chung, Liu, and Yamakawa [CCLY21] show that there does not exist a constant-round post-quantum BBZK for **NP** unless **NP** $\subseteq$ **BQP**. It is worth noting that the result by [CCLY21] holds even for *expected* quantum-polynomial-time (QPT) simulation. Thus, it is *not* merely a analogue of [BL02], but rather represents a qualitatively stronger impossibility result in the post-quantum setting. These two results together essentially provide a complete characterization of the round complexity of BBZK in the post-quantum setting.

Fully-Quantum ZK. This model represents the full-fledged quantum scenario meant to capture the future quantum landscape, where all parties (including the honest ones) and communication channels are allowed to be quantum. Unlike the post-quantum model, the fully-quantum capabilities across all parties in this model opens up new possibilities. One example is the existence of ZK protocols for **QMA**, which is infeasible in the post-quantum setting because classical honest provers cannot utilize quantum **QMA** witnesses.

In contrast to the post-quantum model, our understanding of the round complexity of *fully-quantum* BBZK is rather limited. While there exist fully-quantum BBZK protocols for **NP** and even **QMA** (e.g., [BJSW16, BG20], etc.), they require at least super-constant rounds. Regarding impossibility results, due to the quantum power of the honest parties, it is unclear if the aforementioned results in the post-quantum setting still hold. The only marginally relevant result is the work of [JKMR09], which demonstrates the nonexistence of constant-round public-coin (or three-round but potentially private-coin) post-quantum BBZK proofs for **NP** unless **NP** $\subseteq$ **BQP**; this result is relevant because it holds even if the last message in the protocol is quantum. In summary, the central problem in the area remains widely open:

> **Question:** *Do there exist constant-round fully-quantum BBZK protocols for* **NP** *(or* **QMA***) with strict (or expected) QPT simulation?*

It is worth noting that this question is of interest to the broader theory of quantum computing. It has been an important theme to investigate how quantum power fundamentally alters traditional computational tasks, such as the *unconditional* security of Quantum Key Distribution [BB84] and the incorporation of Oblivious Transfers in MiniQCrypt [GLSV21, BCKM21]. Additionally, quantum communication has led to round compression for commitments [Yan22] and interactive arguments [KW00, KKMV09, BQSY24]. Along this line, this Question is of great significance in understanding whether quantum computing could also alter the nature of zero-knowledge protocols in some fundamentally manner.

Moreover, answers to this Question hold significance for the following reasons:

1. Similar to the impossibility results in the classical and post-quantum settings, answers to this Question would help further our understanding of zero-knowledge and serve as valuable guidance for protocol design in the full-fledged quantum future.
2. Answers to this Question would relate zero-knowledge with the intriguing problem of **BQP** vs **QMA**, while post-quantum protocols could at most relate to **BQP** vs **NP**, as explained above.
3. Recent works (e.g., [CCY21, CCLY22]) have achieved constant-round constructions for post-quantum ZK and even 2PC w.r.t. a relaxed security notion called ε*-simulation*. These results, along with the aforementioned [BS20, CM21] which utilize non-black-box simulation, all extend

to the fully-quantum setting[6]. Without any answers to the above Question, it remains unclear whether these results, when viewed in the fully-quantum setting, are truly optimal (in the sense that the relaxation to ε-simulation or the use of non-black-box simulation is really necessary).

1.1 Our Results

We answer the above Question by establishing the following impossibility result.

Theorem 1. *For any language $\mathcal{L}$, if there exists a constant-round fully-quantum BBZK protocol with expected QPT simulation, then it holds that $\mathcal{L} \in \mathbf{BQP}$.*

It is worth noticing that Thm. 1 rules out *expected* QPT simulation, and thus it can be viewed as the exact analogue of the [CCLY21] impossibility in the fully-quantum context. However, our techniques to establishing Thm. 1 diverge significantly from [CCLY21] due to the fundamentally different nature inherent in fully-quantum protocols. Further discussion on this matter will be provided in Sec. 2.

As mentioned earlier in Item 2, since Thm. 1 is about quantum protocols, it allows us to connect zero-knowledge with the quantum complexity class **QMA**, which we state as the following corollary.

Corollary 1 (of Thm. 1). *There does not exist any constant-round fully-quantum BBZK protocol for* **QMA** *(resp.* **NP***) unless* $\mathbf{QMA} \subseteq \mathbf{BQP}$ *(resp.* $\mathbf{NP} \subseteq \mathbf{BQP}$*).*

Lastly, as previously mentioned in Item 3, Thm. 1 justifies the necessity, *in the fully-quantum context*, of (1) the use of non-black-box simulation in [BS20], (2) the relaxation to ε-simulation in [CCY21, CCLY22], and (3) the non-constant round complexity observed in existing BBZK or secure *quantum* computation protocols with black-box simulation, e.g., [DNS12, BJSW16, BG20, DGJ+20, GLSV21, ACL21] etc.

1.2 More Related Work

In the classical setting, there exist constant-round constructions of ZK (and even MPC) that utilize non-black-box simulation to bypass the [BL02] lower bound, e.g., [Bar01, BG01, Lin03, Pas04, BP12, Goy13, CPS13, PPS15, GOSV14], etc. But compared with black-box simulation, our knowledge of non-black-box simulation is still limited.

We summarize additional impossibility results that are not directly related to the current work. [Kat08] proved that there does not exist a four-round ZK *proof* with black-box simulation for **NP** unless $\mathbf{NP} \subseteq \mathbf{coMA}$. [HPV20] showed that only languages in **coMA** admit a four-round *fully*[7] black-box ZK *argument* based on one-way functions. [KRR17] demonstrated that there does not exist constant-round public-coin ZK proofs for **NP** even with non-black-box simulation under certain assumptions on obfuscation. [FGJ18] showed that there does not exist three-round ZK proofs for **NP** even with non-black-box simulation under the same assumptions.

The recent work by Lombardi, Ma, and Spooner [LMS22] introduced a new notion for simulation called *coherent-runtime expected QPT* simulation. This notion allows the simulator to execute expected QPT procedures coherently, resulting in a superposition over computations with different runtimes, and subsequently "revert" the runtime by executing the same computation in reverse. Utilizing this simulation notion, [LMS22] achieved a set of interesting results that bypass the lower bounds established by [CCLY21].

[6] It means that they also achieve quantum ZK for **QMA** or quantum 2PC for *quantum* functionalities.
[7] This means that not only the simulation but also the construction is black-box (see [RTV04]).

We note that the current work focuses exclusively on the notion of expected QPT as adapted by [CCLY21]. For a detailed discussion on the provenance of (expected) quantum polynomial-time Turing machines, please refer to Section 1.2 of [LMS21], particularly Footnote 2 therein.

1.3 Acknowledgments

We thank Takashi Yamakawa for insightful discussions concerning certain aspects of [CCLY21].

Nai-Hui Chia was supported by NSF Award FET-2243659, NSF Career Award FET-2339116, Google Scholar Award, and DOE Quantum Testbed Finder Award DE-SC0024301.

Kai-Min Chung is supported in part by the 2021 Academia Sinica Investigator Award (AS-IA-110-M02), and NSTC QC project, under Grant no. NSTC 112-2119-M-001-006 -.

Jiahui Liu is supported by DARPA under Agreement No. HR00112020023, NSF CNS-2154149, and a Simons Investigator award.

2 Technical Overview

At a high level, our approach to Thm. 1 aligns with the paradigm established by [BL02, CCLY21] in the post-quantum setting. However, the *fully* quantum nature of the protocol introduces fundamentally different challenges.

In the following subsections, we begin by reviewing the approach proposed by [CCLY21]. Subsequently, we elucidate the obstacles encountered when attempting to generalize their technique to the fully quantum setting. Finally, we present our novel strategies to surmount these obstacles.

2.1 On the [CCLY21] Approach

[CCLY21] establishes that only languages in **BQP** admit constant-round post-quantum BBZK (PQ-BBZK) with expected QPT simulation. At a high level, their approach operates in two main steps:

- **Step 1:** Initially, for any language $\mathcal{L}$, the method begins by assuming the existence of a constant-round PQ-BBZK protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ with a *strict* QPT simulator $\mathcal{S}$. Next, it constructs a specialized malicious verifier $\widetilde{\mathcal{V}}$ based on the honest $\mathcal{V}$. Then, it builds a QPT decider for $\mathcal{L}$ using this $\widetilde{\mathcal{V}}$ and the simulator $\mathcal{S}$, thereby placing $\mathcal{L}$ in **BQP**.
- **Step 2:** This step extends the above result concerning *strict* QPT simulation to *expected* QPT simulation. To achieve this, the authors create another malicious verifier $\widetilde{\mathcal{V}}'$, which effectively runs the honest $\mathcal{V}$ and the aforementioned $\widetilde{\mathcal{V}}$ in superposition. They then prove that if an expected QPT simulator $\mathcal{S}$ exists, simulating the view of $\widetilde{\mathcal{V}}'$, it can be truncated into a strict QPT simulator $\mathcal{S}'$ while still providing sufficient simulation guarantees. Specifically, $\mathcal{S}'$ and $\widetilde{\mathcal{V}}'$ can be leveraged to construct a **BQP** decider using a similar argument as in **Step 1**, thereby ruling out expected QPT simulation as well. This step crucially relies on the quantum nature of $\widetilde{\mathcal{V}}'$ and $\mathcal{S}'$, a characteristic absent in the *classical setting*, where prior work such as [BL02] fails to extend to expected polynomial-time simulation.

Throughout this technical overview, our sole focus will be on **Step 1**. This is because our approach in the fully quantum setting mirrors the two-step structure outlined above, with our **Step 2** closely resembling that of [CCLY21]. Consequently, in the subsequent discussion, we will exclusively recapitulate the techniques pertaining to **Step 1**, as it is the pertinent component for understanding our new ideas later.

The main idea behind **Step 1**, originating from [BL02], involves constructing $\widetilde{\mathcal{V}}$ as a random-aborting variant of the honest $\mathcal{V}$. At each round $k \in [K]$, where K denotes the constant round complexity of the protocol, $\widetilde{\mathcal{V}}$ operates as follows: it initially applies a function H_ε (explained shortly) to the prover's messages $(p_1, \ldots, p_k)$ received up to that point (referred to as the *prefix*). If $H_\varepsilon(p_1, \ldots, p_k) = 0$, $\widetilde{\mathcal{V}}$ halts immediately and outputs a rejection symbol $\perp$; otherwise (i.e., $H_\varepsilon(p_1, \ldots, p_k) = 1$), $\widetilde{\mathcal{V}}$ behaves identically to the honest $\mathcal{V}$. Here, H_ε is a random function[8] that takes input of variable length $k \in [K]$ and maps each $(p_1, \ldots, p_k)$ to 1 (or 0) with a properly chosen probability ε (or $1-\varepsilon$). Subsequently, a **BQP** decider $\mathcal{B}$ can be constructed as follows: given input x, $\mathcal{B}(x)$ simply executes the simulator $\mathcal{S}^{\widetilde{\mathcal{V}}(x)}(x)$ with black-box access to $\widetilde{\mathcal{V}}(x)$ and outputs the result produced by $\mathcal{S}^{\widetilde{\mathcal{V}}(x)}(x)$. To demonstrate that $\mathcal{B}$ indeed functions as a valid **BQP** decider, it suffices to establish the following:

- **Completeness:** For any $x \in \mathcal{L}$, $\mathcal{B}(x)$ outputs acceptance with inverse-polynomial probability $1/\mathsf{poly}(|x|)$;
- **Soundness:** For any $x \notin \mathcal{L}$, $\mathcal{B}(x)$ outputs acceptance with negligible probability $\mathsf{negl}(|x|)$.

To establish completeness, we observe that during the real execution between $\widetilde{\mathcal{V}}(x)$ and the honest prover $\mathcal{P}(x, w)$ with witness w for x, $\widetilde{\mathcal{V}}(x)$ will output acceptance with a probability of at least ε^K. This probability arises because $\widetilde{\mathcal{V}}(x)$ behaves identically to the honest verifier, *conditioned on the function H_ε outputting 1 in each of the K rounds*. Subsequently, owing to the ZK property, $\mathcal{S}^{\widetilde{\mathcal{V}}(x)}(x)$ will output acceptance with a probability of at least $\varepsilon^K - \mathsf{negl}(|x|)$, which can be lower-bounded by an inverse polynomial. This is achievable due to the fact that K is a constant, coupled with a suitable choice of ε.

The proof of soundness is rather intricate. Let us first outline it in the classical context in [BL02], where the idea originated. Assuming for contradiction that $\mathcal{B}(x)$ outputs acceptance with some non-negligible probability $\delta(|x|)$ for some $x \notin \mathcal{L}$, [BL02] illustrates how to transform $\mathcal{B}$ into a malicious prover $\widetilde{\mathcal{P}}$ that induces the honest verifier $\mathcal{V}$ to accept x with probability $\delta(|x|)$, thereby compromising the soundness of the original ZK protocol. The approach involves $\widetilde{\mathcal{P}}(x)$ internally running the $\mathcal{S}^{\widetilde{\mathcal{V}}(x)}(x)$ as $\mathcal{B}(x)$, while externally interacting with the honest verifier $\mathcal{V}(x)$. For each $k \in [K]$, when $\mathcal{S}$ queries its oracle with input $(p_1, \ldots, p_k)$ to determine the next verifier's message v_k, $\widetilde{\mathcal{P}}(x)$ sends p_k to the external verifier $\mathcal{V}(x)$ to obtain v_k, which it then forwards to $\mathcal{S}$ as the response.

Initially, it is unclear whether such a $\widetilde{\mathcal{P}}(x)$ would be effective due to a significant discrepancy: the internal $\mathcal{S}$ might rewind its oracle $\widetilde{\mathcal{V}}$ to obtain the next message for two distinct history transcripts $(p_1, \ldots, p_k) \neq (p'_1, \ldots, p'_k)$, while the external verifier $\mathcal{V}(x)$ anticipates a "straight-line" interaction with $\widetilde{\mathcal{P}}(x)$. [BL02] resolves this issue using the following strategy: through a careful selection of ε, it is very unlikely that $\mathcal{S}$ could find $(p_1, \ldots, p_k) \neq (p'_1, \ldots, p'_k)$ for some $k \in [K]$ so that $H_\varepsilon(p_1, \ldots, p_k) = H_\varepsilon(p'_1, \ldots, p'_k) = 1$. Consequently, if $\mathcal{S}$ has already acquired a message v_k pertaining to a $(p_1, \ldots, p_k)$ that satisfies H_ε, we can confidently presume that all subsequent queries by $\mathcal{S}$ in the form of $(p'_1, \ldots, p'_k)$ do not fulfill the condition imposed by H_ε. According to the definition of $\widetilde{\mathcal{V}}$, responses to such queries do not necessitate $\widetilde{\mathcal{P}}$ to engage with the external $\mathcal{V}$, as the response is simply the symbol $\perp$. Hence, all messages relayed by $\widetilde{\mathcal{P}}$ between the internal $\mathcal{S}$ and the external $\mathcal{V}$ indeed align with a "straight-line" execution.

[8] Although the use of such a random function may render $\widetilde{V}$ inefficient, this can be rectified by replacing H_ε with a q-wise independent hash in the final **BQP** decider. Hence, for the sake of clarity, we will continue to employ H_ε throughout this overview without loss of generality.

The Measure-and-Reprogram (MnR) Technique. The post-quantum setting considered in [CCLY21] introduces a fundamental departure from the classical framework described in [BL02]. A quantum simulator $\mathcal{S}$ has the capability to execute quantum queries to its oracle $\widetilde{\mathcal{V}}$, which in turn prompts quantum queries to the random function H_ε. Notably, a single quantum query to H_ε enables $\mathcal{S}$ to acquire the responses v_k for multiple prefixes $(p_1, \ldots, p_k)$ *in superposition*. Thus, it becomes unclear if the aforementioned idea from [BL02] could work in this setting.

To address this challenge, [CCLY21] utilized the *measure-and-reprogram* (MnR) technique, originally developed to establish the post-quantum security of the Fiat-Shamir transform [DFMS19, DFM20]. First, note that the operation of $\mathcal{S}^{\widetilde{\mathcal{V}}(x)}(x)$ can be conceptualized as that of an oracle machine $\mathsf{SIM}^{H_\varepsilon}(x)$, with H_ε acting as the quantum oracle. Let H_0 denote a null oracle that outputs 0 for all inputs. [CCLY21] propose replacing $\mathsf{SIM}^{H_\varepsilon}(x)$ in the description of $\mathcal{B}(x)$ with a "MnR version" of a new game $\mathsf{SIM}^{H_0}(x)$, which is almost identical to $\mathsf{SIM}^{H_\varepsilon}(x)$, except for the following distinctions:

1. It initializes the oracle as H_0 (instead of H_ε).
2. Assuming SIM makes q quantum queries to its oracle in total, the game begins by randomly selecting K queries out of all these q queries. These selected queries are intended to be measured (see the next step).
3. Each time SIM makes a query intended for measurement, the game measures this query to determine a classical prefix $(p_1, \ldots, p_k)$. The oracle is then "reprogrammed" to output 1 for input $(p_1, \ldots, p_k)$. Subsequent queries are answered using this updated oracle (until it gets reprogrammed again).

By utilizing this revised definition of $\mathcal{B}(x)$, [CCLY21] successfully adapts the arguments from [BL02] to their post-quantum setting as follows:

For completeness, it relies on a so-called *MnR lemma* (established in [DFMS19, DFM20, YZ21]), which asserts that the output of the "MnR version" of SIM^{H_0} does not differ much from the game SIM^{H_0}. Furthermore, SIM^{H_0} does not differ much from the original game $\mathsf{SIM}^{H_\varepsilon}$ due to the sparsity of the random function H_ε. As previously argued, $\mathsf{SIM}^{H_\varepsilon}$ outputs acceptances for $x \in \mathcal{L}$ with a "good" probability due to the ZK guarantee; then, so does the new $\mathcal{B}(x)$ that utilizes the "MnR version" of SIM^{H_0}.

For soundness, first note that the MnR version of SIM^{H_0} initiates with the null oracle H_0. As the execution progresses, it undergoes reprogramming to output 1 solely for the *measured, classical* (instead of super-position) queries made by SIM. This essentially facilitates the recovery of the [BL02] arguments to establish that the messages relayed by $\widetilde{\mathcal{P}}$ between the internal MnR version of SIM^{H_0} and the external verifier $\mathcal{V}$ indeed constitute a "straight-line" execution. This eventually completes the reduction from the soundness of $\mathcal{B}(x)$ to that of the original ZK protocol. (Further discussion on this matter will be provided shortly.)

Benefits of the Classical Nature of the Protocol. In the forthcoming discussion, we emphasize the aspects where the above [CCLY21] techniques rely on the classical nature of the ZK protocol. It is worth noting that while some of these aspects were only implicit in [CCLY21], we need to articulate them more explicitly as they represent the obstacles in our setting where the honest parties are *fully quantum*.

Benefit 1. The most apparent benefit lies in the classical nature of the prover's messages. This is pivotal for adapting the random-aborting technique from [BL02] to the post-quantum setting,

where the H_ε is evaluated on the classical prover messages at each round, determining whether $\widetilde{\mathcal{V}}$ needs to prematurely abort.

Benefit 2. Note that $\widetilde{\mathcal{V}}$ is constructed from the classical honest $\mathcal{V}$, thus also maintaining a classical nature. Once we fix the random tap r of $\widetilde{\mathcal{V}}$, the machine $\widetilde{\mathcal{V}}_r$ becomes deterministic. Specifically, since $\widetilde{\mathcal{V}}_r$ is deterministic, when fed with the same prover's messages $(p_1, \ldots, p_k)$, it will consistently output the same next message v_k. This point is crucial for the final construction of the malicious prover $\widetilde{\mathcal{P}}$.

Recall that $\widetilde{\mathcal{P}}$ internally runs the simulator, which may attempt to rewind the execution. At a certain round k, $\mathcal{S}$ may ask to view the verifier's message multiple times via rewinding. While the MnR technique with the H_0 oracle could limit $\mathcal{S}$ to learning the value v_k for only one prefix $(p_1, \ldots, p_k)$, it cannot prevent $\mathcal{S}$ from requesting to view the response to the same $(p_1, \ldots, p_k)$ again. Now, since $\widetilde{\mathcal{V}}_r$ is deterministic, the response must be v_k again. This implies that $\widetilde{\mathcal{P}}$ only needs to learn the value v_k from the external verifier *only once* to accommodate the internal $\mathcal{S}$.

To further illustrate this advantage, let us momentarily assume that the verifier is not deterministic, meaning that when queried on $(p_1, \ldots, p_k)$, it may provide different v_k values. In such a scenario, to perfectly simulate the environment for the internal $\mathcal{S}$, $\widetilde{\mathcal{P}}$ would need to forward $(p_1, \ldots, p_k)$ externally to learn the (potentially different) response v_k when requested by $\mathcal{S}$. However, note that in this soundness reduction, $\widetilde{\mathcal{P}}$ interacts with the external verifier in a *straight-line* manner. Consequently, $\widetilde{\mathcal{P}}$ is not permitted to query the external verifier twice on the same prefix.

Benefit 3. The last benefit also pertains to the determinism of $\widetilde{\mathcal{V}}_r$, albeit in a more subtle manner. It relates to an inherent feature of the MnR technique that was not explicitly mentioned in the previous discussion. To grasp this issue, let us revisit the steps outlined in Items 2 and 3, where K queries are selected to be measured. Although it can be argued[9] that these measured queries would ultimately form a complete transcript, *there is no guarantee that the prover's messages will appear in the desired order*.

To delve into this further, consider the scenario where the k-th ($k \in [K]$) selected query to be measured from the oracle corresponds exactly to some $(p_1, \ldots, p_k)$, enabling $\widetilde{\mathcal{P}}$ to straightforwardly forward p_k to learn the response v_k. However, this ideal situation does not always occur. For instance, imagine the case where the constructed $\widetilde{\mathcal{P}}$ has yet to send any message to the external $\mathcal{V}$. Ideally, the subsequent move would involve $\widetilde{\mathcal{P}}$ forwarding the first prover message p_1, obtained by measuring the first selected query from the internal SIM^{H_0}, to learn $\mathcal{V}$'s response v_1. However, it is indeed possible that the first measurement results in a query format of (p_1, p_2) instead. Intuitively, this represents a discrepancy where the internal execution of SIM^{H_0} advances to round 2 while the external execution between $\widetilde{\mathcal{P}}$ and $\mathcal{V}$ has not even commenced.

Indeed, this is a known issue in the literature, including the original MnR paper (see [DFM20, Section 4.1]). Typically, it poses no significant problem for post-quantum applications. For instance, [CCLY21] tackled it by leveraging the determinism of the verifier (once the random tap is fixed) as follows: in the scenario described earlier, $\widetilde{\mathcal{P}}$ can pause the internal execution and "synchronize" the external execution as follows: first, $\widetilde{\mathcal{P}}$ sends p_1 (in the measurement outcome (p_1, p_2)) to the external $\mathcal{V}$ to learn and store the response v_1. Then, it sends the p_2 to the external $\mathcal{V}$ to learn v_2. Now that $\widetilde{\mathcal{P}}$ has obtained the message v_2 required for subsequent execution, it can resume the internal execution. Additionally, if a future selected query from SIM^{H_0} measures to p_1, $\widetilde{\mathcal{P}}$ can respond using the v_1 obtained earlier. This strategy works due to the deterministic nature of the

[9] The argument is not immediately evident and requires a close examination of the MnR technique. However, we choose to omit the related intricacies as they are less pertinent to the current discussion.

verifier. Specifically, when the random tap is fixed, the response to p_1 is always v_1, irrespective of whether $\mathcal{V}_r$ has previously been invoked on (p_1, p_2) or not.

We next shift our focus to our proposals in the fully quantum setting, specifically addressing how to handle the aforementioned dependency on the classical nature of the honest parties (or the protocol).

2.2 Non-Programmable EPR Model with Classical Prover Messages

To address the reliance on the prover's classical messages, as discussed in Benefit 1, we introduce an intermediate model for zero-knowledge protocols that partially "de-quantizes" a fully quantum protocol. This model, which we will refer to as the *Non-Programmable EPR* (NPE for short) model, operates as follows. In the NPE model, a trusted third party prepares a polynomial number of EPR pairs. At the start of the protocol, the prover and verifier each possess half of these EPR pairs. Throughout the protocol, they are permitted to utilize these EPR pairs. When defining the zero-knowledge property, the simulator holds the prover's shares of the EPR pairs. It is crucial to emphasize that in the definition of zero-knowledge, the simulator and verifier's EPR shares are still prepared and distributed by the trusted party. This stands in contrast to the widely-used Common Reference String (CRS) model, where the simulator is responsible for generating the CRS when defining zero-knowledge.

We next argue that if there exists a K-round fully-quantum BBZK protocol (in the standard model), then it can be converted into a K-round BBZK protocol in the NPE model, *where all messages sent by the prover are classical*. Roughly, this is because the prover can always use quantum teleportation to transmit the originally quantum prover message. In more detail, whenever the prover needs to send a quantum message p_k, it instead does the following: The prover performs a teleportation measurement on the (originally quantum) message p_k and the EPR registers that are meant to be used for this round, to obtain a classical measurement outcome $\widetilde{p}_k$ (i.e., the teleportation keys) and sends it to the verifier. Using $\widetilde{p}_k$ and the corresponding EPR shares on the verifier's side, the verifier can recover the original message p_k. Then, it behaves as in the original protocol — generating v_k, sending v_k to the prover (note that we do not ask the verifier to perform quantum teleportation), and moving to the next round.

The aforementioned transformation clearly preserves the round complexity of the original quantum protocol. Furthermore, it can be shown that completeness, soundness, and zero-knowledge properties are all maintained. This follows from rather standard techniques, so we will not delve into further detail beyond mentioning the intuition behind the preservation: the EPR pairs are solely used as a means for teleportation, effectively "de-quantizing" the prover's messages; they do not alter the inherent properties, such as soundness or the ZK property, of the original protocol. For more details, please refer to Sec. 5.2 and Sec. 5.3.

With the above claim, it now suffices to establish Thm. 1 in the NPE model and assume that all the prover messages are classical.

Finally, it is worth mentioning that the above de-quantization approach *only* provides us with a well-defined random-aborting behavior for the verifier. However, it remains uncertain whether this model truly offers substantial benefits, as both the verifier and the simulator still perform inherently quantum operations. Moreover, the NPE model comes at a price: it introduces pre-shared entanglement between the prover and the verifier, which may potentially complicate establishing impossibility results. It is unclear whether what we pay justifies what we gain! Looking ahead, we find that the techniques we develop in the sequel actually work even in the presence of pre-shared entanglement.

2.3 Inducing an Order on the Measured Messages

In this section, we introduce new ideas aimed at inducing an order on the measured messages in the MnR game, in order to address the issue mentioned in Benefit 3. Looking ahead, these ideas also prove to be helpful when we address (in Sec. 2.4) the issues mentioned in Benefit 2, as these two benefits are closely related.

We propose that the malicious $\widetilde{\mathcal{V}}$ maintain two registers additionally: a *global counter* register `gc` and a *local counter* register `lc`. Both registers are initialized to 0. Intuitively, the global counter will record the number of times the $\widetilde{\mathcal{V}}$ has been invoked, and the local counter will record at which round the current execution is located.

In particular, consider the k-th round when $\widetilde{\mathcal{V}}$ receives a message p_k. Assume that the current local counter has a value of $j-1$ (for some j) and the current global counter has a value of $k-1$.[10] $\widetilde{\mathcal{V}}$ first increases the global counter from $k-1$ to k, and then behaves by comparing these two counters (before the increase of $k-1$ happens):

1. If $(j-1) \neq (k-1)$, then $\widetilde{\mathcal{V}}$ does not do anything.
2. If $(j-1) = (k-1)$, then $\widetilde{\mathcal{V}}$ behaves as the honest verifier with random-aborting. In particular, she first queries the oracle H to learn $H(p_1, \ldots, p_j)$.[11] Note that the input to H is determined by the local counter value $j-1$, and in this case of $(j-1) = (k-1)$, the value $H(p_1, \ldots, p_j)$ is exactly the value $H(p_1, \ldots, p_k)$.
   The subsequent movements of $\widetilde{\mathcal{V}}$ are controlled by this value:
   (a) If $H(p_1, \ldots, p_j) = 0$, then $\widetilde{\mathcal{V}}$ does not do anything;
   (b) If $H(p_1, \ldots, p_j) = 1$, then $\widetilde{\mathcal{V}}$ generates the response v_k to p_k in the same manner as the honest verifier. After that, $\widetilde{\mathcal{V}}$ increases the local counter from $j-1$ to j.

The most important sentences to notice in the above description are the ones underscored: at each round, the global counter will always be increased; however, *the local counter will be increased only if that round is executed successfully, namely when* $H(p_1, \ldots, p_j) = 1$.

Let us explain the benefits of the above design. First, consider the "straight-line" execution between $\widetilde{\mathcal{V}}$ and the honest prover $\mathcal{P}$. In this real execution, we claim that $\widetilde{\mathcal{V}}$ simply behaves as the random-aborting verifier as the one from [BL02] or [CCLY21]. That is, the use of these extra counters at least does not interfere with the "random-aborting" behavior in this straight-line execution. This is particularly important to ensure that we can establish the completeness of the **BQP** decider, which works in a similar manner as in [BL02, CCLY21]. This can be easily seen by tracking the execution: At the beginning, both counters are set to 0. When the first message p_1 arrives, the global counter first gets increased to $|1\rangle_{\mathtt{gc}}$. Next,

- If $H(p_1) = 0$, then nothing will happen;
- If $H(p_1) = 1$, then the message v_1 is generated, and the local counter is increased to $|1\rangle_{\mathtt{lc}}$.

In summary, if $H(p_1) = 0$ happens, then the counters become $|1\rangle_{\mathtt{gc}}|0\rangle_{\mathtt{lc}}$. According to our definition in Item 1, the execution is essentially "dead" in the sense that nothing will happen in subsequent steps, because the global and local counters are not consistent. This is exactly the same as in [BL02, CCLY21]. If $H(p_1) = 1$ happens, then the counters become $|1\rangle_{\mathtt{gc}}|1\rangle_{\mathtt{lc}}$ and the execution proceeds to the next round properly. This is also the same as in [BL02, CCLY21]. The similar feature holds for every subsequent round, and thus one can see that our counters do not alter the behavior of $\widetilde{\mathcal{V}}$ (as in [BL02, CCLY21]) in the real, straight-line execution with the honest $\mathcal{P}$.

[10] As will become clear shortly, at the beginning of the k-th round, the global counter must have a value of $k-1$.
[11] We remark that this is not a typo. We mean to invoke H on input $(p_1, \ldots, p_j)$ when the local counter is $j-1$.

Next, we turn to the real virtue of the above design — it ensures that the measured messages in the MnR game appear in an increasing order. Let us first show that the first selected (to be measured) query could measure to p_1. Recall that the MnR game starts with the null oracle H_0. Before the first measurement happens, H_0 has not been reprogrammed and thus it outputs 0 on all inputs. In this case, we know that the local counter j must be 0 (recall from Item 2b that the local counter gets increased only in the $H(p_1, \ldots, p_j) = 1$ branch). Thus, the first selected query sent to H_0 must be a superposition of messages of length $j + 1 = 0 + 1 = 1$. Therefore, the measurement outcome must be some message p_1.

Using a similar argument as above, we can indeed show that for the k-th selected (to be measured) query to the oracle, the length of the measured outcome could never exceed k ($\forall k \in [K]$). Finally, since there are only K selected measurement opportunities in the MnR game, if the execution of $\mathsf{SIM}^{H_0}(x)$ finally brings the local counter to value K (which is a necessary condition for the verifier's acceptance), it must be the case that *the k-th selected query to the oracle measures to exactly a length-k prover's message* (see Sec. 6.4.1 for a formal treatment). In other words, the measurements will yield prover messages in the desired order.

Note that the above ideas only assist us in achieving the correct order of the measured messages. However, it remains unclear how to construct the malicious prover $\widetilde{\mathcal{P}}$ to prove the soundness of $\mathcal{B}$ because it is uncertain how to manage rewinding queries by $\mathcal{S}$. This leads us to Sec. 2.4.

2.4 Simulation for Rewinding Queries

We now shift our focus to addressing the challenges mentioned in Benefit 2, particularly regarding how to manage the rewindings required by $\mathcal{S}$ while operating internally within $\widetilde{\mathcal{P}}$.

Technically, the issue can be accurately described as follows: at a certain round k, when $\widetilde{\mathcal{P}}$ receives the (potentially quantum) message v_k from the external $\mathcal{V}$, we need to devise a mechanism that enables $\widetilde{\mathcal{P}}$ to "re-use" this v_k for future rewinding queries made by $\mathcal{S}$ when necessary.

The Intuition. Before delving into the technical intricacies, we aim to provide an intuitive understanding of our approach. Broadly speaking, our goal is to establish an intuition similar to that of [BL02, CCLY21], which helps to extend the "simulator learns nothing new by rewinding" concept to our fully quantum setting. This poses a significant challenge due to the fully quantum interaction between $\mathcal{S}$ and $\widetilde{\mathcal{V}}$, intertwined with intermediate measurements conducted by $\mathcal{S}$ and the MnR game.

To tackle this challenge, we will develop a series of analytical tools to characterize key aspects of the interaction between $\mathcal{S}$ and $\widetilde{\mathcal{V}}$ within the MnR game. These features will serve as crucial threads, enabling us to navigate and maintain a detailed description of the overall state across all registers held by $\mathcal{S}$ and $\widetilde{\mathcal{V}}$ throughout the execution, at an appropriate resolution. As we will demonstrate shortly, this detailed description will reveal a crucial observation: after each query of $\mathcal{S}$, the overall state can always be expressed as a superposition of a "good" branch and a "bad" branch:

- The good branch will mirror the state of the honest verifier in a real execution, at the appropriate round.
- The bad branch comprises some "error" terms that we would ideally like to eliminate, but are unable to do so. Fortunately, we can demonstrate that these error terms possess a structured nature that enables us to assert the following recursive features:
  - Round Slackness: The bad branch consistently lags behind the good branch. Specifically, if the good branch corresponds to a real execution reaching round k, then the bad branch will only contain (in superposition) executions that reached up to round $k - 1$.

- Return Simulatability: In subsequent steps, $\mathcal{S}$ may rewind the execution by invoking $\widetilde{\mathcal{V}}$'s unitary $\widetilde{V}^\dagger$. For these queries, the structure of the bad branch permits us to utilize a "dummy version" $\ddot{V}$ to substitute the unitary $\widetilde{V}$. This $\ddot{V}$ solely adjusts the global and local counter registers without affecting the overall state. Importantly, $\ddot{V}$ guarantees the following: regardless of how many rounds $\mathcal{S}$ rewinds the execution (answered using $\ddot{V}^\dagger$), once the execution (specifically, the good branch) returns to round k (with sufficient $\ddot{V}$ queries by $\mathcal{S}$), the state will be essentially identical to the state when it last reached round k. Additionally, $\mathcal{S}$ cannot discern that $\widetilde{V}$ has been replaced with $\ddot{V}$.
- Error Invariance: The error terms allow us to make the following recursive claim. After the next move by the simulator $\mathcal{S}$, the state can again be partitioned into a good branch and a bad branch. Furthermore, the good branch exhibits the same features as described above, while the bad branch maintains the same Round Slackness and Error Invariance.

Next, we demonstrate how these features lead to the desired construction of $\widetilde{\mathcal{P}}$ for our soundness reduction. The $\widetilde{\mathcal{P}}$ can be outlined as follows: when $\mathcal{S}$ initially requests to observe v_k (and the MnR game measures $H(p_1, \ldots, p_k) = 1$, granting $\mathcal{S}$ permission to observe v_k), $\widetilde{\mathcal{P}}$ will transmit the message p_k to the external $\mathcal{V}$, who will subsequently provide v_k in response. $\widetilde{\mathcal{P}}$ will then relay this v_k to the internal $\mathcal{S}$ (corresponding to the good branch at round k). Subsequently, all queries from $\mathcal{S}$ will be addressed using $\ddot{V}^\dagger$ (or $\ddot{V}$) *until the internal execution returns to round k once more.* This same procedure is repeated for each subsequent round. In essence, when $\mathcal{S}$ seeks to observe the message v_{k+1}, it is provided by the external $\mathcal{V}$ for the first time, and all subsequent queries are answered using $\ddot{V}$ or $\ddot{V}^\dagger$ until $\mathcal{S}$ requests the next message v_{k+2}, and so forth.

Let us analyze the constructed $\widetilde{\mathcal{P}}$. Firstly, note that for each $k \in [K]$, $\widetilde{\mathcal{P}}$ communicates with the external $\mathcal{V}$ precisely once, when $\mathcal{S}$ requests to observe v_k for the first time. All subsequent queries are handled internally by $\widetilde{\mathcal{P}}$ using the dummy unitaries $\ddot{V}$ or $\ddot{V}^\dagger$. Thus, the good branch, *at the conclusion of the execution*, mirrors precisely the final state of the external verifier. Therefore, the remaining task is to demonstrate that the good branch at the conclusion of the execution encompasses sufficient "weight" on acceptance.

Due to the Return Simulatability, the $\mathcal{S}$ inside $\widetilde{\mathcal{P}}$ perceives itself to be operating within the original real MnR game. Hence, it suffices to demonstrate that, at the culmination of the original MnR game, the good branch possesses sufficient weight to warrant acceptance. To this end, the Error Invariance property allows us to uphold the clean good-bad state structure throughout the entire execution. Specifically, this structured state persists at the instant $\mathcal{S}$ concludes its simulation. Now, we assert that the bad branch does not contribute to acceptance whatsoever. This is because: (1) the bad branch, by virtue of the Round Slackness, cannot correspond to the round K, and (2) the verifier will not accept an execution that has not reached the final round K. On the contrary, recalling our initial assumption (for contradicting soundness), the overall state must contain sufficient weight on acceptance. Therefore, only one possibility remains — all the weight for acceptance is "concentrated" within the good branch at the conclusion of the simulation. This concludes our argument establishing the validity of $\widetilde{\mathcal{P}}$.

We now transition to a technical discourse, delving into the mathematical instantiation of the above intuition and ideas.

A Simplified Example. To elucidate our main idea, let us start with an simplified scenario. First, it is crucial to outline certain behaviors of $\mathcal{S}$ and $\widetilde{\mathcal{V}}$ that bear relevance to the ensuing discussion.

Our construction of $\widetilde{\mathcal{V}}$ maintains K registers $\mathtt{p}_1 \ldots \mathtt{p}_K$ intended to store previous messages received from the prover. Additionally, there exists a *message transmission* register $\mathtt{m}$ facilitating the exchange of messages between the prover and verifier. Specifically, to transmit message p_k, $\mathcal{P}$ loads

it into $\mathtt{m}$ and forwards $|p_k\rangle_{\mathtt{m}}$ to the verifier. Upon receiving $|p_k\rangle_{\mathtt{m}}$, the initial step of $\widetilde{V}$ — the unitary operator of our malicious $\widetilde{\mathcal{V}}$ — involves applying a swap operation between register $\mathtt{p_k}$ and $\mathtt{m}$. This signifies $\widetilde{\mathcal{V}}$ relocating message p_k to its appropriate position (i.e., register $\mathtt{p_k}$) within her internal space. Subsequently, $\widetilde{\mathcal{V}}$ executes requisite computations to generate the response message v_k.

We will also utilize a unitary V_{p_k}, representing the operation of the *honest* verifier that derives v_k from message p_k. Notably, in the present model (as per Sec. 2.2), all messages from the honest prover are classical. Hence, we can presume that at round k, the operator of the honest verifier can be represented as $\sum_{p_k} |p_k\rangle\langle p_k|_{\mathtt{p_k}} \otimes V_{p_k}$. That is, when the prover's message is p_k, the unitary V_{p_k} is applied by the honest verifier.

The simulator $\mathcal{S}$ is specified by a "local" operator S, which operates non-trivially on registers $\mathtt{m}$ and $\mathcal{S}$'s working space $\mathtt{s}$. Specifically, $\mathcal{S}$ operates in two steps: (1) it invokes S to prepare a state over registers $\mathtt{m}$ and $\mathtt{s}$, and (2) it makes an oracle call to $\widetilde{V}$ (or $\widetilde{V}^\dagger$ for rewinding). Crucially, $\mathcal{S}$ is not able to observe (let alone modify) the internal registers of $\widetilde{\mathcal{V}}$, as we are focusing on *black-box* simulation.

Additionally, it is important to note that we are operating within the NPE model as described in Sec. 2.2. While this restricts us to dealing only with classical prover messages, both the prover and verifier now possess EPR pairs. Thankfully, we can regard these shares as stored in $\mathcal{S}$'s internal register $\mathtt{s}$ and $\widetilde{\mathcal{V}}$'s internal registers $\mathtt{v}$ respectively; the following derivation proceeds without explicitly referencing these shares.

Now, we are ready to describe our simplified example. Let us assume that the execution is currently at round k, and the overall state of $\widetilde{\mathcal{V}}$ has the following format:

$$|k\rangle_{\mathtt{gc}}|k\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}|\rho\rangle_{\mathtt{v}}, \tag{2.1}$$

where $\mathtt{v}$ represents $\widetilde{\mathcal{V}}$'s other registers, and the $(p_1,\ldots,p_k)$ satisfies the current oracle $\widehat{H}(p_1,\ldots,p_k) = 1$. (Recall that we are in the MnR game where the initial oracle H_0 may get re-programmed during the execution. We use $\widehat{H}$ to denote the current oracle.)

Remark 1. Note that in Expression (2.1), we assume for simplicity that the current global counter and local counter are both equal to k. Also, note that in the full-fledged game, $\widetilde{\mathcal{V}}$'s registers would be in a much more complicated *mixed* state. Here, we assume this *pure* state format for simplicity. We also make similar simplification assumptions for the following description and derivation regarding the simulator's behavior. These assumptions are meant to help the reader understand our ideas more clearly, without being confused by complex notations of secondary importance. We will present a discussion regarding the full-fledged case later in Sec. 2.5.

Now, assume that $\mathcal{S}$ wants to rewind the execution upon receiving input p_k. In particular, let us assume that $\mathcal{S}$ prepares $|p_k\rangle_{\mathtt{m}}$ together with some state $|\phi_{p_k}\rangle_{\mathtt{s}}$ on her internal register $\mathtt{s}$, and then she rewinds the execution by invoking $\widetilde{V}^\dagger$. Next, she applies her local unitary S, and finally calls $\widetilde{V}$ to bring the execution back. Let us track the overall state during this procedure:

$$\begin{aligned}
&\widetilde{V}S\widetilde{V}^\dagger|k\rangle_{\mathtt{gc}}|k\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}|p_k\rangle_{\mathtt{m}}|\phi_{p_k}\rangle_{\mathtt{s}}|\rho\rangle_{\mathtt{v}}\\
&= \widetilde{V}S|k-1\rangle_{\mathtt{gc}}|k-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}|p_k\rangle_{\mathtt{m}}|\phi_{p_k}\rangle_{\mathtt{s}}V_{p_k}^\dagger|\rho\rangle_{\mathtt{v}} &(2.2)\\
&= \widetilde{V}|k-1\rangle_{\mathtt{gc}}|k-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}\big(S|p_k\rangle_{\mathtt{m}}|\phi_{p_k}\rangle_{\mathtt{s}}\big)V_{p_k}^\dagger|\rho\rangle_{\mathtt{v}} &(2.3)\\
&= \widetilde{V}|k-1\rangle_{\mathtt{gc}}|k-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}|p_k\rangle_{\mathtt{m}}|\phi'_{p_k}\rangle_{\mathtt{s}}V_{p_k}^\dagger|\rho\rangle_{\mathtt{v}} +\\
&\qquad\qquad \widetilde{V}|k-1\rangle_{\mathtt{gc}}|k-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}\sum_{p'_k\neq p_k}|p'_k\rangle_{\mathtt{m}}|\phi'_{p'_k}\rangle_{\mathtt{s}}V_{p_k}^\dagger|\rho\rangle_{\mathtt{v}} &(2.4)
\end{aligned}$$

$$= |k\rangle_{\mathtt{gc}}|k\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}|p_k\rangle_{\mathtt{m}}|\phi'_{p_k}\rangle_{\mathtt{s}}|\rho\rangle_{\mathtt{v}} +$$
$$\widetilde{V}|k-1\rangle_{\mathtt{gc}}|k-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_{k-1}}}\sum_{p'_k\neq p_k}|p'_k\rangle_{\mathtt{p_k}}|p_k\rangle_{\mathtt{m}}|\phi'_{p'_k}\rangle_{\mathtt{s}}V^\dagger_{p_k}|\rho\rangle_{\mathtt{v}} \tag{2.5}$$

$$= \underbrace{|k\rangle_{\mathtt{gc}}|k\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}|p_k\rangle_{\mathtt{m}}|\phi'_{p_k}\rangle_{\mathtt{s}}|\rho\rangle_{\mathtt{v}}}_{good} + \underbrace{|k\rangle_{\mathtt{gc}}|k-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_{k-1}}}\sum_{p'_k\neq p_k}|p'_k\rangle_{\mathtt{p_k}}|p_k\rangle_{\mathtt{m}}|\phi'_{p'_k}\rangle_{\mathtt{s}}V^\dagger_{p_k}|\rho\rangle_{\mathtt{v}}}_{bad}, \tag{2.6}$$

where

- Eq. (2.2) follows from the definition of $\widetilde{V}$ — when $(p_1,\ldots,p_k)$ satisfies the current oracle $\widehat{H}$, $\widetilde{V}^\dagger$ will simply decrease the global and local counters, and apply the honest verifier's unitary $V^\dagger_{p_k}$.
- Eq. (2.3) follows from the fact that $\mathcal{S}$'s local unitary S only touches upon registers $\mathtt{m}$ and $\mathtt{s}$.
- Eq. (2.4) follows from decomposing the state on $\mathtt{m}$ and $\mathtt{s}$ in the computational basis on the $\mathtt{m}$ register, i.e., $S|p_k\rangle_{\mathtt{m}}|\phi_{p_k}\rangle_{\mathtt{s}} = \sum_{p'_k}|p'_k\rangle_{\mathtt{m}}|\phi'_{p'_k}\rangle_{\mathtt{s}}$.
- Eq. (2.5) again follows from the definition of $\widetilde{V}$ — when $(p_1,\ldots,p_k)$ satisfies the current oracle $\widehat{H}$, $\widetilde{V}$ will simply decrease the global and local counters, and apply the honest verifier's unitary V_{p_k}, which cancels the $V^\dagger_{p_k}$.
- To see Eq. (2.6), we need to recall that $\widetilde{V}$ indeed first swaps the contents of $\mathtt{m}$ and $\mathtt{p_k}$ as we described earlier. After that, the contents in registers $\mathtt{p_1}\ldots\mathtt{p_k}$ do not satisfy $\widehat{H}$ because $p'_k \neq p_k$. Thus, the $V^\dagger_{p_k}$ and the local counter will be left as they are, and only the global counter gets increased.

Next, we define a "dummy" operator $\ddot{V}$. It works in the identical manner as $\widetilde{V}$, with the only difference that $\ddot{V}$ does not perform the work of V_{p_k}. Now, let us track the same execution but with the $\ddot{V}$ in place of $\widetilde{V}$.

$$\begin{aligned}
&\ddot{V}S\ddot{V}^\dagger|k\rangle_{\mathtt{gc}}|k\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}|p_k\rangle_{\mathtt{m}}|\phi_{p_k}\rangle_{\mathtt{s}}|\rho\rangle_{\mathtt{v}}\\
&= \ddot{V}S|k-1\rangle_{\mathtt{gc}}|k-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}|p_k\rangle_{\mathtt{m}}|\phi_{p_k}\rangle_{\mathtt{s}}|\rho\rangle_{\mathtt{v}}\\
&= \ddot{V}|k-1\rangle_{\mathtt{gc}}|k-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}\big(S|p_k\rangle_{\mathtt{m}}|\phi_{p_k}\rangle_{\mathtt{s}}\big)|\rho\rangle_{\mathtt{v}}\\
&= \ddot{V}|k-1\rangle_{\mathtt{gc}}|k-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}|p_k\rangle_{\mathtt{m}}|\phi'_{p_k}\rangle_{\mathtt{s}}|\rho\rangle_{\mathtt{v}} +\\
&\qquad \ddot{V}|k-1\rangle_{\mathtt{gc}}|k-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}\sum_{p'_k\neq p_k}|p'_k\rangle_{\mathtt{m}}|\phi'_{p'_k}\rangle_{\mathtt{s}}|\rho\rangle_{\mathtt{v}}\\
&= \underbrace{|k\rangle_{\mathtt{gc}}|k\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_k}}|p_k\rangle_{\mathtt{m}}|\phi'_{p_k}\rangle_{\mathtt{s}}|\rho\rangle_{\mathtt{v}}}_{good} + \underbrace{|k\rangle_{\mathtt{gc}}|k-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_k\rangle_{\mathtt{p_1\ldots p_{k-1}}}\sum_{p'_k\neq p_k}|p'_k\rangle_{\mathtt{p_k}}|p_k\rangle_{\mathtt{m}}|\phi'_{p'_k}\rangle_{\mathtt{s}}|\rho\rangle_{\mathtt{v}}}_{bad},
\end{aligned} \tag{2.7}$$

where Eq. (2.7) follows from the similar derivation as we did for Eq. (2.6).

Remark 2. As an astute reader may have already noticed, the derivations above rely on an over-simplification that we must now address. Recall that the unitary V_{p_k} generates message v_k. Right after that, v_k is stored in some register $\mathtt{v_k}$ within the internal space of $\widetilde{\mathcal{V}}$. To actually deliver this message to the prover, $\widetilde{\mathcal{V}}$ needs to apply a swap operator between $\mathtt{m}$ and $\mathtt{v_k}$ to load v_k into $\mathtt{m}$. Thus, $\widetilde{\mathcal{V}}$'s operator $\widetilde{V}$ indeed also needs to interact with register $\mathtt{m}$. However, this interaction is not

reflected in the above derivations. We note that our actual proof in the main body does account for this case. However, doing so requires demonstrating finer-grained properties of both $\mathcal{S}$ and $\widetilde{\mathcal{V}}$, which cannot be accommodated within the limited space of this technical overview. Thus, we have chosen to omit these details here. Nonetheless, we assert that this omission does not pose any real problems for the subsequent discussion.

Now, let us elucidate how the example above illustrates the features outlined in the The Intuition part. Firstly, observe that the branches labeled as *good* in Eq. (2.6) and (2.7) are identical, despite being obtained through different procedures $\widetilde{V}S\widetilde{V}^\dagger$ and $\ddot{V}S\ddot{V}^\dagger$, respectively. This essentially demonstrates what we referred to as Return Simulatability earlier. Although Eq. (2.6) and (2.7) still differ in the bad branches, it is clear that the bad branches in both equations have a local counter value of $k-1$, while the good branch has already progressed to round k (i.e., both the global and local counters are equal to k). This aligns with what we call Round Slackness.

Additionally, recall from our design of counters in Sec. 2.3 that when the global counter differs from the local counter, the unitary $\widetilde{V}$ does nothing. Therefore, the bad branches in Eq. (2.6) and (2.7) are essentially "locked" there, contributing nothing to the subsequent execution. Moreover, notice that in the good branches of both equations, the state $|\rho\rangle_{\mathtt{v}}$ over the verifier's internal register $\mathtt{v}$ turns out to be identical to that in the initial state shown in Expression (2.1). This, along with the uselessness of the bad branches we just explained, essentially means that to handle this "rewinding then returning back to round k" procedure, one does not need to perform the work V_{p_k} again (as per $\widetilde{V}$); the dummy unitary $\ddot{V}$ will suffice.

On the Recursiveness of Error Invariance. Still, there is an obvious discrepancy between the above simplified example and the Error Invariance described in the The Intuition part. Recall that Error Invariance suggests that the good-bad structure will be *recursively* maintained. In our example, we began with an initial state (i.e., Expression (2.1)) that only had a good branch (in that it only contains the $(p_1, \ldots, p_k)$ satisfying H), while to establish Error Invariance, we should have started with a state containing both good and bad branches.

However, the validity of our argument remains intact even if we start with a state containing both good and bad branches. Let us delve into the main intuition. The bad branch in Eq. (2.6) differs from that in Eq. (2.7) only in the presence of $V_{p_k}^\dagger$ in front of $|\rho\rangle_{\mathtt{v}}$. This difference is not coincidental. If we apply $\widetilde{V}S\widetilde{V}^\dagger$ (resp. $\ddot{V}S\ddot{V}^\dagger$) to the state in Eq. (2.6) (resp. Eq. (2.7)), the resulting states will still share an identical good branch, while the bad branches differ by a $V_{p_k}^\dagger$.

To illustrate, consider applying $\widetilde{V}S\widetilde{V}^\dagger$ again to Eq. (2.6). The good branch will evolve in exactly the same manner as the above Eq. (2.6), yielding a new good branch good_1 and a new bad branch bad_1 (with the same $V_{p_k}^\dagger$ hanging there).

The evolution of the original bad branch proceeds as follows: Since the global counter k differs from the local counter $k-1$, the first operator $\widetilde{V}^\dagger$ only reverts the global counter to $k-1$ and does nothing else. Subsequently, $\mathcal{S}$'s local operator S is applied, potentially resulting in a state such as $\sum_{p'_k}|p'_k\rangle_{\mathtt{m}}|\phi''_{p'_k}\rangle_{\mathtt{s}}$ as in Eq. (2.4). The subsequent application of $\widetilde{V}$ leads to a similar evolution as seen from Eq. (2.4) to Eq. (2.6). In particular, this means that the (original) bad branch will eventually leads to a new good branch good_2 and a new bad branch bad_2 (with the same $V_{p_k}^\dagger$ hanging there).

In summary, at the end of the execution, good_1 and good_2 merge into a final good branch, while bad_1 and bad2 merge into a final bad branch, with a $V_{p_k}^\dagger$ preceding the $\mathtt{v}$ register. This illustrates that our argument extends even when starting with a superposition of good and bad branches!

This concludes our analysis of the simplified example. In the following Sec. 2.5, we will briefly address the technical challenges that were omitted from the discussion above, but which will necessitate non-trivial effort and novel ideas in the full-fledged setting.

2.5 On Those under the Rug

Branch-Wise Analysis. The first concern pertains to the assumption of a *pure* format in Expression (2.1). In the real execution, the overall state would be *mixed* due to two types of measurements. Firstly, at each step, $\mathcal{S}$ needs to determine which of the two oracles $\widetilde{V}$ and $\widetilde{V}^\dagger$ to query. This can be modeled by a special register $\mathtt{u}$ within $\mathcal{S}$'s internal space $\mathtt{s}$. After $\mathcal{S}$ applies the local unitary S over $\mathtt{m}$ and $\mathtt{s}$, a superposition such as $\alpha_0|\downarrow\rangle + \alpha_1|\uparrow\rangle$ will be generated on register $\mathtt{u}$. $\mathcal{S}$ then measures this register to decide whether to execute a $\downarrow$-query (i.e., invoking $\widetilde{V}$) or an $\uparrow$-query (i.e., invoking $\widetilde{V}^\dagger$). Secondly, given that we are operating within the MnR game, certain queries to H will prompt measurements on registers $\mathtt{p}_1 \dots \mathtt{p}_j$.

We address this issue through the following approach. First, we claim that these measurements occur at predetermined positions. To see that, notice that $\mathcal{S}$'s measurements always happen right before each oracle query; these are fixed places we know in advance. Additionally, the MnR measurements are executed at locations sampled at the outset of the game (refer to Sec. 3.2). Hence, the entire game can be perceived as a series of unitary operators interspersed with measurements at predefined positions.

To analyze such a procedure, we can instead examine all possible outcomes of each intermediate measurements. For each fixed "outcome sequence" (consisting of the outcomes of all intermediate measurements), we can define a "sub-normalized" version of the game, where each intermediate measurement is replaced with a projection that collapses the register to the predetermined outcome specified in the outcome sequence. Any concerned property of the original game (in our case, it is the final decision of the verifier) is essentially the aggregation of that of all possible "sub-normalized" games. We formalize this property as a *branch-wise equivalence lemma* in Sec. 3.3. Leveraging this lemma, we can indeed adopt the pure-state perspective as demonstrated in the aforementioned simplified example.

Structure of the Local Counter. Next, we address a trickier issue. In the aforementioned simplified example, we assumed that the local counter comprises a classical value $|k\rangle_{\mathtt{lc}}$. However, in the actual execution, the local counter could carry a superposition of values. In such a scenario, it is not immediately evident whether the state resulting from $\widetilde{V}S\widetilde{V}^\dagger$ can be expressed in the clear good-bad format illustrated in Eq. (2.6). Specifically, it is uncertain whether the local counter contains a value smaller than the global counter in the bad branch (i.e., the Round Slackness property). On the other hand, the earlier discussion about Round Slackness w.r.t. that simplified example exhibits a recursive nature — we managed to establish it assuming that *either* the initial state lacks a bad branch, *or* the bad branch is already round-slack (as explained previously for the case where the initial state already contains a bad branch). This recursive argument now seems to place us in an "egg-check dilemma."

We address this issue as follows: Before initiating the derivation depicted in the aforementioned simplified example, we establish a lemma that, in the "sub-normalized" game described earlier, offers a full characterization of the structure of global and local counters (refer to Sec. 7.3 for details). The principal implication of this lemma (see Lem. 14) can be intuitively summarized as follows: Throughout the (sub-normalized) game, the overall state can be expressed as the sum of pure states in superposition. For the branch where the $\mathtt{p}_1 \dots \mathtt{p}_k$ registers contain precisely the values $p_1, \dots, p_k$ satisfying the current oracle H, both the global counter and the local counter equal k. Note that this precisely corresponds to the good branch in the previous discussion. For all other branches (i.e., the bad branch) in the superposition, the local counter is strictly smaller than the global counter; this is exactly what we require for Round Slackness! Essentially, this lemma establishes the Round

Slackness even before we start the derivation (as shown in the simplified example), resolving the egg-chicken dilemma.

Moreover, the clear characterization of the counter structures enables us to conduct the derivation without explicitly tracking the local counter. This is because all branches where the local counter is smaller than the global counter (i.e., where Round Slackness is satisfied) can be grouped under the name of a single bad branch. This presents a significant advantage for presentation: as previously mentioned (and as demonstrated in Lem. 14), the local counter is in a superposition of many values, resulting in the bad branch being a summation of multiple sub-branches. In such a scenario, explicit enumeration of all the sub-branches in the derivation would be necessary. Although the proof would still hold, it would certainly be overly complex to understand.

Commutativity Lemma and Hybrid Argument. There is one more issue worth mentioning here. The simplified example discussed above only addresses the scenario where $\mathcal{S}$ rewinds the execution by one round and subsequently resumes it. In the full-fledged setting, however, there are K rounds, and at any given point, $\mathcal{S}$ can opt to rewind the execution as far back as desired. Moreover, at each move, $\mathcal{S}$ may not necessarily provide the input $|p_k\rangle|\phi_{p_k}\rangle_{\mathtt{s}}$ as illustrated in the example. Instead, it could potentially query the oracle $\widetilde{V}^\dagger$ with a general state ρ over the $\mathtt{p_k}$ and $\mathtt{s}$ registers. Handling this full-fledged setting presents significant challenges in performing the derivation.

We resolve these issues as follows. Firstly, we encapsulate the aforementioned derivation as a general, information-theoretic lemma concerning the commutativity of certain unitary operators. This lemma, presented in Sec. 4, is meticulously crafted so that it can be later applied to handle the general case where $\mathcal{S}$ queries her oracles with a general state over registers $\mathtt{m}$ and $\mathtt{s}$. Essentially, it asserts exactly what we demonstrated in the simplified example: if $\mathcal{S}$ rewinds the execution by one step and then resumes it, the final state resulting from the real execution (utilizing $\widetilde{V}$) and the one from the dummy execution (utilizing $\ddot{V}$) will possess an identical good branch, differing at the bad branch with a *structured* error term.

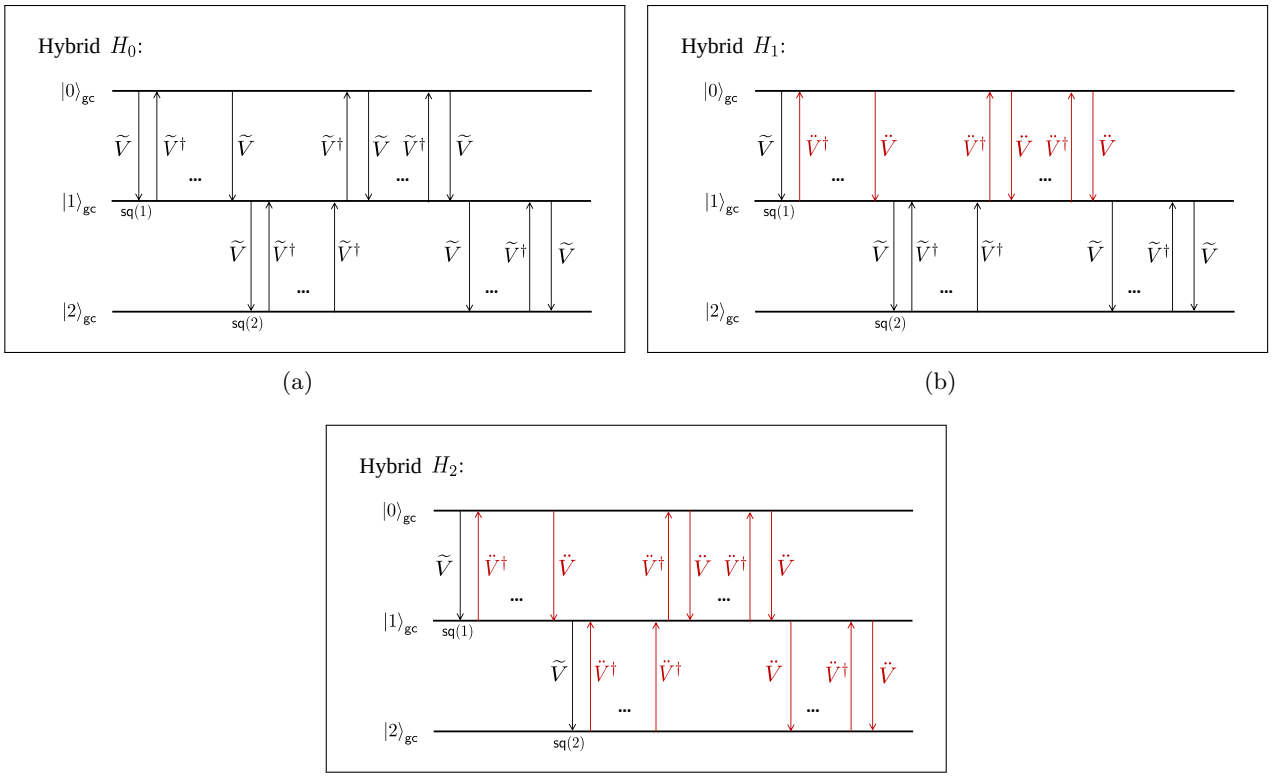
However, it is important to note that this commutativity lemma is established in the rewinding-one-step-back setting. Handling multiple-round rewinding necessitates new ideas. We address this challenge through a careful design of hybrids. Roughly speaking, we create $K+1$ hybrids, with hybrid H_0 representing the real MnR game (utilizing $\widetilde{V}$), and the k-th hybrid (for $k \in [K]$) structured as follows:

Hybrid H_k: This hybrid is identical to H_{k-1}, except for the following difference:

- The first query (made by $\mathcal{S}$) that brings the global counter from $|k-1\rangle_{\mathtt{gc}}$ to $|k\rangle_{\mathtt{gc}}$ is answered with $\widetilde{V}$ (as in the previous hybrid). However, all subsequent queries that bring the global counter from $|k-1\rangle_{\mathtt{gc}}$ to $|k\rangle_{\mathtt{gc}}$ (resp. from $|k\rangle_{\mathtt{gc}}$ to $|k-1\rangle_{\mathtt{gc}}$) are answered with the "dummy" unitary $\ddot{V}$ (resp. $\ddot{V}^\dagger$).

A helpful approach to understand these hybrids is to examine the baby case of $K = 2$, which we also include in the main body as a warm-up example (see Sec. 8). In this scenario, there are only three hybrids H_0, H_1, and H_2. We illustrate them in Fig. 1:

- In hybrid H_0, it can be seen from Fig. 1a that all the $\downarrow$-queries are answered using $\widetilde{V}$ and all the $\uparrow$-queries are answered using $\widetilde{V}^\dagger$.
- The hybrid H_1 shown in Fig. 1b is identical to hybrid H_0 except that the $\downarrow$-queries that bring the global counter from $|0\rangle_{\mathtt{gc}}$ to $|1\rangle_{\mathtt{gc}}$ are answered using the dummy-version unitary $\ddot{V}$, and the $\uparrow$-queries that bring the global counter from $|1\rangle_{\mathtt{gc}}$ to $|0\rangle_{\mathtt{gc}}$ are answered using the dummy-version



(a) (b) (c)

Fig. 1: Illustration of Hybrids H_0, H_1, and H_2

unitary $\ddot{V}^\dagger$. This is except for the query labelled as $\mathsf{sq}(1)$, which represents the first time the global counter successfully reaches value 1 (because it invokes the measurement on the query to H in the MnR game).

– The hybrid H_2 shown in Fig. 1c is identical to hybrid H_1 except that the $\downarrow$-queries that bring the global counter from $|1\rangle_{\mathsf{gc}}$ to $|2\rangle_{\mathsf{gc}}$ are answered using the dummy-version unitary $\ddot{V}$, and the $\uparrow$-queries that bring the global counter from $|2\rangle_{\mathsf{gc}}$ to $|1\rangle_{\mathsf{gc}}$ are answered using the dummy-version unitary $\ddot{V}^\dagger$. This is except for the query labelled as $\mathsf{sq}(2)$, which represents the first time the global counter successfully reaches value 2.

It is evident from Fig. 1c that hybrid H_2 precisely corresponds to the "dummy" MnR game necessary for constructing the malicious prover $\widetilde{\mathcal{P}}$ in our soundness reduction. Therefore, it is sufficient to demonstrate that the acceptance probability remains unchanged across these three hybrids. To establish this for H_0 and H_1, we can readily employ the previously mentioned commutativity lemma. The disparity between these two hybrids only arises in the "first layer" (i.e., the execution between the line of $|0\rangle_{\mathsf{gc}}$ and the line of $|1\rangle_{\mathsf{gc}}$), which corresponds precisely to the one-step-back rewinding scenario addressed by the lemma.

To transition from H_1 to H_2, we can leverage the same lemma. This is facilitated by the fact that in H_1, all the oracles for the first layer have already been substituted with the dummy operator $\ddot{V}$ or $\ddot{V}^\dagger$ (except for the $\mathsf{sq}(1)$ and $\mathsf{sq}(2)$ queries). As the dummy $\ddot{V}$ essentially has no impact except for some counter adjustments, the disparity between H_1 and H_2 can effectively be treated as a one-step-back rewinding in the second layer alone. Consequently, it can be handled by the commutativity lemma once more.

It is conceivable that this argument extends to the scenario with any constant K, where we simply invoke the commutativity lemma to argue the indistinguishability between each pair of H_{k-1} and H_k (for all $k \in [K]$) in a similar manner as the above.

This eventually completes our proof.

2.6 Putting Things Together

Let us assemble all the techniques discussed thus far, providing a complete picture of how we ultimately establish Thm. 1.

We first note that our focus will be on establishing a version of Thm. 1 where the simulator runs in *strict* QPT, rather than *expected* QPT. Once this is accomplished, the extension to expected QPT simulation follows the same technique from [CCLY21]. This extension will be covered in Sec. 10.

For a language $\mathcal{L}$, assuming the existence of a constant-round fully-quantum BBZK protocol $\langle \mathcal{P}, \mathcal{V} \rangle$, our objective is to construct a **BQP** decider $\mathcal{B}$ for the language $\mathcal{L}$. This construction involves leveraging $\mathcal{V}$ along with the ZK simulator $\mathcal{S}$.

Toward that, we first introduce a non-programmable EPR model, where we can assume without loss of generality that all prover messages are classical. Furthermore, we demonstrate that it suffices to establish Thm. 1 in this model. This is covered in Sec. 5.

Subsequently, we define a random-aborting verifier $\widetilde{\mathcal{V}}$, akin to the one introduced in [CCLY21]. However, we augment it with a global counter and a local counter, as discussed in Sec. 2.3. These two counters play a crucial role in the subsequent proofs. The precise definition of our $\widetilde{\mathcal{V}}$ is elucidated in Sec. 6.1.

We then proceed to define the desired **BQP** decider $\mathcal{B}$ in Sec. 6.2. Essentially, it executes the measure-and-reprogram (MnR) version of $\mathcal{S}^{\widetilde{\mathcal{V}}(x)}(x)$, with $\widetilde{\mathcal{V}}$'s random function instantiated by H_0 and serving as the oracle in the MnR game. Our task then becomes demonstrating that $\mathcal{B}$ is both complete and sound.

The proof for completeness is similar to that in [CCLY21], and it is presented in Sec. 6.3.

The proof for soundness is the most intricate aspect of our work. It entails addressing several challenges discussed earlier in Sec. 2.3 to 2.5. In the main body, this is covered in Sec. 6.4 and Sec. 7 to Sec. 9.

3 Preliminaries

Notation. Let λ denote the security parameter throughout the paper. For a positive integer $n \in \mathbb{N}$, $[n]$ denotes a set $\{1, 2 \ldots, n\}$. For a finite set $\mathcal{X}$, $x \xleftarrow{\$} \mathcal{X}$ means that x is uniformly chosen from $\mathcal{X}$. For a finite set $\mathcal{X}$ and a positive integer k, $\mathcal{X}^{\leq k}$ is defined to be $\bigcup_{i \in [k]} \mathcal{X}^i$. For finite sets $\mathcal{X}$ and $\mathcal{Y}$, $\mathcal{F}(\mathcal{X}, \mathcal{Y})$ denotes the set of all functions with domain $\mathcal{X}$ and range $\mathcal{Y}$.

A function $f : \mathbb{N} \to [0, 1]$ is said to be *negligible* if for all polynomial p and sufficiently large $\lambda \in \mathbb{N}$, we have $f(\lambda) < 1/p(\lambda)$; it is said to be *overwhelming* if $1 - f$ is negligible, and said to be *noticeable* if there is a polynomial p such that $f(\lambda) \geq 1/p(\lambda)$ for sufficiently large $\lambda \in \mathbb{N}$. We denote by poly an unspecified polynomial and by negl an unspecified negligible function.

We will use the following convention. If we have a state $|\rho\rangle_{\mathtt{ab}}$ over two registers $\mathtt{a}$ and $\mathtt{b}$ and a unitary operator $U_{\mathtt{a}}$ that acts on registers $\mathtt{a}$ only, we will use $U_{\mathtt{a}}|\rho\rangle_{\mathtt{ab}}$ as an abbreviation for applying $U_{\mathtt{a}} \otimes I_{\mathtt{b}}$ to $|\rho\rangle_{\mathtt{ab}}$ (where $I_{\mathtt{b}}$ is the identity operator on register $\mathtt{b}$). Also, if $U_{\mathtt{b}}$ operates non-trivially only on register $\mathtt{b}$, we will use $|\psi\rangle_a U_{\mathtt{b}} |\phi\rangle_b$ as an abbreviation for applying $I_{\mathtt{a}} \otimes U_{\mathtt{b}}$ to $|\psi\rangle_a |\phi\rangle_b$.

3.1 Technical Lemmas

The following lemma will be used throughout the paper.

Lemma 1 ([Zha12]). *For any sets $\mathcal{X}$ and $\mathcal{Y}$ of classical strings and q-quantum-query algorithm $\mathcal{A}$, we have*

$$\Pr\left[\mathcal{A}^H = 1 : H \leftarrow \mathcal{F}(\mathcal{X},\mathcal{Y})\right] = \Pr\left[\mathcal{A}^H = 1 : H \leftarrow \mathcal{H}_{2q}\right]$$

where $\mathcal{H}_{2q}$ is a family of $2q$-wise independent hash functions from $\mathcal{X}$ to $\mathcal{Y}$.

Differentiating Sparsity Functions as Quantum Oracles.

Lemma 2 ([HRS16, Lemma 3]). *Let $\mathcal{X}$ be a finite set, $\varepsilon \in [0,1]$ be a non-negative real number, and $\mathcal{H}_\varepsilon$ be a distribution over $H_\varepsilon : \mathcal{X} \to \{0,1\}$ such that we have $\Pr[H_\varepsilon(x) = 1] = \varepsilon$ independently for each $x \in \mathcal{X}$. Let $H_0 : \mathcal{X} \to \{0,1\}$ be the function that returns 0 for all inputs $x \in \mathcal{X}$. Then for any algorithm $\mathcal{A}$ that makes at most q quantum queries, we have*

$$\left|\Pr\left[\mathcal{A}^{H_\varepsilon} = 1 : H_\varepsilon \leftarrow \mathcal{H}_\varepsilon\right] - \Pr\left[\mathcal{A}^{H_0} = 1\right]\right| \leq 8q^2\varepsilon.$$

3.2 (Ordered-Query) Measure-and-Reprogram Lemma

We recall the measure-and-reprogram (MnR) lemma developed by [DFMS19, DFM20]. In particular, we need the special "ordered-query" version of the (MnR) lemma formalized in [CCLY21, Lemma 2.11]. We adapt it with some cosmetic changes to suit our applications.

We first give intuitive explanations for notation, which are taken from [CCLY21]. For a quantumly-accessible classical oracle $\mathcal{O}$, we denote by $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x, y)$ to mean that we reprogram $\mathcal{O}$ to output y on input x. For a q-quantum-query algorithm $\mathcal{A}$, function $H : \mathcal{X} \to \mathcal{Y}$, and $\mathbf{y} = (y_1, \ldots, y_k) \in \mathcal{Y}^k$, we denote by $\mathcal{A}[H, \mathbf{y}]$ to mean an algorithm that runs $\mathcal{A}$ w.r.t. an oracle that computes H except that randomly chosen k queries are measured and the oracle is reprogrammed to output y_i on i-th measured query. Formal definitions are given below.

Definition 1 (Reprogramming Oracle [CCLY21, Definition 2.9]). *Let $\mathcal{A}$ be a quantum algorithm with quantumly-accessible oracle $\mathcal{O}$ that is initialized to be an oracle that computes some classical function from $\mathcal{X}$ to $\mathcal{Y}$. At some point in an execution of $\mathcal{A}^{\mathcal{O}}$, we say that we reprogram $\mathcal{O}$ to output $y \in \mathcal{Y}$ on $x \in \mathcal{X}$ if we update the oracle to compute the function $H_{x,y}$ defined by*

$$H^{x,y}(x') := \begin{cases} y & \text{if } x' = x \\ H(x') & \text{otherwise} \end{cases}$$

where H is the function computed by $\mathcal{O}$ before the update. This updated oracle is used in the rest of execution of $\mathcal{A}$. We denote by $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x, y)$ the above reprogramming procedure.

Lemma 3 (Measure-and-Reprogram Lemma [CCLY21, Lemma 2.11]). *Let $\mathcal{X}$ and $\mathcal{Y}$ be sets of classical strings and k be a positive integer. Let $\mathcal{A}$ be a q-quantum-query algorithm with quantum oracle access to an oracle that computes a function from $\mathcal{X}$ to $\mathcal{Y}$ and outputs $\mathbf{x} \in \mathcal{X}^k$ and a (possibly quantum) output ρ. For a function $H : \mathcal{X}^{\leq k} \to \mathcal{Y}$ and $\mathbf{y} = (y_1, \ldots, y_k) \in \mathcal{Y}^k$, we define a MnR game $\mathcal{A}[H, \mathbf{y}]$ in Game 3.1.*

Then, for any positive integer k, any q-quantum query algorithm $\mathcal{A}$, any function $H : \mathcal{X}^{\leq k} \to \mathcal{Y}$, any vectors $\mathbf{x}^* = (x_1^*, \ldots, x_k^*) \in \mathcal{X}^k$ and $\mathbf{y} = (y_1, \ldots, y_k) \in \mathcal{Y}^k$, and any (potentially quantum) predicate Pred, it holds that

$$\Pr\left[\left(\mathbf{x} = \mathbf{x}^*\right) \wedge \left(\mathsf{Pred}(\mathbf{x}, \mathbf{y}, \rho) = 1\right) \ : \ (\mathbf{x}, \rho) \leftarrow \mathcal{A}[H, \mathbf{y}]\right]$$

$$\geq \frac{1}{(2q+1)^{2k}} \cdot \Pr\left[\left(\mathbf{x}=\mathbf{x}^*\right) \wedge \left(\mathsf{Pred}(\mathbf{x},\mathbf{y},\rho)=1\right) \ : \ (\mathbf{x},\rho) \leftarrow \mathcal{A}^{H^{\mathbf{x}^*,\mathbf{y}}}\right].$$

where $H^{\mathbf{x}^*,\mathbf{y}} : \mathcal{X}^{\leq k} \to \mathcal{Y}$ *is defined as*

$$H^{\mathbf{x}^*,\mathbf{y}}(\mathbf{x}') := \begin{cases} y_i & \text{if } \exists i \in [k] \text{ s.t. } \mathbf{x}' = (x_1^*, ..., x_i^*) \\ H(\mathbf{x}') & \text{otherwise} \end{cases}.$$

Remark 3. We remark that the original [CCLY21, Lemma 2.11] defines the output of $\mathcal{A}$ to be $\mathbf{x} \in \mathcal{X}^k$, without the (possibly quantum) ρ part as in our Lem. 3. Our modification is fine since [CCLY21, Lemma 2.11] is indeed a corollary of the "un-ordered" MnR lemma shown in [CCLY21, Lemma 2.10], which allows $\mathcal{A}$ to additionally output a (potentially quantum)[12] part z. The derivation of [CCLY21, Lemma 2.11] from [CCLY21, Lemma 2.10] holds even if one takes the quantum z (i.e., the ρ in our notation) into consideration.

Game 3.1: Measure-and-Reprogram Game $\mathcal{A}[H, \mathbf{y}]$

It works as follows:

1. For each $i \in [k]$, uniformly pick $(j_i, b_i) \in ([q] \times \{0,1\}) \cup \{(\perp, \perp)\}$ conditioned on that there exists at most one $i \in [k]$ such that $j_i = j^*$ for all $j^* \in [q]$.
2. Let s denote the number of j_i's in $\{j_i\}_{i\in[k]}$ that are not $\perp$. We re-label the indices i for pairs $\{(j_i, b_i)\}_{i\in[k]}$ so that $j_1 < j_2 < \ldots < j_s$ and $j_{s+1} = j_{s+2} = \ldots = j_k = \perp$. (See Rmk. 4.)
3. Run $\mathcal{A}^{\mathcal{O}}$ where the oracle $\mathcal{O}$ is initialized to be a quantumly-accessible classical oracle that computes H, and when $\mathcal{A}$ makes its j-th query, the oracle is simulated as follows:
   (a) If $j = j_i$ for some $i \in [s]$, measure $\mathcal{A}$'s query register to obtain $\mathbf{x}'_i = (x'_{i,1}, \ldots, x'_{i,k_i})$ where $k_i \in [k]$ is determined by the measurement outcome, and then behaves according to the value b_i as follows:.
      i. If $b_i = 0$: First reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, \mathbf{x}'_i, y_i)$, and then answer the j_i-th query using the reprogrammed oracle. (See Def. 1 for the definition of $\mathsf{Reprogram}$.)
      ii. If $b_i = 1$: First answer the j_i-th query using the oracle before the reprogramming, and then reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, \mathbf{x}'_i, y_i)$.
   (b) Otherwise (i.e., $j \neq j_i \ \forall i \in [s]$), answer $\mathcal{A}$'s j-th query just using the oracle $\mathcal{O}$ without any measurement or reprogramming.
4. Let $(\mathbf{x} = (x_1, \ldots, x_k), \rho)$ be $\mathcal{A}$'s output.
5. For all $i \in \{s+1, s+2, \ldots, k\}$, set $\mathbf{x}'_i = \mathbf{x}_i$ where $\mathbf{x}_i := (x_1, \cdots, x_i)$.
6. **Output:** The output of this game is defined as follows
   - If it holds for all $i \in [k]$ that $\mathbf{x}'_i$ is a prefix of $\mathbf{x}'_k$, then output $(\mathbf{x}'_k, \rho)$;
   - Otherwise, output $(\perp, \perp)$.

Remark 4 (Regarding Re-labeling). The re-labeling we performed in Step 2 is only cosmetic. It just means to maintain an increasing order for the sequence $\{j_i\}_{i\in[k]}$ and put all the j_i's which are $\perp$ to the very end in this sequence. It does not affect Game 3.1 at all because the order of the j_i's is not used anywhere in this game. But we prefer to maintain an order for $\{j_i\}_{i\in[k]}$ because it will help simplify the presentation of our results and proofs later in this work.

[12] More accurately, [CCLY21, Lemma 2.10] requires z to be a classical string. But [CCLY21, Lemma 2.10] is indeed an adaption of the original MnR lemma from [DFM20, Theorem 6], where z is allowed to be quantum.

3.3 Branch-Wise Equivalence Lemma

We hereby present a lemma pertaining to quantum computing. Although the proof of this lemma is relatively straightforward, it plays a crucial role in streamlining certain aspects of the proof for our main results.

Let us first provide an intuitive description of this lemma. Consider a quantum game wherein unitaries and measurements are alternately applied to an initial pure state. Subsequently, a binary-outcome POVM is applied, determining the final output of the game. We assert that the probability of outputting 1 in such a quantum game can be expressed equivalently as follows:

- Firstly, we examine all possible outcomes of each intermediate measurements (except for the final POVM). For each fixed "outcome sequence" (consisting of the outcomes of all intermediate measurements), we can define a "sub-normalized" version of the game. In this modified version, each intermediate measurement is replaced with a projection that collapses the register to a predetermined outcome (specified in the "outcome sequence"). Finally, a binary-outcome POVM is applied, determining the final output of this "subnormalized" game.
- We then aggregate the probabilities of outputting 1 across all possible "sub-normalized" games.

In the following, we present the formal lemma. To enhance clarity, we will first provide a simplified version of this lemma in Sec. 3.3.1, termed the "baby-case" version. In this context, we consider a quantum game involving the alternating application of two unitaries and two measurements. This simplified scenario serves to illustrate the essence of the lemma and the key concepts underlying its proof. Subsequently, we will present the comprehensive, fully elaborated version in Sec. 3.3.2.

3.3.1 Baby-Case Version

We first present in Game 3.2 the quantum game of alternating applications of two unitaries and measurements.

Game 3.2: Baby-Case Game $G(|\psi\rangle)$ with Alternating Unitary and Measurement

Parameters. Let $|\psi\rangle$ be a pure state over some registers. Let $\mathtt{m}$ be a ℓ-qubit subset of these registers. Let U_1 and U_2 be unitaries over these registers. Let $P = \{E_0, E_1\}$ be a binary-outcome POVM over all the registers.

Game $G(|\psi\rangle)$: on input $|\psi\rangle$, $G(|\psi\rangle)$ performs the following operations:

1. Apply U_1;
2. Measure register $\mathtt{m}$ in the computational basis;
3. Apply U_2;
4. Measure register $\mathtt{m}$ in the computational basis;
5. Apply the POVM P.

Output: the output is defined to be the POVM measurement outcome. We denote it as $\mathsf{OUT}\big(G(|\psi\rangle)\big)$.

Lemma 4 (Branch-Wise Equivalence—Baby Case). *Use the notation in Game 3.2. For each $m_1, m_2 \in \{0,1\}^\ell$, define*

$$|\psi^{(2)}_{m_1,m_2}\rangle := \big(|m_2\rangle\langle m_2|_{\mathtt{m}}\, U_2\big)\big(|m_1\rangle\langle m_1|_{\mathtt{m}}\, U_1\big)|\psi\rangle.$$

Then, it holds that

$$\Pr\big[\mathsf{Out}\big(G(|\psi\rangle)\big)=1\big]=\sum_{m_1,m_2\in\{0,1\}^\ell}\langle\psi^{(2)}_{m_1,m_2}|E_1|\psi^{(2)}_{m_1,m_2}\rangle.$$

Proof of Lem. 4. This lemma can be easily shown by tracking the steps in Game 3.2 with elementary calculation.

Tracking Game 3.2. Right after Step 1, the state $U_1|\psi\rangle$ can be written in the following format

$$U_1|\psi\rangle=\sum_{m_1\in\{0,1\}^\ell}\alpha_{m_1}|m_1\rangle_{\mathtt{m}}|\dot{\psi}^{(1)}_{m_1}\rangle, \tag{3.1}$$

where the α_{m_1}'s are amplitudes satisfies $\sum_{m_1}|\alpha_{m_1}|^2=1$ and $|\dot{\psi}^{(1)}_{m_1}\rangle$'s are pure states over registers other than $\mathtt{m}$. Measuring the $\mathtt{m}$ register (i.e., Step 2) results in the following mixture:

$$\text{with probability } |\alpha_{m_1}|^2, \text{ the overall state collapses to } |m_1\rangle|\dot{\psi}^{(1)}_{m_1}\rangle.$$

For each possible state $|m_1\rangle|\dot{\psi}^{(1)}_{m_1}\rangle$, applying U_2 (i.e., Step 3) yields

$$U_2|m_1\rangle|\dot{\psi}^{(1)}_{m_1}\rangle=\sum_{m_2\in\{0,1\}^\ell}\beta_{m_1,m_2}|m_2\rangle|\dot{\psi}^{(2)}_{m_1,m_2}\rangle, \tag{3.2}$$

where the β_{m_1,m_2}'s are amplitudes satisfies $\sum_{m_2}|\beta_{m_1,m_2}|^2=1$ and $|\dot{\psi}^{(2)}_{m_1,m_2}\rangle$'s are pure states over registers other than $\mathtt{m}$. Now, measuring the $\mathtt{m}$ register (i.e., Step 4) results in the following mixture:

$$\text{with probability } |\beta_{m_1,m_2}|^2, \text{ the overall state collapses to } |m_2\rangle|\dot{\psi}^{2}_{m_1,m_2}\rangle.$$

In summary, we can say that the state right before the POVM (i.e., Step 5) is the following mixture:

$$\text{with probability } |\alpha_{m_1}|^2\cdot|\beta_{m_1,m_2}|^2, \text{ the overall state is } |m_2\rangle|\dot{\psi}^{(2)}_{m_1,m_2}\rangle. \tag{3.3}$$

Analyzing Sub-Normalized States. To relate Game 3.2 with the state $|\psi^{(2)}_{m_1,m_2}\rangle$ defined in Lem. 4, we now define some sub-normalized states. In particular, for each $m_1\in\{0,1\}^\ell$, let

$$\begin{aligned}|\psi^{(1)}_{m_1}\rangle &\coloneqq |m_1\rangle\langle m_1|\,U_1|\psi\rangle &(3.4)\\ &=\alpha_{m_1}|m_1\rangle_{\mathtt{m}}|\dot{\psi}^{(1)}_{m_1}\rangle, &(3.5)\end{aligned}$$

where Eq. (3.5) follows from Eq. (3.1).

Then, it holds for the $|\psi^{(2)}_{m_1,m_2}\rangle$ defined in Lem. 4 that

$$\begin{aligned}|\psi^{(2)}_{m_1,m_2}\rangle &= |m_2\rangle\langle m_2|_{\mathtt{m}}\,U_2|\psi^{(1)}_{m_1}\rangle &(3.6)\\ &=\alpha_{m_1}\,|m_2\rangle\langle m_2|_{\mathtt{m}}\,U_2|m_1\rangle_{\mathtt{m}}|\dot{\psi}^{(1)}_{m_1}\rangle &(3.7)\\ &=\alpha_{m_1}\beta_{m_1,m_2}|m_2\rangle|\dot{\psi}^{(2)}_{m_1,m_2}\rangle, &(3.8)\end{aligned}$$

where Eq. (3.6) follows from Eq. (3.4) and the definition of $|\psi^{(2)}_{m_1,m_2}\rangle$ in Lem. 4, Eq. (3.7) follows from Eq. (3.5), and Eq. (3.8) follows from Eq. (3.2).

Establishing Equivalence. With the above notation, it then follows that

$$\begin{aligned}
\Pr\left[\mathsf{Out}\big(G(|\psi\rangle)\big)=1\right] &= \sum_{m_1,m_2} |\alpha_{m_1}|^2\cdot|\beta_{m_1,m_2}|^2\cdot\langle m_2,\dot{\psi}^{(2)}_{m_1,m_2}|E_1|m_2,\dot{\psi}^{(2)}_{m_1,m_2}\rangle && (3.9)\\
&= \sum_{m_1,m_2} \alpha^*_{m_1}\cdot\alpha_{m_1}\cdot\beta^*_{m_1,m_2}\cdot\beta_{m_1,m_2}\cdot\langle m_2,\dot{\psi}^{(2)}_{m_1,m_2}|E_1|m_2,\dot{\psi}^{(2)}_{m_1,m_2}\rangle\\
&= \sum_{m_1,m_2} \langle m_2,\dot{\psi}^{(2)}_{m_1,m_2}|\alpha^*_{m_1}\beta^*_{m_1,m_2}E_1\alpha_{m_1}\beta_{m_1,m_2}|m_2,\dot{\psi}^{(2)}_{m_1,m_2}\rangle\\
&= \sum_{m_1,m_2} \langle\psi^{(2)}_{m_1,m_2}|E_1|\psi^{(2)}_{m_1,m_2}\rangle, && (3.10)
\end{aligned}$$

where Eq. (3.9) follows from Expression (3.3), Eq. (3.10) follows from Eq. (3.8).

This finishes the proof of Lem. 4.

□

3.3.2 Full-Fledged Version

We now extend the baby-case version into the full-fledged scenario. In particular, we generalize the baby case in two aspects: (1) we allow alternating applications of multiple (instead of two) unitaries and measurements, and (2) the intermediate measurements may performed over different registers.

We present the general quantum game in Game 3.3 and the lemma in Lem. 5.

Game 3.3: Game $G_k(|\psi\rangle)$ with Alternating Unitary and Measurement

Parameters. Let k be an positive integer. Let $|\psi\rangle$ be a pure state over some registers. For each $i\in[k]$, let $\mathtt{m}_i$ be a ℓ_i-qubit subset of these registers. Let $\{U_i\}_{i\in[k]}$ be k unitaries. Let $P=\{E_0,E_1\}$ be a binary-outcome POVM over all the registers.

Game $G_k(|\psi\rangle)$: on input $|\psi\rangle$, $G_k(|\psi\rangle)$ iterates the following operations for each $i\in[k]$:

1. Apply U_i;
2. Measure register $\mathtt{m}_i$ in the computational basis;

Output: finally, apply the POVM P and output the measurement outcome. We denote it as $\mathsf{OUT}(G_k(|\phi\rangle))$.

Lemma 5 (Branch-Wise Equivalence). *Use the notation in Game 3.3. For each possible vector $\mathbf{m}=(m_1,\ldots,m_k)$ where $m_i\in\{0,1\}^{\ell_i}$ for all $i\in[k]$, define a sub-normalized state*

$$|\psi^{(k)}_{\mathbf{m}}\rangle := \left(\prod_{i=1}^{k}|m_i\rangle\langle m_i|_{\mathtt{m}_i}U_i\right)|\psi\rangle = \big(|m_k\rangle\langle m_k|_{\mathtt{m}_k}U_k\big)\big(|m_{k-1}\rangle\langle m_{k-1}|_{\mathtt{m}_{k-1}}U_{k-1}\big)\cdots\big(|m_1\rangle\langle m_1|_{\mathtt{m}_1}U_1\big)|\psi\rangle.$$

Then, it holds that

$$\Pr\left[\mathsf{Out}\big(G_k(|\psi\rangle)\big)=1\right]=\sum_{\mathbf{m}}\langle\psi^{(k)}_{\mathbf{m}}|E_1|\psi^{(k)}_{\mathbf{m}}\rangle,$$

where the summation over $\mathbf{m}$ means to sum over all $\mathbf{m}=(m_1,\ldots,m_k)\in\{0,1\}^{\ell_1}\times\ldots\times\{0,1\}^{\ell_k}$.

Proof of Lem. 5. We prove this lemma via a mathematical induction over the number k of measurements (excluding the final POVM).

Base Case ($k = 1$)**.** This corresponds to game $G_1(|\psi\rangle)$ which consists of only one unitary U_1 and one measurement over register m_1 (before the final POVM). In this case, it is obvious that Lem. 5 holds. Indeed, Lem. 4 can be understood as a special case of $k = 2$, modulo that the two measurement there were performed over the same register $\mathtt{m}$ (i.e., $\mathtt{m}_1 = \mathtt{m}_2$ in the Lem. 5 notation).

Induction Step ($k \geq 2$)**.** We now assume the lemma holds for $k-1$, and prove it for k.

Consider the first iteration of $G_k(|\psi\rangle)$. Right after Step 1 of the first iteration, the state becomes $U_1|\psi\rangle$. Such a state can be written in the following format

$$U_1|\psi\rangle = \sum_{m_1 \in \{0,1\}^\ell} \alpha_{m_1} |m_1\rangle_{\mathtt{m}_1} |\dot{\psi}^{(1)}_{m_1}\rangle, \tag{3.11}$$

where the α_{m_1}'s are amplitudes satisfies $\sum_{m_1} |\alpha_{m_1}|^2 = 1$ and the $|\dot{\psi}^{(1)}_{m_1}\rangle$'s are pure states over registers other than $\mathtt{m}_1$. Measuring the $\mathtt{m}_1$ register (i.e., Step 2 in the first iteration) results in the following mixture:

$$\text{with probability } |\alpha_{m_1}|^2, \text{ the overall state collapses to } |m_1\rangle_{\mathtt{m}_1} |\dot{\psi}^{(1)}_{m_1}\rangle.$$

Therefore, the output of the game can be described as:

$$\Pr\big[\mathsf{Out}\big(G_k(|\psi\rangle)\big) = 1\big] = \sum_{m_1 \in \{0,1\}^\ell} |\alpha_{m_1}|^2 \cdot \Pr\Big[\mathsf{Out}\big(G_{2:k}(|m_1\rangle_{\mathtt{m}_1} |\dot{\psi}^{(1)}_{m_1}\rangle)\big) = 1\Big], \tag{3.12}$$

where $G_{2:k}(|m_1\rangle_{\mathtt{m}_1} |\dot{\psi}^{(1)}_{m_1}\rangle)$ denote the remaining party of $G_k(|\psi\rangle)$ (i.e., right after the the first iteration). For clearness, we present this game in Game 3.4.

Game 3.4: Game $G_{2:k}(|m_1\rangle_{\mathtt{m}_1} |\dot{\psi}^{(1)}_{m_1}\rangle)$

Parameters. Same as in Game 3.3.

Game $G_{2:k}(|m_1\rangle_{\mathtt{m}_1} |\dot{\psi}^{(1)}_{m_1}\rangle)$**:** on input $|m_1\rangle_{\mathtt{m}_1} |\dot{\psi}^{(1)}_{m_1}\rangle$, iterate the following operations for each $i \in \{2, 3, \ldots, k\}$:

1. Apply U_i;
2. Measure register $\mathtt{m}_i$ in the computational basis;

Output: apply the POVM P and output the measurement outcome.

Now, it is important to notice that Game 3.4 can be viewed as a version of Game 3.3 with parameter $k-1$, because it includes $k-1$ iterations of alternating unitaries and measurements (and then the final POVM). Therefore, we can invoke our induction assumption. In particular, for each possible state $|m_1\rangle_{\mathtt{m}_1} |\dot{\psi}^{(1)}_{m_1}\rangle$, our induction assumption implies that

$$\Pr\Big[\mathsf{Out}\big(G_{2:k}(|m_1\rangle_{\mathtt{m}_1} |\dot{\psi}^{(1)}_{m_1}\rangle)\big) = 1\Big] = \sum_{\mathbf{m}'} \langle \dot{\psi}^{(k-1)}_{\mathbf{m}'} | E_1 | \dot{\psi}^{(k-1)}_{\mathbf{m}'} \rangle, \tag{3.13}$$

where the summation over $\mathbf{m}'$ means to sum over all $\mathbf{m}' = (m_2, \ldots, m_k) \in \{0,1\}^{\ell_2} \times \ldots \times \{0,1\}^{\ell_k}$, and the state $|\psi^{(k-1)}_{\mathbf{m}'}\rangle$ is defined as

$$|\dot{\psi}^{(k-1)}_{\mathbf{m}'}\rangle := \left(\prod_{i=2}^{k} |m_i\rangle\langle m_i|_{\mathtt{m}_i} U_i \right) |m_1\rangle_{\mathtt{m}_1} |\dot{\psi}^{(1)}_{m_1}\rangle. \tag{3.14}$$

On the other hand, we note that

$$|\psi^{(k)}_{\mathbf{m}}\rangle = \left(\prod_{i=1}^{k} |m_i\rangle\langle m_i|_{\mathtt{m}_i} U_i\right)|\psi\rangle \tag{3.15}$$

$$= \left(\prod_{i=2}^{k} |m_i\rangle\langle m_i|_{\mathtt{m}_i} U_i\right) |m_1\rangle\langle m_1|_{\mathtt{m}_1} U_1|\psi\rangle$$

$$= \alpha_{m_1}\left(\prod_{i=2}^{k} |m_i\rangle\langle m_i|_{\mathtt{m}_i} U_i\right)|m_1\rangle_{\mathtt{m}_1}|\dot{\psi}^{(1)}_{m_1}\rangle \tag{3.16}$$

$$= \alpha_{m_1}|\dot{\psi}^{(k-1)}_{\mathbf{m}'}\rangle, \tag{3.17}$$

where Eq. (3.15) is the definition of $|\psi^{(k)}_{\mathbf{m}}\rangle$ (see Lem. 5), Eq. (3.16) follows from Eq. (3.11), and Eq. (3.17) follows from Eq. (3.14).

With the above notation, we show the final derivation

$$\Pr\left[\mathsf{Out}\left(G_k(|\psi\rangle)\right) = 1\right] = \sum_{m_1\in\{0,1\}^\ell} |\alpha_{m_1}|^2 \cdot \Pr\left[\mathsf{Out}\left(G_{2:k}(|m_1\rangle_{\mathtt{m}_1}|\dot{\psi}^{(1)}_{m_1}\rangle)\right) = 1\right] \tag{3.18}$$

$$= \sum_{m_1\in\{0,1\}^\ell} |\alpha_{m_1}|^2 \cdot \sum_{\mathbf{m}'} \langle\dot{\psi}^{(k-1)}_{\mathbf{m}'}|E_1|\dot{\psi}^{(k-1)}_{\mathbf{m}'}\rangle \tag{3.19}$$

$$= \sum_{\mathbf{m}',m_1\in\{0,1\}^\ell} \langle\dot{\psi}^{(k-1)}_{\mathbf{m}'}|\alpha^*_{m_1}E_1\alpha_{m_1}|\dot{\psi}^{(k-1)}_{\mathbf{m}'}\rangle$$

$$= \sum_{\mathbf{m}} \langle\psi^{(k)}_{\mathbf{m}}|E_1|\psi^{(k)}_{\mathbf{m}}\rangle, \tag{3.20}$$

where Eq. (3.18) follows from Eq. (3.12), Eq. (3.19) follows from Eq. (3.13), and Eq. (3.20) follows from Eq. (3.17).

This completes the proof of the induction step.

This completes the proof of Lem. 5.

□

4 Error-Invariant Commutativity Lemma

In this section, we present an information-theoretic lemma regarding the commutativity of some unitary operators. As will become evident later, this lemma will serve as a clean abstraction of the behaviors of the ZK simulator and a malicious verifier we designed for the impossibility results.

4.1 Statement and Interpretation

Lemma 6 (Error-Invariant Commutativity Lemma). *Let W_0, W_1, and U_0 be unitary operators over $\mathcal{H}_{\mathtt{m}} \otimes \mathcal{H}_{\mathtt{t}} \otimes \mathcal{H}_{\mathtt{s}} \otimes \mathcal{H}_{\mathtt{o}}$ satisfying the following requirements:*

- *W_1 acts non-trivially only on $\mathcal{H}_{\mathtt{m}} \otimes \mathcal{H}_{\mathtt{t}} \otimes \mathcal{H}_{\mathtt{o}}$, and is identity on $\mathcal{H}_{\mathtt{s}}$.*
- *W_0 is the swap operator between registers $\mathtt{m}$ and $\mathtt{t}$, and is identity on $\mathcal{H}_{\mathtt{s}} \otimes \mathcal{H}_{\mathtt{o}}$.*
- *U_1 acts non-trivially only on $\mathcal{H}_{\mathtt{m}} \otimes \mathcal{H}_{\mathtt{t}} \otimes \mathcal{H}_{\mathtt{o}}$, and is identity on $\mathcal{H}_{\mathtt{s}}$.*

Let $\mathcal{H}_{\mathtt{a}}$ be a Hilbert space of ℓ qubits. Let $a \in \{0,1\}^\ell$ be a fixed classical string. Define the following unitary operators on $\mathcal{H}_{\mathtt{a}} \otimes \mathcal{H}_{\mathtt{m}} \otimes \mathcal{H}_{\mathtt{t}} \otimes \mathcal{H}_{\mathtt{s}} \otimes \mathcal{H}_{\mathtt{o}}$:

- $W := |a\rangle\langle a|_{\mathtt{a}} \otimes W_1 + \sum_{a' \neq a} |a'\rangle\langle a'|_{\mathtt{a}} \otimes W_0$, *where W_0 and W_1 are as specified above.*
- $\widetilde{W} := |a\rangle\langle a|_{\mathtt{a}} \otimes I_{\mathtt{mtso}} + \sum_{a' \neq a} |a'\rangle\langle a'|_{\mathtt{a}} \otimes W_0$, *where $I_{\mathtt{mtso}}$ is the identity operator on $\mathcal{H}_{\mathtt{m}} \otimes \mathcal{H}_{\mathtt{t}} \otimes \mathcal{H}_{\mathtt{s}} \otimes \mathcal{H}_{\mathtt{o}}$ and W_0 are as specified above.*
- $U := |a\rangle\langle a|_{\mathtt{a}} \otimes U_1 + \sum_{a' \neq a} |a'\rangle\langle a'|_{\mathtt{a}} \otimes I_{\mathtt{mtso}}$, *where U_1 is as specified above and $I_{\mathtt{mtso}}$ is the identity operator on $\mathcal{H}_{\mathtt{m}} \otimes \mathcal{H}_{\mathtt{t}} \otimes \mathcal{H}_{\mathtt{s}} \otimes \mathcal{H}_{\mathtt{o}}$.*
- *S is an operator that acts non-trivially only on $\mathcal{H}_{\mathtt{a}} \otimes \mathcal{H}_{\mathtt{s}}$, and is identity on $\mathcal{H}_{\mathtt{m}} \otimes \mathcal{H}_{\mathtt{t}} \otimes \mathcal{H}_{\mathtt{o}}$.*

These operators satisfy the following property:

- *Let $\{|\rho_{a'}^{(\mathsf{in})}\rangle_{\mathtt{mtso}}\}_{a' \in \{0,1\}^\ell}$ be a sequence of (potentially sub-normalized) pure states over registers $\mathtt{m}$, $\mathtt{t}$, $\mathtt{s}$, and $\mathtt{o}$. Define states $|\eta_0^{(\mathsf{in})}\rangle$, $|\eta_1^{(\mathsf{in})}\rangle$, $|\eta_0^{(\mathsf{out})}\rangle$, and $|\eta_1^{(\mathsf{out})}\rangle$ as follows:*

$$|\eta_0^{(\mathsf{in})}\rangle := |a\rangle_{\mathtt{a}}|\rho_a^{(\mathsf{in})}\rangle_{\mathtt{mtso}} + \sum_{a' \in \{0,1\}^\ell \setminus \{a\}} |a'\rangle_{\mathtt{a}} W_0 U_1^\dagger W_1^\dagger |\rho_{a'}^{(\mathsf{in})}\rangle_{\mathtt{mtso}} \tag{4.1}$$

$$|\eta_1^{(\mathsf{in})}\rangle := |a\rangle_{\mathtt{a}}|\rho_a^{(\mathsf{in})}\rangle_{\mathtt{mtso}} + \sum_{a' \in \{0,1\}^\ell \setminus \{a\}} |a'\rangle_{\mathtt{a}} W_0 |\rho_{a'}^{(\mathsf{in})}\rangle_{\mathtt{mtso}} \tag{4.2}$$

$$|\eta_0^{(\mathsf{out})}\rangle := WUSU^\dagger W^\dagger \cdot |\eta_0^{(\mathsf{in})}\rangle \tag{4.3}$$

$$|\eta_1^{(\mathsf{out})}\rangle := \widetilde{W}S\widetilde{W}^\dagger \cdot |\eta_1^{(\mathsf{in})}\rangle \tag{4.4}$$

Then, there exists a sequence of (potentially sub-normalized) pure states $\{|\rho_{a'}^{(\mathsf{out})}\rangle_{\mathtt{mtso}}\}_{a' \in \{0,1\}^\ell}$ so that the states $|\eta_0^{(\mathsf{out})}\rangle$ and $|\eta_1^{(\mathsf{out})}\rangle$ defined above can be written in the following format:

$$|\eta_0^{(\mathsf{out})}\rangle = |a\rangle_{\mathtt{a}}|\rho_a^{(\mathsf{out})}\rangle_{\mathtt{mtso}} + \sum_{a' \in \{0,1\}^\ell \setminus \{a\}} |a'\rangle_{\mathtt{a}} W_0 U_1^\dagger W_1^\dagger |\rho_{a'}^{(\mathsf{out})}\rangle_{\mathtt{mtso}} \tag{4.5}$$

$$|\eta_1^{(\mathsf{out})}\rangle = |a\rangle_{\mathtt{a}}|\rho_a^{(\mathsf{out})}\rangle_{\mathtt{mtso}} + \sum_{a' \in \{0,1\}^\ell \setminus \{a\}} |a'\rangle_{\mathtt{a}} W_0 |\rho_{a'}^{(\mathsf{out})}\rangle_{\mathtt{mtso}} \tag{4.6}$$

4.2 Proof of Lem. 6

In this proof, we assume for simplicity that each of the registers $\mathtt{m}$, $\mathtt{t}$, $\mathtt{s}$, and $\mathtt{o}$ consists of ℓ qubits. This assumption is without loss of generality because the subsequent derivation works regardless of the length of these registers.

We first note that since S acts non-trivially only on $\mathcal{H}_{\mathtt{a}} \otimes \mathcal{H}_{\mathtt{s}}$, it holds that for any $a' \in \{0,1\}^\ell$, $s \in \{0,1\}^\ell$, and any pure state $|\rho_{a',s}\rangle_{\mathtt{mto}}$,

$$S|a'\rangle_{\mathtt{a}}|s\rangle_{\mathtt{s}}|\rho_{a',s}\rangle_{\mathtt{mto}} = \sum_{a^*,s^* \in \{0,1\}^\ell} \beta_{a^*,s^*}^{a',s} |a^*\rangle_{\mathtt{a}}|s^*\rangle_{\mathtt{s}}|\rho_{a',s}\rangle_{\mathtt{mto}}, \tag{4.7}$$

where each $\beta_{a^*,s^*}^{a',s}$ is a complex number that depends on (a', s, a^*, s^*).[13] Note that since S may not be unitary, the β values may not be normalized. But this does not affect our proof as we work with sub-normalized states anyway.

In the following, we derive Eq. (4.5) and (4.6) one by one.

[13] It is worth noting that Eq. (4.7) holds for *any* $a' \in \{0,1\}^\ell$, including the case where a' equals to the a we fixed.

4.2.1 Deriving Eq. (4.5)

We first derive Eq. (4.5):

$$
\begin{aligned}
|\eta_0^{(\mathsf{out})}\rangle &= WUSU^\dagger W^\dagger \cdot |\eta_0^{(\mathsf{in})}\rangle && (4.8)\\
&= WUSU^\dagger W^\dagger \cdot \Big(|a\rangle_{\mathtt{a}}|\rho_a^{(\mathsf{in})}\rangle_{\mathtt{mtso}} + \sum_{a'\in\{0,1\}^\ell\setminus\{a\}} |a'\rangle_{\mathtt{a}} W_0 U_1^\dagger W_1^\dagger |\rho_{a'}^{(\mathsf{in})}\rangle_{\mathtt{mtso}}\Big) && (4.9)\\
&= WUSU^\dagger W^\dagger \cdot \Big(|a\rangle_{\mathtt{a}} \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}}|\rho_{a,s}^{(\mathsf{in})}\rangle_{\mathtt{mto}} \; + \\
&\qquad \sum_{a'\in\{0,1\}^\ell\setminus\{a\}} |a'\rangle_{\mathtt{a}} W_0 U_1^\dagger W_1^\dagger \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}}|\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}}\Big)\\
&= WUSU^\dagger W^\dagger \cdot \Big(|a\rangle_{\mathtt{a}} \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}}|\rho_{a,s}^{(\mathsf{in})}\rangle_{\mathtt{mto}} \; + \\
&\qquad \sum_{a'\in\{0,1\}^\ell\setminus\{a\}} |a'\rangle_{\mathtt{a}} \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}} W_0 U_1^\dagger W_1^\dagger |\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}}\Big) && (4.10)\\
&= WUSU^\dagger \cdot \Big(|a\rangle_{\mathtt{a}} \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}} W_1^\dagger |\rho_{a,s}^{(\mathsf{in})}\rangle_{\mathtt{mto}} \; + \\
&\qquad \sum_{a'\in\{0,1\}^\ell\setminus\{a\}} |a'\rangle_{\mathtt{a}} \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}} U_1^\dagger W_1^\dagger |\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}}\Big) && (4.11)\\
&= WUS \cdot \Big(|a\rangle_{\mathtt{a}} \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}} U_1^\dagger W_1^\dagger |\rho_{a,s}^{(\mathsf{in})}\rangle_{\mathtt{mto}} \; + \\
&\qquad \sum_{a'\in\{0,1\}^\ell\setminus\{a\}} |a'\rangle_{\mathtt{a}} \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}} U_1^\dagger W_1^\dagger |\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}}\Big) && (4.12)\\
&= WUS \cdot \Big(\sum_{a'\in\{0,1\}^\ell} |a'\rangle_{\mathtt{a}} \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}} U_1^\dagger W_1^\dagger |\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}}\Big)\\
&= WUS \cdot \Big(\sum_{a',s\in\{0,1\}^\ell} |a'\rangle_{\mathtt{a}}|s\rangle_{\mathtt{s}} U_1^\dagger W_1^\dagger |\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}}\Big)\\
&= WU \cdot \Big(\sum_{a',s\in\{0,1\}^\ell} S|a'\rangle_{\mathtt{a}}|s\rangle_{\mathtt{s}} U_1^\dagger W_1^\dagger |\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}}\Big)\\
\text{(by Eq. (4.7))} \quad &= WU \cdot \Big(\sum_{a',s\in\{0,1\}^\ell} \sum_{a^*,s^*\in\{0,1\}^\ell} \beta_{a^*,s^*}^{a',s} |a^*\rangle_{\mathtt{a}}|s^*\rangle_{\mathtt{s}} U_1^\dagger W_1^\dagger |\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}}\Big)\\
&= WU \cdot \sum_{a',s\in\{0,1\}^\ell} \Big(\sum_{s^*\in\{0,1\}^\ell} \beta_{a,s^*}^{a',s} |a\rangle_{\mathtt{a}}|s^*\rangle_{\mathtt{s}} U_1^\dagger W_1^\dagger |\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}} \; + \\
&\qquad \sum_{a^*\in\{0,1\}^\ell\setminus\{a\}} \sum_{s^*\in\{0,1\}^\ell} \beta_{a^*,s^*}^{a',s} |a^*\rangle_{\mathtt{a}}|s^*\rangle_{\mathtt{s}} U_1^\dagger W_1^\dagger |\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}}\Big)\\
&= W \cdot \sum_{a',s\in\{0,1\}^\ell} \Big(\sum_{s^*\in\{0,1\}^\ell} \beta_{a,s^*}^{a',s} |a\rangle_{\mathtt{a}}|s^*\rangle_{\mathtt{s}} W_1^\dagger |\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}} \; + \\
&\qquad \sum_{a^*\in\{0,1\}^\ell\setminus\{a\}} \sum_{s^*\in\{0,1\}^\ell} \beta_{a^*,s^*}^{a',s} |a^*\rangle_{\mathtt{a}}|s^*\rangle_{\mathtt{s}} U_1^\dagger W_1^\dagger |\rho_{a',s}^{(\mathsf{in})}\rangle_{\mathtt{mto}}\Big) && (4.13)
\end{aligned}
$$

$$= \sum_{a',s\in\{0,1\}^\ell} \Bigg(\sum_{s^*\in\{0,1\}^\ell} \beta^{a',s}_{a,s^*} |a\rangle_{\mathtt{a}} |s^*\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \; + \sum_{a^*\in\{0,1\}^\ell\setminus\{a\}} \sum_{s^*\in\{0,1\}^\ell} \beta^{a',s}_{a^*,s^*} |a^*\rangle_{\mathtt{a}} |s^*\rangle_{\mathtt{s}} W_0 U_1^\dagger W_1^\dagger |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \Bigg) \tag{4.14}$$

$$= |a\rangle_{\mathtt{a}} \sum_{a',s,s^*\in\{0,1\}^\ell} \beta^{a',s}_{a,s^*} |s^*\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \; + \sum_{a^*\in\{0,1\}^\ell\setminus\{a\}} |a^*\rangle_{\mathtt{a}} W_0 U_1^\dagger W_1^\dagger \sum_{a',s,s^*\in\{0,1\}^\ell} \beta^{a',s}_{a^*,s^*} |s^*\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \tag{4.15}$$

$$= |a\rangle_{\mathtt{a}} |\rho^{(\mathsf{out})}_{a}\rangle_{\mathtt{mtso}} + \sum_{a^*\in\{0,1\}^\ell\setminus\{a\}} |a^*\rangle_{\mathtt{a}} W_0 U_1^\dagger W_1^\dagger |\rho^{(\mathsf{out})}_{a^*}\rangle_{\mathtt{mtso}} \tag{4.16}$$

$$= |a\rangle_{\mathtt{a}} |\rho^{(\mathsf{out})}_{a}\rangle_{\mathtt{mtso}} + \sum_{a'\in\{0,1\}^\ell\setminus\{a\}} |a'\rangle_{\mathtt{a}} W_0 U_1^\dagger W_1^\dagger |\rho^{(\mathsf{out})}_{a'}\rangle_{\mathtt{mtso}}, \tag{4.17}$$

where Eq. (4.8) follows from Eq. (4.3), Eq. (4.9) follows from Eq. (4.1), Eq. (4.10) follows from the fact that all the unitary operators W_0, $U_1^\dagger$, and $W_1^\dagger$ act as identity on $\mathcal{H}_{\mathtt{s}}$, Eq. (4.11) follows from the definition of $W^\dagger$ and the fact that $W_0^\dagger$ and $W_1^\dagger$ act as identity on $\mathcal{H}_{\mathtt{s}}$, Eq. (4.12) follow from the definition of $U^\dagger$ and the fact that $U_1^\dagger$ acts as identity on $\mathcal{H}_{\mathtt{s}}$, Eq. (4.13) follow from the definition of U and the fact that U_1 acts as identity on $\mathcal{H}_{\mathtt{s}}$, Eq. (4.14) follow from the definition of W and the fact that W_0 and W_1 act as identity on $\mathcal{H}_{\mathtt{s}}$, and Eq. (4.15) follows from standard algebraic calculation and the fact that all the unitary operators W_0, $U_1^\dagger$, and $W_1^\dagger$ act as identity on $\mathcal{H}_{\mathtt{s}}$, Eq. (4.16) follows by defining

$$\forall\; a^* \in \{0,1\}^\ell, \quad |\rho^{(\mathsf{out})}_{a^*}\rangle_{\mathtt{mtso}} \coloneqq \sum_{a',s,s^*\in\{0,1\}^\ell} \beta^{a',s}_{a^*,s^*} |s^*\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}}, \tag{4.18}$$

and Eq. (4.17) follows simply by renaming a^* to a'.

This concludes the proof of Eq. (4.5).

4.2.2 Deriving Eq. (4.6)

Finally, we derive Eq. (4.6):

$$|\eta_1^{(\mathsf{out})}\rangle = \widetilde{W} S \widetilde{W}^\dagger \cdot |\eta_1^{(\mathsf{in})}\rangle \tag{4.19}$$

$$= \widetilde{W} S \widetilde{W}^\dagger \cdot \Big(|a\rangle_{\mathtt{a}} |\rho^{(\mathsf{in})}_{a}\rangle_{\mathtt{mtso}} + \sum_{a'\in\{0,1\}^\ell\setminus\{a\}} |a'\rangle_{\mathtt{a}} W_0 |\rho^{(\mathsf{in})}_{a'}\rangle_{\mathtt{mtso}} \Big) \tag{4.20}$$

$$= \widetilde{W} S \widetilde{W}^\dagger \cdot \Big(|a\rangle_{\mathtt{a}} \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a,s}\rangle_{\mathtt{mto}} \; + \sum_{a'\in\{0,1\}^\ell\setminus\{a\}} |a'\rangle_{\mathtt{a}} W_0 \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \Big)$$

$$= \widetilde{W} S \cdot \Big(|a\rangle_{\mathtt{a}} \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a,s}\rangle_{\mathtt{mto}} \; + \sum_{a'\in\{0,1\}^\ell\setminus\{a\}} |a'\rangle_{\mathtt{a}} \sum_{s\in\{0,1\}^\ell} |s\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \Big) \tag{4.21}$$

$$
\begin{aligned}
&= \widetilde{W} S \cdot \Big(\sum_{a' \in \{0,1\}^\ell} |a'\rangle_{\mathtt{a}} \sum_{s \in \{0,1\}^\ell} |s\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \Big) \\
&= \widetilde{W} \cdot \Big(\sum_{a',s \in \{0,1\}^\ell} S|a'\rangle_{\mathtt{a}}|s\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \Big) \\
\text{(by Eq. (4.7))} \quad &= \widetilde{W} \cdot \Big(\sum_{a',s \in \{0,1\}^\ell} \sum_{a^*,s^* \in \{0,1\}^\ell} \beta^{a',s}_{a^*,s^*} |a^*\rangle_{\mathtt{a}}|s^*\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \Big) \\
&= \widetilde{W} \cdot \sum_{a',s \in \{0,1\}^\ell} \Big(\sum_{s^* \in \{0,1\}^\ell} \beta^{a',s}_{a,s^*} |a\rangle_{\mathtt{a}}|s^*\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \; + \\
&\qquad \sum_{a^* \in \{0,1\}^\ell \setminus \{a\}} \sum_{s^* \in \{0,1\}^\ell} \beta^{a',s}_{a^*,s^*} |a^*\rangle_{\mathtt{a}}|s^*\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \Big) \\
&= \sum_{a',s \in \{0,1\}^\ell} \Big(\sum_{s^* \in \{0,1\}^\ell} \beta^{a',s}_{a,s^*} |a\rangle_{\mathtt{a}}|s^*\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \; + \\
&\qquad \sum_{a^* \in \{0,1\}^\ell \setminus \{a\}} \sum_{s^* \in \{0,1\}^\ell} \beta^{a',s}_{a^*,s^*} |a^*\rangle_{\mathtt{a}}|s^*\rangle_{\mathtt{s}} W_0 |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \Big) \qquad (4.22) \\
&= |a\rangle_{\mathtt{a}} \sum_{a',s,s^* \in \{0,1\}^\ell} \beta^{a',s}_{a,s^*} |s^*\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \; + \\
&\qquad \sum_{a^* \in \{0,1\}^\ell \setminus \{a\}} |a^*\rangle_{\mathtt{a}} W_0 \sum_{a',s,s^* \in \{0,1\}^\ell} \beta^{a',s}_{a^*,s^*} |s^*\rangle_{\mathtt{s}} |\rho^{(\mathsf{in})}_{a',s}\rangle_{\mathtt{mto}} \qquad (4.23) \\
&= |a\rangle_{\mathtt{a}} |\rho^{(\mathsf{out})}_{a}\rangle_{\mathtt{mtso}} + \sum_{a^* \in \{0,1\}^\ell \setminus \{a\}} |a^*\rangle_{\mathtt{a}} W_0 |\rho^{(\mathsf{out})}_{a^*}\rangle_{\mathtt{mtso}} \qquad (4.24) \\
&= |a\rangle_{\mathtt{a}} |\rho^{(\mathsf{out})}_{a}\rangle_{\mathtt{mtso}} + \sum_{a' \in \{0,1\}^\ell \setminus \{a\}} |a'\rangle_{\mathtt{a}} W_0 |\rho^{(\mathsf{out})}_{a'}\rangle_{\mathtt{mtso}}, \qquad (4.25)
\end{aligned}
$$

where Eq. (4.19) follows from Eq. (4.4), Eq. (4.20) follows from Eq. (4.2), Eq. (4.21) follows from the definition of $\widetilde{W}^\dagger$, Eq. (4.22) follows from the definition of $\widetilde{W}$ and the fact that W_0 acts as identity on $\mathcal{H}_{\mathtt{s}}$, and Eq. (4.23) follows from standard algebraic calculation and the fact that W_0 acts as identity on $\mathcal{H}_{\mathtt{s}}$, Eq. (4.24) follows from *the same definition of* $\{|\rho^{(\mathsf{out})}_{a^*}\rangle\}_{a^* \in \{0,1\}^\ell}$ as shown in Expression (4.18), and Eq. (4.25) follows by renaming a^* to a'.

This concludes the proof of Eq. (4.6).

This eventually completes the proof of Lem. 6.

5 The Models for Quantum Zero-Knowledge

5.1 Quantum Black-Box Zero-Knowledge Protocols for NP

We define quantum black-box zero-knowledge proofs (and arguments) for **NP**. This model is similar to the standard notion of *post-quantum* black-box zero-knowledge proofs (and arguments) for **NP** (e.g., see [CCLY21]) with the following differences:

– The (honest) prover and verifier could be quantum polynomial-time machines. In particular, their communication channel is also quantum.

In the following, we present the formal definitions. We first define quantum interactive proofs (and arguments) for **NP** (in Sec. 5.1.1), and then define zero-knowledge property (in Sec. 5.1.2).

5.1.1 Quantum Interactive Proofs/Arguments for NP

For an **NP** language $\mathcal{L}$ and $x \in \mathcal{L}$, $\mathcal{R}_{\mathcal{L}}(x)$ is the set that consists of all (classical) witnesses w such that the verification machine for $\mathcal{L}$ accepts (x, w).

A quantum interactive protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ is modeled as an interaction between *interactive quantum polynomial-time machines*[14] $\mathcal{P}$ (dubbed the prover) and $\mathcal{V}$ (dubbed the verifier). We denote by $\langle \mathcal{P}(x_{\mathcal{P}}), \mathcal{V}(x_{\mathcal{V}}) \rangle(x)$ an execution of the protocol where x is a common input, $x_{\mathcal{P}}$ is the prover's private input, and $x_{\mathcal{V}}$ is the verifier's private input. We denote by $\rho \leftarrow \mathsf{OUT}_{\mathcal{V}} \langle \mathcal{P}(x_{\mathcal{P}}), \mathcal{V}(x_{\mathcal{V}}) \rangle(x)$ the final output of $\mathcal{V}$, where ρ is a single-qubit state over a special register d of $\mathcal{V}$. We also define a quantum predicate $\mathsf{Acc}(\cdot)$ which, on input a single-qubit state ρ, measures ρ in the computational basis and outputs the measurement outcome. Intuitive, the output of Acc indicates if the the verifier accepts (outputting 1) or rejects (outputting 0).

Definition 2 (Quantum Interactive Proofs/Arguments). *A quantum interactive proof or argument $\langle \mathcal{P}, \mathcal{V} \rangle$ for an* **NP** *language $\mathcal{L}$ is a quantum interactive protocol between a QPT prover $\mathcal{P}$ and a QPT verifier $\mathcal{V}$ that satisfy the following requirements:*

- **Completeness.** *For each $x \in \mathcal{L}$ and each $w \in \mathcal{R}_{\mathcal{L}}(x)$, it holds that*

$$\Pr\left[\mathsf{Acc}(\rho) = 1 \ : \ \rho \leftarrow \mathsf{OUT}_{\mathcal{V}} \langle \mathcal{P}(w), \mathcal{V} \rangle(x)\right] \geq 1 - \mathsf{negl}(\lambda).$$

- **Statistical/Computational Soundness.** *The protocol is statistically (resp. computationally)* sound *if for any unbounded (resp. non-uniform QPT) cheating prover $\mathcal{P}^*$ and any $x \in \{0,1\}^\lambda \backslash \mathcal{L}$, it holds that*

$$\Pr\left[\mathsf{Acc}(\rho) = 1 \ : \ \rho \leftarrow \mathsf{OUT}_{\mathcal{V}} \langle \mathcal{P}^*, \mathcal{V} \rangle(x)\right] = \mathsf{negl}(\lambda).$$

  The protocol is dubbed an interactive proof (resp. argument) if the soundness property is statistical (resp. computational).

Remark 5. We note that considering the function $\mathsf{Acc}(\cdot)$ as part of the verifier is a valid alternative. However, we prefer to use the above formalism to treat $\mathsf{Acc}(\cdot)$ as an "external" measurement. This approach aims to facilitate our subsequent presentation, where we will assume without loss of generality that the verifier $\mathcal{V}$ does not conduct any measurements.

Remark 6. In a plain standard quantum interactive protocol, $\mathcal{V}$ and $\mathcal{P}$ do not pre-share any entanglement at the beginning of the protocol.

5.1.2 Black-Box Quantum Zero-Knowledge

Next, we proceed to define the quantum zero-knowledge property. Toward that, we first need to be more accurate about the malicious verifiers and the way the simulator interacts with the malicious verifier.

Number of Rounds. We say that a quantum interactive proof/argument has K rounds if the messages exchanged between $\mathcal{P}$ and $\mathcal{V}$ are of the following structure

$$(p_1, v_1, p_2, v_2, \ldots, p_{K-1}, v_{K-1}, p_K, v_K),$$

where p_1 is the first message $\mathcal{P}$ sends to $\mathcal{V}$, and v_1 is $\mathcal{V}$'s response to p_1, and p_2 is the next message $\mathcal{P}$ sends to $\mathcal{V}$ and so one. Some remarks regarding this message structure follow:

[14] We refer to the literature (e.g., [HSS11]) for the standard notion of *quantum interactive machines*.

– All the p_i's and v_i's could be quantum messages, as we allow quantum communication in quantum interactive proofs/arguments.

– It is without loss of generality to assume that $\mathcal{P}$ sends the first message (i.e., p_1).

– Typically, the last message should be from the prover to the verifier. However, in our model, we ask the verifier to send the last message v_K. We explain the reason in Rmk. 7.

– We remark that different authors may use the term "round" differently. In this work, we use it to refer to *a pair* of adjacent messages. For example, (p_1, v_1) constitutes the first round, and (p_2, v_2) constitutes the second round and so on. This is why we say that the protocol, where $2K$ messages are actually exchanged, has K rounds.

Remark 7 (On the last message v_K). In this work, we require the honest verifier to send its final decision qubit (i.e., register $\mathtt{d}$) to the prover, which is exactly the last message v_K. It is important to note that this requirement does not impact completeness or soundness. However, it does influence the definition of zero-knowledge. Specifically, it could potentially make the simulator's task easier, leading to a weaker definition of the zero-knowledge property. Nevertheless, since our goal is to establish impossibility results, this assumption could only make our results stronger.

A related issue arises: If we ask the honest $\mathcal{V}$ to sends its final decision qubit as message v_K, how do we define the final output of $\mathcal{V}$? This can be handled as follows. When the message v_K is generated in register $\mathtt{m}$, we ask $\mathcal{V}$ to keep a copy of it in a designated register $\mathtt{d}$ in its working space $\mathtt{w}$. Technically, $\mathcal{V}$ applies a CNOT gate on the register containing the decision qubit v_K and the register $|0\rangle_{\mathtt{d}}$ where the former register is the control register, and then $\mathcal{V}$ sends the $|v_K\rangle_{\mathtt{m}}$ to the prover as the last message. Note that this effectively collapses the decision qubit $|v_K\rangle_{\mathtt{d}}$, but this is fine since the predicate $\mathsf{Acc}(\cdot)$ anyway performs computational measurement on the decision qubit (see Def. 2).

Malicious Verifiers. For a formal definition of black-box quantum zero-knowledge, we give a model of quantum malicious verifiers against quantum interactive protocols. A malicious verifier $\widetilde{\mathcal{V}}$ is specified by a sequence of unitary $\widetilde{V}_\lambda$ over the internal register $\mathtt{v}$ and the message register $\mathtt{m}$ (whose details are explained later) and an auxiliary input ρ_λ indexed by the security parameter $\lambda \in \mathbb{N}$. We say that $\widetilde{\mathcal{V}}$ is non-uniform QPT if the sizes of $\widetilde{V}_\lambda$ and ρ_λ are polynomial in λ. In the rest of this paper, λ is always is always set to be the length of the statement x to be proven, and thus we omit λ for notation simplicity.

Its internal register $\mathtt{v}$ consists of the instance register $\mathtt{ins}$, auxiliary input register $\mathtt{aux}$, and verifier's working register $\mathtt{w}$. Part of the working space $\mathtt{w}$ is designated as the output register $\mathtt{out}$. $\widetilde{\mathcal{V}}$ interacts with an honest prover $\mathcal{P}$ on a common input x and $\mathcal{P}$'s private input $w \in \mathcal{R}_{\mathcal{L}}(x)$ in the following manner:

1. Register $\mathtt{ins}$ is initialized to x, $\mathtt{aux}$ is initialized to ρ, and $\mathtt{w}$ and $\mathtt{m}$ are initialized to be $\mathbf{0}$ of sufficient length.
2. $\mathcal{P}$ (with private input w) and $\widetilde{\mathcal{V}}$ run the K-round protocol as follows. On round $k \in [K]$,
   (a) $\mathcal{P}$ sends message p_k by loading it in register $\mathtt{m}$. (Physically, one can think that register $\mathtt{m}$ was in the hand of $\mathcal{P}$ at the beginning of this round; $\mathcal{P}$ loads message p_k in $\mathtt{m}$ and sends $\mathtt{m}$ to $\widetilde{\mathcal{V}}$.)
   (b) $\widetilde{\mathcal{V}}$ applies the unitary $\widetilde{V}$, which generates $\widetilde{\mathcal{V}}$'s response v_k and loads it in register $\mathtt{m}$. (Physically, one can think that register $\mathtt{m}$ is then sent to $\mathcal{P}$; This is how we model "$\widetilde{\mathcal{V}}$ sends message v_k to $\mathcal{P}$".)
3. At the end of the execution, $\widetilde{\mathcal{V}}$ outputs the state in register $\mathtt{out}$ as her final output.

We denote by $\langle \mathcal{P}(w), \widetilde{\mathcal{V}}(\rho)\rangle(x)$ the above execution and by $\rho_{\mathsf{out}} \leftarrow \mathsf{OUT}_{\widetilde{\mathcal{V}}}\langle \mathcal{P}(w), \widetilde{\mathcal{V}}(\rho)\rangle(x)$ the final output ρ_{out} of $\widetilde{\mathcal{V}}$, which is a quantum state over $\mathtt{out}$.

Black-Box Simulator. We now describe formally how the simulator $\mathcal{S}$ interacts with a malicious verifier $\widetilde{\mathcal{V}}$ defined above. We note that there are multiple ways to do so and we choose the following one without loss of generality, which is particularly suitable for our proof of impossibility.

A quantum black-box simulator $\mathcal{S}$ is modeled as a quantum oracle Turing machine (e.g., see [BBBV97]). We say that $\mathcal{S}$ is expected-QPT (resp. strict-QPT) if the expected (resp. maximum) number of steps is polynomial in the input length, counting an oracle access as a unit step.

Note that we focus on black-box simulation. Thus, $\mathcal{S}$ can only query $\widetilde{\mathcal{V}}$ via her unitary $\widetilde{V}$ and its inverse $\widetilde{V}^\dagger$, but does not get to see the internal register registers of $\widetilde{\mathcal{V}}$. Also, recall that $\mathtt{m}$ is used to exchange messages as explained above. Therefore, for simulation, $\mathtt{m}$ is considered as being held by $\mathcal{S}$.

In more detail, a black-box simulator $\mathcal{S}$ is defined by a "local" unitary S. It is granted oracle access to $\widetilde{V}$ and $\widetilde{V}^\dagger$. It makes use of the following registers (in addition to $\widetilde{\mathcal{V}}$'s internal registers that she could only access through $\widetilde{V}$ or $\widetilde{V}^\dagger$)

- Register $\mathtt{ins}$ initialized to $|x\rangle_{\mathtt{ins}}$.
- Register $\mathtt{m}$ to exchange messages with the malicious verifier $\widetilde{\mathcal{V}}$;
- Register $\mathtt{s}$, which is her "local" working space;
- A query-type register $\mathtt{u}$ in a 2-dimension Hilbert space spanned by the basis $\{|\downarrow\rangle_{\mathtt{u}}, |\uparrow\rangle_{\mathtt{u}}\}$ (explained shortly).

$\mathcal{S}$'s behavior is iterations of the following three steps:

1. Apply the local operator S on registers $\mathtt{m}$, $\mathtt{u}$, and $\mathtt{s}$;
2. Measure the $\mathtt{u}$ register to learn the type of the next query (see the next bullet);
3. Make the oracle query according to the measurement outcome from the last step—if the measurement outcome is $|\downarrow\rangle_{\mathtt{u}}$, it query the $\widetilde{V}$ oracle on $\mathtt{m}$; otherwise (i.e., the measurement outcome is $|\uparrow\rangle_{\mathtt{u}}$), it query the $\widetilde{V}^\dagger$ oracle on $\mathtt{m}$.

$\mathcal{S}$ keeps repeating the above three steps until she decides to stop. At that moment, the internal state of $\widetilde{\mathcal{V}}$ (i.e., contents of register $\mathtt{out}$) is defined to be the simulation output. Notation-wise, we denote the above procedure by $\rho_{\mathsf{out}} \leftarrow \mathcal{S}^{\widetilde{\mathcal{V}}(x;\rho)}(x)$ the final output ρ_{out} of $\widetilde{\mathcal{V}}$, where ρ_{out} the final state $\mathtt{out}$ (tracing out all other registers after the execution).

Black-Box Quantum ZK. With the above notations, we now present the definition of black-box quantum zero-knowledge in Def. 3.

Definition 3 (Black-Box Quantum ZK Proofs/Arguments). *A black-box quantum zero-knowledge proof (resp. argument) for an* **NP** *language $\mathcal{L}$ is a quantum interactive proof (resp. argument) $\langle \mathcal{P}, \mathcal{V}\rangle$ for $\mathcal{L}$ (as per Def. 2) that additionally satisfies the following property:*

- **Black-Box Quantum Zero-Knowledge.** *There exists an expected-QPT simulator $\mathcal{S}$ such that for each non-uniform QPT malicious verifier $\widetilde{\mathcal{V}}$ with an auxiliary input ρ, it holds that*

$$\left\{\mathsf{OUT}_{\widetilde{\mathcal{V}}}\langle \mathcal{P}(w), \widetilde{\mathcal{V}}(\rho)\rangle(x)\right\}_{\lambda\in\mathbb{N}, x\in\mathcal{L}\cap\{0,1\}^\lambda, w\in\mathcal{R}_{\mathcal{L}}(x)} \overset{c}{\approx} \left\{\mathcal{S}^{\widetilde{\mathcal{V}}(x;\rho)}(x)\right\}_{\lambda\in\mathbb{N}, x\in\mathcal{L}\cap\{0,1\}^\lambda, w\in\mathcal{R}_{\mathcal{L}}(x)}.$$

5.2 Quantum ZK in the Non-Programmable EPR Model

In this section, we define a new model for quantum zero-knowledge, which we refer to as the *Non-Programmable EPR* (NPE for short) model. Before we dive into the formal definitions, we first provide an intuitive explanation of this model and how it helps us establish our impossibility results for the (standard) QZK (as per Sec. 5.1).

The NPE model is an intermediate model where $\mathcal{P}$ and $\mathcal{V}$ pre-share EPR pairs issued by some trusted third party. These EPR pairs can be used to "de-quantize" the prover's messages. We will show that given a fully quantum BBZK protocol, we can transform it into a BBZK protocol in the NPE model with the same round complexity. This allows us to focus on the NPE model when establishing our impossibility result.

We next proceed to the formal definitions.

Non-Programmable EPR Model. As explained above, QZK in this non-programmable EPR model is almost identical to the standard QZK we defined in Sec. 5.1, except for how the prover's message is transmitted. Therefore, instead of presenting all the details (which include repeated descriptions of the same steps as we did for the standard QZK model), we choose to describe the non-programmable EPR model by only highlighting its differences from the standard QZK model.

The non-programmable EPR model (NPE) ZK is identical to the QZK protocol as we defined in Sec. 5.1, except for the following differences

- We assume there is a trusted party Trust who prepares polynomially many EPR pairs
$$(e_1^{(1)}, e_2^{(1)}), \ldots, (e_1^{(n)}, e_2^{(n)}),$$
where $n(\lambda)$ is a fixed polynomial specified by the protocol.
- At the beginning of the protocol, the first halves of these EPR pairs $(e_1^{(1)}, \ldots, e_1^{(n)})$ are given to the honest prover $\mathcal{P}$ as a part of its input; the second halves of these EPR pairs $(e_2^{(1)}, \ldots, e_2^{(n)})$ are given to the honest verifier $\mathcal{V}$ as a part of its input.
- The computation models for the honest prover $\mathcal{P}$ and honest verifier $\mathcal{V}$ are identical to that in the standard QZK defined in Sec. 5.1, except that $\mathcal{P}$ and $\mathcal{V}$ now can make use of the EPR pairs in their input during the execution, as long as the computation is QPT.
We note that we can continue to use the same interface between $\mathcal{P}$ and $\mathcal{V}$ as described in Sec. 5.1.2. That is, though they additionally hold EPR pairs, the protocol still has the generic form that $\mathcal{P}$ and $\mathcal{V}$ exchanges (possibly) quantum messages $(p_1, v_1, \ldots, p_K, v_K)$. The only modifications would be in the specific descriptions of their local unitary, which are anyway left unspecified in the generic description provided in Sec. 5.1.
Similarly, we denote by
$$\langle \mathcal{P}\big(w, (e_1^{(1)}, \ldots, e_1^{(n)})\big), \mathcal{V}(e_2^{(1)}, \ldots, e_2^{(n)})\rangle(x)$$
the above execution, and by
$$\rho \leftarrow \mathsf{OUT}_{\mathcal{V}}\langle \mathcal{P}\big(w, (e_1^{(1)}, \ldots, e_1^{(n)})\big), \mathcal{V}(e_2^{(1)}, \ldots, e_2^{(n)})\rangle(x)$$
the final output of $\mathcal{V}$, where ρ is a single-qubit state over a special register $\mathtt{d}$ of $\mathcal{V}$. We re-use the same definition of $\mathsf{Acc}(\cdot)$ as in Sec. 5.1.
- The modeling of the malicious verifier $\widetilde{\mathcal{V}}$ and the simulator $\mathcal{S}$ also remains the same as in Sec. 5.1.2, except that $\mathcal{S}$ now gets the $(e_1^{(1)}, \ldots, e_1^{(n)})$ as additional input (similar as the honest prover). We emphasize the following points regarding the simulator $\mathcal{S}$.

- $\mathcal{S}$ does not get to choose the EPR pairs. That is, the EPR pairs are always sampled by the trusted party. $\mathcal{S}$ gets $(e_1^{(1)},\ldots,e_1^{(n)})$ as input, and the other halves $(e_2^{(1)},\ldots,e_2^{(n)})$ are given to the (potentially malicious) verifier $\widetilde{\mathcal{V}}$. The latter is considered as being stored in $\widetilde{\mathcal{V}}$'s internal registers, so $\mathcal{S}$ cannot see or modify these shares hold by $\widetilde{\mathcal{V}}$ directly (unless through oracle access to $\widetilde{\mathcal{V}}$'s unitary $\widetilde{V}$ and its inverse $\widetilde{V}^\dagger$).
- We note that we can continue to use the same interface between $\mathcal{S}$ and $\widetilde{\mathcal{V}}$ as described in Sec. 5.1.2. In terms of notation, we can consider $(e_1^{(1)},\ldots,e_1^{(n)})$ to be stored in a designated region of $\mathcal{S}$'s working space $\mathtt{s}$, and $(e_2^{(1)},\ldots,e_2^{(n)})$ to be stored in a corresponding region of $\widetilde{\mathcal{V}}$'s working space $\mathtt{w}$. Consequently, the generic description of how $\mathcal{S}$ interacts with $\widetilde{\mathcal{V}}$ remains unchanged—the only modifications would be in the specific descriptions of $\mathcal{S}$'s local unitary S and $\widetilde{\mathcal{V}}$'s unitary $\widetilde{V}$, which are anyway left unspecified in the generic description provided in Sec. 5.1.2.

  Similarly, we use $\rho_{\mathsf{out}} \leftarrow \mathcal{S}^{\widetilde{\mathcal{V}}\left(x,(e_2^{(1)},\ldots,e_2^{(n)});\rho\right)}\left(x,(e_1^{(1)},\ldots,e_1^{(n)})\right)$ to denote the final output ρ_{out} of $\widetilde{\mathcal{V}}$, where ρ_{out} the final state on $\widetilde{\mathcal{V}}$'s internal register $\mathtt{out}$ (tracing out all other registers after the execution).

With the above notation, we now present the formal definition for QZK in the NPE model in Def. 4.

Definition 4 (Black-Box QZK in the NPE Model). *Let $n(\lambda)$ be a fixed polynomial of the security parameter λ. A black-box quantum zero-knowledge proof (resp. argument) for an* **NP** *language $\mathcal{L}$, in the n-pair non-programmable EPR model, is a quantum interactive protocol between a QPT prover $\mathcal{P}$ and a QPT verifier $\mathcal{V}$ that satisfy the following requirements:*

– **Completeness.** *For any $x \in \mathcal{L}\cap\{0,1\}^\lambda$, any $w \in \mathcal{R}_{\mathcal{L}}(x)$, and any n EPR pairs $\{e_1^{(i)},e_2^{(i)}\}_{i\in[n]}$ generated by* Trust*, it holds that*

$$\Pr\left[\mathsf{Acc}(\rho)=1 \ : \ \rho \leftarrow \mathsf{OUT}_{\mathcal{V}}\langle\mathcal{P}\big(w,(e_1^{(1)},\ldots,e_1^{(n)})\big),\mathcal{V}(e_2^{(1)},\ldots,e_2^{(n)})\rangle(x)\right] \geq 1-\mathsf{negl}(\lambda).$$

– **Statistical/Computational Soundness.** *The protocol is statistically (resp. computationally)* sound *if for any unbounded (resp. non-uniform QPT) cheating prover $\mathcal{P}^*$ and any $x \in \{0,1\}^\lambda\backslash\mathcal{L}$, and any n EPR pairs $\{e_1^{(i)},e_2^{(i)}\}_{i\in[n]}$ generated by* Trust*, it holds that*

$$\Pr\left[\mathsf{Acc}(\rho)=1 \ : \ \rho \leftarrow \mathsf{OUT}_{\mathcal{V}}\langle\mathcal{P}^*(e_1^{(1)},\ldots,e_1^{(n)}),\mathcal{V}(e_2^{(1)},\ldots,e_2^{(n)})\rangle(x)\right] = \mathsf{negl}(\lambda).$$

The protocol is dubbed a proof (resp. argument) if the soundness property is statistical (resp. computational).

– **Black-Box Quantum Zero-Knowledge.** *There exists an expected-QPT simulator $\mathcal{S}$ such that for any non-uniform QPT malicious verifier $\widetilde{\mathcal{V}}$ with an auxiliary input ρ and any n EPR pairs $\{e_1^{(i)},e_2^{(i)}\}_{i\in[n]}$ generated by* Trust*, it holds that*

$$\begin{aligned}&\left\{\mathsf{OUT}_{\widetilde{\mathcal{V}}}\langle\mathcal{P}\big(w,(e_1^{(1)},\ldots,e_1^{(n)})\big),\widetilde{\mathcal{V}}\big((e_2^{(1)},\ldots,e_2^{(n)}),\rho\big)\rangle(x)\right\}_{\lambda\in\mathbb{N},x\in\mathcal{L}\cap\{0,1\}^\lambda,w\in\mathcal{R}_{\mathcal{L}}(x)}\\ \overset{c}{\approx}\ &\left\{\mathcal{S}^{\widetilde{\mathcal{V}}\left(x,(e_2^{(1)},\ldots,e_2^{(n)});\rho\right)}\left(x,(e_1^{(1)},\ldots,e_1^{(n)})\right)\right\}_{\lambda\in\mathbb{N},x\in\mathcal{L}\cap\{0,1\}^\lambda,w\in\mathcal{R}_{\mathcal{L}}(x)}.\end{aligned}$$

Henceforth, we use n-NPE-BBQZK as the abbreviation for black-box quantum zero-knowledge arguments in the n-pair non-programmable EPR model.

5.3 Standard BBQZK Implies NPE-BBQZK

In this part, we show in lem. 7 that if (standard) BBQZK exists, then BBQZK in the NPE model also exists. Moreover, the prover in the latter protocol only needs to send *classical* messages. Looking forward, lem. 7 allows us to switch our attention to the NPE model—to prove that there does not exist constant-round black-box quantum zero-knowledge, we only need to show that such protocols do not exist in the NPE model.

Lemma 7. *For a number K, assume that there exists a K-round black-box QZK proof (resp. argument) as per Def. 3, where we assume w.l.o.g. that each messages exchanged in this protocol are of the same length of $\ell(\lambda)$ qubits. Then, there exists a K-round black-box QZK proof (resp. argument) in the $(K \cdot \ell)$-pair NPE model as per Def. 4, where all the messages sent by the prover is classical.*

Proof of Lem. 7. Consider a K-round BBQZK protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ as defined in Def. 3. We assume without loss of generality that all the p_k's and v_k's message are of the same length $\ell(\lambda)$ qubits, where $\ell(\lambda)$ is some polynomial of the security parameter λ. To establish Lem. 7, we show a compiler that converts $\langle \mathcal{P}, \mathcal{V} \rangle$ to a new protocol $\langle \mathcal{P}, \mathcal{V} \rangle_{\text{NPE}}$ in the $(K \cdot \ell)$-pair NPE model, with all prover's messages being classical.

At a high level, the compiler simply runs the original $\langle \mathcal{P}, \mathcal{V} \rangle$ with only one difference: all the prover's messages p_k's are sent to the verifier by quantum teleportation. Note that each p_k consists of ℓ qubits. Thus, $(K \cdot \ell)$ EPR pairs suffice for teleporting all prover's messages. We present the formal description of the compiler in Prot. 1, and then prove that the resulting protocol $\langle \mathcal{P}, \mathcal{V} \rangle_{\text{NPE}}$ does satisfies the requirements in Lem. 7.

Protocol 1: NPE-BBQZK Protocol $\langle \mathcal{P}, \mathcal{V} \rangle_{\text{NPE}}$

Let $n(\lambda) = K \cdot \ell(\lambda)$. Then, a K-round QZK in the NPE model is identical to the original QZK protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ except for the following differences:

- Before the protocol starts, the trusted party prepares n EPR pairs

$$(e_1^{(1)}, e_2^{(1)}), \ldots, (e_1^{(n)}, e_2^{(n)}).$$

  The first halves of these EPR pairs $(e_1^{(1)}, \ldots, e_1^{(n)})$ are given to the honest prover $\mathcal{P}$ as a part of its input (in addition to its original input x and w); the second halves of these EPR pairs $(e_2^{(1)}, \ldots, e_2^{(n)})$ are given to the honest verifier $\mathcal{V}$ as a part of its input (in addition to its original input x).

- $\mathcal{P}$ ane $\mathcal{V}$ then behaves as in the original execution of $\langle \mathcal{P}(w), \mathcal{V} \rangle(x)$, except for the following differences:
  - When $\mathcal{P}$ wants to send message p_k ($\forall k \in [K]$), it does not send p_k directly. Instead, it uses quantum teleportation to transmit it, consuming the EPR shares $(e_1^{((k-1)\ell+1)}, \ldots, e_1^{(k\ell)})$. In more detail, $\mathcal{P}$ performs the teleportation measurements over the ℓ-qubit message p_k and the ℓ-qubit EPR shares $(e_1^{((k-1)\ell+1)}, \ldots, e_1^{(k\ell)})$. This leads to ℓ pairs of (classical) teleportation keys $\{(a_k^{(i)}, b_k^{(i)})\}_{i \in [\ell]}$. The honest prover $\mathcal{P}$ then sends $\widetilde{p}_k = \{(a_k^{(i)}, b_k^{(i)})\}_{i \in [\ell]}$ to the verifier.
  - When the honest verifier $\mathcal{V}$ receives $\widetilde{p}_k = \{(a_k^{(i)}, b_k^{(i)})\}_{i \in [\ell]}$, it first recover the quantum message p_k using the teleportation keys contained in $\widetilde{p}_k$ and the corresponding halves of the EPR pairs $(e_2^{((k-1)\ell+1)}, \ldots, e_2^{(k\ell)})$. Then, $\mathcal{V}$ behaves in the same manner as in the original protocol to generate message v_k and sends it to the prover.

- At the end of the execution, the verifier output whatever the original $\mathcal{V}$ would output in the original $\langle \mathcal{P}(w), \mathcal{V} \rangle(x)$ protocol.

From the description of Prot. 1, it is easy to see that the protocol $\langle \mathcal{P}, \mathcal{V} \rangle_{\text{NPE}}$ has K rounds, consumes $(K \cdot \ell)$ EPR pairs, and all the prover's messages $\{\widetilde{p}_k\}_{k\in[K]}$ are classical. Also, the completeness follows direct from the completeness of the original $\langle \mathcal{P}, \mathcal{V} \rangle$ protocol and the correctness guarantee of quantum teleportation.

To prove soundness, we assume that there is a malicious QPT prover $\widetilde{\mathcal{P}}_{\text{NPE}}$ that breaks the soundness of $\langle \mathcal{P}, \mathcal{V} \rangle_{\text{NPE}}$ shown in Prot. 1. Then, we can construct another malicious QPT prover $\widetilde{\mathcal{P}}$ that breaks the soundness of the original protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ as follows. Machine $\widetilde{\mathcal{P}}$ samples the EPR pairs by herself, and emulates machine $\widetilde{\mathcal{P}}_{\text{NPE}}$ internally, on input $(e_1^{(1)}, \ldots, e_1^{(n)})$. When the internal $\widetilde{\mathcal{P}}_{\text{NPE}}$ sends the message $\widetilde{p}_k$, $\widetilde{\mathcal{P}}$ first recover the original quantum message p_k using $(e_2^{(1)}, \ldots, e_2^{(n)})$ (note that this is possible because $\widetilde{\mathcal{P}}$ sampled the EPR pairs), and then forward p_k to the external honest verifier $\mathcal{V}$; When the external $\mathcal{V}$ sends message v_k, machine $\widetilde{\mathcal{P}}$ simply forward it to the internal $\widetilde{\mathcal{P}}_{\text{NPE}}$. It is straightforward that if $\widetilde{\mathcal{P}}_{\text{NPE}}$ breaks the soundness of $\langle \mathcal{P}, \mathcal{V} \rangle_{\text{NPE}}$, then the constructed $\widetilde{\mathcal{P}}$ breaks the soundness of the original protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ as well.

The zero-Knowledge property follows from a similar argument as for soundness above. In more detail, the black-box ZK simulator $\mathcal{S}_{\text{NPE}}$ works as follows:

1. Given a malicious verifier $\widetilde{\mathcal{V}}_{\text{NPE}}$ for the protocol Prot. 1, it first constructs a malicious verifier $\widetilde{\mathcal{V}}$ for the original protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ as follows:
   - Machine $\widetilde{\mathcal{V}}$ samples the EPR pairs by herself, and emulates machine $\widetilde{\mathcal{V}}_{\text{NPE}}$ internally, on input $(e_2^{(1)}, \ldots, e_2^{(n)})$. When the external $\widetilde{\mathcal{P}}$ sends the message p_k $(\forall k \in [K])$, $\widetilde{\mathcal{V}}$ performs teleportation measurements on p_k and the EPR shares $(e_1^{((k-1)\ell+1)}, \ldots, e_1^{(k\ell)})$, which yield the classical keys $\widetilde{p}_k = \{(a_k^{(i)}, b_k^{(i)})\}_{i\in[\ell]}$. $\widetilde{\mathcal{V}}$ forward $\widetilde{p}_k$ to the internal $\widetilde{\mathcal{V}}_{\text{NPE}}$. When the internal $\widetilde{\mathcal{V}}_{\text{NPE}}$ sends v_k, $\widetilde{\mathcal{V}}$ forwards it to the external $\mathcal{P}$. At the end of the execution, $\widetilde{\mathcal{V}}$ outputs whatever the internal $\widetilde{\mathcal{V}}_{\text{NPE}}$ outputs.
2. $\mathcal{S}_{\text{NPE}}$ then invokes the simulator $\mathcal{S}$ of the original protocol $\widetilde{\mathcal{V}}$, providing the $\widetilde{\mathcal{V}}$ constructed above as the oracle required by $\mathcal{S}$. Finally, $\mathcal{S}_{\text{NPE}}$ outputs whatever $\mathcal{S}^{\widetilde{\mathcal{V}}}$ outputs.

We first argue that the above $\mathcal{S}_{\text{NPE}}$ only makes black-box access to $\widetilde{\mathcal{V}}_{\text{NPE}}$. This can be seen by noticing that the construction of $\widetilde{\mathcal{V}}$ requires only black-box access to $\widetilde{\mathcal{V}}_{\text{NPE}}$, and that the simulator $\mathcal{S}$ of the original protocol makes only black-box to the $\widetilde{\mathcal{V}}$ constructed in black-box from $\widetilde{\mathcal{V}}_{\text{NPE}}$.

As for the indistinguishability of the simulation, first notice that in a real execution between the original honest prover $\mathcal{P}(x, w)$ and the above constructed $\widetilde{\mathcal{V}}$, the final output of $\widetilde{\mathcal{V}}$ is identically distributed as the final output of $\widetilde{\mathcal{V}}_{\text{NPE}}$ obtained from an execution with the honest Prot. 1 prover $\mathcal{P}_{\text{NPE}}(x, w)$. This is because $\widetilde{\mathcal{V}}$ perfectly emulates the execution for the internal $\widetilde{\mathcal{V}}_{\text{NPE}}$, and eventually outputs whatever the latter outputs. Then, by the ZK property of the original protocol, the output of $\mathcal{S}^{\widetilde{\mathcal{V}}}$ is computationally indistinguishable with $\widetilde{\mathcal{V}}$'s output in the real execution with $\mathcal{P}(x, w)$, which, as we just argued, is identical to $\widetilde{\mathcal{V}}_{\text{NPE}}$'s output from the real execution with $\mathcal{P}_{\text{NPE}}(x, w)$. Also notice that $\mathcal{S}_{\text{NPE}}$ is defined to output whatever $\mathcal{S}^{\widetilde{\mathcal{V}}}$ outputs. Therefore, the final output of $\mathcal{S}_{\text{NPE}}$ is computationally indistinguishable with $\widetilde{\mathcal{V}}_{\text{NPE}}$'s output from the real execution with $\mathcal{P}_{\text{NPE}}(x, w)$.

This completes the proof of Lem. 7.

□

6 Impossibility of Constant-Round Black-Box Quantum Zero-Knowledge

In this section, we prove the following Thm. 2.

Theorem 2. *For a language $\mathcal{L}$, if there exists a constant-round black-box quantum zero-knowledge argument (as per Def. 3), then it holds that $\mathcal{L} \in \mathbf{BQP}$.*

It follows from Lem. 7 that to prove Thm. 2, it suffices to establish the following Thm. 3. The rest of this section is devoted to proving Thm. 3.

Theorem 3. *For a language $\mathcal{L}$, if there exists a constant-round black-box quantum zero-knowledge argument in the non-programmable EPR model (as per Def. 4) with prover's messages being classical, then it holds that $\mathcal{L} \in \mathbf{BQP}$.*

Proof Structure. At a high level, our proof for Thm. 3 follows the paradigm established in [BL02, CCLY21]. Given a fixed language $\mathcal{L}$, we begin by assuming the existence of a constant-round BBQZK protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ in the non-programmable EPR model, where the prover's messages are classical. We then proceed to construct a malicious verifier $\widetilde{\mathcal{V}}$, who behaves identically to the honest $\mathcal{V}$ except for the fact that $\widetilde{\mathcal{V}}$ aborts at each round with a carefully chosen probability of $1-\varepsilon$. To elaborate further, $\widetilde{\mathcal{V}}$ employs a random oracle H_ε that outputs 1 with probability ε; at each round, $\widetilde{\mathcal{V}}$ continues the execution only if H_ε outputs 1 on input the prover's messages $\widetilde{\mathcal{V}}$ received thus far.

Since the protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ is zero-knowledge, there must exist a black-box simulator $\mathcal{S}$ that, when given oracle access to $\widetilde{\mathcal{V}}$, is able to simulate the output of $\widetilde{\mathcal{V}}$ in the real execution. We then demonstrate that the execution $\mathcal{S}^{\widetilde{\mathcal{V}}}$ can be converted into a bounded-error quantum polynomial-time decider $\mathcal{B}$ for the language $\mathcal{L}$, thereby establishing that $\mathcal{L} \in \mathbf{BQP}$.

We make two important remarks regarding the above procedure:

1. **On the Efficiency of $\widetilde{\mathcal{V}}$.** Note that the final (QPT) decider $\mathcal{B}$ is constructed from $\mathcal{S}^{\widetilde{\mathcal{V}}}$. However, the $\widetilde{\mathcal{V}}$ described above is not QPT yet, because the random oracle H_ε does not have a polynomial-size description.
   We can address this issue in exactly the same manner as [CCLY21]. Specifically, the oracle H_ε can be replaced with a $2q$-wise independent hash function, where q is the number of queries made by the original $\mathcal{S}^{\widetilde{\mathcal{V}}}$ machine to H_ε. It then follows from Lem. 1 that the constructed **BQP** decider B will function just as effectively. In the subsequent discussion, we will primarily focus on the scenario where $\widetilde{\mathcal{V}}$ uses H_ε. For the replacement of H_ε with a $2q$-wise independent hash, we refer to Sec. 10 for more details.
2. **On the Expected QPT Running Time of $\mathcal{S}$.** Notice that the simulator in Def. 3 runs in *expected* QPT time. Consequently, our impossibility results shown in Thm. 2 (and Thm. 3) aim to rule out expected QPT simulation. However, the above paradigm does not work if $\mathcal{S}$ runs in *expected* QPT, as it would result in the decider $\mathcal{B}$ running in expected QPT, which does not meet the efficiency requirement (i.e., strictly QPT) for being a valid **BQP** decider.
   Once again, we resolve this issue in the same manner as [CCLY21]. Specifically, we first utilize the above paradigm to demonstrate the impossibility for *strictly* QPT simulation only. Then, we extend this impossibility to expected QPT simulation using the identical argument as presented in [CCLY21, Section 3.3]. For a formal treatment of this issue, we refer to Sec. 10.

We now proceed to the formal proof of Thm. 3. Given the complexity of this proof and its fulfillment across several sections, we provide an overview of the contents of the related sections:

– In Sec. 6.1, we define the malicious $\widetilde{\mathcal{V}}$; in Sec. 6.2, we define the **BQP** decider $\mathcal{B}$. It is important to note that the $\widetilde{\mathcal{V}}$ and $\mathcal{B}$ defined so far are subject to the efficiency issues mentioned in Items 1 and 2. However, we will proceed with the proof while ignoring these issues, as they will be addressed in Sec. 10 as explained above.

– We next prove that $\mathcal{B}$ is indeed a valid **BQP** decider. This requires us to demonstrate:

  - **Completeness:** On input $x \in \mathcal{L}$, $\mathcal{B}(x)$ accepts with $1/\mathsf{poly}(\lambda)$ probability. This is established in Sec. 6.3.
  - **Soundness:** On input $x \notin \mathcal{L}$, $\mathcal{B}(x)$ accepts with $\mathsf{negl}(\lambda)$ probability. This proof is intricate. In Sec. 6.4, we complete the proof for soundness assuming the important technical lemma Lem. 11. Then, we conclude the proof of Lem. 11 in Sec. 7 to 9.

– In Sec. 10, we address the efficiency issues mentioned in Items 1 and 2. This completes the proof of Thm. 3.

6.1 A Malicious Verifier

We start by considering a language $\mathcal{L}$. For this $\mathcal{L}$, assume that there exists a K-round black-box quantum zero-knowledge argument $\langle \mathcal{P}, \mathcal{V} \rangle$ in the non-programmable EPR model as per Def. 4, where all the prover's message are classical.

Let us recall that structure of the protocol as defined in Sec. 5.2. The protocol consists $2K$ messages

$$(p_1, v_1, p_2, v_2, \ldots, p_K, v_K).$$

That is, the protocol has K rounds in total. In round $k \in [K]$, $\mathcal{P}$ sends message p_k and $\mathcal{V}$ responds with v_k. Since we assumed that the prover's messages are classical, all the messages $(p_1, \ldots, p_k)$ are classical strings; but the messages $(v_1, \ldots, v_K)$ could be quantum. W.l.o.g., we assume that all these massages are of the same length $\ell(\lambda)$, which is a polynomial in λ.

We now proceed to define the particular malicious verifier $\widetilde{\mathcal{V}}$.

6.1.1 $\widetilde{\mathcal{V}}$'s registers

$\widetilde{\mathcal{V}}$ starts by initializing the following registers:

$$|x\rangle_{\mathtt{ins}} |0\rangle_{\mathtt{gc}} |0\rangle_{\mathtt{lc}} |\mathbf{0}\rangle_{\mathtt{p}_1} \ldots |\mathbf{0}\rangle_{\mathtt{p}_K} |\mathbf{0}\rangle_{\mathtt{v}_1} \ldots |\mathbf{0}\rangle_{\mathtt{v}_K} |\mathbf{0}\rangle_{\mathtt{m}} |\bot\rangle_{\mathtt{t}_1} \ldots |\bot\rangle_{\mathtt{t}_K} |e_2^{(1)}, \ldots, e_2^{(n)}, \mathbf{0}\rangle_{\mathtt{w}} |H\rangle_{\mathtt{aux}}, \tag{6.1}$$

where the meaning of each register is explained below:

– Register `ins` stores the statement (or instance) x of language $\mathcal{L}$, which is a common input to both the verifier and the prover.

– Register `gc` is called the *global counter* register and register `lc` is called the *local counter* register. The functionality of these two registers will become clear later when we describe the behavior of $\widetilde{\mathcal{V}}$ in Sec. 6.1.2.

– Register $\mathtt{p}_1 \ldots \mathtt{p}_K$ (resp. $\mathtt{v}_1 \ldots \mathtt{v}_K$) are to store all the prover's messages (resp. verifier's messages). The exact meaning of these registers will become clear later when we describe the behavior of $\widetilde{\mathcal{V}}$ in Sec. 6.1.2.

– Register `m` is used to exchange messages between the prover and the verifier. Again, the exact meaning of `m` will become clear later when we describe the behavior of $\widetilde{\mathcal{V}}$ in Sec. 6.1.2.

– Register $\mathtt{t}_1 \ldots \mathtt{t}_k$ will be used when the verifier decides to abort in a particular rounds. See Sec. 6.1.2 for how these registers are used.

– Register $\mathtt{w}$ is $\widetilde{\mathcal{V}}$'s working space. It contains suffices 0 states that will be used as ancilla qubits during the computation. Also, recall that we are in the non-programmable EPR model, which means that the verifier gets the second halves of n EPR pairs $(e_2^{(1)}, \ldots, e_2^{(n)})$. We choose to put then in the $\mathtt{w}$ register as well. We remark that the subsequent proof does not need to refer to these EPR shares explicitly.

– Register $\mathtt{aux}$ stores the truth table of a function $H : \mathcal{M}^{\leq K} \to \{0,1\}$, where $\mathcal{M} = \{0,1\}^\ell$ (note that each p_k is a classical string of length ℓ). This H comes from certain distributions that we will specify later in our proof. Here, we remark that this H is the only inefficient part of $\widetilde{\mathcal{V}}$, and we will eventually replace it with a $2q$-wise independent hash function as mentioned in Item 1.

6.1.2 $\widetilde{V}$'s Unitary.

In this part, we define the malicious verifier $\widetilde{\mathcal{V}}$ by specifying her unitary $\widetilde{V}$. We present $\widetilde{V}$ in two equivalent manners. The first description (shown in Algo. 6.1) is formulated to help the reader understand the behavior of $\widetilde{\mathcal{V}}$ at the operational level. However, it is initially unclear whether the description in Algo. 6.1 can be implemented as a unitary circuit. Particularly, the $\widetilde{V}$ in Algo. 6.1, at first glance, appears to utilize controlled gates on the function H. This poses a problem because in later parts of the proof, we need to regard H as a quantum oracle. (It is not known whether one can implement controlled gates on H in the model where H is provided as an oracle.)

Therefore, in Algo. 6.2, we provide a functionally equivalent description of Algo. 6.1. It is evident from Algo. 6.2 that the $\widetilde{V}$ defined in Algo. 6.1 can indeed be implemented as a unitary that utilizes H as a quantum oracle, without the need for any controlled gates on it.

In subsequent parts of the proofs, we primarily utilize Algo. 6.1 as it illustrates the operational meaning more effectively. However, we may also reference Algo. 6.2 in certain instances where it aids in clarifying matters (e.g., in the proof of Lem. 14 and the proof of lem. 10).

Operationally Clear Description. We start by defining two unitaries that works on the global counter and local counter registers defined above.

Note that both $\mathtt{gc}$ and $\mathtt{lc}$ registers are initialized to 0. We set $C = 2^\lambda$,[15] and define the unitary U_{gc} as the following mapping

$$U_{gc} : |i\rangle_{\mathtt{gc}} \mapsto |i+1 \mod C\rangle_{\mathtt{gc}}.$$

We also define the unitary to increase the local counter U_{lc} as follows:

$$U_{lc} : |i\rangle_{\mathtt{lc}} \mapsto |i+1 \mod (K+1)\rangle_{\mathtt{lc}}.$$

We remark that modulus $(k+1)$ is good enough for us, because we will show later that the value of the local counter will stay in the set $\{0\} \cup [K]$.

With the above notation in hand, we now describe $\widetilde{\mathcal{V}}$'s unitary $\widetilde{V}$ in Algo. 6.1.

Algorithm 6.1: Unitary $\widetilde{V}$ for the Malicious Verifier $\widetilde{\mathcal{V}}$

Helper Unitaries. Before we described $\widetilde{V}$, we first define some helper unitaries. For each $k \in [K]$:

– A_k**: swap** $\mathtt{p_k}$ **and** $\mathtt{m}$**.** This unitary swaps the contents of $\mathtt{p_k}$ and $\mathtt{m}$.

– B_k**: apply** V**'s unitary.** This unitary perform the following operation in superposition[a]:

[15] We remark that it suffices to set C to any super-polynomial function on λ. We choose 2^λ only for concreteness.

- If $H(p_1, \ldots, p_k) = 1$, then apply the honest verifier's unitary V to generate message v_k. We remark that the generated v_k is now stored in register $\mathtt{v_k}$.
- If $H(p_1, \ldots, p_k) = 0$, then do nothing;

– C_k**: deliver** v_k**.** This unitary perform the following operation in superposition:

- If $H(p_1, \ldots, p_k) = 1$, then swap the contents of registers $\mathtt{m}$ and $\mathtt{v_k}$.
- If $H(p_1, \ldots, p_k) = 0$, then swap the contents of registers $\mathtt{m}$ and $\mathtt{t_k}$.

– C'_k**: swap** $\mathtt{m}$ **and** $\mathtt{t_k}$**.** This unitary is the swap operator between registers $\mathtt{m}$ and $\mathtt{t_k}$.

– D_k**: increase local counter.** This unitary perform the following operation in superposition:

- If $H(p_1, \ldots, p_k) = 1$, then apply the unitary U_{lc} to increase the local counter by 1.
- If $H(p_1, \ldots, p_k) = 0$, then do nothing.

$\widetilde{\mathcal{V}}$**'s Unitary** $\widetilde{V}$**.** On each query, it compares the global counter value k with the local counter value j and behaves accordingly. In particular:

Case $k = j$**:** It first applies the unitary U_{gc} to increase the global counter by 1, and then behaves according to the (increased) global counter value $k + 1$. In particular:

– If $k + 1 \in [K]$, then it applies $D_{k+1}C_{k+1}B_{k+1}A_{k+1}$ as defined above.

– Otherwise (i.e., $k + 1 \notin [K]$), it does nothing (i.e., applies the identity operator).

Case $(k \neq j)$**:** It first applies the unitary U_{gc} to increase the global counter by 1, and then behaves according to the (increased) global counter value $k + 1$. In particular:

– If $k + 1 \in [K]$, then it applies $C'_{k+1}A_{k+1}$ as defined above.

– Otherwise (i.e., $k + 1 \notin [K]$), it does nothing (i.e., applies the identity operator).

$\widetilde{\mathcal{V}}$**'s Output:** It output all the registers. Notation-wise, we write the final output as $\big(\mathbf{p} = (p_1, \ldots, p_K), \rho\big)$, where $p_1, \ldots, p_K$ is the contents in register $\mathtt{p_1} \ldots \mathtt{p_K}$ and ρ is the state over the remaining registers at halt.

[a] Henceforth, when we refer to $(p_1, \ldots, p_k)$, we mean the values stored in registers $\mathtt{p_1} \ldots \mathtt{p_k}$.

Notation. We will make use of the following notations. We write $\widetilde{\mathcal{V}}^H$ to refer to the malicious verifier $\widetilde{\mathcal{V}}$ when the function in register $\mathtt{aux}$ is instantiated by H (see Expression (6.1)). We use $\langle \mathcal{P}(w), \widetilde{\mathcal{V}}^H \rangle(x)$ to denote the execution of the protocol between $\widetilde{\mathcal{V}}^H$ and the honest prover $\mathcal{P}$, where the comment input is x and the honest prover holds a witness w as its private input. We use $(\mathbf{p}, \rho) \leftarrow \mathsf{OUT}_{\widetilde{\mathcal{V}}} \langle \mathcal{P}(w), \widetilde{\mathcal{V}}^H \rangle(x)$ to denote the final output of $\widetilde{\mathcal{V}}$.

Remark 8 (Omitting EPR Shares). Notice that we are currently in the non-programmable EPR model, and thus $\mathcal{P}$ (resp. $\widetilde{\mathcal{V}}$) also takes the EPR shares $(e_1^{(1)}, \ldots, e_1^{(n)})$ (resp. $(e_2^{(1)}, \ldots, e_2^{(n)})$) as a part of its input. Henceforth, we omit these EPR pairs in our notation for succinctness, e.g., in $\langle \mathcal{P}(w), \widetilde{\mathcal{V}}^H \rangle(x)$. This is fine because our proof never needs to explicitly refer to these EPR shares.

On Implementing $\widetilde{V}$ as an Unitary. As discussed at the beginning of Sec. 6.1.2 , we now show in Algo. 6.2 that the $\widetilde{V}$ defined in Algo. 6.1 can indeed be implemented as a unitary with quantum-oracle access to H.

Algorithm 6.2: Unitary Implementation of $\widetilde{V}$

At each round $k \in [K]$, it behaves in the following three steps:

Step 1: It applies U_{gc} to increase the global counter from $k-1$ to k. (Recall that right before round k starts, the global counter is $k-1$.)

Step 2: It behaves according to the current (i.e., already increased) global counter value k:

1. If $k \in [K]$, then it applies the unitary A_k as defined in Algo. 6.1;
2. Otherwise (i.e., $k \notin [K]$), it does not do anything (i.e., it applies the identity operator).

Step 3: It queries the (quantum) oracle H to learn the value $H(p_1, \ldots, p_{j+1})$ and store this value in a temporary register `tmp`. We note that j is the value of the current local counter and $(p_1, \ldots, p_{j+1})$ are the values stored in registers $\mathtt{p}_1 \ldots \mathtt{p}_{j+1}$. Also note that in later proofs, the local counter register may contain a superposition of values; that is why we need *quantum* oracle access to H.

Step 4: It behaves according to the current (i.e., already increased) global counter value k:

1. If $k \in [K]$, then it behaves by comparing the local counter value j with $k-1$ (recall again that $k-1$ is the global counter value at the beginning of round k, before the U_{gc} at the very beginning is applied):
   (a) If $(k-1) = j$, then it applies the $D_k C_k B_k$ defined in Algo. 6.1. Note that these operators make use of the value $H(p_1, \ldots, p_k)$ as the control (qu)bit. But this is exactly the value stored in register `tmp` defined in **Step 3** (note that in this case $j+1$ is exactly k). Thus, $D_k C_k B_k$ can be implemented using `tmp` as the control register.
   (b) Otherwise (i.e., $(k-1) \neq j$), it applies the C'_k defined in Algo. 6.1.
2. Otherwise (i.e., $k \notin [K]$), it does not do anything (i.e., it applies the identity operator)

The equivalence between Algo. 6.1 and Algo. 6.2 can be easily seen by comparing their respective descriptions. Moreover, it is evident from the description of Algo. 6.2 that it can be implemented as a unitary circuit with quantum oracle access to H — particularly, the query to H is elevated to **Step 3** and is not controlled by any register.

6.1.3 Understanding $\widetilde{\mathcal{V}}$ with Two examples

To gain a better understanding of the unitary $\widetilde{V}$ defined in Algo. 6.1, let us consider two examples where we instantiate the function H differently. We recommend that the reader not skip these examples, as we will introduce important notations that will be utilized in the subsequent proofs as well.

The First Example. In this example, consider a function H that always output 1 on all input. For this such an H, it is not hard to see that the execution $\langle \mathcal{P}(w), \widetilde{\mathcal{V}}^H \rangle(x)$ is effectively identical to the $\langle \mathcal{P}(w), \mathcal{V} \rangle(x)$ (i.e., the real protocol between honest prover and verifier)—in each round k, only the branch corresponding to $H(p_1, \ldots, p_k) = 1$ would happen; if one track the execution, it

is easy to see that only the work performed by B_k actually matters (while A_k, C_k, and D_k are there to deliver messages or maintain the counters), and B_k is nothing but emulating the honest verifier $\mathcal{V}$ (in the $H(p_1,\ldots,p_k)=1$ branch). Therefore, the execution in this case is indeed a perfect emulation of the real $\langle \mathcal{P}(w),\mathcal{V}\rangle(x)$. In particular, $\widetilde{\mathcal{V}}^H$'s final output will be accepted by Acc with the same probability $1-\mathsf{negl}(\lambda)$ as in the honest execution (see the **Completeness** requirement in Def. 4).

Let us now turn to the local counter register. In this example, it is also straightforward that the local counter register will contain the a classical value K at the end of the execution $\langle \mathcal{P}(w),\widetilde{\mathcal{V}}^H\rangle(x)$. This is simply because the local counter will increase by 1 at the end of each round (due to the unitary D_k), and there are K rounds in total.

Moreover, we also know that the value $(p_1,\ldots,p_K)$ contained in the $\mathbf{p}$ part of the final output of $\widetilde{\mathcal{V}}^H$ will be accepted by H. In particular, it holds that $\overline{H}(\mathbf{p})=\mathbf{1}$, where $\mathbf{1}$ is the vector containing K repetition of value 1 and $\overline{H}$ is the vector-valued version of H as defined in the following Def. 5.

Definition 5 (Vector-Valued Function). *Let $\mathcal{M}:=\{0,1\}^\ell$. Let H be a function from $\mathcal{M}^{\le K}$ to $\{0,1\}$. Then, we define the vector-version $\overline{H}$ as follows:*

$$\forall \mathbf{p}=(p_1,\ldots,p_K)\in\mathcal{M}^K,\quad \overline{H}(\mathbf{p}):=\big(H(p_1),H(p_1,p_2),\ldots,H(p_1,\ldots,p_K)\big).$$

The above observations for this example can be summarized as follows. For any $x\in\mathcal{L}$ and any $w\in\mathcal{R}_{\mathcal{L}}(x)$, it holds that

$$\begin{aligned}
&\Pr\Big[\mathsf{Pred}(\rho)=1\wedge\overline{H}(\mathbf{p})=\mathbf{1}\ :\ (\mathbf{p},\rho)\leftarrow\mathsf{OUT}_{\widetilde{\mathcal{V}}}\langle\mathcal{P}(w),\widetilde{\mathcal{V}}^H\rangle(x)\Big]\\
=\ &\Pr[\mathsf{Acc}(\rho)=1\ :\ \rho\leftarrow\mathsf{OUT}_{\mathcal{V}}\langle\mathcal{P}(w),\mathcal{V}\rangle(x)]\\
=\ &1-\mathsf{negl}(\lambda),
\end{aligned}$$

where the $\mathsf{Pred}(\cdot)$ is a quantum predicate defined in the following Def. 6.

Definition 6 (Predicate Pred). *We define $\mathsf{Pred}(\cdot)$ as a quantum predicate over the second part (i.e., the ρ) in the output of $\widetilde{\mathcal{V}}$. In particular, $\mathsf{Pred}(\rho)=1$ if and only if the following conditions hold:*

1. *Measure the local counter register $\mathtt{lc}$ in the computational basis and the measurement outcome is K. (Recall that the $\mathtt{lc}$ register is contained in ρ.)*
2. *Measure register $\mathtt{d}$ in the computational basis and the measurement outcome is 1. (Recall that the $\mathtt{d}$ register is a designated register in $\mathtt{w}$ containing the verifier's final decision qubit (see also Rmk. 7). It is contained in ρ because $\mathtt{w}$ is so.)*

The Second Example. A more interesting example is when H is sampled from a family of random functions that output 1 with probability ε. Formally, we define the following family of functions.

Definition 7 (Family of ε-Random Functions). *Let $\mathcal{M}:=\{0,1\}^\ell$. For a real number $\varepsilon\in[0,1]$ and a positive integer K, let $\mathcal{H}_{\varepsilon,K}$ be a distribution over $H_{\varepsilon,K}:\mathcal{M}^{\le K}\to\{0,1\}$ such that we have*

$$\Pr[H_{\varepsilon,K}(p_1,\ldots,p_k)=1]=\varepsilon$$

independently for each $(p_1,\ldots,p_k)\in\mathcal{M}^{\le K}$. (In the sequel, the value K is always fixed to the number of rounds. Thus, we omit the K from the subscript of $H_{\varepsilon,K}$ and simply write is as H_ε.)

Consider a H_ε that sampled from the $\mathcal{H}_\varepsilon$ defined in Def. 7. Let us compare the the execution $\langle \mathcal{P}(w), \widetilde{\mathcal{V}}^{H_\varepsilon} \rangle(x)$ and the honest execution $\langle \mathcal{P}(w), \mathcal{V} \rangle(x)$. Due to a similar argument as we did in the previous example with the all-accepting function H, it is easy to see that the execution $\langle \mathcal{P}(w), \widetilde{\mathcal{V}}^{H_\varepsilon} \rangle(x)$, when conditioned on the fact that $H_\varepsilon(p_1, \ldots, p_k) = 1$ for all $k \in [K]$, is identical to the real execution $\langle \mathcal{P}(w), \mathcal{V} \rangle(x)$ between the honest parties. In this case (i.e., the conditioned fact happens), all the observations we made in the first example (regarding the accepting probability, the local counter, the predicate Pred) remain valid.

On the other hand, it follows from Def. 7 that the event "$H_\varepsilon(p_1, \ldots, p_k) = 1$ for all $k \in [K]$" happen with probability exactly ε^K over the randomly sampling of H_ε.

Therefore, the following holds: For any $x \in \mathcal{L}$ and any $w \in \mathcal{R}_{\mathcal{L}}(x)$, it holds that

$$\begin{aligned}
&\Pr\left[\mathsf{Pred}(\rho) = 1 \wedge \overline{H}_\varepsilon(\mathbf{p}) = \mathbf{1} \ : \ \begin{array}{l} H_\varepsilon \leftarrow \mathcal{H}_\varepsilon \\ (\mathbf{p}, \rho) \leftarrow \mathsf{OUT}_{\widetilde{\mathcal{V}}} \langle \mathcal{P}(w), \widetilde{\mathcal{V}}^{H_\varepsilon} \rangle(x) \end{array}\right] \\
&\geq \varepsilon^K \cdot \Pr[\mathsf{Acc}(\rho) = 1 \ : \ \rho \leftarrow \mathsf{OUT}_{\mathcal{V}} \langle \mathcal{P}(w), \mathcal{V} \rangle(x)] \\
&= \varepsilon^K \cdot \big(1 - \mathsf{negl}(\lambda)\big) \\
&= \varepsilon^K - \mathsf{negl}(\lambda), \qquad (6.2)
\end{aligned}$$

where $\overline{H}_\varepsilon$ is the vector-valued version of H_ε as defined in Def. 5 and $\mathsf{Pred}(\cdot)$ is the quantum predicate defined in Def. 6.

6.1.4 Interaction between $\mathcal{S}$ and $\widetilde{\mathcal{V}}$

Recall from Sec. 5.1 and 5.2 that the simulator $\mathcal{S}$ is granted oracle access to $\widetilde{V}$ and $\widetilde{V}^\dagger$. She does not get to see the internal registers of $\widetilde{\mathcal{V}}$ (except for register $\mathtt{m}$). She has a local operator S, and after each oracle query, her behavior consists of applying S and then measuring a special $\mathtt{u}$ register to determine the type of the next query.

As for notations, for any function H, we use $\mathsf{SIM}^H(x)$ to refer to the execution of $\mathcal{S}^{\widetilde{\mathcal{V}}(x)}(x)$ where the function in $\widetilde{\mathcal{V}}$'s $\mathtt{aux}$ register is instantiated by H. (Similar as in Rmk. 8, we omit the EPR shares from our notation.) Also, recall from Sec. 5.1 that the output of the simulator is defined to be the output of the malicious verifier at the end of simulation. For the particular $\widetilde{\mathcal{V}}$ defined in Algo. 6.1, the output of the simulation can be written as $(\mathbf{p}, \rho) \leftarrow \mathsf{SIM}^H(x)$, where $(\mathbf{p}, \rho)$ has the same meaning as defined toward the end of Algo. 6.1.

With the above notations, let us know make three simply but useful claims.

Claim 4. *For any $x \in \mathcal{L}$, it holds that*

$$\Pr\left[\mathsf{Pred}(\rho) = 1 \wedge \overline{H}_\varepsilon(\mathbf{p}) = \mathbf{1} \ : \ \begin{array}{l} H_\varepsilon \leftarrow \mathcal{H}_\varepsilon \\ (\mathbf{p}, \rho) \leftarrow \mathsf{SIM}^{H_\varepsilon}(x) \end{array}\right] \geq \frac{\varepsilon^K}{4} - \mathsf{negl}(\lambda).$$

Proof of Claim 4. This claim simply follows from the ZK property of the protocol and Eq. (6.2).

In more detail,

$$\begin{aligned}
&\Pr\left[\mathsf{Pred}(\rho) = 1 \wedge \overline{H}_\varepsilon(\mathbf{p}) = \mathbf{1} \ : \ \begin{array}{l} H_\varepsilon \leftarrow \mathcal{H}_\varepsilon \\ (\mathbf{p}, \rho) \leftarrow \mathsf{SIM}^{H_\varepsilon}(x) \end{array}\right] \\
\geq\ &\Pr\left[\mathsf{Pred}(\rho) = 1 \wedge \overline{H}_\varepsilon(\mathbf{p}) = \mathbf{1} \ : \ \begin{array}{l} H_\varepsilon \leftarrow \mathcal{H}_\varepsilon \\ (\mathbf{p}, \rho) \leftarrow \mathsf{OUT}_{\widetilde{\mathcal{V}}} \langle \mathcal{P}(w), \widetilde{\mathcal{V}}^{H_\varepsilon} \rangle(x) \end{array}\right] - \mathsf{negl}(\lambda) \qquad (6.3) \\
\geq\ &\varepsilon^K - \mathsf{negl}(\lambda), \qquad (6.4)
\end{aligned}$$

where Eq. (6.3) follows from the ZK property of the $\langle \mathcal{P}, \mathcal{V} \rangle$ protocol, and Inequality (6.4) follows form Eq. (6.2).

Then, Claim 4 simply follows from Inequality (6.4) and the fact that $\varepsilon^K > \frac{\varepsilon^K}{4}$.

□

Remark 9 (On Tightness of Claim 4). One may wonder why we state Claim 4 with a lower bound of $\frac{\varepsilon^K}{4} - \mathsf{negl}(\lambda)$, given that the proof of Claim 4 already achieves a tighter lower bound of $\varepsilon^K - \mathsf{negl}(\lambda)$ (i.e., Inequality (6.4)). This is because Claim 4 serves two purposes in the later part of our proof:

1. First, Claim 4 will be used when establishing the completeness property of the **BQP** decider (in Lem. 8). For this purpose, we can indeed use Claim 4 with the tighter lower bound of $\varepsilon^K - \mathsf{negl}(\lambda)$.
2. Second, Claim 4 will also be used when extending our impossibility results from strictly QPT simulation to *expected* QPT simulation in Sec. 10. For this purpose, the tighter lower bound of $\varepsilon^K - \mathsf{negl}(\lambda)$ does not suffice, and we have to use the bound of $\frac{\varepsilon^K}{4} - \mathsf{negl}(\lambda)$.

Therefore, we choose to state Claim 4 with the lower bound of $\frac{\varepsilon^K}{4} - \mathsf{negl}(\lambda)$.

Claim 5 (Classical Global Counter). *For any function H, throughout the execution of $\mathsf{SIM}^H(x)$, the global counter register will always contain a classical value.*

Proof of Claim 5. Note that $\mathcal{S}$ can only modify the content in the global counter register through invoking her oracle $\widetilde{V}$ or $\widetilde{V}^\dagger$. Then, Claim 5 can be easily seen from the definition of $\widetilde{V}$ in Algo. 6.1—every invocation of $\widetilde{V}$ (resp. $\widetilde{V}^\dagger$) will trigger U_{gc} (resp. $U_{gc}^\dagger$) exactly once, increasing (resp. decreasing) the global counter by 1. Thus, the global counter will always contain a classical value, decoherent from other registers.

This completes the proof of Claim 5.

□

Claim 6. *For any function H, the execution $\mathsf{SIM}^H(x)$ can be seen as an oracle-aided algorithm where H plays the role of the oracle. If the simulator $\mathcal{S}$ makes q quieres to her oracle $\widetilde{V}$ and $\widetilde{V}^\dagger$ in total, then the oracle H will be queries at most $2Kq$ times.*

Proof of Claim 6. First, note that $\mathsf{SIM}^H(x)$ can indeed be seen as an oracle-aided algorithm where H plays the role of the oracle. This follows from the description of $\widetilde{\mathcal{V}}$, who only makes use of the I/O behavior of H.

As for the number of queries, it suffices to show that each query of $\mathcal{S}$ (to her oracle $\widetilde{V}$ or $\widetilde{V}^\dagger$) will invoke at most $2K$ queries to H. This is true by definition of $\widetilde{V}$ Algo. 6.1: From the descritpion there, unitaries B_k, C_k, and D_k all makes H queries. However, we remark that this can be done by query the H once and store the output of H in a seperate register that can be used (as a control register) by B_k, C_k, and D_k. In the manner, each $\widetilde{V}$ or $\widetilde{V}^\dagger$ query only invoke 2 quries to H (note that we need one more query to uncompute each query to H). Then, Claim 6 follows because $2q \leq 2Kq$ for all positive interger K.

This completes the proof of Claim 6.

□

Remark 10 (On Tightness of Claim 6). The above proof of Claim 6 indeed establishes a tighter upper bound on the number of queries — it shows that SIM^H can make at most $2q$ queries to H. One may wonder why we choose to claim the upper bound as $2Kq$ in the statement of Claim 6. Indeed, our Claim 6 can be treated as the analogue of **Observation 3** on Page 20 of [CCLY21],

where the upper bound is $2Kq$. We choose to use the same bound because this allows us to reuse the same parameter settings (and certain calculations) in [CCLY21].

One may also wonder why we can save the multiplicative factor K in this bound while [CCLY21] cannot in their **Observation 3**. This is due to a difference in the behavior of the malicious verifier at the last round. In our case, the malicious verifier $\widetilde{\mathcal{V}}$'s behavior at the last round is syntactically the same as other rounds. In particular, she only queries H once as we argued in the above proof of Claim 6 (plus one more query for uncomputing). In contrast, the malicious verifier in [CCLY21] behaves differently at the last round. Specifically, at the last round, the [CCLY21] verifier will invoke H for K times to learn $H(m_1, \ldots, m_i)$ for all $i \in [K]$ and check if all these values are 1 (see the top of Page 20 of [CCLY21]). We remark that our $\widetilde{\mathcal{V}}$ does not need to perform such checks at the last round. This is because (looking ahead) our proof will perform a fine-grained analysis of the measure-and-reprogram game with respect to SIM^H to establish certain properties of the prover's messages $(p_1, \ldots, p_K)$, which will effectively achieve the same effect as the checks performed by the [CCLY21] verifier at the last round.

6.2 BQP Decider

In this section, we define the **BQP** decider for the language $\mathcal{L}$.

First, let us take a closer look at the execution of SIM^H. As we established in Claim 6, SIM^H can be treated as an oracle-aided execution, where H plays the role of the oracle. Our first step toward constructing the **BQP** decider $\mathcal{B}$ is to put SIM^H into the format of a measure-and-reprogram (MnR) game, as discussed in Sec. 3.2.

MnR Game with ZK Simulator. We first define the MnR game w.r.t. the simulator $\mathcal{S}$ and $\widetilde{\mathcal{V}}(x)$ defined in previous sections. This game is simply an instantiation of MnR game defined in Game 3.1 with the following notation:

1. We treat the execution of the simulator (with oracle access to $\widetilde{V}$ and $\widetilde{V}^\dagger$) as an oracle machine SIM^H that has quantum oracle access to a random function $H : \mathcal{X}^{\leq K} \to \mathcal{Y}$, where $\mathcal{X} = \{0,1\}^\ell$ and $\mathcal{Y} = \{0,1\}$. Let $\mathbf{y} = (y_1, \ldots, y_K) \in \mathcal{Y}^K$.
2. Recall that the output of SIM^H is defined to be the output $\widetilde{\mathcal{V}}$ at halt, which is splitted as $(\mathbf{p}, \rho)$ (see also Algo. 6.1).
3. Recall from Claim 6 that SIM as an oracle machine makes at most $2Kq$ queries to H in total.

Since this game will be particularly important for the sequel sections, we choose to present it in full detail in Game 6.1.

Game 6.1: Measure-and-Reprogram Game $\mathcal{S}^{\widetilde{\mathcal{V}}}[H, \mathbf{y}](x)$

It works as follows

1. For each $i \in [K]$, uniformly pick $(j_i, b_i) \in ([2Kq] \times \{0,1\}) \cup \{(\bot, \bot)\}$ conditioned on that there exists at most one $i \in [K]$ such that $j_i = j^*$ for all $j^* \in [2Kq]$.
2. Let s denote the number of j_i's in $\{j_i\}_{i\in[K]}$ that are not $\bot$. We re-label the indices i for pairs $\{(j_i, b_i)\}_{i\in[K]}$ so that $j_1 < j_2 < \ldots < j_s$ and $j_{s+1} = j_{s+2} = \ldots = j_K = \bot$.
3. Run the oracle machine $\mathsf{SIM}(x)$ with oracle $\mathcal{O}$, which is initialized to be a quantumly-accessible classical oracle that computes H, and when $\mathsf{SIM}(x)$ makes its j-th query, the oracle is simulated as follows:

(a) If $j = j_i$ for some $i \in [K]$, measure this query to obtain $\mathbf{p}_i = (p_{i,1}, \ldots, p_{i,z_i})$ where $z_i \in [K]$ is determined by the measurement outcome, and then behaves according to the value b_i as follows:

i. If $b_i = 0$: First reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, \mathbf{p}_i, y_i)$, and then answer the j_i-th query using the reprogrammed oracle.

ii. If $b_i = 1$: First answer the j_i-th query using the oracle before the reprogramming, and then reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, \mathbf{p}_i, y_i)$.

(b) Otherwise (i.e., $j \neq j_i \ \forall i \in [s]$), answer the j-th query just using the oracle $\mathcal{O}$ without any measurement or reprogramming.

4. Let $(\mathbf{p}^* = (p_1^*, \ldots, p_K^*), \rho_{\mathsf{out}})$ be the output of $\mathcal{S}^{\widetilde{\mathcal{V}}}$ at halt (as per Item 2).
5. For all $i \in \{s+1, s+2, \ldots, K\}$, set $\mathbf{p}_i = \mathbf{p}_i^*$ where $\mathbf{p}_i^* := (p_1^*, \ldots, p_i^*)$.
6. **Output:** The output of this game is defined as follows
   - If it holds for all $i \in [K]$ that $\mathbf{p}_i$ is a prefix of $\mathbf{p}_K$, then output $(\mathbf{p}_K, \rho_{\mathsf{out}})$;
   - Otherwise, output $(\bot, \bot)$.

Decider for BQP. With the above notation in hand, we proceed to define the **BQP** decider $\mathcal{B}$. Roughly speaking, $\mathcal{B}$ on input x simply runs the MnR game $\mathcal{S}^{\widetilde{\mathcal{V}}}[H, \mathbf{y}](x)$ as defined in Game 6.1 by setting H to an all-zero function and setting $\mathbf{y}$ to an all-1 vector. It accepts (resp. rejects) if the verifier $\widetilde{\mathcal{V}}$ accepts (resp. rejects) at the end of the MnR game.

The formal description of $\mathcal{B}$ is presented in Algo. 6.3.

Algorithm 6.3: BQP Decider $\mathcal{B}(x)$

On input a classical string x, machine $\mathcal{B}(x)$ works as follows:

1. **(Parameter Setting.)** Let $\mathbf{1} := (1, \ldots, 1)$ containing K copies of 1. Let $\mathcal{M} := \{0,1\}^\ell$ and $H_0 : \mathcal{M}^{\leq K} \to \{0,1\}$ be the zero-function, i.e., $H_0(p_1, \ldots, p_i) = 0$ for all $(p_1, \ldots, p_i) \in \mathcal{M}^{\leq K}$.
2. **(MnR Game with $\mathcal{S}^{\widetilde{V}}$.)** Execute $(\mathbf{p}, \rho) \leftarrow \mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$ as per Game 6.1. That is, $\mathcal{B}(x)$ executes Game 6.1 with $H := H_0$ and $\mathbf{y} := (1, \ldots, 1)$ and denotes the output of this procedure as $(\mathbf{p}, \rho)$.
3. **(Output.)** It outputs 1 if and only if $\mathsf{Pred}(\rho) = 1$, where $\mathsf{Pred}(\rho)$ is the predicate defined in Def. 6.

Next, we argue that $\mathcal{B}$ is a valid **BQP** decider. Toward that, it suffices to establish the following Lem. 8 and 9. We present the proof for Lem. 8 in Sec. 6.3. The proof for Lem. 9 is the most technically involved part of this work. It will be covered in Sec. 6.4 and Sec. 7 to 9.

Lemma 8 (Completeness). *For all $\mathcal{L} \in$ **BQP** and all $x \in \mathcal{L} \cap \{0,1\}^\lambda$, it holds for the $\mathcal{B}$ defined in Algo. 6.3 that*

$$\Pr[\mathcal{B}(x) = 1] \geq \frac{1}{8 \cdot (4Kq+1)^{2K}} - \mathsf{negl}(\lambda),$$

where q is the number of oracle queries made by the $\mathcal{S}$ (to her oracle $\widetilde{V}$ or $\widetilde{V}^\dagger$) during the execution of $\mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$ in Step 2 of $\mathcal{B}(x)$ (i.e., Algo. 6.3).

Lemma 9 (Soundness). *For all $\mathcal{L} \in \mathbf{BQP}$ and all $x \in \{0,1\}^\lambda \setminus \mathcal{L}$, it holds for the $\mathcal{B}$ defined in Algo. 6.3 that*

$$\Pr[\mathcal{B}(x) = 1] = \mathsf{negl}(\lambda).$$

6.3 Proof of Completeness

In this subsection, we prove Lem. 8.

We first set some parameters that will be used in this proof. Recall that q denotes the number of oracle queries made by the $\mathcal{S}$ (to her oracle $\widetilde{V}$ or $\widetilde{V}^\dagger$) during the execution of $\mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$ in Step 2 of $\mathcal{B}(x)$ (i.e., Algo. 6.3). For this q, we define

$$\varepsilon_q := \frac{1}{256K^2q^2(4Kq+1)^{2K}}. \tag{6.5}$$

In the sequel, let $\varepsilon \leq \varepsilon_q$ be an arbitrary noticeable function in λ.

Notation. We first recall some notations that have already been defined previously. They will be used in the sequel derivation:

- We will make use of the family of ε-random functions $\mathcal{H}_\varepsilon$ defined in Def. 7. Note that this ε is the particular noticeable $\varepsilon \leq \varepsilon_q$ we fixed above. When a probability is taken over the random sampling of $H_\varepsilon \leftarrow \mathcal{H}_\varepsilon$, we will not explicitly include this expression; instead, we simply write H_ε under the "Pr" symbol to indicate the sampling procedure.
- For a function H_ε, we will make use of its vector-valued version $\overline{H}_\varepsilon$ as defined in Def. 5.
- We will make use of $\mathcal{S}^{\widetilde{V}}[H_\varepsilon, \mathbf{1}](x)$, which is the MnR game Game 6.1 instantiated with $H := H_\varepsilon$ and $\mathbf{y} := \mathbf{1}$ containing K repetitions of 1's.
- When we write $\boldsymbol{\beta}$ in under the "Pr" symbol, it means we sample $\boldsymbol{\beta}$ from a distribution D_ε over $\{0,1\}^K$, where each bit of β takes the value 1 with probability ε independently.
- For a function H, we will make use of the symbol SIM^H, which is the execution of $\mathcal{S}$ with oracle access to $\widetilde{\mathcal{V}}$ whose random functions is instantiated by H. We treat the $\mathcal{S}^{\widetilde{\mathcal{V}}}$ as an oracle-aided machine with H playing the role of the oracle (as explained in Claim 6).
- For $\mathbf{x} = (x_1, \ldots, x_K) \in \{0,1\}^{K\ell}$, $\mathbf{y} = (y_1, \ldots, y_K) \in \{0,1\}^K$, and a function $H : (\{0,1\}^\ell)^{\leq K} \to \{0,1\}$, recall the function $H^{\mathbf{x},\mathbf{y}}$ defined in Lem. 3:

$$\forall i \in [K]\ \forall \mathbf{x}'_i = (x'_1, \ldots, x'_i), \quad H^{\mathbf{x},\mathbf{y}}(\mathbf{x}'_i) := \begin{cases} y_i & \text{if } \mathbf{x}'_i = (x_1, ..., x_i) \\ H(\mathbf{x}'_i) & \text{otherwise} \end{cases}.$$

With the above notations in hand, we now establish Lem. 8.

First, notice that

$$\Pr[\mathcal{B}(x) = 1] = \Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathcal{S}^{\widetilde{V}}[H_0, \mathbf{1}](x)\right] \tag{6.6}$$

$$\geq \Pr_{H_\varepsilon}[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathcal{S}^{\widetilde{V}}[H_\varepsilon, \mathbf{1}](x)] - 32K^2q^2\varepsilon \tag{6.7}$$

where Eq. (6.6) follows from the definition and notation in Algo. 6.3, and Inequality (6.7) follows from Lem. 2 and the fact that in the MnR game, there are $2Kq$ queries to the oracle H_0 (or H_ε) in total (see Claim 6).

Next, we derive a lower bound for the first term in the RHS of Inequality (6.7):

$$\begin{aligned}
&\Pr_{H_\varepsilon}[\mathsf{Pred}(\rho)=1 \ : \ (\mathbf{p},\rho)\leftarrow \mathcal{S}^{\widetilde{V}}[H_\varepsilon,\mathbf{1}](x)] \\
&= \sum_{\mathbf{p}^*\in\{0,1\}^{K\ell}\cup\{\perp\}} \Pr_{H_\varepsilon}[\mathbf{p}=\mathbf{p}^*\wedge\mathsf{Pred}(\rho)=1 \ : \ (\mathbf{p},\rho)\leftarrow \mathcal{S}^{\widetilde{V}}[H_\varepsilon,\mathbf{1}](x)] && (6.8)\\
&\geq \frac{1}{(4Kq+1)^{2K}}\cdot \sum_{\mathbf{p}^*\in\{0,1\}^{K\ell}\cup\{\perp\}} \Pr_{H_\varepsilon}[\mathbf{p}=\mathbf{p}^*\wedge\mathsf{Pred}(\rho)=1 \ : \ (\mathbf{p},\rho)\leftarrow \mathsf{SIM}^{H_\varepsilon^{\mathbf{p}^*,\mathbf{1}}}(x)] && (6.9)\\
&= \frac{\varepsilon^{-K}}{(4Kq+1)^{2K}}\cdot \sum_{\mathbf{p}^*\in\{0,1\}^{K\ell}\cup\{\perp\}} \Pr_{H_\varepsilon,\boldsymbol{\beta}}[\mathbf{p}=\mathbf{p}^*\wedge\mathsf{Pred}(\rho)=1\wedge\boldsymbol{\beta}=\mathbf{1} \ : \ (\mathbf{p},\rho)\leftarrow \mathsf{SIM}^{H_\varepsilon^{\mathbf{p}^*,\boldsymbol{\beta}}}(x)] \\
&= \frac{\varepsilon^{-K}}{(4Kq+1)^{2K}}\cdot \sum_{\mathbf{p}^*\in\{0,1\}^{K\ell}\cup\{\perp\}} \Pr_{H_\varepsilon}[\mathbf{p}=\mathbf{p}^*\wedge\mathsf{Pred}(\rho)=1\wedge\overline{H}_\varepsilon(\mathbf{p})=\mathbf{1} \ : \ (\mathbf{p},\rho)\leftarrow \mathsf{SIM}^{H_\varepsilon}(x)] \\
&= \frac{\varepsilon^{-K}}{(4Kq+1)^{2K}}\cdot \Pr_{H_\varepsilon}[\mathsf{Pred}(\rho)=1\wedge\overline{H}_\varepsilon(\mathbf{p})=\mathbf{1} \ : \ (\mathbf{p},\rho)\leftarrow \mathsf{SIM}^{H_\varepsilon}(x)] && (6.10)\\
&\geq \frac{\varepsilon^{-K}}{(4Kq+1)^{2K}}\cdot\left(\frac{\varepsilon^K}{4}-\mathsf{negl}(\lambda)\right), && (6.11)
\end{aligned}$$

where Eq. (6.8) follows from the Law of Total Probability, Eq. (6.9) follows from Lem. 3, Eq. (6.10) follows again from the Law of Total Probability, and Inequality (6.11) follows from Claim 4,

Finally, it holds that

$$\begin{aligned}
\Pr[\mathcal{B}(x)=1] &\geq \frac{\varepsilon^{-K}}{(4Kq+1)^{2K}}\cdot\left(\frac{\varepsilon^K}{4}-\mathsf{negl}(\lambda)\right)-32K^2q^2\varepsilon && (6.12)\\
&\geq \frac{1}{4\cdot(4Kq+1)^{2K}}-\mathsf{negl}(\lambda)-32K^2q^2\varepsilon \\
&\geq \frac{1}{8\cdot(4Kq+1)^{2K}}-\mathsf{negl}(\lambda), && (6.13)
\end{aligned}$$

where Inequality (6.12) follows from Inequalities (6.7) and (6.11), and Inequality (6.13) follows from our parameter setting that $\varepsilon\leq\varepsilon_q$ with the ε_q defined in Expression (6.5).

This completes the proof of Lem. 8.

6.4 Proof of Soundness

In this section, we prove Lem. 9.

At a high level, we assume for contradiction that Lem. 9 is false and show how to construct a malicious prover $\widetilde{\mathcal{P}}$ breaking the soundness of the original protocol $\langle\mathcal{P},\mathcal{V}\rangle$. Toward that, we first need to make some modifications to the machine $\mathcal{B}$ and the MnR game $\mathcal{S}^{\widetilde{\mathcal{V}}}$, making use of some particular properties of the $\widetilde{\mathcal{V}}$ we defined earlier. This is covered in Sec. 6.4.1 to 6.4.3. Next, we will present the malicious prover $\widetilde{\mathcal{P}}$ and prove that it can indeed break the soundness. This is done in Sec. 6.4.4.

6.4.1 Cleaning the MnR Game

6.4.1.1 Structure of z_i's

In the sequel, we focus on the MnR game $\mathcal{S}^{\widetilde{\mathcal{V}}}[H_0,\mathbf{1}](x)$, which is exactly the game Game 6.1 with H instantiated by the all-zero function H_0 and $\mathbf{y}$ instantiated by the vector $\mathbf{1}$ of K repetition of value 1.

Recall that the length z_i of the measurement outcome $(p_{i,1}, \ldots, p_{i,z_i})$ defined in Step 3 of the MnR game $\mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$ is determined by the measurement. In the following Lem. 10, we show that the length values z_i's will indeed exhibit a non-decreasing order. This property will be crucial for the later parts of the proof.

For notational convenience, let us first give a name to $\mathcal{S}$'s query to her oracle $\widetilde{V}$ (or $\widetilde{V}^\dagger$) that invokes the measurement in Step 3 of the MnR game $\mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$.

Definition 8 (Special Queries). *During the execution of $\mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$, note that the j_i-th query to $\mathcal{O}$ must be invoked by $\mathcal{S}$'s query to verifier $\widetilde{\mathcal{V}}$. We call this query of $\mathcal{S}$ to $\widetilde{\mathcal{V}}$ (which in turn invokes the j_i-th query to $\mathcal{O}$) as the* i-th special query *and denote it by* $\mathsf{sq}(i)$.

Lemma 10. *For the s defined in Step 2 and $\{z_t\}_{t\in[s]}$ defined in defined in Step 3 of game $\mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$, it holds that*[16]

$$\forall t \in [s],\ z_t \leq \max\{z_1, z_2 \ldots, z_{t-1}\} + 1. \quad (\textit{In particular, } z_1 = 1 \textit{ if } s \geq 1.)$$

Proof. In the following proof, (arbitrarily) fix a $t \in [s]$.

Consider the moment of the execution when $\mathcal{S}$ is about to make the $\mathsf{sq}(t)$ query (i.e., the query that will invoke the measurement of z_t). Let us denote the overall state, at this moment, across all the registers as ρ. Let us denote $z := \max\{z_1, z_2, \ldots, z_{t-1}\}$.

At this moment, we know by definition of Game 6.1 that the oracle H_0 has been re-programmed to output 1 for input vectors of length *at most* z. In particular, if we denote the oracle at this moment as $\widehat{H}$, then it outputs 0 on all vectors $(p_1, \ldots, p_{z+1})$ of length $z+1$.

Next, we claim that

$$\forall j \in \{z+1, z+2, \ldots, K\},\ \mathrm{Tr}[|j\rangle\langle j|_{\mathtt{lc}}\, \rho] = 0,$$

where $|j\rangle\langle j|_{\mathtt{lc}}$ is the projector that projects register $\mathtt{lc}$ to value j. That is, we claim that the $\mathtt{lc}$ register of state ρ does not have any weights on value $j \geq z+1$ (or equivalently, $j \leq z$). This is because:

1. by the definition of $\widetilde{V}$ (see Algo. 6.1), for $\mathtt{lc}$ to contain the value $z+1$ or greater, it is necessary that there exists some $(p_1, \ldots, p_{z+1})$ such that $H(p_1, \ldots, p_{z+1}) = 1$;
2. as we just argued, it holds for the current oracle $\widehat{H}$ that $\widehat{H}(p_1, \ldots, p_{z+1}) = 0$ for all possible $(p_1, \ldots, p_{z+1})$.

Now, let us example how the $\mathsf{sq}(t)$ query will be process using the description of $\widetilde{\mathcal{V}}$ in Algo. 6.2. In particular, according to **Step 3** of Algo. 6.2, the maximum length of each "branch" in the super-position query to H is upper bounded by $j+1$. Thus, the value z_t determined by measuring this super-position will not exceed $j+1$ either. It then follows from the above argument that $z_t \leq j+1 \leq z+1$, satisfying the requirement of Lem. 10.

Finally, we remark that the above argument implies $z_1 \leq 1$ as a corner case. On the other hand, if $s \geq 1$, then z_1 by definition lies in $[K]$. Thus, it follows that $z_1 = 1$.

This completes the proof of Lem. 10.

□

We next show in Corollary 2 a powerful corollary of Lem. 10.

[16] In Lem. 10 and its proof, for the corner case $t = 1$, the max is taken over $\emptyset$. We naturally define $\max(\emptyset) = 0$.

Corollary 2 (of Lem. 10). *For the game $\mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$, define the following event E_{Bad}*

$$E_{\mathsf{Bad}} \coloneqq (s \neq K) \vee (z_1 \neq 1) \vee (z_2 \neq 2) \vee \ldots \vee (z_K \neq K).$$

It then follows that

$$\Pr\Big[\mathsf{Pred}(\rho) = 1 \ \wedge \ E_{\mathsf{Bad}} \ : \ (\mathbf{p}, \rho) \leftarrow \mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)\Big] = 0.$$

Proof of Corollary 2. We first define another "bad" event $\widetilde{E}_{\mathsf{Bad}}$

$$\widetilde{E}_{\mathsf{Bad}} \coloneqq \big(\max\{z_1, \ldots, z_s\} < K\big).$$

Then, by Lem. 10, it is not hard to see that the event that "$\widetilde{E}_{\mathsf{Bad}}$ happens" implies the event that "E_{Bad} happens."

Therefore, to prove Corollary 2, it suffices to prove the following inequality:

$$\Pr\Big[\mathsf{Pred}(\rho) = 1 \ \wedge \ \widetilde{E}_{\mathsf{Bad}} \ : \ (\mathbf{p}, \rho) \leftarrow \mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)\Big] = 0 \tag{6.14}$$

The proof for Inequality (6.14) is very similar to that for Lem. 10. We include it below for the sake of completeness.

Let $z \coloneqq \max\{z_1, \ldots, z_s\}$. If event $\widetilde{E}_{\mathsf{Bad}}$ happens, we know that $z < K$. Now, consider the moment when the $\mathcal{S}$ halts. At this moment, we know by definition of game $\mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$ that the oracle H_0 has been re-programmed to output 1 for input vectors of length *at most* $z < K$. In particular, if we denote the oracle at this moment as $\widehat{H}$, then it outputs 0 on all vectors $(p_1, \ldots, p_K)$ of length K.

Next, we claim that

$$\mathrm{Tr}[|K\rangle\langle K|_{\mathtt{lc}}\, \rho] = 0, \tag{6.15}$$

where $|K\rangle\langle K|_{\mathtt{lc}}$ is the projector that projects register $\mathtt{lc}$ to value K. That is, we claim that the $\mathtt{lc}$ register of the final state ρ at $\mathcal{S}$'s halt does not have any weights on value K. This is because:

1. by the definition of $\widetilde{V}$ (see Algo. 6.1), for $\mathtt{lc}$ to contain the value K, it is necessary that there exists some $(p_1, \ldots, p_K)$ such that $H(p_1, \ldots, p_K) = 1$;
2. as we just argued, it holds for the current oracle $\widehat{H}$ that $\widehat{H}(p_1, \ldots, p_K) = 0$ for all possible $(p_1, \ldots, p_K)$.

In this case, Eq. (6.15) implies that $\mathsf{Pred}(\rho) = 0$ (recall the definition of Pred in Algo. 6.3).

This completes the proof of Corollary 2.

□

Corollary 2 is powerful in the sense that it allows us to "clean" the game $\mathcal{S}^{\widetilde{V}}[H_0, \mathbf{1}](x)$ in the following manner: during the execution of $\mathcal{S}^{\widetilde{V}}[H_0, \mathbf{1}](x)$, once the event E_{Bad} occurs, we can immediately abort the execution (possibly prematurely), without reducing the probability of $\mathsf{Pred}(\rho) = 1$. We formalize this observation as the following Game 6.2 (where we highlight its difference with $\mathcal{S}^{\widetilde{V}}[H_0, \mathbf{1}](x)$ in red color) and Corollary 3.

Game 6.2: Measure-and-Reprogram Game $\ddot{\mathcal{S}}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$
It works as follows

1. For each $i \in [K]$, uniformly pick $(j_i, b_i) \in ([2Kq] \times \{0,1\})$ ~~$\cup\{(\bot,\bot)\}$~~ conditioned on that there exists at most one $i \in [K]$ such that $j_i = j^*$ for all $j^* \in [2Kq]$.
2. ~~Let s denote the number of j_i's in $\{j_i\}_{i\in[k]}$ that are not $\bot$. We re-label the indices i for pairs $\{(j_i,b_i)\}_{i\in[K]}$ so that $j_1 < j_2 < \ldots < j_s$ and $j_{s+1} = j_{s+2} = \ldots = j_K = \bot$.~~
We re-label the indices i for pairs $\{(j_i, b_i)\}_{i\in[K]}$ so that $j_1 < j_2 < \ldots < j_K$.
3. Run the oracle machine $\mathcal{S}^{\widetilde{\mathcal{V}}}$ with oracle $\mathcal{O}$, which is initialized to be a quantumly-accessible classical oracle that computes H_0, and when $\mathcal{S}^{\widetilde{\mathcal{V}}}$ makes its j-th query, the oracle is simulated as follows:
   (a) If $j = j_i$ for some $i \in [K]$, measure this query to obtain $\mathbf{p}_i = (p_{i,1}, \ldots, p_{i,z_i})$ where $z_i \in [K]$ is determined by the measurement outcome. It halts immediately and outputs $(\bot, \bot)$ if any of the following events happen:
      i. $z_i \neq i$, *or*
      ii. $\mathbf{p}_{i-1}$ is not a prefix of $\mathbf{p}_i$.

      The $\mathsf{sq}(i)$ queries are defined in the same manner as in Def. 8.
      It then behaves according to the value b_i as follows:
      i. If $b_i = 0$: First reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, \mathbf{p}_i, 1)$, and then answer the j_i-th query using the reprogrammed oracle.
      ii. If $b_i = 1$: First answer the j_i-th query using the oracle before the reprogramming, and then reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, \mathbf{p}_i, 1)$.
   (b) Otherwise (i.e., $j \neq j_i \ \forall i \in [K]$), answer the j-th query just using the oracle $\mathcal{O}$ without any measurement or reprogramming.
4. Let $(\mathbf{p}^* = (p_1^*, \ldots, p_K^*), \rho_{\mathsf{out}})$ be the output of $\mathcal{S}^{\widetilde{\mathcal{V}}}$ at halt (as per Item 2).
5. ~~For all $i \in \{s+1, s+2, \ldots, K\}$, set $\mathbf{p}_i = \mathbf{p}_i^*$ where $\mathbf{p}_i^* := (p_1^*, \ldots, p_i^*)$.~~
6. **Output:** directly output $(\mathbf{p}_K, \rho_{\mathsf{out}})$ ~~The output of this game is defined as follows~~
   – ~~If it holds for all $i \in [K]$ that $\mathbf{p}_i$ is a prefix of $\mathbf{p}_K$, then output $(\mathbf{p}_K, \rho_{\mathsf{out}})$;~~
   – ~~Otherwise, output $(\bot, \bot)$.~~

Corollary 3 (of Corollary 2). *For the $\ddot{\mathcal{S}}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$ defined in Game 6.2 , it holds that*

$$\Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \ddot{\mathcal{S}}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)\right] \geq \Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)\right].$$

Proof of Corollary 3. We prove this corollary by considering the red-color changes in Game 6.2 one by one.

In Step 1, we simply prevent the value j_i from taking the value $\bot$. This follows immediately from Corollary 2—if there exists some $j_i = \bot$, then it must follow that $s \neq K$, which is a part of the E_{Bad} defined in Corollary 2. Thus, we can safely ignore this case without reducing the probability of $\mathsf{Pred}(\rho) = 1$.

The changes made in Steps 2 and 5 are direct consequences of the adjustment made in Step 1.

The introduction of Step 3(a)i can also be elucidated by Corollary 2. Specifically, the occurrence of the event $z_i \neq i$ is encompassed within the definition of E_{Bad} outlined in Corollary 2. Consequently, we can safely disregard this scenario without diminishing the probability of $\mathsf{Pred}(\rho) = 1$.

The inclusion of Step 3(a)ii essentially involves relocating the checks conducted in the original Step 6 to an earlier stage, specifically within the current Step 3(a)ii.

This finishes the proof of Corollary 3.

□

Remark 11 (The Type of $\mathsf{sq}(i)$ *Query).* We remark that in Game 6.2, the $\mathsf{sq}(i)$ query must be a $\widetilde{V}$ query (instead of being a $\widetilde{V}^\dagger$ query). This follows from a similar argument as we did in the proof of Lem. 10. In more detail, note that the $\mathsf{sq}(i)$ query by definition invokes the measurement of $\mathtt{p}_1 \dots \mathtt{p}_i$ in Game 6.2, if the game is not aborted in Step 3a. If the $\mathsf{sq}(i)$ query is invoked by a $\widetilde{V}^\dagger$ query, then by the definition of $\widetilde{V}$ (see Algo. 6.1), both the global counter and the local counter must be of the same value i *right before this query*. However, this is simply impossible—using the argument as we did in the proof of Lem. 10, the $\mathtt{lc}$ register has zero weights on value i before the $\mathsf{sq}(i)$ query is processed. Therefore, the $\mathsf{sq}(i)$ query must be a $\widetilde{V}$ query.

6.4.1.2 Simplification Assumptions

Henceforth, we make three assumptions regarding the behavior of the simulator $\mathcal{S}$ that would greatly simplify the remaining proof. We first present two assumptions which are easy to state and validate.

- **Assumption 1:** When the global counter is $|0\rangle_{\mathtt{gc}}$,[17] $\mathcal{S}$ will not make a $\widetilde{V}^\dagger$ query.
- **Assumption 2:** When the global counter is $|K\rangle_{\mathtt{gc}}$, $\mathcal{S}$ will not make a $\widetilde{V}$ query.

We argue that these assumptions will not decrease the probability of $\mathsf{Pred}(\rho) = 1$ for Game 6.2. To see that, consider that scenario where the $\mathcal{S}$ makes a $\widetilde{V}^\dagger$ (resp. $\widetilde{V}$) query when global counter is $|0\rangle_{\mathtt{gc}}$ (resp. $|K\rangle_{\mathtt{gc}}$). By the definition in Algo. 6.1, the effect of this $\widetilde{V}^\dagger$ (resp. $\widetilde{V}$) query is to apply the identity operator (i.e., do nothing). Thus, such a query has no effects on the overall states at all and can be ignored. More accurately, one can define another machine $\mathcal{S}^*$ that behaves identically to $\mathcal{S}$ except that when $\mathcal{S}$ tries to make a $\widetilde{V}^\dagger$ (resp. $\widetilde{V}$) query at $|0\rangle_{\mathtt{gc}}$ (resp. $|K\rangle_{\mathtt{gc}}$), $\mathcal{S}^*$ does not do anything and pretend that this query has been answered. The probability of $\mathsf{Pred}(\rho) = 1$ for Game 6.2 w.r.t. this new $\mathcal{S}^*$ is then identical to that of the original Game 6.2.

Next, we state and validate the third assumption.

- **Assumption 3:** For each $k \in \{0, 1, \dots, K-1\}$, *before the* $\mathsf{sq}(k+1)$ *query is made*, we allow $\mathcal{S}$ to operate directly on registers $\mathtt{t}_z$ for all $z \in \{k+1, k+2, \dots, K\}$. Then, for each $k \in [K]$, before the $\mathsf{sq}(k+1)$ query is made, $\mathcal{S}$ will not try to make a $\widetilde{V}$-query when the global counter is $|k\rangle_{\mathtt{gc}}$.

We now argue that **Assumption 3** will not decrease the probability of $\mathsf{Pred}(\rho) = 1$ for Game 6.2. Consider the scenario where before the $\mathsf{sq}(k+1)$ query is made, $\mathcal{S}$ makes a $\widetilde{V}$ when the global counter is $|k\rangle_{\mathtt{gc}}$. At this moment, via the same argument as in the proof of Lem. 10 and Corollary 2, we know that the local counter register does not have any weights on value $k+1$. Therefore, the only effect of this application of $\widetilde{V}$, according to Algo. 6.1, it to increase the global counter to $k+1$, apply A_{k+1}, and then apply $C_{k+1,0}$. From $\mathcal{S}$'s view point, this is nothing but swapping the registers $\mathtt{t}_{\mathtt{k+1}}$ and $\mathtt{m}$. Moreover, note that after this query, the global counter has been increase to the value $k+1$ while the local counter does not increase at all. Therefore, future $\widetilde{V}$ queries (before $\mathsf{sq}(k+1)$) will always been answered by $C'_{k+1}A_{k+1}$ according to the global counter value $k+1$, *until* $\mathcal{S}$ *makes enough* $\widetilde{V}^\dagger$ *queries to bring the global counter back to value* k. Therefore, instead of allowing $\mathcal{S}$ to perform these queries, one can define another machine $\mathcal{S}^*$ that behaves identically to $\mathcal{S}$ except

[17] Note that due to Claim 5, we do not need to consider the case where the global counter is a superposition.

that when $\mathcal{S}$ tries to make a $\widetilde{V}$ query at $|g\rangle_{\mathtt{gc}}$ ($\forall g \in \{k, k+1, \ldots, K-1\}$) before the $\mathsf{sq}(k+1)$ query, $\mathcal{S}^*$ dose not make this query; instead it operates directly on register $\mathtt{t}_{\mathtt{g+1}}$. The probability of $\mathsf{Pred}(\rho) = 1$ for Game 6.2 w.r.t. this new $\mathcal{S}^*$ is then identical to that of the original Game 6.2. This is exactly what we allowed in **Assumption 3**.

6.4.2 Defining the Real Game

Looking into $\mathcal{S}^{\widetilde{\mathcal{V}}}$. Next, we re-state the game $\ddot{\mathcal{S}}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)$ defined in Game 6.2. This time, we will look into the execution of $\mathcal{S}^{\widetilde{\mathcal{V}}}$ and their registers. This is different from previous games where we always treat $\mathcal{S}^{\widetilde{\mathcal{V}}}$ as a "black-box" and only focus on how it accesses the oracle H_0. We denote this game as $\mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}})$ and present it in Game 6.3.

It is straightforward that Game 6.3 is identical to Game 6.2 with only cosmetic changes. Therefore, it holds that

$$\Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \ddot{\mathcal{S}}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}](x)\right] = \Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}})\right]. \tag{6.16}$$

Game 6.3: Game $\mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}})$

For each $i \in [K]$, uniformly pick $(j_i, b_i) \in [2Kq] \times \{0,1\}$ conditioned on that there exists at most one $i \in [K]$ such that $j_i = j^*$ for all $j^* \in [2Kq]$. We re-label the indices i for $\{(j_i, b_i)\}_{i\in[K]}$ so that $j_1 < j_2 < \ldots < j_K$. Initialize $H_0^{(0)} := H_0$ and $\mathbf{p}_0 = \emptyset$ and run the oracle machine $\mathcal{S}^{\widetilde{\mathcal{V}}}(x)$ in the following manner:

For each $i \in [K]$:

- **For $\mathcal{S}$'s Queries between $\mathsf{sq}(i-1)$ and $\mathsf{sq}(i)$:** Answer such queries with $\widetilde{V}$ and $\widetilde{V}^\dagger$ accordingly, where $\widetilde{\mathcal{V}}$ uses $H_0^{(i-1)}$ as the oracle. (see below for the definitions of $H_0^{(i-1)}$ for $i \geq 2$.)
- **For Query $\mathsf{sq}(i)$:** Note that by definition (and Rmk. 11), the $\mathsf{sq}(i)$ query is a $\widetilde{V}$-query and it invokes a measurement over registers $\mathtt{p_1} \ldots \mathtt{p_i}$, resulting in the measurement outcome $\mathbf{p}_i = (p_1, \ldots, p_i)$. If $\mathbf{p}_{i-1}$ is not a prefix of $\mathbf{p}_i$, the execution halts immediately with output $(\perp, \perp)$. Otherwise, it defines $H_0^{(i)} := H_0^{\mathbf{p}_i, \mathbf{1}}$ and then behaves according to the value b_i:
  - If $b_i = 0$, applies $\widetilde{V}$ using $H_0^{(i)}$ as the oracle.
  - If $b_i = 1$, applies $\widetilde{V}$ using $H_0^{(i-1)}$ as the oracle.

Output: Let $(\mathbf{p}^*, \rho_{\mathsf{out}})$ be the output of $\mathcal{S}^{\widetilde{\mathcal{V}}}$ at halt. This game $\mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}})$ outputs $(\mathbf{p}_K, \rho_{\mathsf{out}})$.

6.4.3 Defining the Dummy Game

A Dummy Version of $\widetilde{\mathcal{V}}$. We first define a unitary $\ddot{V}$, which should be treated as the "dummy version" of $\widetilde{\mathcal{V}}$'s unitary $\widetilde{V}$. We present it in Algo. 6.4, highlighting its difference with $\widetilde{V}$ (defined in Algo. 6.1) in red color.

Algorithm 6.4: Unitary $\ddot{V}$ for a Dummy-Version of $\widetilde{\mathcal{V}}$

Helper Unitaries. We first define additional helper unitaries. For each $k \in [K]$:

- $\ddot{C}_k$**: swap $\mathtt{m}$ and $\mathtt{t_k}$.** This unitary perform the following operation in superposition:

- If $H(p_1, \ldots, p_k) = 1$, then do nothing.
- If $H(p_1, \ldots, p_k) = 0$, then swap the contents of registers $\mathtt{t_k}$ and $\mathtt{m}$.

Unitary $\ddot{V}$. On each query, it compares the global counter value k with the local counter value j and behaves accordingly. In particular:

Case $(k = j)$: It first applies the unitary U_{gc} to increase the global counter by 1, and then behaves according to the (increased) global counter value $k + 1$. In particular:

- If $k+1 \in [K]$, then it applies $D_{k+1}\ddot{C}_{k+1}A_{k+1}$, where A_{k+1} and D_{k+1} is defined in Algo. 6.1 and $\ddot{C}_{k+1}$ is defined above.
- Otherwise (i.e., $k + 1 \notin [K]$), it does nothing (i.e., applies the identity operator).

Case $(k \neq j)$: It first applies the unitary U_{gc} to increase the global counter by 1, and then behaves according to the (increased) global counter value $k + 1$. In particular:

- If $k + 1 \in [K]$, then it applies $C'_{k+1}A_{k+1}$, where A_{k+1} and C_{k+1} is defined in Algo. 6.1
- Otherwise (i.e., $k + 1 \notin [K]$), it does nothing (i.e., applies the identity operator).

A Dummy MnR Game. We now define another game $\mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}})$. This game will essentially serve as a malicious prover $\widetilde{\mathcal{P}}$ that could break the soundness of the original ZK protocol, leading to our desired contradiction. Intuitively, $\mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}})$ behaves identical to $\mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}})$ except for that it answers $\mathcal{S}$'s queries between $\mathsf{sq}(i-1)$ and $\mathsf{sq}(i)$ using the dummy version of $\widetilde{\mathcal{V}}$ as we defined in Algo. 6.4. We present the game in Game 6.4 and highlight its difference $\mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}})$ (in Game 6.3) in red color.

Game 6.4: Game $\mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}})$

For each $i \in [K]$, uniformly pick $(j_i, b_i) \in [2Kq] \times \{0, 1\}$ conditioned on that there exists at most one $i \in [K]$ such that $j_i = j^*$ for all $j^* \in [2Kq]$. We re-label the indices i for $\{(j_i, b_i)\}_{i\in[K]}$ so that $j_1 < j_2 < \ldots < j_K$. Initialize $H_0^{(0)} := H_0$ and $\mathbf{p}_0 = \emptyset$ and run the oracle machine $\mathcal{S}^{\widetilde{\mathcal{V}}}(x)$ in the following manner:

For each $i \in [K]$:

- **For $\mathcal{S}$'s Queries between $\mathsf{sq}(i-1)$ and $\mathsf{sq}(i)$:** Answer such queries with $\ddot{V}$ and $\ddot{V}^\dagger$ accordingly, using $H_0^{(i-1)}$ as the oracle. (Note that $\ddot{V}$ is defined in Algo. 6.4.)
- **For Query $\mathsf{sq}(i)$:** Answered in the same manner as in Game 6.3.

Output: Let $(\mathbf{p}^*, \rho_{\mathsf{out}})$ be the output of $\mathcal{S}^{\widetilde{\mathcal{V}}}$ at halt. This game $\mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}})$ outputs $(\mathbf{p}_K, \rho_{\mathsf{out}})$.

The following is the most involved lemma. We defer its proof to Sec. 7. In the following, we show (in Sec. 6.4.4) how to finish the current proof of Lem. 9 assuming that Lem. 11 holds.

Lemma 11. *For the $\mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}})$ defined in Game 6.4 and the $\mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}})$ defined in Game 6.3, it holds that*

$$\Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}})\right] = \Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}})\right].$$

6.4.4 Finishing the Proof of Lem. 9

In this part, we finish the proof of Lem. 9 assuming that Lem. 11 holds.

We start by describing the malicious prover $\widetilde{\mathcal{P}}$ that we will use to break the soundness of the original protocol $\langle \mathcal{P}, \mathcal{V} \rangle$.

Algorithm 6.5: Malicious Prover $\widetilde{\mathcal{P}}(x)$

On input x, $\widetilde{\mathcal{P}}(x)$ internally emulates Game 6.4 with the help of an external honest verifier $\mathcal{V}$ in the following manner.

Registers: Note that to emulate Game 6.4, $\widetilde{\mathcal{P}}$ needs to prepare registers as in Expression (6.1). To do that, $\widetilde{\mathcal{P}}$ prepares the following registers

$$|x\rangle_{\mathtt{ins}}|0\rangle_{\mathtt{gc}}|0\rangle_{\mathtt{lc}}|\mathbf{0}\rangle_{\mathtt{p}_1}\dots|\mathbf{0}\rangle_{\mathtt{p}_K}|\mathbf{0}\rangle_{\mathtt{m}}|\bot\rangle_{\mathtt{t}_1}\dots|\bot\rangle_{\mathtt{t}_K}|H_0\rangle_{\mathtt{aux}} \tag{6.17}$$

Comparing Expression (6.17) with Expression (6.1), one can see that registers $\mathtt{v}_1 \dots \mathtt{v}_K$ and $\mathtt{w}$ are missing. As we will explain shortly, these registers can be think of being held by the external honest verifier $\mathcal{V}$.

Execution: $\widetilde{\mathcal{P}}$ emulates Game 6.4 in the following manner. For each $i \in [K]$,

- **For $\mathcal{S}$'s Queries between $\mathsf{sq}(i-1)$ and $\mathsf{sq}(i)$:** $\widetilde{\mathcal{P}}$ answers such queries with in exactly the same manner as Game 6.4. We emphasize that such queries will be answered using the $\ddot{V}$ or $\ddot{V}^\dagger$. Note that $\ddot{V}$ by definition (see Algo. 6.4) will only make use of the registers shown in Expression (6.17). Therefore, $\widetilde{\mathcal{P}}$ could emulate them internally, without touching upon the external honest $\mathcal{V}$'s registers.
- **For Query $\mathsf{sq}(i)$:** $\widetilde{\mathcal{P}}$ first behaves in the same manner as Game 6.4 until she obtains the measurement outcome $\mathbf{p}_i = (p_1, \dots, p_i)$. Also, same as Game 6.4, if the value $\mathbf{p}_{i-1}$ obtained earlier is not a prefix of the current $\mathbf{p}_i$, $\widetilde{\mathcal{P}}$ aborts the execution directly.
  If the execution has not been aborted so far, the next step in Game 6.4 is to apply the unitary $\widetilde{V}$. We emphasize that this is the (only) step that $\widetilde{\mathcal{P}}$ cannot finish by herself internally, because the $\widetilde{V}$ operator defined in Algo. 6.1 (in particular, the operator B_i) needs to work on the external honest $\mathcal{V}$'s registers $\mathtt{v}_\mathtt{i}$ and $\mathtt{w}$. To do that, $\widetilde{\mathcal{P}}$ simply puts p_i in to the $\mathtt{m}$ register and sends it to the external $\mathcal{V}$ (note that p_i is a classical message). Then, by definition, the honest $\mathcal{V}$ will compute the (possibly quantum) response v_2, put it in the $\mathtt{m}$ register, and send $\mathtt{m}$ back to $\widetilde{\mathcal{P}}$. Using this returned $\mathtt{m}$ register, $\widetilde{\mathcal{P}}$ can easily finish the remaining steps in exactly the same manner as Game 6.4.

Output: We do not define any output for $\widetilde{\mathcal{P}}$. (In Game 6.4, the ρ_{out} part in the output is on the register $\mathtt{w}$. But this register is held by the external honest $\mathcal{V}$ in the current game.)

From the above description, it is easy to see that the view of $\mathcal{S}$ in Algo. 6.5 is identical to that in Game 6.4. Thus, during the execution $\langle \widetilde{\mathcal{P}}, \mathcal{V} \rangle(x)$, the overall state across the joint registers held by both the malicious $\widetilde{\mathcal{P}}$ (as defined in Algo. 6.5) and the honest $\mathcal{V}$ evolves in exactly the same manner as in Game 6.4. Therefore, it follows that

$$\Pr\Big[\mathsf{Acc}(\rho) = 1 \ : \ \rho \leftarrow \mathsf{OUT}_{\mathcal{V}}\langle \widetilde{\mathcal{P}}, \mathcal{V} \rangle(x)\Big] \geq \Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}})\Big], \tag{6.18}$$

where the '$>$' part in the "$\geq$" sign in Inequality (6.18) is due to the following reason: Note that the predicate Pred checks if the $\mathtt{lc}$ register contains K *in addition to the check that the verifier's*

decision bit is 1 (see Def. 6); however, the LHS in Inequality (6.18) only checks if the verifier's decision bit is 1.

Deriving the Final Contradiction. Now, we are ready to finish the proof of Lem. 9. Toward that, we assume for contradiction that Lem. 9 does not hold. Formally, we assume that there exist a $\mathcal{L} \in \mathbf{BQP}$, a $x \in \{0,1\}^\lambda \setminus \mathcal{L}$, and a polynomial $\delta(\cdot)$ so that for infinitely many λ it holds that

$$\Pr[\mathcal{B}(x) = 1] \geq \frac{1}{\delta(\lambda)}. \tag{6.19}$$

It then follows that

$$\begin{aligned}
&\Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Out}_{\mathcal{V}}\big(\langle \widetilde{\mathcal{P}}, \mathcal{V}\rangle(x)\big)\Big] \\
\geq\ &\Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}})\Big] && (6.20)\\
=\ &\Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}})\Big] && (6.21)\\
=\ &\Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \ddot{\mathcal{S}}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}]\Big] && (6.22)\\
\geq\ &\Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathcal{S}^{\widetilde{\mathcal{V}}}[H_0, \mathbf{1}]\Big] && (6.23)\\
=\ &\Pr[\mathcal{B}(x) = 1] && (6.24)\\
\geq\ &\frac{1}{\delta(\lambda)}, && (6.25)
\end{aligned}$$

where Inequality (6.20) follows form Inequality (6.18), Eq. (6.21) follows from Lem. 11, Eq. (6.22) follows from Eq. (6.16), Inequality (6.23) follows from Corollary 3, Inequality (6.24) follows from the definition of $\mathcal{B}(x)$ (see Algo. 6.3), and Inequality (6.25) follows from Inequality (6.19).

Note that Inequality (6.25) contradicts the soundness of the protocol $\langle \mathcal{P}, \mathcal{V}\rangle$.

This eventually completes the proof of Lem. 9.

7 Relating the Dummy and Real Games

In this section (and the next two sections), we establish Lem. 11.

7.1 Branch-Wise Equivalence Suffices

We first notice that Game 6.3 can be treated as a sequence of alternating unitaries and measurements. In particular, there are two types of measurements (and $(q+K)$ measurements are made in total):

1. **Type-1:** Recall that the simulator $\mathcal{S}$ simply alternates between her local unitary S and the measurement on register $\mathtt{u}$, which determines the type of her next query (i.e., either $\widetilde{V}$ or $\widetilde{V}^\dagger$). Note that there are q such measurements on $\mathtt{u}$ since $\mathcal{S}$ makes q queries in total.
2. **Type-2:** When $\mathcal{S}$ makes the $\mathtt{sq}(i)$-th query, it triggers a measurement on registers $\mathtt{p}_1 \dots \mathtt{p}_i$. Note that there are K such measurements (if the game is not aborted prematurely).

Also, notice that in Game 6.3, the values $\{(j_i, b_i)\}_{i\in[K]}$ are sampled at the very beginning and after that the timing of all the measurements are fixed. It then follows from Lem. 5 that the accepting probability $\Pr[\mathsf{Pred}(\rho) = 1]$ w.r.t. Game 6.3 can be equivalently expressed as the summation of

all accepting probabilities w.r.t. a "sub-normalized" version of Game 6.3 where all the $(q+K)$ measurements are replaced with projections. Formally, we defined the "sub-normalized" Game 6.3 (and similarly for Game 6.4) in Game 7.1.

Game 7.1: Sub-Normalized Games $\mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})$ and $\mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})$

Parameters. Fix $J = \{(j_i, b_i)\}_{i\in[K]}$ where each $j_i \in [2Kq]$, each $b_i \in \{0,1\}$, and $j_1 < j_2 < \ldots < j_K$. Also fix a classical string of $(q+K)$ symbols $\mathsf{pat} = (u_1, u_2, \ldots, u_q, p_1, p_2, \ldots, p_K)$ where each $u_i \in \{\uparrow, \downarrow\}$, and each $p_i \in \{0,1\}^\ell$. The following games are parameterized by the fixed (J, pat).

Game $\mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})$. It behaves identically to Game 6.3 except for the following differences:

1. It does not sample the (j_i, b_i) pairs; instead, it uses the $\{(j_i, b_i)\}_{i\in[K]}$ contained in J.
2. When $\mathcal{S}$ needs to make the i-th measurement on $\mathtt{u}$ (corresponding to Type 1 above), it instead applies the projector $|u_i\rangle\langle u_i|$ to $\mathtt{u}$; (Note that the value u_i is contained in pat.)
3. When the measurement on registers $\mathtt{p_1}\ldots\mathtt{p_i}$ is triggered (corresponding to Type 2 above), it instead applies the projector $|p_1, \ldots, p_i\rangle\langle p_1, \ldots, p_i|$ registers $\mathtt{p_1}\ldots\mathtt{p_i}$; (Note that the values $p_1, \ldots, p_i$ are contained in pat.)

Game $\mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})$. This is defined in a similar way as the above. Namely, It behaves identically to Game 6.4 except for the following differences:

1. It does not sample the (j_i, b_i) pairs; instead, it uses the $\{(j_i, b_i)\}_{i\in[K]}$ contained in J.
2. When $\mathcal{S}$ needs to make the i-th measurement on $\mathtt{u}$ (corresponding to Type 1 above), it instead applies the projector $|u_i\rangle\langle u_i|$ to $\mathtt{u}$; (Note that the value u_i is contained in pat.)
3. When the measurement on registers $\mathtt{p_1}\ldots\mathtt{p_i}$ is triggered (corresponding to Type 2 above), it instead applies the projector $|p_1, \ldots, p_i\rangle\langle p_1, \ldots, p_i|$ registers $\mathtt{p_1}\ldots\mathtt{p_i}$; (Note that the values $p_1, \ldots, p_i$ are contained in pat.)

It then follows from Lem. 5 that

$$\begin{aligned}&\Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}})\Big] \\ &= \frac{1}{\binom{2Kq}{K}\cdot 2^K} \cdot \sum_{J,\mathsf{pat}} \Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})\Big],\end{aligned}$$

and that

$$\begin{aligned}&\Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}})\Big] \\ &= \frac{1}{\binom{2Kq}{K}\cdot 2^K} \cdot \sum_{J,\mathsf{pat}} \Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})\Big],\end{aligned}$$

where the summations are taken over all possible J and pat satisfying the requirements in Game 7.1. (Note that the multiplicative factor $1/\left(\binom{2Kq}{K}\cdot 2^K\right)$ is to compensate for the fact that the J is fixed in Game 7.1, instead of being sampled randomly as in the original real Game 6.3 and dummy Game 6.4.)

Therefore, to prove Lem. 11, it suffices to prove the following Lem. 12.

Lemma 12 (Branch-Wise Equivalence Suffices). *For all (J, pat) satisfying the requirements in Game 7.1, it holds that*

$$\Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})\Big] = \Pr\Big[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})\Big].$$

The proof of Lem. 12 is involved. In the following Sec. 7.2, we prove Lem. 12 making use of a technical lemma (i.e., Lem. 13). Then, the later proofs (i.e., Sec. 7.3, Sec. 7.4, Sec. 8, and Sec. 9) are devoted to establishing Lem. 13.

7.2 Proving Lem. 12: Define Hybrids

Henceforth, we consider a fixed (J, pat) pair satisfying the requirements in Game 7.1. We first define a sequence of hybrids $H_0^{(J,\mathsf{pat})}, H_1^{(J,\mathsf{pat})}, \ldots, H_K^{(J,\mathsf{pat})}$.

Hybrid $H_0^{(J,\mathsf{pat})}$. This is exactly the execution of $\mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})$. Therefore, it holds that

$$\Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow H_0\right] = \Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})\right]. \tag{7.1}$$

Hybrid $H_k^{(J,\mathsf{pat})}$ $(k \in [K])$. This hybrid is identical to $H_{k-1}^{(J,\mathsf{pat})}$, except for the following difference:

- The first query (made by $\mathcal{S}$) that brings the global counter from $|k-1\rangle_{\mathsf{gc}}$ to $|k\rangle_{\mathsf{gc}}$ (i.e., query $\mathsf{sq}(k)$) is answered with $\widetilde{V}$ (as in the previous hybrid). However, all subsequent queries that brings the global counter from $|k-1\rangle_{\mathsf{gc}}$ to $|k\rangle_{\mathsf{gc}}$ (resp. from $|k\rangle_{\mathsf{gc}}$ to $|k-1\rangle_{\mathsf{gc}}$) are answered with the "dummy" unitary $\ddot{V}$ (resp. $\ddot{V}^\dagger$).

This completes the description of the hybrids.

Intuition. Intuitively, we use these hybrids to replace the $\widetilde{V}$ (and $\widetilde{V}^\dagger$) to its dummy version $\ddot{V}$ (and $\ddot{V}^\dagger$) "one layer by one layer," where by "layer" we mean the execution that brings the global counter from some value k to $k+1$. Note that such replacement does not affect the special $\mathsf{sq}(k)$ queries, which are always answered by $\widetilde{V}$ across all the hybrids.

To further aid in interpreting these hybrids, we recommend referring to Fig. 2 on Page 69, where we illustrate these hybrids for the simplified case of $K = 2$. In more detail,

- Fig. 2a is the real game $\mathsf{Real}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})$ where all the $\downarrow$ queries (resp. $\uparrow$ queries) are answered by $\widetilde{V}$ (resp. $\widetilde{V}^\dagger$);
- In Fig. 2b (corresponding to $H_1^{(J,\mathsf{pat})}$), all the $\downarrow$ queries (resp. $\uparrow$ queries) in the first "layer" (i.e., the area between the horizontal line for $|0\rangle_{\mathsf{gc}}$ and the horizontal line for $|1\rangle_{\mathsf{gc}}$) are replaced with $\ddot{V}$ (resp. $\ddot{V}^\dagger$), except for the $\mathsf{sq}(1)$ query, which is still answered by $\widetilde{V}$.
- In Fig. 2c (corresponding to $H_2^{(J,\mathsf{pat})}$), all the $\downarrow$ queries (resp. $\uparrow$ queries) in the first two layers are replaced with $\ddot{V}$ (resp. $\ddot{V}^\dagger$), except for the $\mathsf{sq}(1)$ and $\mathsf{sq}(2)$ query, which is still answered by $\widetilde{V}$.

Note that in this baby case of $K = 2$, the hybrid $H_2^{(J,\mathsf{pat})}$ illustrated by Fig. 2c is the last hybrid, and obviously, it is exactly the dummy game $\mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})$.

From the definition of hybrids (and the intuitive explanation above), it is easy to see that hybrid $H_K^{(J,\mathsf{pat})}$ is exactly the execution of $\mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})$. Thus, it holds that

$$\Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow H_K^{(J,\mathsf{pat})}\right] = \Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow \mathsf{Dummy}(\mathcal{S}, \widetilde{\mathcal{V}}, J, \mathsf{pat})\right]. \tag{7.2}$$

It then following from Eq. (7.1) and Eq. (7.2) that to prove Lem. 12, it suffices to establish the following Lem. 13.

Lemma 13. *For all* (J, pat) *satisfying the requirements in Game 7.1 and all* $k \in [K]$*, it holds that*

$$\Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow H_{k-1}^{(J,\mathsf{pat})}\right] = \Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow H_{k}^{(J,\mathsf{pat})}\right].$$

Proof Structure of Lem. 13. The remaining part of the proof focuses on establishing Lem. 13. Its proof is quite intricate and spans several sections: Sec. 7.3, Sec. 7.4, Sec. 8, and Sec. 9. Here is an overview of what each section covers:

- First, we remark that the hybrids described above is stated in a manner for ease of understanding; however, this presentation is not conducive to the mathematical derivations needed to establish Lem. 13. Therefore, we need to restate these hybrids in a more mathematically friendly form. To accomplish this, we introduce a lemma characterizing the structure of the local counter and global counter registers in Sec. 7.3. Armed with this lemma, we proceed to provide an alternative description of the hybrids in Sec. 7.4. This new description is essentially equivalent but offers a more suitable framework for the subsequent mathematical analysis.
- Having established the alternative description of the hybrids in Sec. 7.4, we are now prepared to demonstrate the indistinguishability between adjacent hybrids, as mandated by Lem. 13. However, given the complexity of this proof, we opt to initially focus on the simplified case of $K = 2$, aiming to elucidate the core concept in a more streamlined context. This treatment is presented in Sec. 8.
- Ultimately, we provide the full proof for the general case (for an arbitrary constant K). This is elucidated in Sec. 9.

7.3 Prove Lem. 13: Structure of Counters

Intuition. Let us first provide a high-level overview of this part. We assert that within any hybrid $H_k^{(J,\mathsf{pat})}$, throughout its execution, the global counter and local counter will exhibit a nice structure as defined in the following Lem. 14. Intuitively, Lem. 14 can be interpreted as follows: during the execution of $H_k^{(J,\mathsf{pat})}$, the overall state can be expressed as the sum of pure states in superposition, and within each superposition, the local counter value does not exceed the global counter value. Furthermore, for the superposition where the $\mathtt{p}_1 \dots \mathtt{p}_i$ registers contain precisely the values $p_1, \dots, p_i$ (specified in pat), both the global counter and the local counter equal i.

Looking ahead, Lem. 14 bears significance for the subsequent proof for the following reason. As per Algo. 6.1, the behavior of $\widetilde{\mathcal{V}}$ is (partially) dictated by the global and local counters. In other words, it necessitates comparing these two counters to determine the appropriate unitaries to employ in Algo. 6.1. Hence, if we aim to monitor the evolution of the overall state throughout the execution, we must monitor these two counters. It is not immediately evident whether this is feasible, as the global and local counter registers might be in a complex superposition that defies a neat mathematical expression.

Now, armed with Lem. 14, we possess a complete and clear understanding of the structure of these two counters. This structured framework enables us to derive a "finer-grained" description of the unitaries $\widetilde{V}$ and $\ddot{V}$ simply by examining the global counter, which, as noted in Claim 5, maintains a classical value throughout the execution. As will become clear later in Sec. 8 and 9, such a "finer-grained" characterization serves as the linchpin for tracking the overall states throughout the hybrid executions. This capability, in turn, facilitates the establishment of the indistinguishability conditions stipulated in Lem. 13.

We next proceed to the formal treatment.

Lemma 14 (Counter Structure.). *For each pair* (J, pat) *satisfying the requirements in Game 7.1, each* $k \in [K] \cup \{0\}$*, each hybrid* $H_k^{(J,\mathsf{pat})}$*, let* $|\phi^{(t,i)}\rangle$ *denote the overall state right after the* $\mathcal{S}$*'s query that leads to the global counter's* t*-th arrival at value* i*. Then, the following holds:*

1. *If this arrival at value* i *is due to the* $\mathsf{sq}(i)$ *query* and *the value* $b_i = 1$*, then the state* $|\phi^{(t,i)}\rangle$ *can be written in the following format:*

$$|\phi^{(t,i)}\rangle = |i\rangle_{\mathtt{gc}}|i-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_i}|\rho^{(t,i)}\rangle. \tag{7.3}$$

2. *In all the other cases (i.e., either this arrival at value* i *is not caused by the* $\mathsf{sq}(i)$ *query or it is but* $b_i = 0$*), the state* $|\phi^{(t,i)}\rangle$ *can be written in the following format:*

$$\begin{aligned}|\phi^{(t,i)}\rangle = |i\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_i}|\rho^{(t,i)}\rangle \ + \\ |i\rangle_{\mathtt{gc}}\sum_{j=0}^{i-1}|j\rangle_{\mathtt{lc}}|p_1,\ldots,p_j\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_j}\sum_{p'_{j+1}\neq p_{j+1}}|p'_{j+1}\rangle_{\mathtt{p}_{j+1}}|\rho^{(t,i)}_{p'_{j+1}}\rangle,\end{aligned} \tag{7.4}$$

where the summation over $p'_{j+1} \neq p_{j+1}$ *means to take the summation over* $p'_{j+1} \in \{0,1\}^\ell \backslash \{p_{j+1}\}$.

For ease of understanding, we expand the above succinct expression in Eq. (7.4) as follows:

$$\begin{aligned}|\phi^{(t,i)}\rangle =& |i\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_i}|\rho^{(t,i)}\rangle \ + \\ & |i\rangle_{\mathtt{gc}}|i-1\rangle_{\mathtt{lc}}|p_1,\ldots,p_{i-1}\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_{i-1}}\sum_{p'_i\neq p_i}|p'_i\rangle_{\mathtt{p}_i}|\rho^{(t,i)}_{p'_i}\rangle \ + \\ & |i\rangle_{\mathtt{gc}}|i-2\rangle_{\mathtt{lc}}|p_1,\ldots,p_{i-2}\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_{i-2}}\sum_{p'_{i-1}\neq p_{i-1}}|p'_{i-1}\rangle_{\mathtt{p}_{i-1}}|\rho^{(t,i)}_{p'_{i-1}}\rangle \ + \\ & \cdots \\ & |i\rangle_{\mathtt{gc}}|1\rangle_{\mathtt{lc}}|p_1\rangle_{\mathtt{p}_1}\sum_{p'_2\neq p_2}|p'_2\rangle_{\mathtt{p}_2}|\rho^{(t,i)}_{p'_2}\rangle \ + \\ & |i\rangle_{\mathtt{gc}}|0\rangle_{\mathtt{lc}}\sum_{p'_1}|p'_1\rangle_{\mathtt{p}_1}|\rho^{(t,i)}_{p'_1\neq p_1}\rangle.\end{aligned}$$

Proof of Lem. 14. We prove this lemma via mathematical induction over the operations performed by the simulator $\mathcal{S}$. That is, we first prove that the initial state at the very beginning of the MnR game is of the format shown in Eq. (7.4) (i.e., the **Base Case**), and then show that each movement of $\mathcal{S}$ will lead to a new state of the same format shown in Eq. (7.3) or Eq. (7.3), depending on which case between Cases 1 and 2 will happen (i.e., the **Induction Step**).

In the sequel, we consider a fixed pair (J, pat) and $H_k^{(J,\mathsf{pat})}$. We emphasize that the lower-case "k" denotes the hybrid index. Do not confuse it with the capitalized "K" which denotes the round complexity of the original $\langle \mathcal{P}, \mathcal{V}\rangle$.

Base Case. This corresponds to the state at the very beginning of the MnR game. By definition, the initial sate is of the format $|0\rangle_{\mathtt{gc}}|0\rangle_{\mathtt{lc}}|\rho\rangle$. It satisfies the format shown in Eq. (7.4) (by setting $|\rho^{(0,0)}\rangle := |\rho\rangle$).

Induction Step. We assume that the lemma is true for the global counter's t-th arrival at value i, namely, the current overall state $|\phi^{(t,i)}\rangle$ satisfies the format shown in Lem. 14. We prove that the simulator's next query, no matter what this query is, will lead to a new state that satisfies the format shown in Lem. 14 as well.

Toward that, let us first classify the possible movements of $\mathcal{S}$ when the state is $|\phi^{(t,i)}\rangle$.

– Type 1: after applying the local operator S and measuring register $\mathtt{u}$, $\mathcal{S}$ makes the $\mathsf{sq}(i+1)$ query for some $i \in [K]$. (Recall that when the global counter is i and $\mathcal{S}$ is about to make a $\mathsf{sq}(j)$ query, then it must be the $\mathsf{sq}(i+1)$ query.)

– Type 2: after applying the local operator S and measuring register $\mathtt{u}$, $\mathcal{S}$ makes a $\widetilde{V}$ query that is not the $\mathsf{sq}(i+1)$ query. (This could happen only if the global counter $i \geq k$)

– Type 3: after applying the local operator S and measuring register $\mathtt{u}$, $\mathcal{S}$ makes a $\widetilde{V}^\dagger$ query. (This could happen only if the global counter $i > k$)

– Type 4: after applying the local operator S and measuring register $\mathtt{u}$, $\mathcal{S}$ makes a $\ddot{V}$ query. (This could happen if the global counter $i < k$.)

– Type 5: after applying the local operator S and measuring register $\mathtt{u}$, $\mathcal{S}$ makes a $\ddot{V}^\dagger$ query. (This could happen if the global counter $i \leq k$.)

The proof for the above five cases are very similar. In the following, we only show the proofs for Type 1, Type 2, and Type 3. They are the most representative cases because Type 1 illustrates why the statement of Lem. 14 contains two separate cases (i.e., Cases 1 and 2), and Type 2 and Type 3 cover the both the "going-up" $\uparrow$-query and the "going-down" $\downarrow$-query.

Proof of Type 1: From our induction assumption, we know that right before the $\mathsf{sq}(i+1)$ query, the state $|\phi^{(t,i)}\rangle$ is of the format shown in either Eq. (7.3) or Eq. (7.4).

Next, we further claim that the state $|\phi^{(t,i)}\rangle$ must be of the format shown in Eq. (7.4) (i.e., we know for sure that we are in Case 2). Let us prove this: Assume for contradiction that this is not the case, then $|\phi^{(t,i)}\rangle$ is of the format shown in Eq. (7.3) and in particular, the local counter register contains the (classical) value $i-1$. Now, since the next query is the $\mathsf{sq}(i+1)$ query, it will invoke the a measurement on the query made to the oracle H oracle query. Then, from **Step 3** in Algo. 6.2, we know that H will be queried on registers $\mathtt{p}_1 \dots \mathtt{p}_i$ and thus the measurement outcome $\mathbf{p}_{i+1}$ is of length exactly $z_{i+1} = i$. However, this contradicts the requirement at Step 3a (or otherwise the game is aborted prematurely at this step) of Game 6.2. Thus, the state $|\phi^{(t,i)}\rangle$ must be of the format shown in Eq. (7.4).

Therefore, the $\mathsf{sq}(i+1)$ will be handled as follows:

1. U_{gc} is first applied to increase the global counter from $|i\rangle_{\mathtt{gc}}$ to $|i+1\rangle_{\mathtt{gc}}$.
2. The A_{i+1} is applied;
3. The query to H will be made, which invokes the projection on the registers $\mathtt{p}_1 \dots \mathtt{p}_i$ to value $|p_1, \dots, p_{i+1}\rangle_{\mathtt{p}_1 \dots \mathtt{p}_{i+1}}$. (Recall that we are currently in the "sub-normalized" game $H_k^{(J,\mathsf{pat})}$. Thus, the measurement has been replaced with this projection.) Also, as we just argued, the initial state $|\phi^{(t,i)}\rangle$ was of the format shown in Eq. (7.4). Thus, all the branches in Eq. (7.4), except for the first one, will be "killed" by the projector $|p_1, \dots, p_{i+1}\rangle_{\mathtt{p}_1 \dots \mathtt{p}_{i+1}}$. Therefore, the resulting state would be of the following format:

$$|i+1\rangle_{\mathtt{gc}} |i\rangle_{\mathtt{lc}} |p_1, \dots, p_i\rangle_{\mathtt{p}_1, \dots, \mathtt{p}_i} |p_{i+1}\rangle_{\mathtt{p}_{i+1}} |\rho\rangle. \tag{7.5}$$

4. Next, the behavior depends on the the local counter value (in superposition) and the value b_{i+1} specified in pat. In particular:
   (a) If $b_{i+1} = 0$: the operator $D_{i+1}C_{i+1}B_{i+1}$ (defined in Algo. 6.1) will be applied to Expression (7.5), resulting in the following state:

$$D_{i+1}C_{i+1}B_{i+1} |i+1\rangle_{\mathtt{gc}} |i\rangle_{\mathtt{lc}} |p_1, \dots, p_i\rangle_{\mathtt{p}_1, \dots, \mathtt{p}_i} |p_{i+1}\rangle_{\mathtt{p}_{i+1}} |\rho\rangle$$

$$= |i+1\rangle_{\mathtt{gc}}|i+1\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_i}|p_{i+1}\rangle_{\mathtt{p}_{i+1}}C_{i+1,0}B_{i+1,0}|\rho\rangle \tag{7.6}$$

$$= |i+1\rangle_{\mathtt{gc}}|i+1\rangle_{\mathtt{lc}}|p_1,\ldots,p_{i+1}\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_{i+1}}|\rho^{(z,i+1)}\rangle \tag{7.7}$$

where Eq. (7.6) follows from the definition of D_{i+1}, C_{i+1}, and B_{i+1} (see Algo. 6.1), and Eq. (7.7) follows by defining $|\rho^{(z,i+1)}\rangle \coloneqq C_{i+1,0}B_{i+1,0}|\rho\rangle$.
Obviously, Eq. (7.7) satisfies the format of Eq. (7.4) with t updated to z and i updated to $i+1$ (i.e., we assume this is the global counter's z-th arrival at value $i+1$).

(b) If $b_{i+1}=1$: In this case, first the query will be answered and then the oracle is programmed. Note that before the oracle is programmed, it must hold that $H(p_1,\ldots,p_{i+1})=0$. Therefore, by definition of $\widetilde{V}$, only the operator $C_{i+1,0}$ will be effectively applied, resulting in the following state:

$$\begin{aligned}&|i+1\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_i}|p_{i+1}\rangle_{\mathtt{p}_{i+1}}C_{i+1,0}|\rho\rangle\\ =\ &|i+1\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_{i+1}\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_{i+1}}|\rho^{(z,i+1)}\rangle,\end{aligned} \tag{7.8}$$

where Eq. (7.8) follows by defining $|\rho^{(z,i+1)}\rangle \coloneqq C_{i+1,0}|\rho\rangle$.
Obviously, Eq. (7.8) satisfies the format of Eq. (7.3) with t updated to z and i updated to $i+1$ (i.e., we assume this is the global counter's z-th arrival at value $i+1$).

This finishes the proof of Type 1.

Proof of Type 2: From our induction assumption, we know that right before the $\mathsf{sq}(i+1)$ query, the state $|\phi^{(t,i)}\rangle$ is of the format shown in either Eq. (7.3) or Eq. (7.4).

Next, we further claim that the state $|\phi^{(t,i)}\rangle$ must be of the format shown in Eq. (7.4) (i.e., we know for sure that we are in Case 2). Let us prove this: Assume for contradiction that this is not the case, then the last query is exactly the $\mathsf{sq}(i)$ query. At this moment, the $\mathsf{sq}(i+1)$ query has not been made. Then, it follows from **Assumption 3** that the next query is not a $\widetilde{V}$ query. However, this contradicts the fact that this Type 2 is a $\widetilde{V}$ query.

Therefore, in the following, we only need to show the derivation starting from the state $|\phi^{(t,i)}\rangle$ shown in Eq. (7.4).

In this type, $\mathcal{S}$ first applies the local operator S. Then, she measures the $\mathtt{u}$ register. Recall that we are in the "sub-normalized" game $H_k^{(J,\mathsf{pat})}$, and thus this measurement is replaced with a projector $|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}$ (it must be a $\downarrow$ because in this case the coming query is $\widetilde{V}$). Finally, $\mathcal{S}$ makes a $\widetilde{V}$ query. Notation-wise, we assume this leads to the global counter's z-th arrival at value $i+1$.

In summary, the state $|\phi^{(t,i)}\rangle$ evolves as follows

$$\begin{aligned}|\phi^{(z,i+1)}\rangle &= \widetilde{V}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\phi^{(t,i)}\rangle\\ &= \widetilde{V}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S\Big(|i\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_i}|\rho^{(t,i)}\rangle\ +\\ &\qquad |i\rangle_{\mathtt{gc}}\sum_{j=0}^{i-1}|j\rangle_{\mathtt{lc}}|p_1,\ldots,p_j\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_j}\sum_{p'_{j+1}\neq p_{j+1}}|p'_{j+1}\rangle_{\mathtt{p}_{j+1}}|\rho^{(t,i)}_{p'_{j+1}}\rangle\Big)\end{aligned} \tag{7.9}$$

$$\begin{aligned}&= \widetilde{V}\Big(|i\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_i}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\rho^{(t,i)}\rangle\ +\\ &\qquad |i\rangle_{\mathtt{gc}}\sum_{j=0}^{i-1}|j\rangle_{\mathtt{lc}}|p_1,\ldots,p_j\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_j}\sum_{p'_{j+1}\neq p_{j+1}}|p'_{j+1}\rangle_{\mathtt{p}_{j+1}}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\rho^{(t,i)}_{p'_{j+1}}\rangle\Big)\end{aligned} \tag{7.10}$$

$$= D_{i+1}C_{i+1}B_{i+1}A_{i+1}|i+1\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p_1},\ldots,\mathtt{p_i}}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\rho^{(t,i)}\rangle\;+$$

$$C'_{i+1}A_{i+1}|i+1\rangle_{\mathtt{gc}}\sum_{j=0}^{i-1}|j\rangle_{\mathtt{lc}}|p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}}\sum_{p'_{j+1}\neq p_{j+1}}|p'_{j+1}\rangle_{\mathtt{p_{j+1}}}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\rho^{(t,i)}_{p'_{j+1}}\rangle \quad (7.11)$$

$$= D_{i+1}C_{i+1}B_{i+1}|i+1\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p_1},\ldots,\mathtt{p_i}}A_{i+1}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\rho^{(t,i)}\rangle\;+$$

$$|i+1\rangle_{\mathtt{gc}}\sum_{j=0}^{i-1}|j\rangle_{\mathtt{lc}}|p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}}\sum_{p'_{j+1}\neq p_{j+1}}|p'_{j+1}\rangle_{\mathtt{p_{j+1}}}C'_{i+1}A_{i+1}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\rho^{(t,i)}_{p'_{j+1}}\rangle \quad (7.12)$$

$$= D_{i+1}C_{i+1}B_{i+1}|i+1\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p_1},\ldots,\mathtt{p_i}}\sum_{p'_{i+1}\in\{0,1\}^{\ell}}|p'_{i+1}\rangle_{\mathtt{p_{i+1}}}|\rho^{(t,i)}_{p'_{i+1}}\rangle\;+$$

$$|i+1\rangle_{\mathtt{gc}}\sum_{j=0}^{i-1}|j\rangle_{\mathtt{lc}}|p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}}\sum_{p'_{j+1}\neq p_{j+1}}|p'_{j+1}\rangle_{\mathtt{p_{j+1}}}C'_{i+1}A_{i+1}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\rho^{(t,i)}_{p'_{j+1}}\rangle \quad (7.13)$$

$$= D_{i+1}C_{i+1}B_{i+1}|i+1\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p_1},\ldots,\mathtt{p_i}}\Bigg(|p_{i+1}\rangle_{\mathtt{p_{i+1}}}|\rho^{(t,i)}_{p_{i+1}}\rangle\;+$$

$$\sum_{p'_{i+1}\neq p_{i+1}}|p'_{i+1}\rangle_{\mathtt{p_{i+1}}}|\rho^{(t,i)}_{p'_{i+1}}\rangle\Bigg)\;+$$

$$|i+1\rangle_{\mathtt{gc}}\sum_{j=0}^{i-1}|j\rangle_{\mathtt{lc}}|p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}}\sum_{p'_{j+1}\neq p_{j+1}}|p'_{j+1}\rangle_{\mathtt{p_{j+1}}}C'_{i+1}A_{i+1}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\rho^{(t,i)}_{p'_{j+1}}\rangle$$

$$= D_{i+1}|i+1\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p_1},\ldots,\mathtt{p_i}}\Bigg(|p_{i+1}\rangle_{\mathtt{p_{i+1}}}C_{i+1,1}B_{i+1,1}|\rho^{(t,i)}_{p_{i+1}}\rangle\;+$$

$$\sum_{p'_{i+1}\neq p_{i+1}}|p'_{i+1}\rangle_{\mathtt{p_{i+1}}}C_{i+1,0}B_{i+1,0}|\rho^{(t,i)}_{p'_{i+1}}\rangle\Bigg)\;+$$

$$|i+1\rangle_{\mathtt{gc}}\sum_{j=0}^{i-1}|j\rangle_{\mathtt{lc}}|p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}}\sum_{p'_{j+1}\neq p_{j+1}}|p'_{j+1}\rangle_{\mathtt{p_{j+1}}}C'_{i+1}A_{i+1}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\rho^{(t,i)}_{p'_{j+1}}\rangle \quad (7.14)$$

$$= D_{i+1}|i+1\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p_1},\ldots,\mathtt{p_i}}\Bigg(|p_{i+1}\rangle_{\mathtt{p_{i+1}}}|\rho^{(z,i+1)}\rangle\;+$$

$$\sum_{p'_{i+1}\neq p_{i+1}}|p'_{i+1}\rangle_{\mathtt{p_{i+1}}}|\rho^{(z,i+1)}_{p'_{i+1}}\rangle\Bigg)\;+$$

$$|i+1\rangle_{\mathtt{gc}}\sum_{j=0}^{i-1}|j\rangle_{\mathtt{lc}}|p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}}\sum_{p'_{j+1}\neq p_{j+1}}|p'_{j+1}\rangle_{\mathtt{p_{j+1}}}|\rho^{(z,i+1)}_{p'_{j+1}}\rangle \quad (7.15)$$

$$= |i+1\rangle_{\mathtt{gc}}|i+1\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p_1},\ldots,\mathtt{p_i}}|p_{i+1}\rangle_{\mathtt{p_{i+1}}}|\rho^{(z,i+1)}\rangle\;+$$

$$|i+1\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p_1},\ldots,\mathtt{p_i}}\sum_{p'_{i+1}\neq p_{i+1}}|p'_{i+1}\rangle_{\mathtt{p_{i+1}}}|\rho^{(z,i+1)}_{p'_{i+1}}\rangle\;+$$

$$|i+1\rangle_{\mathtt{gc}}\sum_{j=0}^{i-1}|j\rangle_{\mathtt{lc}}|p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}}\sum_{p'_{j+1}\neq p_{j+1}}|p'_{j+1}\rangle_{\mathtt{p_{j+1}}}|\rho^{(z,i+1)}_{p'_{j+1}}\rangle \quad (7.16)$$

$$= |i+1\rangle_{\mathtt{gc}}|i+1\rangle_{\mathtt{lc}}|p_1,\ldots,p_{i+1}\rangle_{\mathtt{p_1},\ldots,\mathtt{p_{i+1}}}|\rho^{(z,i+1)}\rangle\;+$$

$$|i+1\rangle_{\mathtt{gc}} \sum_{j=0}^{i} |j\rangle_{\mathtt{lc}} |p_1,\ldots,p_j\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_j} \sum_{p'_{j+1}\neq p_{j+1}} |p'_{j+1}\rangle_{\mathtt{p}_{j+1}} |\rho^{(z,i+1)}_{p'_{j+1}}\rangle, \quad (7.17)$$

where

- Eq. (7.9) follows from our induction assumption that the starting state $|\phi^{(t,i)}\rangle$ is as shown in Eq. (7.4).
- Eq. (7.10) follows from the fact that $|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}} S$ does not operate on the $\mathtt{gc}$, $\mathtt{lc}$, and $\mathtt{p}_1 \ldots \mathtt{p}_i$ registers.
- Eq. (7.11) follows from the definition of $\widetilde{V}$ (see Algo. 6.1).
- Eq. (7.12) follows from the fact that C'_{i+1} and A_{i+1} do not act on registers $\mathtt{p}_1 \ldots \mathtt{p}_i$.
- Eq. (7.13) follows by expanding the $\mathtt{p}_{i+1}$ register under the computational basis for the state:

$$A_{i+1}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\, S|\rho^{(t,i)}\rangle = \sum_{p'_{i+1}\in\{0,1\}^{\ell}} |p'_{i+1}\rangle_{\mathtt{p}_{i+1}} |\rho^{(t,i)}_{p'_{i+1}}\rangle.$$

- Eq. (7.14) follows from the following definitions:
  - Define $C_{i+1,1}$ and $B_{i+1,1}$ as the parts of C_{i+1} and B_{i+1} (see Algo. 6.1) w.r.t. to the $H(p_1,\ldots,p_{i+1}) = 1$ branch, and define $C_{i+1,0}$ and $B_{i+1,0}$ as the parts of C_{i+1} and B_{i+1} (see Algo. 6.1) w.r.t. the $H(p_1,\ldots,p_{i+1}) = 0$ branch. (Therefore, none of $C_{i+1,1}$, $C_{i+1,0}$, $B_{i+1,1}$, or $B_{i+1,0}$ act on registers $\mathtt{p}_1 \ldots \mathtt{p}_{i+1}$.)
- Eq. (7.15) follows by defining

$$\begin{aligned}
|\rho^{(z,i+1)}\rangle &:= C_{i+1,1}B_{i+1,1}|\rho^{(t,i)}_{p_{i+1}}\rangle \\
|\rho^{(z,i+1)}_{p'_{i+1}}\rangle &:= C_{i+1,0}B_{i+1,0}|\rho^{(t,i)}_{p'_{i+1}}\rangle \quad (\forall p'_{i+1}\neq p_{i+1}) \\
|\rho^{(z,i+1)}_{p'_{j+1}}\rangle &:= C'_{i+1}A_{i+1}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\, S|\rho^{(t,i)}_{p'_{j+1}}\rangle \quad (\forall j\in\{0,1,\ldots,i-1\},\ \forall p'_{j+1}\neq p_{j+1}).
\end{aligned}$$

- Eq. (7.16) follows by the definition of D_{i+1} (see Algo. 6.1).

Obviously, Eq. (7.17) satisfies the format of Eq. (7.4) with t updated to z and i updated to $i+1$. This finishes the proof of Type 2.

Proof of Type 3: From our induction assumption, we know that right before the $\mathtt{sq}(i+1)$ query, the state $|\phi^{(t,i)}\rangle$ is of the format shown in either Eq. (7.3) or Eq. (7.4).

In the following, we show the derivation assuming the starting state $|\phi^{(t,i)}\rangle$ is of the Eq. (7.4) format, because the other case (i.e., $|\phi^{(t,i)}\rangle$ is of the Eq. (7.3) format) can be established using the same derivation.

In this type, $\mathcal{S}$ first applies the local operator S. Then, she measures the $\mathtt{u}$ register. Recall that we are in the "sub-normalized" game $H_k^{(J,\mathsf{pat})}$, and thus this measurement is replaced with a projector $|{\uparrow}\rangle\langle{\uparrow}|_{\mathtt{u}}$ (it must be a $\uparrow$ because in this case the coming query is $\widetilde{V}^{\dagger}$). Finally, $\mathcal{S}$ makes a $\widetilde{V}^{\dagger}$ query. Notation-wise, we assume this leads to the global counter's z-th arrival at value $i-1$.

In summary, the state $|\phi^{(t,i)}\rangle$ evolves as follows

$$\begin{aligned}
|\phi^{(z,i-1)}\rangle &= \widetilde{V}^{\dagger}\,|{\uparrow}\rangle\langle{\uparrow}|_{\mathtt{u}}\, S|\phi^{(t,i)}\rangle \\
&= \widetilde{V}^{\dagger}\,|{\uparrow}\rangle\langle{\uparrow}|_{\mathtt{u}}\, S\Big(|i\rangle_{\mathtt{gc}}|i\rangle_{\mathtt{lc}}|p_1,\ldots,p_i\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_i}|\rho^{(t,i)}\rangle + \\
\end{aligned}$$

$$|i\rangle_{\mathtt{gc}} \sum_{j=0}^{i-1} |j\rangle_{\mathtt{lc}} |p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}} \sum_{p'_{j+1}\neq p_{j+1}} |p'_{j+1}\rangle_{\mathtt{p_{j+1}}} |\rho^{(t,i)}_{p'_{j+1}}\rangle \Bigg) \tag{7.18}$$

$$= \widetilde{V}^\dagger \Bigg(|i\rangle_{\mathtt{gc}} |i\rangle_{\mathtt{lc}} |p_1,\ldots,p_i\rangle_{\mathtt{p_1},\ldots,\mathtt{p_i}} |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S|\rho^{(t,i)}\rangle +$$

$$|i\rangle_{\mathtt{gc}} \sum_{j=0}^{i-1} |j\rangle_{\mathtt{lc}} |p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}} \sum_{p'_{j+1}\neq p_{j+1}} |p'_{j+1}\rangle_{\mathtt{p_{j+1}}} |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S|\rho^{(t,i)}_{p'_{j+1}}\rangle \Bigg) \tag{7.19}$$

$$= U^\dagger_{gc} A^\dagger_i B^\dagger_i C^\dagger_i D^\dagger_i |i\rangle_{\mathtt{gc}} |i\rangle_{\mathtt{lc}} |p_1,\ldots,p_i\rangle_{\mathtt{p_1},\ldots,\mathtt{p_i}} |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S|\rho^{(t,i)}\rangle +$$

$$U^\dagger_{gc} A^\dagger_i C'^\dagger_i |i\rangle_{\mathtt{gc}} \sum_{j=0}^{i-1} |j\rangle_{\mathtt{lc}} |p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}} \sum_{p'_{j+1}\neq p_{j+1}} |p'_{j+1}\rangle_{\mathtt{p_{j+1}}} |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S|\rho^{(t,i)}_{p'_{j+1}}\rangle \tag{7.20}$$

$$= |i-1\rangle_{\mathtt{gc}} |i-1\rangle_{\mathtt{lc}} |p_1,\ldots,p_{i-1}\rangle_{\mathtt{p_1},\ldots,\mathtt{p_{i-1}}} A^\dagger_i |p_i\rangle_{\mathtt{p_i}} B^\dagger_{i,1} C^\dagger_{i,1} |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S|\rho^{(t,i)}\rangle +$$

$$|i-1\rangle_{\mathtt{gc}} \sum_{j=0}^{i-1} |j\rangle_{\mathtt{lc}} |p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}} \sum_{p'_{j+1}\neq p_{j+1}} A^\dagger_i |p'_{j+1}\rangle_{\mathtt{p_{j+1}}} C'^\dagger_i |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S|\rho^{(t,i)}_{p'_{j+1}}\rangle \tag{7.21}$$

$$= |i-1\rangle_{\mathtt{gc}} |i-1\rangle_{\mathtt{lc}} |p_1,\ldots,p_{i-1}\rangle_{\mathtt{p_1},\ldots,\mathtt{p_{i-1}}} A^\dagger_i |p_i\rangle_{\mathtt{p_i}} B^\dagger_{i,1} C^\dagger_{i,1} |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S|\rho^{(t,i)}\rangle +$$

$$|i-1\rangle_{\mathtt{gc}} |i-1\rangle_{\mathtt{lc}} |p_1,\ldots,p_{i-1}\rangle_{\mathtt{p_1},\ldots,\mathtt{p_{i-1}}} \sum_{p'_i\neq p_i} A^\dagger_i |p'_i\rangle_{\mathtt{p_i}} C'^\dagger_i |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S|\rho^{(t,i)}_{p'_i}\rangle$$

$$|i-1\rangle_{\mathtt{gc}} \sum_{j=0}^{i-2} |j\rangle_{\mathtt{lc}} |p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}} \sum_{p'_{j+1}\neq p_{j+1}} |p'_{j+1}\rangle_{\mathtt{p_{j+1}}} A^\dagger_i C'^\dagger_i |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S|\rho^{(t,i)}_{p'_{j+1}}\rangle \tag{7.22}$$

$$= |i-1\rangle_{\mathtt{gc}} |i-1\rangle_{\mathtt{lc}} |p_1,\ldots,p_{i-1}\rangle_{\mathtt{p_1},\ldots,\mathtt{p_{i-1}}} |\rho^{(z,i-1)}\rangle +$$

$$|i-1\rangle_{\mathtt{gc}} \sum_{j=0}^{i-2} |j\rangle_{\mathtt{lc}} |p_1,\ldots,p_j\rangle_{\mathtt{p_1},\ldots,\mathtt{p_j}} \sum_{p'_{j+1}\neq p_{j+1}} |p'_{j+1}\rangle_{\mathtt{p_{j+1}}} |\rho^{(z,i-1)}_{p'_{j+1}}\rangle, \tag{7.23}$$

where

- Eq. (7.18) follows from our induction assumption that the starting state $|\phi^{(t,i)}\rangle$ is as shown in Eq. (7.4).
- Eq. (7.19) follows from the fact that $|\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S$ does not operate on the $\mathtt{gc}$, $\mathtt{lc}$, and $\mathtt{p_1}\ldots\mathtt{p_i}$ registers.
- Eq. (7.20) follows from the definition of $\widetilde{V}$ (see Algo. 6.1).
- Eq. (7.21) follows from the definition of D_i and U_{gc} (see Algo. 6.1), the fact that A_i does not act on registers $\mathtt{p_1}\ldots\mathtt{p_{i-1}}$, the fact that C'_i does not act on registers $\mathtt{p_1}\ldots\mathtt{p_i}$, and the following definitions.
  - Define $C_{i,1}$ and $B_{i,1}$ as the part of C_i and B_i (see Algo. 6.1) corresponding to the $H(p_1,\ldots,p_i) = 1$ branch (and thus $C_{i,1}$ and $B_{i,1}$ do not act on registers $\mathtt{p_1}\ldots\mathtt{p_i}$).
- Eq. (7.22) follows by isolating the $(j = i-1)$ term and that A_i does not act on registers $\mathtt{p_1}\ldots\mathtt{p_{i-1}}$.
- Eq. (7.23) follows by defining

$$|\rho^{(z,i-1)}\rangle := A^\dagger_i |p_i\rangle_{\mathtt{p_i}} B^\dagger_{i,1} C^\dagger_{i,1} |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S|\rho^{(t,i)}\rangle + \sum_{p'_i\neq p_i} A^\dagger_i |p'_i\rangle_{\mathtt{p_i}} C'^\dagger_i |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} S|\rho^{(t,i)}_{p'_i}\rangle$$

$$|\rho^{(z,i-1)}_{p'_{j+1}}\rangle := A_i^\dagger C_i'^\dagger \, |\uparrow\rangle\langle\uparrow|_{\mathtt{u}} \, S|\rho^{(t,i)}_{p'_{j+1}}\rangle \quad \big(\forall j \in \{0,1,\ldots,i-2\},\ \forall p'_{j+1} \neq p_{j+1}\big).$$

Obviously, Eq. (7.23) satisfies the format of Eq. (7.4) with t updated to z and i updated to $i-1$. This finishes the proof of Type 3.

This finishes the proof of Lem. 14.

□

7.4 Prove Lem. 13: Re-Stating the Hybrids

As explained at the beginning of Sec. 7.3, we now introduce an alternative characterization of the hybrids. This perspective will prove valuable for the forthcoming mathematical derivations in Sec. 8 and 9.

Formally, we first define some auxiliary unitaries in Algo. 7.1. Subsequently, we present the new description for the hybrids in Game 7.2. Finally, in Corollary 4, we demonstrate the equivalence of this new description to the original one introduced in Sec. 7.2.

For a fixed (J, pat), recall that pat contain a fix sequence $\mathbf{p} = (p_1, \ldots, p_K)$. For such a fixed $\mathbf{p}$, we re-define (in Algo. 7.1) the unitaries $\{(A_k, B_k, C_k, D_k)\}_{k\in[K]}$ *that depend on this* $\mathbf{p}$. We emphasize that these unitaries were originally defined in Algo. 6.1. Now, we re-load them w.r.t. a fix $\mathbf{p}$. (Technically, we should have include $\mathbf{p}$ in the superscript, such as $A_k^{\mathbf{p}}$, to indicate the dependence on $\mathbf{p}$. But we choose to omit it for succinct notation.)

Algorithm 7.1: Re-Define Verifier's Unitaries for Fixed $(p_1, \ldots, p_K)$

For the $\mathbf{p} = (p_1, \ldots, p_K)$ contained in (J, pat), we keep the U_{gc}, U_{lc}, and A_k the same as in Algo. 6.1, but we re-define the unitaries B_k, C_k, $\ddot{C}_k$, and D_k $(\forall k \in [K])$ as follows:

1. Let $B_k = |p_1,\ldots,p_k\rangle\langle p_1,\ldots,p_k|_{\mathtt{p}_1\ldots\mathtt{p}_k} \otimes B_{k,1} + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle\langle p'_1,\ldots,p'_k|_{\mathtt{p}_1\ldots\mathtt{p}_k} \otimes I$. Note that $B_{k,1}$ is exactly the honest verifier's unitary V to generate message v_k (see Algo. 6.1), which acts non-trivially only on $\mathtt{v_k}$ and $\mathtt{w}$, and works as identity on other registers.
2. Let $C_k = |p_1,\ldots,p_k\rangle\langle p_1,\ldots,p_k|_{\mathtt{p}_1\ldots\mathtt{p}_k} \otimes C_{k,1} + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle\langle p'_1,\ldots,p'_k|_{\mathtt{p}_1\ldots\mathtt{p}_k} \otimes C_{k,0}$. Note that $C_{k,1}$ is the swap operator between registers $\mathtt{m}$ and $\mathtt{v_k}$, and $C_{k,0}$ is the swap operator between registers $\mathtt{m}$ and $\mathtt{t_k}$.
3. Let $\ddot{C}_k = |p_1,\ldots,p_k\rangle\langle p_1,\ldots,p_k|_{\mathtt{p}_1\ldots\mathtt{p}_k} \otimes I + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle\langle p'_1,\ldots,p'_k|_{\mathtt{p}_1\ldots\mathtt{p}_k} \otimes C_{k,0}$. (Recall the definition of $C_{k,0}$ from Item 2.)
4. Let $D_k = |p_1,\ldots,p_k\rangle\langle p_1,\ldots,p_k|_{\mathtt{p}_1\ldots\mathtt{p}_k} \otimes D_{k,1} + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle\langle p'_1,\ldots,p'_k|_{\mathtt{p}_1\ldots\mathtt{p}_k} \otimes I$. Note that $D_{k,1}$ is the local-counter increasing unitary U_{lc}, which acts non-trivially only on $\mathtt{lc}$.

It will be beneficial to monitor the registers on which the unitaries (in Algo. 7.1) have non-trivial effects. For the reader's ease, we summarize this in Table 1. For a unitary listed in the table, it operates as the identity on registers not indicated in its row.

Next, we re-state the hybrids $\{H_k^{(J,\mathsf{pat})}\}_{k\in[K]\cup\{0\}}$ in Game 7.2 and prove in Corollary 4 that they are indeed equivalent to the original version defined in Sec. 7.2.

Game 7.2: Re-Define the Hybrids $\{H_k^{(J,\mathsf{pat})}\}_{k\in[K]\cup\{0\}}$

Parameters. Use the same (J, pat) as defined Game 7.1. Use the unitaries defined in Algo. 7.1. We emphasize that this game inherits the "sub-normalized" nature of Game 7.1 for the fixed (J, pat)

Table 1: Registers on which the unitaries acts non-trivially

Unitary operators	Non-trivial registers	Unitary operators	Non-trivial registers
A_k	$\mathtt{p_k}, \mathtt{m}$	$\ddot{C}_k$	$\mathtt{p_1} \dots \mathtt{p_k}, \mathtt{t_k}, \mathtt{m}$
B_k	$\mathtt{p_1} \dots \mathtt{p_k}, \mathtt{v_k}, \mathtt{w}$	D_k	$\mathtt{p_1} \dots \mathtt{p_k}, \mathtt{lc}$
$B_{k,1}$	$\mathtt{v_k}, \mathtt{w}$	$D_{k,1}$	$\mathtt{lc}$
C_k	$\mathtt{p_1} \dots \mathtt{p_k}, \mathtt{v_k}, \mathtt{t_k}, \mathtt{m}$	U_{gc}	$\mathtt{gc}$
$C_{k,1}$	$\mathtt{v_k}, \mathtt{m}$	S	$\mathtt{m}, \mathtt{u}, \mathtt{s}$
$C_{k,0}$	$\mathtt{t_k}, \mathtt{m}$		

(this was already there in its original version defined in Sec. 7.2), where all the measurements are replaced with projectors to the corresponding value contained in pat.

Hybrid $H_k^{(J,\mathsf{pat})}$ ($k \in [K] \cup \{0\}$). The hybrid behaves as follows for all $i \in [K]$:

1. **When $\mathcal{S}$ makes the $\mathsf{sq}(i)$ Query:** Note that by definition, this is the first $\downarrow$-query that brings the global counter from $|i-1\rangle_{\mathtt{gc}}$ to $|i\rangle_{\mathtt{gc}}$. The hybrid behaves according to the value b_i contained in J:
   - If $b_i = 0$: answers this query by applying the following operator the overall state:
   $$D_i C_i B_i \, |p_1, \dots, p_i\rangle\langle p_1, \dots, p_i|_{\mathtt{p_1 \dots p_i}} \, A_i U_{gc},$$
   where recall that the values $p_1, \dots, p_i$ are specified in pat.
   - If $b_i = 1$: answers this query by applying the following operator the overall state:
   $$C_{i,0} \, |p_1, \dots, p_i\rangle\langle p_1, \dots, p_i|_{\mathtt{p_1 \dots p_i}} \, A_i U_{gc},$$
   where recall that the values $p_1, \dots, p_i$ are specified in pat.
2. **When $\mathcal{S}$ makes a qeury between the $\mathsf{sq}(j-1)$ and $\mathsf{sq}(j)$ queries:** it behaves based on the query type and the value of the global counter value $z \in [K] \cup \{0\}$:
   (a) **For $\downarrow$-query and $z < k$:** This corresponds to a $\ddot{V}$-query bringing the global counter from $|z\rangle_{\mathtt{gc}}$ to $|z+1\rangle_{\mathtt{gc}}$. The hybrid answers it by applying $D_{z+1}\ddot{C}_{z+1}A_{z+1}U_{gc}$ to the overall state.
   (b) **For $\downarrow$-query and $z \geq k$:** This corresponds to a $\widetilde{V}$-query bringing the global counter from $|z\rangle_{\mathtt{gc}}$ to $|z+1\rangle_{\mathtt{gc}}$. The hybrid answers it by applying $D_{z+1}C_{z+1}B_{z+1}A_{z+1}U_{gc}$ to the overall state.
   (c) **For $\uparrow$-query and $z \leq k$:** This corresponds to a $\ddot{V}^\dagger$-query bringing the global counter from $|z\rangle_{\mathtt{gc}}$ to $|z-1\rangle_{\mathtt{gc}}$. The hybrid answers it by applying $U_{gc}^\dagger A_z^\dagger \ddot{C}_z^\dagger D_z^\dagger$ to the overall state.
   (d) **For $\uparrow$-query and $z > k$:** This corresponds to a $V^\dagger$-query bringing the global counter from $|z\rangle_{\mathtt{gc}}$ to $|z-1\rangle_{\mathtt{gc}}$. The hybrid answers it by applying $U_{gc}^\dagger A_z^\dagger B_z^\dagger C_z^\dagger D_z^\dagger$ to the overall state.

Corollary 4 (of Lem. 14). *The hybrids $\{H_k^{(J,\mathsf{pat})}\}_{k\in[K]\cup\{0\}}$ defined in Game 7.2 are equivalent to those defined in Sec. 7.2.*

Proof of Corollary 4. In the following, consider a fixed pair (J, pat) and hybrid $H_k^{(J,\mathsf{pat})}$.

The main idea of this proof as follows. First, note that the original $H_k^{(J,\mathsf{pat})}$ (in Sec. 7.2) makes use of the $\widetilde{V}$ and $\ddot{V}$, which by definition determine what to do by comparing the global counter and the local counter (see Algo. 6.1 and 6.4). On the other hand, Lem. 14 provides a full characterization of the relation among the global counter, the local counters, and the $\mathtt{p}_1 \dots \mathtt{p}_K$ registers throughout the execution, which essentially allows the hybrid to determine what to do by checking the (classical) value contained in the global counter only. This can be easily seen by comparing the original $H_k^{(J,\mathsf{pat})}$ (in Sec. 7.2) and the one defined in Game 7.2 (together with the re-named unitaries in Algo. 7.1). In the following, we show why they are equivalent for a representative type of query as an example. Other types of queries can be argued in the same manner.

A Representative Example. Consider the case when $\mathcal{S}$ makes the $\mathsf{sq}(i)$ query for some $i \in [K]$. By definition (and Rmk. 11), this is a $\widetilde{V}$ query that brings the global counter from $|i-1\rangle_{\mathtt{gc}}$ to $|i\rangle_{\mathtt{gc}}$. This query is handled as follows in the original $H_k^{(J,\mathsf{pat})}$ (in Sec. 7.2) according to the definition of $\widetilde{\mathcal{V}}$ in Algo. 6.2:

1. U_{gc} is first applied to increase the global counter from $|i-1\rangle_{\mathtt{gc}}$ to $|i\rangle_{\mathtt{gc}}$.
2. The A_i is applied;
3. Then the query to H will be made, which invokes the projection on the registers $\mathtt{p}_1 \dots \mathtt{p}_i$ to value $|p_1, \dots, p_i\rangle_{\mathtt{p}_1 \dots \mathtt{p}_i}$. (Recall that we are currently in the "sub-normalized" game $H_k^{(J,\mathsf{pat})}$. Thus, the measurement on register $\mathtt{p}_1 \dots \mathtt{p}_i$ has been replaced with this projection.) It then follows from Lem. 14 that this projection will retain only the "branch" corresponding to $|i-1\rangle_{\mathtt{lc}}$ and "kill" all other branches in the superposition. Thus, the current overall state must be of the following format
$$|i\rangle_{\mathtt{gc}}|i-1\rangle_{\mathtt{lc}}|p_1, \dots, p_{i-1}\rangle_{\mathtt{p}_1,\dots,\mathtt{p}_{i-1}}|p_i\rangle_{\mathtt{p}_i}|\rho\rangle. \tag{7.24}$$
4. Next, the behavior depends on the the local counter value (in superposition) and the value b_i specified in pat. In particular:
   (a) If $b_i = 0$: In this case, first the oracle will be programmed and then the query will be answered using the programmed oracle. By definition of $\widetilde{V}$, the operator $D_iC_iB_i$ (defined in Algo. 6.1) will be applied to Expression (7.24).
   (b) If $b_i = 1$: In this case, first the query will be answered and then the oracle is programmed. Note that before the oracle is programmed, it must hold that $H(p_1, \dots, p_i) = 0$. Therefore, by definition of $\widetilde{V}$, only the operator $C_{i,0}$ will be effectively applied.

In summary, the above four steps can be unified as

- If $b_i = 0$: apply $D_iC_iB_i\,|p_1, \dots, p_i\rangle\langle p_1, \dots, p_i|_{\mathtt{p}_1 \dots \mathtt{p}_i}\,A_iU_{gc}$.
- If $b_i = 1$: apply $C_{i,0}\,|p_1, \dots, p_i\rangle\langle p_1, \dots, p_i|_{\mathtt{p}_1 \dots \mathtt{p}_i}\,A_iU_{gc}$.

This is exactly what happens in Game 7.2.

This completes the proof of Corollary 4.

□

8 Proving Lem. 13 (Warm-Up Case)

In this section, we prove Lem. 13 for the special case $K = 2$.

In this setting, there are only three hybrids to consider, namely $H_0^{(J,\mathsf{pat})}$, $H_1^{(J,\mathsf{pat})}$, and $H_2^{(J,\mathsf{pat})}$. To provide better intuition, we illustrate them in Fig. 2, with difference between adjacent hybrids

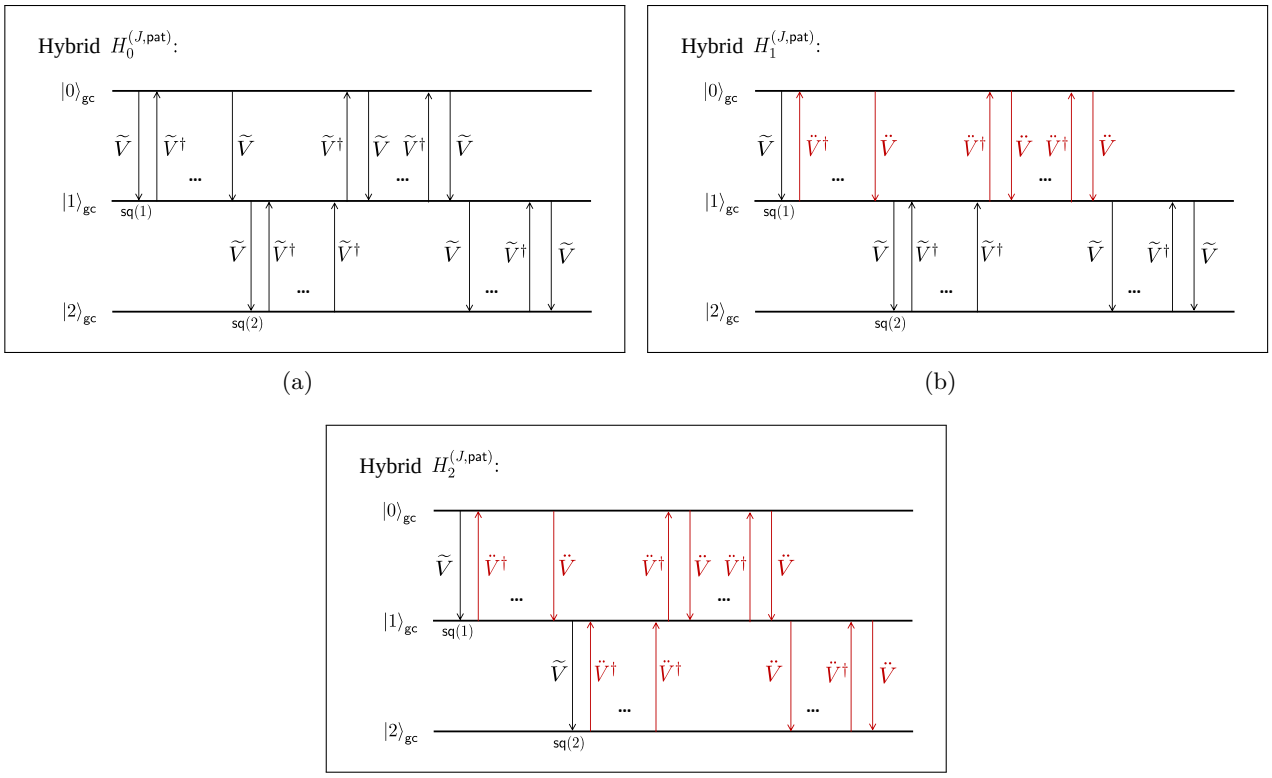


Fig. 2: Illustration of Hybrids $H_0^{(J,\mathsf{pat})}$, $H_1^{(J,\mathsf{pat})}$, and $H_2^{(J,\mathsf{pat})}$

highlighted in red color. In more detail, Fig. 2 illustrates these three hybrids for some particular (J, pat):

- In hybrid $H_0^{(J,\mathsf{pat})}$, it can be seen from Fig. 2a that all the $\downarrow$-queries are answered using $\widetilde{V}$ and all the $\uparrow$-queries are answered using $\widetilde{V}^\dagger$.
- The hybrid $H_1^{(J,\mathsf{pat})}$ shown in Fig. 2b is identical to hybrid $H_0^{(J,\mathsf{pat})}$ except that the $\downarrow$-queries that bring the global counter from $|0\rangle_{\mathsf{gc}}$ to $|1\rangle_{\mathsf{gc}}$ (except for the $\mathsf{sq}(1)$ query) are answered using the "dummy-version" unitary$\ddot{V}$, and the $\uparrow$-queries that bring the global counter from $|1\rangle_{\mathsf{gc}}$ to $|0\rangle_{\mathsf{gc}}$ are answered using the "dummy-version" unitary $\ddot{V}^\dagger$.
- The hybrid $H_2^{(J,\mathsf{pat})}$ shown in Fig. 2c is identical to hybrid $H_1^{(J,\mathsf{pat})}$ except that the $\downarrow$-queries that bring the global counter from $|1\rangle_{\mathsf{gc}}$ to $|2\rangle_{\mathsf{gc}}$ (except for the $\mathsf{sq}(2)$ query) are answered using the "dummy-version" unitary$\ddot{V}$, and the $\uparrow$-queries that bring the global counter from $|2\rangle_{\mathsf{gc}}$ to $|1\rangle_{\mathsf{gc}}$ are answered using the "dummy-version" unitary $\ddot{V}^\dagger$.

For these hybrids, to prove Lem. 13, it now suffices to establish the following Lem. 15 and 16.

Lemma 15. *For all (J, pat) satisfying the requirements in Game 7.1, it holds that*

$$\Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow H_0^{(J,\mathsf{pat})}\right] = \Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow H_1^{(J,\mathsf{pat})}\right].$$

Lemma 16. *For all (J, pat) satisfying the requirements in Game 7.1, it holds that*

$$\Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow H_1^{(J,\mathsf{pat})}\right] = \Pr\left[\mathsf{Pred}(\rho) = 1 \ : \ (\mathbf{p}, \rho) \leftarrow H_2^{(J,\mathsf{pat})}\right].$$

In the sequel, we prove these two lemmas in Sec. 8.1 and 8.2 respectively. We emphasize that in the following, we use the notation established in Sec. 7.4.

8.1 Proving Lem. 15

We present a lemma (Lem. 17) that characterizes how the overall state evolves in $H_0^{(J,\mathsf{pat})}$ and $H_1^{(J,\mathsf{pat})}$. This lemma is the major workhorse that implies Lem. 15. In the sequel, we first show how to establish Lem. 15 assuming that Lem. 17 holds. After that, we will present the proof of Lem. 17.

Lemma 17 (Invariance in $H_0^{(J,\mathsf{pat})}$ and $H_1^{(J,\mathsf{pat})}$). *Assume that during the execution of $H_0^{(J,\mathsf{pat})}$ (and $H_1^{(J,\mathsf{pat})}$), the global counter reaches value 1 for T times in total. For each $t \in [T]$, there exist (possibly sub-normalized) pure states $\{|\rho_{p_1'}^{(t)}\rangle\}_{p_1' \in \{0,1\}^\ell}$ so that the following holds: in hybrid $H_0^{(J,\mathsf{pat})}$ (resp. $H_1^{(J,\mathsf{pat})}$), when the global counter reaches value 1 for the t-th time, the overall state can be written as $|\phi^{(t)}\rangle$ (resp. $|\psi^{(t)}\rangle$) defined as follows*

$$|\phi^{(t)}\rangle = |p_1\rangle_{\mathsf{p}_1}|\rho_{p_1}^{(t)}\rangle \ + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathsf{p}_1} C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger |\rho_{p_1'}^{(t)}\rangle, \tag{8.1}$$

$$|\psi^{(t)}\rangle = |p_1\rangle_{\mathsf{p}_1}|\rho_{p_1}^{(t)}\rangle \ + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathsf{p}_1} C_{1,0} |\rho_{p_1'}^{(t)}\rangle. \tag{8.2}$$

where the p_1 is defined in pat *and the unitaries $B_{1,1}$, $C_{1,0}$, and $C_{1,1}$ are as defined in Algo. 7.1.*

Finishing the Proof of Lem. 15. Let us consider the last time when the global counter reaches value 1. By Lem. 17, the overall states in $H_0^{(J,\mathsf{pat})}$ and $H_1^{(J,\mathsf{pat})}$ would be of the following format respectively

$$|\phi^{(T)}\rangle = \overbrace{|p_1\rangle_{\mathsf{p}_1}|\rho_{p_1}^{(T)}\rangle}^{\mathsf{Good}} \ + \overbrace{\sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathsf{p}_1} C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger |\rho_{p_1'}^{(T)}\rangle}^{\mathsf{Bad}}, \tag{8.3}$$

$$|\psi^{(T)}\rangle = \overbrace{|p_1\rangle_{\mathsf{p}_1}|\rho_{p_1}^{(T)}\rangle}^{\mathsf{Good}} \ + \overbrace{\sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathsf{p}_1} C_{1,0} |\rho_{p_1'}^{(T)}\rangle}^{\mathsf{Bad}}. \tag{8.4}$$

Since this is the last time the global counter reaches value 1, the two hybrids $H_0^{(J,\mathsf{pat})}$ and $H_1^{(J,\mathsf{pat})}$ by definition behave identically after that (see Fig. 2). That is, the states $|\phi^{(T)}\rangle$ and $|\psi^{(T)}\rangle$ will go through the same quantum procedure (let us denote it by $\mathcal{F}$) and finally measured by the predicate Pred.

Next, note that the states $|\phi^{(T)}\rangle$ and $|\psi^{(T)}\rangle$ share the identical "branch" corresponding to $|p_1\rangle_{\mathsf{p}_1}$ (labeled as Good in the above equations). Also, the Bad branch will not contribute any shares to the accepting probability of the predicate Pred—this is because the quantum procedure $\mathcal{F}$ will not change the fact that $p_1' \neq p_1$ for the Bad branch, and recall that, as a consequence of Lem. 14, Pred will output 0 if a branch in the superposition does not have any non-zero amplitude for $|p_1\rangle_{\mathsf{p}_1}$.

In summary, only the Good branch matters for the remaining execution (i.e., that after the last time when the global counter reaches value 1), contributing to the event $\mathsf{Pred}(\rho) = 1$. Therefore, Lem. 15 holds simply because $|\phi^{(T)}\rangle$ and $|\psi^{(T)}\rangle$ share the same Good branch.

Proving Lem. 17. Now, the only task remaining is to prove Lem. 17. We will demonstrate this lemma through mathematical induction on the number $t \in [T]$, indicating the time at which the global counter reaches the value 1. Throughout this proof, we will monitor the evolution of the overall states in both $H_0^{(J,\mathsf{pat})}$ and $H_1^{(J,\mathsf{pat})}$ simultaneously. This is done in Sec. 8.1.1 and 8.1.2 respectively.

8.1.1 Base Case ($t = 1$)

We first derive how the overall state evolves in $H_0^{(J,\mathsf{pat})}$.

This case corresponds to the very first time the global counter reaches 1. By definition, this is due to the $\mathsf{sq}(1)$ query. In more detail, $\mathcal{S}$ will first apply her local unitary S, followed by the projector $|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}$. Next, according to the notation in Game 7.2,

- If $b_1 = 0$, then the operator $D_1C_1B_1\,|p_1\rangle\langle p_1|_{\mathtt{p}_1}\,A_1U_{gc}$ will be applied;
- If $b_1 = 1$, then the operator $C_{1,0}\,|p_1\rangle\langle p_1|_{\mathtt{p}_1}\,A_1U_{gc}$ will be applied;

In the following, we show the proof for $b_1 = 0$ only; the other case (i.e., $b_1 = 1$) can be established using the same argument.

If we assume that the initial state across all the registers are $|\rho\rangle$, then the state $|\phi^{(1)}\rangle$ will be as follows:

$$\begin{aligned}
|\phi^{(1)}\rangle &= D_1C_1B_1\,|p_1\rangle\langle p_1|_{\mathtt{p}_1}\,A_1U_{gc}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\rho\rangle \\
&= D_1C_1B_1|p_1\rangle_{\mathtt{p}_1}|\rho_{p_1}\rangle && (8.5)\\
&= |p_1\rangle_{\mathtt{p}_1}D_{1,1}C_{1,1}B_{1,1}|\rho_{p_1}\rangle && (8.6)\\
&= |p_1\rangle_{\mathtt{p}_1}|\rho_{p_1}^{(1)}\rangle, && (8.7)
\end{aligned}$$

where

- Eq. (8.5) follows by defining $|\rho_{p_1}\rangle \coloneqq \langle p_1|_{\mathtt{p}_1}A_1U_{gc}\,|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}\,S|\rho\rangle$;
- Eq. (8.6) follows from the definition of B_1 C_1, and D_1 (see Algo. 7.1);
- Eq. (8.7) follows by defining $|\rho_{p_1}^{(1)}\rangle \coloneqq D_{1,1}C_{1,1}B_{1,1}|\rho_{p_1}\rangle$.

It is straightforward that the $|\phi^{(1)}\rangle$ shown in Eq. (8.7) satisfies the format shown in Eq. (8.1) when $t = 1$.

Also, note that the hybrids $H_0^{(J,\mathsf{pat})}$ and $H_1^{(J,\mathsf{pat})}$ are identical so far and thus $|\phi^{(1)}\rangle = |\psi^{(1)}\rangle$. Therefore, it follows from Eq. (8.7) that

$$|\psi^{(1)}\rangle = |p_1\rangle_{\mathtt{p}_1}|\rho_{p_1}^{(1)}\rangle.$$

Such a $|\psi^{(1)}\rangle$ satisfies the format shown in Eq. (8.2) with $t = 1$ as well.

This finishes the proof for the base case $t = 1$.

8.1.2 Induction Step ($t \geq 2$)

We assume that $|\phi^{(t-1)}\rangle$ and $|\psi^{(t-1)}\rangle$ satisfy Lem. 17, and show in the following that Lem. 17 holds when the global counter reaches 1 for the t-th time.

We establish this claim by considering the following MECE (mutually exclusive and collectively exhaustive) cases:

1. **Case 1:** The t-th arrival at value 1 is due to an immediate $\uparrow\downarrow$ after the $(t-1)$-th arrival. That is, after the $(t-1)$-th arrival, $\mathcal{S}$ first makes a $\uparrow$ query, bringing the global counter to 0, and then makes an $\downarrow$ query, bringing the global counter back to 1.
2. **Case 2:** The t-th arrival at value 1 is due to an immediate $\downarrow\uparrow$ after the $(t-1)$-th arrival. That is, after the $(t-1)$-th arrival, $\mathcal{S}$ first makes a $\downarrow$ query, bringing the global counter to 2, and then makes an $\uparrow$ query, bringing the global counter back to 1.

8.1.2.1 Proof for Case 1

We first describe formally how $|\phi^{(t-1)}\rangle$ (resp. $|\psi^{(t-1)}\rangle$) evolves into $|\phi^{(t)}\rangle$ (resp. $|\psi^{(t)}\rangle$) in Case 1:

1. $\mathcal{S}$'s local unitary S is applied, followed by the projector $|\uparrow\rangle\langle\uparrow|_{\mathtt{u}}$. Note that this step is identical for both $H_0^{(J,\mathsf{pat})}$ and $H_1^{(J,\mathsf{pat})}$.

   Remark 12 (Hiding the Projector on Register $\mathtt{u}$*).* In the sequel, we overload the notation S to think of it as already including the projection $|\uparrow\rangle\langle\uparrow|_{\mathtt{u}}$, and thus do not spell the $|\uparrow\rangle\langle\uparrow|_{\mathtt{u}}$ out explicitly. We remark that this will not affect our proof, because what matters for S in the following proof is the registers on which it operates, and the original S already operates non-trivially on register $\mathtt{u}$ (see Table 1).

2. An $\uparrow$-query is made, bringing the global counter from 1 to 0. According to the notation in Game 7.2:
   - In $H_0^{(J,\mathsf{pat})}$, this corresponds to applying $U_{gc}^\dagger A_1^\dagger B_1^\dagger C_1^\dagger D_1^\dagger$;
   - In $H_1^{(J,\mathsf{pat})}$, this corresponds to applying $U_{gc}^\dagger A_1^\dagger \ddot{C}_1^\dagger D_1^\dagger$.
3. $\mathcal{S}$ will apply her local operation S again (followed by $|\downarrow\rangle\langle\downarrow|_{\mathtt{u}}$ which we hide as per Rmk. 12). Note that this step is again identical for both $H_0^{(J,\mathsf{pat})}$ and $H_1^{(J,\mathsf{pat})}$.
4. An $\downarrow$-query is made, bringing the global counter from 0 back to 1, which is the global counter's t-th arrival at value 1. According to the notation in Game 7.2:
   - In $H_0^{(J,\mathsf{pat})}$, this corresponds to applying $D_1 C_1 B_1 A_1 U_{gc}$;
   - In $H_1^{(J,\mathsf{pat})}$, this corresponds to applying $D_1 \ddot{C}_1 A_1 U_{gc}$.

It follows from the above description (and Rmk. 12) that the states $|\phi^{(t)}\rangle$ and $|\psi^{(t)}\rangle$ can be written as:

$$|\phi^{(t)}\rangle = D_1 C_1 B_1 A_1 U_{gc} S U_{gc}^\dagger A_1^\dagger B_1^\dagger C_1^\dagger D_1^\dagger S |\phi^{(t-1)}\rangle, \tag{8.8}$$

$$|\psi^{(t)}\rangle = D_1 \ddot{C}_1 A_1 U_{gc} S U_{gc}^\dagger A_1^\dagger \ddot{C}_1^\dagger D_1^\dagger S |\psi^{(t-1)}\rangle. \tag{8.9}$$

High-Level Idea for the Sequel. Recall that our eventual goal is to prove that the states $|\phi^{(t)}\rangle$ and $|\psi^{(t)}\rangle$ are of the format shown in Eq. (8.1) and (8.2) in Lem. 17. At a high level, we prove it by applying Lem. 6 to Eq. (8.8) and (8.9). But we first need to perform some preparation work, putting Eq. (8.8) and (8.9) into a format that is more "compatible" with Lem. 6. In the sequel, we first perform the preparation work in Claims 7 and 8. Then, we show on Page 76 how to use Claims 7 and 8 to complete the proof for Case 1.

Claim 7. *There exist (possibly sub-normalized) pure states* $\{\dot{\rho}_{p_1'}^{(t-1)}\}_{p_1'\in\{0,1\}^\ell}$ *so that the following holds*

$$|\phi^{(t)}\rangle = D_1C_1B_1U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger B_1^\dagger C_1^\dagger\Big(|p_1\rangle_{\mathtt{p}_1}|\dot{\rho}_{p_1}^{(t-1)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}}|p_1'\rangle_{\mathtt{p}_1}C_{1,0}B_{1,1}^\dagger C_{1,1}^\dagger|\dot{\rho}_{p_1'}^{(t-1)}\rangle\Big) \tag{8.10}$$

$$|\psi^{(t)}\rangle = D_1\ddot{C}_1U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger \ddot{C}_1^\dagger\Big(|p_1\rangle_{\mathtt{p}_1}|\dot{\rho}_{p_1}^{(t-1)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}}|p_1'\rangle_{\mathtt{p}_1}C_{1,0}|\dot{\rho}_{p_1'}^{(t-1)}\rangle\Big), \tag{8.11}$$

where $S^{\mathtt{p}_1/\mathtt{m}}$ *is identical to* S *except that it treats register* $\mathtt{p}_1$ *as register* $\mathtt{m}$.

Proof of Claim 7. First, notice that

$$\begin{aligned}
A_1U_{gc}SU_{gc}^\dagger A_1^\dagger &= A_1U_{gc}SA_1^\dagger U_{gc}^\dagger && (8.12)\\
&= A_1U_{gc}A_1^\dagger S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger && (8.13)\\
&= A_1A_1^\dagger U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger && (8.14)\\
&= U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger, && (8.15)
\end{aligned}$$

where

- Eq. (8.12) and (8.14) follows from the fact that U_{gc} acts on different registers from A_1 (see Table 1);
- Eq. (8.13) from the fact that S acts as the identity operator on $\mathtt{p}_1$ (see Table 1) and that A_1 is nothing but a swap operator between $\mathtt{m}$ and $\mathtt{p}_1$ (see Algo. 6.1).

Proving Eq. (8.10). We now show the derivation for $|\phi^{(t)}\rangle$:

$$\begin{aligned}
&|\phi^{(t)}\rangle\\
&= D_1C_1B_1A_1U_{gc}SU_{gc}^\dagger A_1^\dagger B_1^\dagger C_1^\dagger D_1^\dagger S|\phi^{(t-1)}\rangle && (8.16)\\
&= D_1C_1B_1U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger B_1^\dagger C_1^\dagger D_1^\dagger S|\phi^{(t-1)}\rangle && (8.17)\\
&= D_1C_1B_1U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger B_1^\dagger C_1^\dagger D_1^\dagger S\Big(|p_1\rangle_{\mathtt{p}_1}|\rho_{p_1}^{(t-1)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}}|p_1'\rangle_{\mathtt{p}_1}C_{1,0}B_{1,1}^\dagger C_{1,1}^\dagger|\rho_{p_1'}^{(t-1)}\rangle\Big) && (8.18)\\
&= D_1C_1B_1U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger B_1^\dagger C_1^\dagger D_1^\dagger\Big(|p_1\rangle_{\mathtt{p}_1}S|\rho_{p_1}^{(t-1)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}}|p_1'\rangle_{\mathtt{p}_1}SC_{1,0}B_{1,1}^\dagger C_{1,1}^\dagger|\rho_{p_1'}^{(t-1)}\rangle\Big) && (8.19)\\
&= D_1C_1B_1U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger B_1^\dagger C_1^\dagger\Big(|p_1\rangle_{\mathtt{p}_1}D_{1,1}^\dagger S|\rho_{p_1}^{(t-1)}\rangle+\\
&\qquad\qquad\sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}}|p_1'\rangle_{\mathtt{p}_1}SC_{1,0}B_{1,1}^\dagger C_{1,1}^\dagger|\rho_{p_1'}^{(t-1)}\rangle\Big) && (8.20)\\
&= D_1C_1B_1U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger B_1^\dagger C_1^\dagger\Big(|p_1\rangle_{\mathtt{p}_1}D_{1,1}^\dagger S|\rho_{p_1}^{(t-1)}\rangle+\\
&\qquad\qquad\sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}}|p_1'\rangle_{\mathtt{p}_1}C_{1,0}S^{\mathtt{t}_1/\mathtt{m}}B_{1,1}^\dagger C_{1,1}^\dagger|\rho_{p_1'}^{(t-1)}\rangle\Big) && (8.21)\\
&= D_1C_1B_1U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger B_1^\dagger C_1^\dagger\Big(|p_1\rangle_{\mathtt{p}_1}D_{1,1}^\dagger S|\rho_{p_1}^{(t-1)}\rangle+
\end{aligned}$$

$$\sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger S^{\mathtt{t}_1/\mathtt{m}} |\rho_{p_1'}^{(t-1)}\rangle \Bigg) \tag{8.22}$$

$$= D_1 C_1 B_1 U_{gc} S^{\mathtt{p}_1/\mathtt{m}} U_{gc}^\dagger B_1^\dagger C_1^\dagger \Bigg(|p_1\rangle_{\mathtt{p}_1} |\dot{\rho}_{p_1}^{(t-1)}\rangle \; + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger |\dot{\rho}_{p_1'}^{(t-1)}\rangle \Bigg), \tag{8.23}$$

where

- Eq. (8.16) follows from Eq. (8.8);
- Eq. (8.17) follows from Eq. (8.15);
- Eq. (8.18) follows from our induction assumption;
- Eq. (8.19) from from the fact that S acts as the identity operator on $\mathtt{p}_1$ (see Table 1);
- Eq. (8.20) from from the definition of D_1 (see Algo. 7.1);
- Eq. (8.21) from the fact that S acts as the identity operator on $\mathtt{t}_1$ (see Table 1) and $C_{1,0}$ is nothing but a swap operator between $\mathtt{t}_1$ and $\mathtt{m}$ (see Algo. 7.1); (Note that $S^{\mathtt{t}_1/\mathtt{m}}$ is defined to be an operator that is identical to S except that it treats $\mathtt{t}_1$ as $\mathtt{m}$.)
- Eq. (8.22) follows from the fact that $S^{\mathtt{t}_1/\mathtt{m}}$ acts non-trivially on different registers from $B_{1,1}$ and $C_{1,1}$ (see Table 1);
- Eq. (8.23) follows by defining

$$|\dot{\rho}_{p_1}^{(t-1)}\rangle \coloneqq D_{1,1}^\dagger S |\rho_{p_1}^{(t-1)}\rangle \quad \text{and} \quad |\dot{\rho}_{p_1'}^{(t-1)}\rangle \coloneqq S^{\mathtt{t}_1/\mathtt{m}} |\rho_{p_1'}^{(t-1)}\rangle \;\; (\forall p_1' \in \{0,1\}^\ell \setminus \{p_1\}). \tag{8.24}$$

Eq. (8.23) finishes the proof of Eq. (8.10) in Claim 7.

Proving Eq. (8.11). We now show the derivation for $|\psi^{(t)}\rangle$. This is almost identical to the above proof for Eq. (8.10). Nevertheless, we present it for the sake of completeness.

$$|\psi^{(t)}\rangle = D_1 \ddot{C}_1 A_1 U_{gc} S U_{gc}^\dagger A_1^\dagger \ddot{C}_1^\dagger D_1^\dagger S |\psi^{(t-1)}\rangle \tag{8.25}$$

$$= D_1 \ddot{C}_1 U_{gc} S^{\mathtt{p}_1/\mathtt{m}} U_{gc}^\dagger \ddot{C}_1^\dagger D_1^\dagger S |\psi^{(t-1)}\rangle \tag{8.26}$$

$$= D_1 \ddot{C}_1 U_{gc} S^{\mathtt{p}_1/\mathtt{m}} U_{gc}^\dagger \ddot{C}_1^\dagger D_1^\dagger S \Bigg(|p_1\rangle_{\mathtt{p}_1} |\rho_{p_1}^{(t-1)}\rangle \; + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0} |\rho_{p_1'}^{(t-1)}\rangle \Bigg) \tag{8.27}$$

$$= D_1 \ddot{C}_1 U_{gc} S^{\mathtt{p}_1/\mathtt{m}} U_{gc}^\dagger \ddot{C}_1^\dagger \Bigg(|p_1\rangle_{\mathtt{p}_1} D_{1,1}^\dagger S |\rho_{p_1}^{(t-1)}\rangle \; + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0} S^{\mathtt{t}_1/\mathtt{m}} |\rho_{p_1'}^{(t-1)}\rangle \Bigg) \tag{8.28}$$

$$= D_1 \ddot{C}_1 U_{gc} S^{\mathtt{p}_1/\mathtt{m}} U_{gc}^\dagger \ddot{C}_1^\dagger \Bigg(|p_1\rangle_{\mathtt{p}_1} |\dot{\rho}_{p_1}^{(t-1)}\rangle \; + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0} |\dot{\rho}_{p_1'}^{(t-1)}\rangle \Bigg), \tag{8.29}$$

where

- Eq. (8.25) follows from Eq. (8.9);
- Eq. (8.26) follows from Eq. (8.15);
- Eq. (8.27) follows from our induction assumption;
- Eq. (8.28) follows from a similar argument as we did to derive Eq. (8.22) from Eq. (8.18);
- Eq. (8.29) follows from *the same definitions* of $|\dot{\rho}_{p_1}^{(t-1)}\rangle$ and $|\dot{\rho}_{p_1'}^{(t-1)}\rangle$ as shown in Expression (8.24).

Eq. (8.29) finishes the proof of Eq. (8.11) in Claim 7.

This completes the proof of Claim 7.

□

Claim 8. *Let $S^{\mathtt{p}_1/\mathtt{m}}$ and $\{\dot{\rho}^{(t-1)}_{p'_1}\}_{p'_1\in\{0,1\}^\ell}$ be as defined in Claim 7. Let*

$$|\gamma_0^{(t-1)}\rangle := |p_1\rangle_{\mathtt{p}_1}|\dot{\rho}^{(t-1)}_{p_1}\rangle + \sum_{p'_1\in\{0,1\}^\ell\setminus\{p_1\}} |p'_1\rangle_{\mathtt{p}_1} C_{1,0}B^\dagger_{1,1}C^\dagger_{1,1}|\dot{\rho}^{(t-1)}_{p'_1}\rangle \tag{8.30}$$

$$|\gamma_1^{(t-1)}\rangle := |p_1\rangle_{\mathtt{p}_1}|\dot{\rho}^{(t-1)}_{p_1}\rangle + \sum_{p'_1\in\{0,1\}^\ell\setminus\{p_1\}} |p'_1\rangle_{\mathtt{p}_1} C_{1,0}|\dot{\rho}^{(t-1)}_{p'_1}\rangle \tag{8.31}$$

$$|\gamma_0^{(t)}\rangle := C_1B_1U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U^\dagger_{gc}B^\dagger_1C^\dagger_1|\gamma_0^{(t-1)}\rangle \tag{8.32}$$

$$|\gamma_1^{(t)}\rangle := \ddot{C}_1U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U^\dagger_{gc}\ddot{C}^\dagger_1|\gamma_1^{(t-1)}\rangle. \tag{8.33}$$

Then, there exist (possibly sub-normalized) pure states $\{|\dot{\rho}^{(t)}_{p'_1}\rangle\}_{p'_1\in\{0,1\}^\ell}$ so that the following holds:

$$|\gamma_0^{(t)}\rangle = |p_1\rangle_{\mathtt{p}_1}|\dot{\rho}^{(t)}_{p_1}\rangle + \sum_{p'_1\in\{0,1\}^\ell\setminus\{p_1\}} |p'_1\rangle_{\mathtt{p}_1} C_{1,0}B^\dagger_{1,1}C^\dagger_{1,1}|\dot{\rho}^{(t)}_{p'_1}\rangle \tag{8.34}$$

$$|\gamma_1^{(t)}\rangle = |p_1\rangle_{\mathtt{p}_1}|\dot{\rho}^{(t)}_{p_1}\rangle + \sum_{p'_1\in\{0,1\}^\ell\setminus\{p_1\}} |p'_1\rangle_{\mathtt{p}_1} C_{1,0}|\dot{\rho}^{(t)}_{p'_1}\rangle. \tag{8.35}$$

Proof of Claim 8. This claim follows from an application of Lem. 6, with the notation correspondence listed in Table 2. We provide a detailed explanation below.

Table 2: Notation Correspondence between Lem. 6 and Claim 8

Registers		Operators		Random Variables	
In Lem. 6	In Claim 8	In Lem. 6	In Claim 8	In Lem. 6	In Claim 8
$\mathtt{a}$	$\mathtt{p}_1$	W_1	$C_{1,1}$	$\|a\rangle_{\mathtt{a}}$	$\|p_1\rangle_{\mathtt{p}_1}$
$\mathtt{m}$	$\mathtt{m}$	W_0	$C_{1,0}$	$\|a'\rangle_{\mathtt{a}}$	$\|p'_1\rangle_{\mathtt{p}_1}$
$\mathtt{t}$	$\mathtt{t}_1$	W	C_1	$\|\rho^{(\mathsf{in})}_a\rangle_{\mathtt{mtso}}$	$\|\dot{\rho}^{(t-1)}_{p_1}\rangle$
$\mathtt{s}$	$\mathtt{u}$, $\mathtt{s}$, $\mathtt{gc}$	$\widetilde{W}$	$\ddot{C}_1$	$\|\rho^{(\mathsf{in})}_{a'}\rangle_{\mathtt{mtso}}$	$\|\dot{\rho}^{(t-1)}_{p'_1}\rangle$
$\mathtt{o}$	other registers	U_1	$B_{1,1}$	$\|\eta^{(\mathsf{in})}_0\rangle$	$\|\gamma^{(t-1)}_0\rangle$
		U	B_1	$\|\eta^{(\mathsf{in})}_1\rangle$	$\|\gamma^{(t-1)}_1\rangle$
		S	$U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U^\dagger_{gc}$	$\|\rho^{(\mathsf{out})}_a\rangle_{\mathtt{mtso}}$	$\|\dot{\rho}^{(t)}_{p_1}\rangle$
				$\|\rho^{(\mathsf{out})}_{a'}\rangle_{\mathtt{mtso}}$	$\|\dot{\rho}^{(t)}_{p'_1}\rangle$
				$\|\eta^{(\mathsf{out})}_0\rangle$	$\|\gamma^{(t)}_0\rangle$
				$\|\eta^{(\mathsf{out})}_1\rangle$	$\|\gamma^{(t)}_1\rangle$

First, we argue that the premises in Lem. 6 are satisfied with the notation listed in Table 2:

– Lem. 6 requires that W_1 should work as the identity operator on register $\mathtt{s}$. In terms of the Claim 8 notation, this is satisfied by $C_{1,1}$ (playing the role of W_1) who works as identity on registers $\mathtt{u}$, $\mathtt{s}$, and $\mathtt{gc}$ (playing the role of registers $\mathtt{s}$). (Recall $C_{1,1}$ from Table 1.)

– Lem. 6 requires that W_0 should be a swap operator between $\mathtt{m}$ and $\mathtt{t}$. In terms of the Claim 8 notation, this is satisfied by $C_{1,0}$ (playing the role of W_0), who is a swap operator between registers $\mathtt{m}$ and $\mathtt{t}_1$ (playing the role of $\mathtt{m}$ and $\mathtt{t}$ respectively). (Recall $C_{1,0}$ from Algo. 7.1.)

– Lem. 6 requires that $\widetilde{W}$ is the identity operator on branch $|a\rangle_{\mathtt{a}}$ and is identical to W_0 on branches $|a'\rangle_{\mathtt{a}}$ with $a' \neq a$. In terms of the Claim 8 notation, this is satisfied by $\ddot{C}_1$ (playing the role of $\widetilde{W}$), who is the identity operator on branch $|p_1\rangle_{\mathtt{p}_1}$ (playing the role of $|a\rangle_{\mathtt{a}}$) and is identical to $C_{1,0}$ (playing the role of W_0) on branches $|p_1'\rangle_{\mathtt{p}_1}$ (playing the role of $|a'\rangle_{\mathtt{a}}$) with $p_1' \neq p_1$. (Recall $\ddot{C}_1$ from Algo. 7.1)

– Lem. 6 requires that U_1 should work as identity on register $\mathtt{s}$. In terms of the Claim 8 notation, this is satisfied by $B_{1,1}$ (playing the role of U_1), who works as identity on registers $\mathtt{u}$, $\mathtt{s}$, and $\mathtt{gc}$ (playing the role of register $\mathtt{s}$). (Recall $B_{1,1}$ from Table 1.)

– Lem. 6 requires that S should act non-trivially *only* on registers $\mathtt{a}$ and $\mathtt{s}$. In terms of the Claim 8 notation, this is satisfied by $U_{gc}S^{\mathtt{p}_1/\mathtt{m}}U_{gc}^\dagger$ (playing the role of S), who does not touch registers beyond $\mathtt{p}_1$, $\mathtt{u}$, $\mathtt{s}$, and $\mathtt{gc}$ (playing the role of registers $\mathtt{a}$ and $\mathtt{s}$). (Recall S from Table 1 and the fact that $S^{\mathtt{p}_1/\mathtt{m}}$ is identical to S except that it treats $\mathtt{p}_1$ as $\mathtt{m}$.)

Finally, we apply Lem. 6 (with the notation in Table 2) to the $|\gamma_0^{(t-1)}\rangle$ and $|\gamma_1^{(t-1)}\rangle$ defined in Eq. (8.30) and (8.31) (playing the role of $|\eta_0^{(\mathsf{in})}\rangle$ and $|\eta_1^{(\mathsf{in})}\rangle$ in Lem. 6). This implies the existence of (possibly sub-normalized) pure states $\{|\dot{\rho}_{p_1'}^{(t)}\rangle\}_{p_1'\in\{0,1\}^\ell}$ (playing the role of $\{|\rho_{a'}^{(\mathsf{out})}\rangle_{\mathtt{mtso}}\}_{a'\in\{0,1\}^\ell}$ in Lem. 6) such that the following holds

$$|\gamma_0^{(t)}\rangle = |p_1\rangle_{\mathtt{p}_1}|\dot{\rho}_{p_1}^{(t)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0}B_{1,1}^\dagger C_{1,1}^\dagger |\dot{\rho}_{p_1'}^{(t)}\rangle$$
$$|\gamma_1^{(t)}\rangle = |p_1\rangle_{\mathtt{p}_1}|\dot{\rho}_{p_1}^{(t)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0}|\dot{\rho}_{p_1'}^{(t)}\rangle,$$

which are exactly Eq. (8.34) and (8.35) in Claim 8.

This completes the proof of Claim 8.

□

Finishing the Proof for Case 1. With Claims 7 and 8 at hand, we now finish the proof for Case 1.

Proof for Eq. (8.1). We first establish Eq. (8.1):

$$|\phi^{(t)}\rangle = D_1|\gamma_0^{(t)}\rangle \tag{8.36}$$
$$= D_1\left(|p_1\rangle_{\mathtt{p}_1}|\dot{\rho}_{p_1}^{(t)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0}B_{1,1}^\dagger C_{1,1}^\dagger |\dot{\rho}_{p_1'}^{(t)}\rangle\right) \tag{8.37}$$
$$= |p_1\rangle_{\mathtt{p}_1} D_{1,1}|\dot{\rho}_{p_1}^{(t)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0}B_{1,1}^\dagger C_{1,1}^\dagger |\dot{\rho}_{p_1'}^{(t)}\rangle \tag{8.38}$$
$$= |p_1\rangle_{\mathtt{p}_1}|\rho_{p_1}^{(t)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0}B_{1,1}^\dagger C_{1,1}^\dagger |\dot{\rho}_{p_1'}^{(t)}\rangle, \tag{8.39}$$

where

- Eq. (8.36) follows from Eq. (8.10) in Claim 7 and Eq. (8.32) in Claim 8;
- Eq. (8.37) follows from Eq. (8.34) in Claim 8;
- Eq. (8.38) follows from the definition of D_1 (see Algo. 7.1);
- Eq. (8.39) follows by defining

$$|\rho_{p_1}^{(t)}\rangle \coloneqq D_{1,1}|\dot{\rho}_{p_1}^{(t)}\rangle \quad \text{and} \quad |\rho_{p_1'}^{(t)}\rangle \coloneqq |\dot{\rho}_{p_1'}^{(t)}\rangle \ \ (\forall p_1' \in \{0,1\}^\ell \setminus \{p_1\}). \tag{8.40}$$

Note that Eq. (8.39) is exactly Eq. (8.1) in Lem. 17.

Proof for Eq. (8.2). Next, we establish Eq. (8.2):

$$\begin{aligned}
|\psi^{(t)}\rangle &= D_1|\gamma_1^{(t)}\rangle && (8.41)\\
&= D_1\Big(|p_1\rangle_{\mathtt{p}_1}|\dot{\rho}_{p_1}^{(t)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0}|\dot{\rho}_{p_1'}^{(t)}\rangle\Big) && (8.42)\\
&= |p_1\rangle_{\mathtt{p}_1} D_{1,1}|\dot{\rho}_{p_1}^{(t)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0}|\dot{\rho}_{p_1'}^{(t)}\rangle && (8.43)\\
&= |p_1\rangle_{\mathtt{p}_1}|\rho_{p_1}^{(t)}\rangle + \sum_{p_1'\in\{0,1\}^\ell\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0}|\rho_{p_1'}^{(t)}\rangle, && (8.44)
\end{aligned}$$

where

- Eq. (8.41) follows from Eq. (8.11) in Claim 7 and Eq. (8.33) in Claim 8;
- Eq. (8.42) follows from Eq. (8.35) in Claim 8;
- Eq. (8.43) follows from the definition of D_1 (see Algo. 7.1);
- Eq. (8.44) follows from *the same definitions* of $|\rho_{p_1}^{(t)}\rangle$ and $|\rho_{p_1'}^{(t)}\rangle$ in Expression (8.40).

Note that Eq. (8.44) is exactly Eq. (8.2) in Lem. 17.

This eventually completes the proof for Case 1.

8.1.2.2 Proof for Case 2

We first remark that in Case 2, the computation that brings $|\phi^{(t-1)}\rangle$ to $|\phi^{(t)}\rangle$ is identical to that brings $|\psi^{(t-1)}\rangle$ to $|\psi^{(t)}\rangle$, because $H_0^{(J,\mathsf{pat})}$ and $H_1^{(J,\mathsf{pat})}$ are identical when the global counter "jumps" between 1 and 2. (This can be also seen pictorially by comparing Fig. 2a and Fig. 2b.) In the following, we refer to this computation as Λ. That is, we have

$$|\phi^{(t)}\rangle = \Lambda|\phi^{(t-1)}\rangle \tag{8.45}$$
$$|\psi^{(t)}\rangle = \Lambda|\psi^{(t-1)}\rangle. \tag{8.46}$$

Structure of Λ. While the exact format of Λ will not be substantial, our proof of Case 2 will rely on certain properties of Λ, which we formalize in the following Claim 9.

Claim 9. *For the operator Λ defined above, there exist two operators Λ_0 and Λ_1 so that*

– *both* Λ_1 *and* Λ_0 *act as the identity operator on* $\mathtt{p}_1$*;*

– Λ_0 *acts as the identity operator on* $\mathtt{t}_1$*;*

and the following holds

$$\Lambda = |p_1\rangle\langle p_1|_{\mathtt{p}_1} \otimes \Lambda_1 + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle\langle p_1'|_{\mathtt{p}_1} \otimes \Lambda_0, \tag{8.47}$$

$$\Lambda_0 C_{1,0} = C_{1,0} \Lambda_0^{\mathtt{t}_1/\mathtt{m}}, \tag{8.48}$$

$$\Lambda_0 C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger = C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger \Lambda_0^{\mathtt{t}_1/\mathtt{m}}, \tag{8.49}$$

where $\Lambda_0^{\mathtt{t}_1/\mathtt{m}}$ *is identical to* Λ_0 *except that it treats* $\mathtt{t}_1$ *as* $\mathtt{m}$*.*

Proof of Claim 9. First, note that Λ can be written as $\Lambda = \Gamma_z \Gamma_{z-1} \cdots \Gamma_1$ (for some integer z), where each Γ_i ($i \in [z]$) comes from the set of operators $\{S, A_2, B_2, C_2, D_2, U_{gc}, |p_1, p_2\rangle\langle p_1, p_2|_{\mathtt{p}_1\mathtt{p}_2}\}$. This is because Λ only corresponds to the operations that happen when the global counter is no less than value 1 (see Game 7.2 and Fig. 2b). We remark that $\mathcal{S}$ may also apply projectors on $\mathtt{u}$, but we consider it as a part of S as per Rmk. 12.

Therefore, to prove Claim 9, it suffices to show that for each operator

$$\Gamma \in \{|p_1, p_2\rangle\langle p_1, p_2|_{\mathtt{p}_1\mathtt{p}_2}, U_{gc}, S, A_2, B_2, C_2, D_2\},$$

there exist two operators Γ_0 and Γ_1 so that

– both Γ_1 and Γ_0 act as the identity operator on $\mathtt{p}_1$;

– Γ_0 acts as the identity operator on $\mathtt{t}_1$;

and the following holds

$$\Gamma = |p_1\rangle\langle p_1|_{\mathtt{p}_1} \otimes \Gamma_1 + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle\langle p_1'|_{\mathtt{p}_1} \otimes \Gamma_0,$$

$$\Gamma_0 C_{1,0} = C_{1,0} \Gamma_0^{\mathtt{t}_1/\mathtt{m}},$$

$$\Gamma_0 C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger = C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger \Gamma_0^{\mathtt{t}_1/\mathtt{m}}.$$

In the following, we prove it for each possible Γ.

First, notice that the above is true for $\Gamma = |p_1, p_2\rangle\langle p_1, p_2|_{\mathtt{p}_1\mathtt{p}_2}$, simply because such a Γ can be written in the following format:

$$\Gamma = |p_1, p_2\rangle\langle p_1, p_2|_{\mathtt{p}_1\mathtt{p}_2} = |p_1\rangle\langle p_1|_{\mathtt{p}_1} \otimes \Gamma_1 + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle\langle p_1'|_{\mathtt{p}_1} \otimes \Gamma_0,$$

with $\Gamma_1 := |p_2\rangle\langle p_2|_{\mathtt{p}_2}$ and $\Gamma_0 := I$, and such a Γ_0 vacuously satisfies the requires $\Gamma_0 C_{1,0} = C_{1,0} \Gamma_0^{\mathtt{t}_1/\mathtt{m}}$ and $\Gamma_0 C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger = C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger \Gamma_0^{\mathtt{t}_1/\mathtt{m}}$.

The above is true for $\Gamma = U_{gc}$ as well, because U_{gc} acts non-trivially only on register $\mathtt{gc}$.

The above is also true for $\Gamma = S$ because (1) S does not act on $\mathtt{p}_1$, (2) S does not work on $\mathtt{t}_1$ and $C_{1,0}$ is nothing but a swap operator between $\mathtt{m}$ and $\mathtt{t}_1$, and (3) $S^{\mathtt{t}_1/\mathtt{m}}$ acts non-trivially on different registers from $B_{1,1}$ and $C_{1,1}$ (see Table 1).

The above is true for $\Gamma = A_2$ as well, because A_2 is nothing but the swap operator between registers $\mathtt{p}_2$ and $\mathtt{m}$ (see Algo. 6.1).

The only cases left are $\Gamma \in \{B_2, C_2, D_2\}$. The proof for these cases are (almost) identical. In the following, we only prove it for $\Gamma = C_2$.

$$\begin{aligned}
\Gamma = C_2 &= |p_1,p_2\rangle\langle p_1,p_2|_{\mathtt{p}_1\mathtt{p}_2} \otimes C_{2,1} + \sum_{(p_1',p_2')\in\{0,1\}^{2\ell}\setminus\{(p_1,p_2)\}} |p_1',p_2'\rangle\langle p_1',p_2'|_{\mathtt{p}_1\mathtt{p}_2} \otimes C_{2,0} && (8.50)\\
&= |p_1\rangle\langle p_1|_{\mathtt{p}_1} \otimes |p_2\rangle\langle p_2|_{\mathtt{p}_2} \otimes C_{2,1} + \sum_{(p_1',p_2')\in\{0,1\}^{2\ell}\setminus\{(p_1,p_2)\}} |p_1'\rangle\langle p_1'|_{\mathtt{p}_1} \otimes |p_2'\rangle\langle p_2'|_{\mathtt{p}_2} \otimes C_{2,0}\\
&= |p_1\rangle\langle p_1|_{\mathtt{p}_1} \otimes |p_2\rangle\langle p_2|_{\mathtt{p}_2} \otimes C_{2,1} + |p_1\rangle\langle p_1|_{\mathtt{p}_1} \otimes \Big(\sum_{p_2'\in\{0,1\}^{\ell}\setminus\{p_2\}} |p_2'\rangle\langle p_2'|_{\mathtt{p}_2}\Big) \otimes C_{2,0}\\
&\qquad + \sum_{p_1'\in\{0,1\}^{\ell}\setminus\{p_1\}} |p_1'\rangle\langle p_1'|_{\mathtt{p}_1} \otimes \Big(\sum_{p_2'\in\{0,1\}^{\ell}} |p_2'\rangle\langle p_2'|_{\mathtt{p}_2}\Big) \otimes C_{2,0}\\
&= |p_1\rangle\langle p_1|_{\mathtt{p}_1} \otimes \Gamma_1 + \sum_{p_1'\in\{0,1\}^{\ell}\setminus\{p_1\}} |p_1'\rangle\langle p_1'|_{\mathtt{p}_1} \otimes \Gamma_0, && (8.51)
\end{aligned}$$

where Eq. (8.50) follows from the definition of C_2 (see Algo. 7.1), and Eq. (8.51) follows by defining Γ_0 and Γ_1 as follows

$$\begin{aligned}
\Gamma_1 &\coloneqq |p_2\rangle\langle p_2|_{\mathtt{p}_2} \otimes C_{2,1} + \Big(\sum_{p_2'\in\{0,1\}^{\ell}\setminus\{p_2\}} |p_2'\rangle\langle p_2'|_{\mathtt{p}_2}\Big) \otimes C_{2,0}\\
\Gamma_0 &\coloneqq \Big(\sum_{p_2'\in\{0,1\}^{\ell}} |p_2'\rangle\langle p_2'|_{\mathtt{p}_2}\Big) \otimes C_{2,0}.
\end{aligned}$$

Clearly, such a Γ_0 satisfies the requires $\Gamma_0 C_{1,0} = C_{1,0}\Gamma_0^{\mathtt{t}_1/\mathtt{m}}$ and $\Gamma_0 C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger = C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger \Gamma_0^{\mathtt{t}_1/\mathtt{m}}$, because (1) $C_{2,0}$ is nothing but a swap operator between $\mathtt{m}$ and $\mathtt{t}_2$, and (2) $C_{2,0}^{\mathtt{t}_1/\mathtt{m}}$ acts non-trivially on different registers from $B_{1,1}$ and $C_{1,1}$ (see Table 1).

This finishes the proof of Claim 9.

□

Finishing the Proof for Case 2. With Claim 9 in hand, we now show how to finish the proof for Case 2.

Proof of Eq. (8.1). First, we establish Eq. (8.1):

$$\begin{aligned}
|\phi^{(t)}\rangle &= \Lambda|\phi^{(t-1)}\rangle && (8.52)\\
&= \Lambda\Big(|p_1\rangle_{\mathtt{p}_1}|\rho_{p_1}^{(t-1)}\rangle + \sum_{p_1'\in\{0,1\}^{\ell}\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger |\rho_{p_1'}^{(t-1)}\rangle\Big) && (8.53)\\
&= |p_1\rangle_{\mathtt{p}_1}\Lambda_1|\rho_{p_1}^{(t-1)}\rangle + \sum_{p_1'\in\{0,1\}^{\ell}\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1}\Lambda_0 C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger |\rho_{p_1'}^{(t-1)}\rangle && (8.54)\\
&= |p_1\rangle_{\mathtt{p}_1}\Lambda_1|\rho_{p_1}^{(t-1)}\rangle + \sum_{p_1'\in\{0,1\}^{\ell}\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger \Lambda_0^{\mathtt{t}_1/\mathtt{m}} |\rho_{p_1'}^{(t-1)}\rangle && (8.55)\\
&= |p_1\rangle_{\mathtt{p}_1}|\rho_{p_1}^{(t)}\rangle + \sum_{p_1'\in\{0,1\}^{\ell}\setminus\{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0} B_{1,1}^\dagger C_{1,1}^\dagger |\rho_{p_1'}^{(t)}\rangle, && (8.56)
\end{aligned}$$

where

– Eq. (8.52) follows from Eq. (8.45);

– Eq. (8.53) follows from our induction assumption;

– Eq. (8.54) follows from Eq. (8.47) in Claim 9;

– Eq. (8.55) follows from Eq. (8.49) in Claim 9;

– Eq. (8.56) follows by defining

$$|\rho_{p_1}^{(t)}\rangle := \Lambda_1 |\rho_{p_1}^{(t-1)}\rangle \quad \text{and} \quad |\rho_{p_1'}^{(t)}\rangle := \Lambda_0^{\mathtt{t}_1/\mathtt{m}} |\rho_{p_1'}^{(t-1)}\rangle \ \ (\forall p_1' \in \{0,1\}^\ell \setminus \{p_1\}). \tag{8.57}$$

Clearly, Eq. (8.56) is of the same format as Eq. (8.1) in Lem. 17.

Proof of Eq. (8.2). Next, we present the derivation for Eq. (8.2):

$$\begin{aligned}
|\psi^{(t)}\rangle &= \Lambda |\psi^{(t-1)}\rangle && (8.58)\\
&= \Lambda\Big(|p_1\rangle_{\mathtt{p}_1} |\rho_{p_1}^{(t-1)}\rangle \ + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0} |\rho_{p_1'}^{(t-1)}\rangle \Big) && (8.59)\\
&= |p_1\rangle_{\mathtt{p}_1} \Lambda_1 |\rho_{p_1}^{(t-1)}\rangle \ + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathtt{p}_1} \Lambda_0 C_{1,0} |\rho_{p_1'}^{(t-1)}\rangle && (8.60)\\
&= |p_1\rangle_{\mathtt{p}_1} \Lambda_1 |\rho_{p_1}^{(t-1)}\rangle \ + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0} \Lambda_0^{\mathtt{t}_1/\mathtt{m}} |\rho_{p_1'}^{(t-1)}\rangle && (8.61)\\
&= |p_1\rangle_{\mathtt{p}_1} |\rho_{p_1}^{(t)}\rangle \ + \sum_{p_1' \in \{0,1\}^\ell \setminus \{p_1\}} |p_1'\rangle_{\mathtt{p}_1} C_{1,0} |\rho_{p_1'}^{(t)}\rangle, && (8.62)
\end{aligned}$$

where

– Eq. (8.58) follows from Eq. (8.46),

– Eq. (8.59) follows from our induction assumption,

– Eq. (8.60) follows from Eq. (8.47) in Claim 9,

– Eq. (8.61) follows from Eq. (8.48) in Claim 9,

– Eq. (8.62) follows from *the same definitions* of $|\rho_{p_1}^{(t)}\rangle$ and $|\rho_{p_1'}^{(t)}\rangle$ as shown in Expression (8.57).

Clearly, Eq. (8.62) is of the same format as Eq. (8.2) in Lem. 17.

This finishes the proof for Case 2.

Finally, we remark that our proof for the base case in Sec. 8.1.1 and the proof for the induction step in Sec. 8.1.2 together finish the proof of Lem. 17, which in turn finishes the proof of Lem. 15 eventually.

8.2 Proof of Lem. 16

Due to a similar argument as we did at the beginning of Sec. 8.1, we claim that: to prove Lem. 16, it suffices to establish the following Lem. 18.

Lemma 18 (Invariance in $H_1^{(J,\mathsf{pat})}$ and $H_2^{(J,\mathsf{pat})}$). *Assume that during the execution of $H_1^{(J,\mathsf{pat})}$ (and $H_2^{(J,\mathsf{pat})}$), the global counter reaches value 2 for T times in total. For each $t \in [T]$, there*

exist (possibly sub-normalized) pure states $\{|\rho^{(t)}_{p'_1,p'_2}\rangle\}_{p'_1,p'_2\in\{0,1\}^\ell\times\{0,1\}^\ell}$ *so that the following holds: in hybrid* $H_1^{(J,\mathsf{pat})}$ *(resp.* $H_2^{(J,\mathsf{pat})}$*), when the global counter reaches value* 2 *for the* t*-th time, the overall state can be written as* $|\phi^{(t)}\rangle$ *(resp.* $|\psi^{(t)}\rangle$*) defined as follows*

$$|\phi^{(t)}\rangle = |p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\rho^{(t)}_{p_1,p_2}\rangle \ + \sum_{(p'_1,p'_2)\neq(p_1,p_2)} |p'_1\rangle_{\mathtt{p}_1}|p'_2\rangle_{\mathtt{p}_2} C_{2,0}B^\dagger_{2,1}C^\dagger_{2,1}|\rho^{(t)}_{p'_1,p'_2}\rangle, \tag{8.63}$$

$$|\psi^{(t)}\rangle = |p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\rho^{(t)}_{p_1,p_2}\rangle \ + \sum_{(p'_1,p'_2)\neq(p_1,p_2)} |p'_1\rangle_{\mathtt{p}_1}|p'_2\rangle_{\mathtt{p}_2} C_{2,0}|\rho^{(t)}_{p'_1,p'_2}\rangle, \tag{8.64}$$

where the summations are taken over all $(p'_1,p'_2)\in\{0,1\}^\ell\times\{0,1\}^\ell\setminus\{(p_1,p_2)\}$ *(abbreviated as* $(p'_1,p'_2)\neq(p_1,p_2)$ *in the above), the* (p_1,p_2) *are defined in* pat*, and the unitaries* $B_{2,1}$*,* $C_{2,0}$*, and* $C_{2,1}$ *are as defined in Algo. 7.1.*

Proving Lem. 18. Similar as the proof for Lem. 17, we establish Lem. 18 through mathematical induction on the number $t\in[T]$, indicating the time at which the global counter reaches the value 2. Throughout this proof, we will monitor the evolution of the overall states in both $H_1^{(J,\mathsf{pat})}$ and $H_2^{(J,\mathsf{pat})}$ simultaneously. This is done in Sec. 8.2.1 and 8.2.2 respectively.

8.2.1 Base Case ($t=1$)

We first derive how the overall state evolves in $H_1^{(J,\mathsf{pat})}$.

This case corresponds to the very first time the global counter reaches 2. By definition, this is due to the $\mathsf{sq}(2)$ query. We assume w.l.o.g. that the overall state right before this query is some pure state $|\rho\rangle$.

Remark 13. Actually, we know the exact format of this $|\rho\rangle$ from the already-established Lem. 17. That is, this state must be of the format $|\rho\rangle = |p_1\rangle_{\mathtt{p}_1}|\rho_{p_1}\rangle + \sum_{p'_1\in\{0,1\}^\ell\setminus\{p_1\}}|p'_1\rangle_{\mathtt{p}_1}C_{1,0}|\rho_{p'_1}\rangle$. However, we remark that the exact format of $|\rho\rangle$ is not useful anywhere in this proof, and thus we do not need to refer to this long expression in the following derivation.

By definition, the $\mathsf{sq}(2)$ query corresponds to the following procedure: $\mathcal{S}$ will first apply her local unitary S (followed by the projector $|{\downarrow}\rangle\langle{\downarrow}|_{\mathtt{u}}$ which we hide as per Rmk. 12). Next, according to the notation in Game 7.2,

- If $b_2=0$, then the operator $D_2C_2B_2\,|p_1,p_2\rangle\langle p_1,p_2|_{\mathtt{p}_1\mathtt{p}_2}\,A_2U_{gc}$ will be applied;
- If $b_2=1$, then the operator $C_{2,0}\,|p_1,p_2\rangle\langle p_1,p_2|_{\mathtt{p}_1\mathtt{p}_2}\,A_2U_{gc}$ will be applied;

In the following, we show the proof for $b_2=0$ only; the other case (i.e., $b_2=1$) can be established using the same argument.

It follows from the above discussion that the state $|\phi^{(1)}\rangle$ in $H_1^{(J,\mathsf{pat})}$ can be written as follows:

$$\begin{aligned}
|\phi^{(1)}\rangle &= D_2C_2B_2\,|p_1,p_2\rangle\langle p_1,p_2|_{\mathtt{p}_1\mathtt{p}_2}\,A_2U_{gc}S|\rho\rangle \\
&= D_2C_2B_2|p_1,p_2\rangle_{\mathtt{p}_1\mathtt{p}_2}|\rho_{p_1,p_2}\rangle && (8.65)\\
&= |p_1,p_2\rangle_{\mathtt{p}_1\mathtt{p}_2}D_{2,1}C_{2,1}B_{2,1}|\rho_{p_1,p_2}\rangle && (8.66)\\
&= |p_1,p_2\rangle_{\mathtt{p}_1\mathtt{p}_2}|\rho^{(1)}_{p_1,p_2}\rangle, && (8.67)
\end{aligned}$$

where

– Eq. (8.65) follows by defining $|\rho_{p_1,p_2}\rangle := \langle p_1, p_2|_{\mathtt{p}_1\mathtt{p}_2} A_2 U_{gc} S|\rho\rangle$;

– Eq. (8.66) follows from the definitions of B_2, C_2, and D_2 (see Algo. 7.1);

– Eq. (8.67) follows by defining $|\rho^{(1)}_{p_1,p_2}\rangle := D_{2,1}C_{2,1}B_{2,1}|\rho_{p_1,p_2}\rangle$.

It is straightforward that the $|\phi^{(1)}\rangle$ shown in Eq. (8.67) satisfies the format shown in Eq. (8.63) when $t = 1$.

Also, note that the game $H_1^{(J,\mathsf{pat})}$ and $H_2^{(J,\mathsf{pat})}$ are identical so far and thus $|\phi^{(1)}\rangle = |\psi^{(1)}\rangle$. Therefore, it follows from Eq. (8.67) that

$$|\psi^{(1)}\rangle = |p_1, p_2\rangle_{\mathtt{p}_1\mathtt{p}_2}|\rho^{(1)}_{p_1,p_2}\rangle.$$

Such a $|\psi^{(1)}\rangle$ satisfies the format shown in Eq. (8.64) with $t = 1$ as well.

This finishes the proof for the base case $t = 1$.

8.2.2 Induction Step ($t \geq 2$)

We assume $|\phi^{(t-1)}\rangle$ and $|\psi^{(t-1)}\rangle$ satisfy Lem. 18, and show in the following that Lem. 18 holds when the global counter reaches 2 for the t-th time.

We first describe formally how $|\phi^{(t-1)}\rangle$ (resp. $|\psi^{(t-1)}\rangle$) evolves into $|\phi^{(t)}\rangle$ (resp. $|\psi^{(t)}\rangle$):

1. $\mathcal{S}$'s local unitary S is applied (followed by the projector $|\uparrow\rangle\langle\uparrow|_{\mathtt{u}}$ which we hide as per Rmk. 12). Note that this step is identical for both $H_1^{(J,\mathsf{pat})}$ and $H_2^{(J,\mathsf{pat})}$.
2. An $\uparrow$-query is made, bringing the global counter from 2 to 1. According to the notation in Game 7.2:
   – In $H_1^{(J,\mathsf{pat})}$, this corresponds to applying $U_{gc}^\dagger A_2^\dagger B_2^\dagger C_2^\dagger D_2^\dagger$;
   – In $H_2^{(J,\mathsf{pat})}$, this corresponds to applying $U_{gc}^\dagger A_2^\dagger \ddot{C}_2^\dagger D_2^\dagger$.
3. At this point, there are two cases to consider:
   (a) $\mathcal{S}$ can make a $\downarrow$-query immediately, bringing the global counter back to 2; *or*
   (b) $\mathcal{S}$ can rewind the verifier $\widetilde{\mathcal{V}}$ further, bringing the global counter to 0.

   Defining Operator Λ: Looking ahead, our proof can handle these two cases at one stroke. To do that, we denote the execution from this point until (exclusively) *the next* $\downarrow$ *query that brings the global counter to value* 2 (i.e., the exact query that yields the global counter's t-th arrival at value 2) as an operator Λ. Λ might be $\mathcal{S}$ local operator S *only*, corresponding to Case 3a; Λ might also be the combonation of some other operators, representing the operations performed in Case 3b before the global counter's next arrival at value 2.
4. After the application of the operator Λ defined above, an $\downarrow$-query is made, bringing the global counter from 1 back to 2, which is the global counter's t-th arrival at value 2. According to the notation in Game 7.2:
   – In $H_0^{(J,\mathsf{pat})}$, this corresponds to applying $D_2C_2B_2A_2U_{gc}$;
   – In $H_1^{(J,\mathsf{pat})}$, this corresponds to applying $D_2\ddot{C}_2A_2U_{gc}$.

It follows from the above description that the states $|\phi^{(t)}\rangle$ and $|\psi^{(t)}\rangle$ can be written as:

$$|\phi^{(t)}\rangle = D_2C_2B_2A_2U_{gc}\Lambda U_{gc}^\dagger A_2^\dagger B_2^\dagger C_2^\dagger D_2^\dagger S|\phi^{(t-1)}\rangle, \tag{8.68}$$

$$|\psi^{(t)}\rangle = D_2\ddot{C}_2A_2U_{gc}\Lambda U_{gc}^\dagger A_2^\dagger \ddot{C}_2^\dagger D_2^\dagger S|\psi^{(t-1)}\rangle. \tag{8.69}$$

We first make a useful claim that specifies the registers on which the Λ defined above acts non-trivially.

Claim 10 (Non-Trivial Registers for Λ). *The Λ defined in Step 3 acts non-trivially only on registers* $\mathtt{m}$, $\mathtt{u}$, $\mathtt{s}$, $\mathtt{p}_1$, $\mathtt{t}_1$, $\mathtt{lc}$, *and* $\mathtt{gc}$ *(i.e., it works as the identity operator on the other registers).*

Proof of Claim 10. By definition, Λ only contains operations that happen when the global counter is 1 or smaller. This is best illustrated by Fig. 2b—Λ only contains operations that happen in between the line corresponding to $|0\rangle_{\mathtt{gc}}$ and the line corresponding to $|1\rangle_{\mathtt{gc}}$. That is, all the $\downarrow$ and $\uparrow$ queries are answered by the "dummy" operators $\ddot{V}$ and $\ddot{V}^\dagger$. (One exception is the $\widetilde{V}$ corresponding $\mathtt{sq}(1)$. But this does not matter in the current proof—Before the application of operator Λ, the global counter has already been 2 (for $t-1$ times), which implies that the $\mathtt{sq}(1)$ query has already happened and thus cannot be a part of the operator Λ.)

Therefore, according to our notation in Game 7.2, operator Λ can be written as $\Lambda = \Gamma_z \Gamma_{z-1} \cdots \Gamma_1$ (for some integer z) where each Γ_i ($i \in [z]$) comes from the set of operators $\{S, A_1, \ddot{C}_1, D_1, U_{gc}\}$. It then follows from Table 1 that Λ acts non-trivially only on registers $\mathtt{m}$, $\mathtt{u}$, $\mathtt{s}$, $\mathtt{p}_1$, $\mathtt{t}_1$, $\mathtt{lc}$, and $\mathtt{gc}$.

This completes the proof of Claim 10.

□

High-Level Idea for the Sequel. Recall that our eventual goal is to prove that the states $|\phi^{(t)}\rangle$ and $|\psi^{(t)}\rangle$ are of the format shown in Eq. (8.63) and (8.64) in Lem. 18. At a high level, we prove it by applying Lem. 6 to Eq. (8.68) and (8.69). But we first need to perform some preparation work, putting Eq. (8.68) and (8.69) into a format that is more "compatible" with Lem. 6. In the sequel, we first perform the preparation work in Claims 11 and 12. Then, we show on Page 87 how to use Claims 11 and 12 to complete the proof for the induction step.

The following Claim 11 can be viewed as an analogue of Claim 7.

Claim 11. *There exist (possibly sub-normalized) pure states* $\{\dot{\rho}^{(t-1)}_{p_1',p_2'}\}_{(p_1',p_2')\in\{0,1\}^\ell\times\{0,1\}^\ell}$ *so that the following holds*

$$|\phi^{(t)}\rangle = D_2 C_2 B_2 U_{gc} \Lambda^{\mathtt{p}_2/\mathtt{m}} U_{gc}^\dagger B_2^\dagger C_2^\dagger \Bigg(|p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\dot{\rho}^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)} |p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2} C_{2,0} B_{2,1}^\dagger C_{2,1}^\dagger |\dot{\rho}^{(t-1)}_{p_1',p_2'}\rangle \Bigg) \tag{8.70}$$

$$|\psi^{(t)}\rangle = D_2 \ddot{C}_2 U_{gc} \Lambda^{\mathtt{p}_2/\mathtt{m}} U_{gc}^\dagger \ddot{C}_2^\dagger \Bigg(|p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\dot{\rho}^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)} |p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2} C_{2,0} |\dot{\rho}^{(t-1)}_{p_1',p_2'}\rangle \Bigg) \tag{8.71}$$

where $\Lambda^{\mathtt{p}_2/\mathtt{m}}$ is identical to the Λ defined in Step 3, except that it treats register $\mathtt{p}_2$ as $\mathtt{m}$, and the summations are taken over all $(p_1',p_2') \in \{0,1\}^\ell \times \{0,1\}^\ell \backslash \{(p_1,p_2)\}$ (abbreviated as $(p_1',p_2') \neq (p_1,p_2)$ in the above).

Proof of Claim 11. First, notice that

$$A_2 U_{gc} \Lambda U_{gc}^\dagger A_2^\dagger = A_2 U_{gc} \Lambda A_2^\dagger U_{gc}^\dagger \tag{8.72}$$

$$= A_2 U_{gc} A_2^\dagger \Lambda^{\mathtt{p}_2/\mathtt{m}} U_{gc}^\dagger \tag{8.73}$$

$$= A_2 A_2^\dagger U_{gc} \Lambda^{\mathtt{p}_2/\mathtt{m}} U_{gc}^\dagger \tag{8.74}$$

$$= U_{gc} \Lambda^{\mathtt{p}_2/\mathtt{m}} U_{gc}^\dagger, \tag{8.75}$$

where

– Eq. (8.72) and (8.74) follows from the fact that U_{gc} acts non-trivially on different registers from A_2;

– Eq. (8.73) from the fact that Λ acts as the identity operator on $\mathtt{p}_2$ (see Claim 10) and that A_2 is nothing but a swap operator between $\mathtt{m}$ and $\mathtt{p}_2$ (see Algo. 6.1).

Proving Eq. (8.70). We now show the derivation for $|\phi^{(t)}\rangle$:

$$|\phi^{(t)}\rangle = D_2C_2B_2A_2U_{gc}\Lambda U_{gc}^\dagger A_2^\dagger B_2^\dagger C_2^\dagger D_2^\dagger S|\phi^{(t-1)}\rangle \tag{8.76}$$

$$= D_2C_2B_2U_{gc}\Lambda^{\mathtt{p}_2/\mathtt{m}}U_{gc}^\dagger B_2^\dagger C_2^\dagger D_2^\dagger S|\phi^{(t-1)}\rangle \tag{8.77}$$

$$= D_2C_2B_2U_{gc}\Lambda^{\mathtt{p}_2/\mathtt{m}}U_{gc}^\dagger B_2^\dagger C_2^\dagger D_2^\dagger S\Big(|p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\rho^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)}|p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}B_{2,1}^\dagger C_{2,1}^\dagger|\rho^{(t-1)}_{p_1',p_2'}\rangle\Big) \tag{8.78}$$

$$= D_2C_2B_2U_{gc}\Lambda^{\mathtt{p}_2/\mathtt{m}}U_{gc}^\dagger B_2^\dagger C_2^\dagger D_2^\dagger\Big(|p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}S|\rho^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)}|p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}SC_{2,0}B_{2,1}^\dagger C_{2,1}^\dagger|\rho^{(t-1)}_{p_1',p_2'}\rangle\Big) \tag{8.79}$$

$$= D_2C_2B_2U_{gc}\Lambda^{\mathtt{p}_2/\mathtt{m}}U_{gc}^\dagger B_2^\dagger C_2^\dagger\Big(|p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}D_{2,1}^\dagger S|\rho^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)}|p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}SC_{2,0}B_{2,1}^\dagger C_{2,1}^\dagger|\rho^{(t-1)}_{p_1',p_2'}\rangle\Big) \tag{8.80}$$

$$= D_2C_2B_2U_{gc}\Lambda^{\mathtt{p}_2/\mathtt{m}}U_{gc}^\dagger B_2^\dagger C_2^\dagger\Big(|p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}D_{2,1}^\dagger S|\rho^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)}|p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}S^{\mathtt{t}_2/\mathtt{m}}B_{2,1}^\dagger C_{2,1}^\dagger|\rho^{(t-1)}_{p_1',p_2'}\rangle\Big) \tag{8.81}$$

$$= D_2C_2B_2U_{gc}\Lambda^{\mathtt{p}_2/\mathtt{m}}U_{gc}^\dagger B_2^\dagger C_2^\dagger\Big(|p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}D_{2,1}^\dagger S|\rho^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)}|p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}B_{2,1}^\dagger C_{2,1}^\dagger S^{\mathtt{t}_2/\mathtt{m}}|\rho^{(t-1)}_{p_1',p_2'}\rangle\Big) \tag{8.82}$$

$$= D_2C_2B_2U_{gc}\Lambda^{\mathtt{p}_2/\mathtt{m}}U_{gc}^\dagger B_2^\dagger C_2^\dagger\Big(|p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\dot\rho^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)}|p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}B_{2,1}^\dagger C_{2,1}^\dagger|\dot\rho^{(t-1)}_{p_1',p_2'}\rangle\Big), \tag{8.83}$$

where

– Eq. (8.76) follows from Eq. (8.68);

– Eq. (8.77) follows from Eq. (8.75);

– Eq. (8.78) follows from our induction assumption;

– Eq. (8.79) from from the fact that S acts as the identity operator on $\mathtt{p}_1$ and $\mathtt{p}_2$ (see Table 1);

– Eq. (8.80) from the definition of D_2 (see Algo. 7.1);

– Eq. (8.81) from the fact that S acts as the identity operator on $\mathtt{t}_2$ (see Table 1) and $C_{2,0}$ is nothing but a swap operator between $\mathtt{t}_2$ and $\mathtt{m}$ (see Algo. 7.1); (Note that $S^{\mathtt{t}_2/\mathtt{m}}$ is defined to be an operator that is identical to S except that it treats $\mathtt{t}_2$ as $\mathtt{m}$.)

– Eq. (8.82) follows from the fact that $S^{\mathtt{t}_2/\mathtt{m}}$ acts non-trivially on different registers from $B_{2,1}$ and $C_{2,1}$ (see Table 1);

– Eq. (8.83) follows by defining

$$|\dot{\rho}^{(t-1)}_{p_1,p_2}\rangle := D^\dagger_{2,1} S |\rho^{(t-1)}_{p_1,p_2}\rangle \;\text{ and }\; |\dot{\rho}^{(t-1)}_{p'_1,p'_2}\rangle := S^{\mathtt{t}_2/\mathtt{m}} |\rho^{(t-1)}_{p'_1,p'_2}\rangle \;\; \big(\forall (p'_1,p'_2) \in \big(\{0,1\}^\ell \times \{0,1\}^\ell\big) \backslash \{(p_1,p_2)\}\big). \tag{8.84}$$

Eq. (8.83) finishes the proof of Eq. (8.70) in Claim 11.

Proving Eq. (8.71). We now show the derivation for $|\psi^{(t)}\rangle$. This is almost identical to the above proof for Eq. (8.70). Nevertheless, we present it for the sake of completeness.

$$|\psi^{(t)}\rangle = D_2 \ddot{C}_2 A_2 U_{gc} \Lambda U^\dagger_{gc} A^\dagger_2 \ddot{C}^\dagger_2 D^\dagger_2 S |\psi^{(t-1)}\rangle \tag{8.85}$$

$$= D_2 \ddot{C}_2 U_{gc} \Lambda^{\mathtt{p}_2/\mathtt{m}} U^\dagger_{gc} \ddot{C}^\dagger_2 D^\dagger_2 S |\psi^{(t-1)}\rangle \tag{8.86}$$

$$= D_2 \ddot{C}_2 U_{gc} \Lambda^{\mathtt{p}_2/\mathtt{m}} U^\dagger_{gc} \ddot{C}^\dagger_2 D^\dagger_2 S \Bigg(|p_1\rangle_{\mathtt{p}_1} |p_2\rangle_{\mathtt{p}_2} |\rho^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p'_1,p'_2)\neq(p_1,p_2)} |p'_1\rangle_{\mathtt{p}_1} |p'_2\rangle_{\mathtt{p}_2} C_{2,0} |\rho^{(t-1)}_{p'_1,p'_2}\rangle \Bigg) \tag{8.87}$$

$$= D_2 \ddot{C}_2 U_{gc} \Lambda^{\mathtt{p}_2/\mathtt{m}} U^\dagger_{gc} \ddot{C}^\dagger_2 \Bigg(|p_1\rangle_{\mathtt{p}_1} |p_2\rangle_{\mathtt{p}_2} D^\dagger_{2,1} S |\rho^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p'_1,p'_2)\neq(p_1,p_2)} |p'_1\rangle_{\mathtt{p}_1} |p'_2\rangle_{\mathtt{p}_2} C_{2,0} S^{\mathtt{t}_2/\mathtt{m}} |\rho^{(t-1)}_{p'_1,p'_2}\rangle \Bigg) \tag{8.88}$$

$$= D_2 \ddot{C}_2 U_{gc} \Lambda^{\mathtt{p}_2/\mathtt{m}} U^\dagger_{gc} \ddot{C}^\dagger_2 \Bigg(|p_1\rangle_{\mathtt{p}_1} |p_2\rangle_{\mathtt{p}_2} |\dot{\rho}^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p'_1,p'_2)\neq(p_1,p_2)} |p'_1\rangle_{\mathtt{p}_1} |p'_2\rangle_{\mathtt{p}_2} C_{2,0} |\dot{\rho}^{(t-1)}_{p'_1,p'_2}\rangle \Bigg), \tag{8.89}$$

where

– Eq. (8.85) follows from Eq. (8.69);

– Eq. (8.86) follows from Eq. (8.75);

– Eq. (8.87) follows from our induction assumption;

– Eq. (8.88) follows from a similar argument as we did to derive Eq. (8.82) from Eq. (8.78);

– Eq. (8.89) follows from *the same definitions* of $|\dot{\rho}^{(t-1)}_{p_1,p_2}\rangle$ and $|\dot{\rho}^{(t-1)}_{p'_1,p'_2}\rangle$ as shown in Expression (8.84).

Eq. (8.89) finishes the proof of Eq. (8.71) in Claim 11.

This finishes the proof of Claim 11.

□

The following Claim 12 can be treated as an analogue of Claim 8.

Claim 12. *Let* $\Lambda^{\mathtt{p}_2/\mathtt{m}}$ *and* $\{\dot{\rho}^{(t-1)}_{p'_1,p'_2}\}_{(p'_1,p'_2)\in\{0,1\}^\ell\times\{0,1\}^\ell}$ *be as defined in Claim 11. Let*

$$|\gamma^{(t-1)}_0\rangle := |p_1\rangle_{\mathtt{p}_1} |p_2\rangle_{\mathtt{p}_2} |\dot{\rho}^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p'_1,p'_2)\neq(p_1,p_2)} |p'_1\rangle_{\mathtt{p}_1} |p'_2\rangle_{\mathtt{p}_2} C_{2,0} B^\dagger_{2,1} C^\dagger_{2,1} |\dot{\rho}^{(t-1)}_{p'_1,p'_2}\rangle \tag{8.90}$$

$$|\gamma_1^{(t-1)}\rangle := |p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\dot{\rho}^{(t-1)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)} |p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2} C_{2,0}|\dot{\rho}^{(t-1)}_{p_1',p_2'}\rangle \tag{8.91}$$

$$|\gamma_0^{(t)}\rangle := C_2 B_2 U_{gc} \Lambda^{\mathtt{p}_2/\mathtt{m}} U_{gc}^\dagger B_2^\dagger C_2^\dagger |\gamma_0^{(t-1)}\rangle \tag{8.92}$$

$$|\gamma_1^{(t)}\rangle := \ddot{C}_2 U_{gc} \Lambda^{\mathtt{p}_2/\mathtt{m}} U_{gc}^\dagger \ddot{C}_2^\dagger |\gamma_1^{(t-1)}\rangle. \tag{8.93}$$

Then, there exist (possibly sub-normalized) pure states $\{\dot{\rho}^{(t)}_{p_1',p_2'}\}_{(p_1',p_2')\in\{0,1\}^\ell\times\{0,1\}^\ell}$ *so that the following holds:*

$$|\gamma_0^{(t)}\rangle = |p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\dot{\rho}^{(t)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)} |p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2} C_{2,0} B_{2,1}^\dagger C_{2,1}^\dagger |\dot{\rho}^{(t)}_{p_1',p_2'}\rangle \tag{8.94}$$

$$|\gamma_1^{(t)}\rangle = |p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\dot{\rho}^{(t)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)} |p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2} C_{2,0} |\dot{\rho}^{(t)}_{p_1',p_2'}\rangle. \tag{8.95}$$

Proof of Claim 12. This claim follows from an application of Lem. 6, with the notation correspondence listed in Table 3. We provide a detailed explanation below.

Table 3: Notation Correspondence between Lem. 6 and Claim 12

Registers		Operators		Random Variables	
In Lem. 6	In Claim 12	In Lem. 6	In Claim 12	In Lem. 6	In Claim 12
$\mathtt{a}$	$\mathtt{p}_1$, $\mathtt{p}_2$	W_1	$C_{2,1}$	$\|a\rangle_{\mathtt{a}}$	$\|p_1,p_2\rangle_{\mathtt{p}_1\mathtt{p}_2}$
$\mathtt{m}$	$\mathtt{m}$	W_0	$C_{2,0}$	$\|a'\rangle_{\mathtt{a}}$	$\|p_1',p_2'\rangle_{\mathtt{p}_1\mathtt{p}_2}$
$\mathtt{t}$	$\mathtt{t}_2$	W	C_2	$\|\rho_a^{(\mathsf{in})}\rangle_{\mathtt{mtso}}$	$\|\dot{\rho}^{(t-1)}_{p_1,p_2}\rangle$
$\mathtt{s}$	$\mathtt{u}$, $\mathtt{s}$, $\mathtt{t}_1$, $\mathtt{gc}$, $\mathtt{lc}$	$\widetilde{W}$	$\ddot{C}_2$	$\|\rho_{a'}^{(\mathsf{in})}\rangle_{\mathtt{mtso}}$	$\|\dot{\rho}^{(t-1)}_{p_1',p_2'}\rangle$
$\mathtt{o}$	other registers	U_1	$B_{2,1}$	$\|\eta_0^{(\mathsf{in})}\rangle$	$\|\gamma_0^{(t-1)}\rangle$
		U	B_2	$\|\eta_1^{(\mathsf{in})}\rangle$	$\|\gamma_1^{(t-1)}\rangle$
		S	$U_{gc}\Lambda^{\mathtt{p}_2/\mathtt{m}}U_{gc}^\dagger$	$\|\rho_a^{(\mathsf{out})}\rangle_{\mathtt{mtso}}$	$\|\dot{\rho}^{(t)}_{p_1,p_2}\rangle$
				$\|\rho_{a'}^{(\mathsf{out})}\rangle_{\mathtt{mtso}}$	$\|\dot{\rho}^{(t)}_{p_1',p_2'}\rangle$
				$\|\eta_0^{(\mathsf{out})}\rangle$	$\|\gamma_0^{(t)}\rangle$
				$\|\eta_1^{(\mathsf{out})}\rangle$	$\|\gamma_1^{(t)}\rangle$

First, we argue that the premises in Lem. 6 are satisfied with the notation listed in Table 3:

- Lem. 6 requires that W_1 should work as the identity operator on register $\mathtt{s}$. In terms of the Claim 12 notation, this is satisfied by $C_{2,1}$ (playing the role of W_1) who works as identity on registers $\mathtt{u}$, $\mathtt{s}$, $\mathtt{t}_1$, $\mathtt{gc}$, and $\mathtt{lc}$ (playing the role of registers $\mathtt{s}$). (Recall $C_{2,1}$ from Table 1.)
- Lem. 6 requires that W_0 should be the swap operator between $\mathtt{m}$ and $\mathtt{t}$. In terms of the Claim 12 notation, this is satisfied by $C_{2,0}$ (playing the role of W_0), who is the swap operator between registers $\mathtt{m}$ and $\mathtt{t}_2$ (playing the role of $\mathtt{m}$ and $\mathtt{t}$ respectively). (Recall $C_{2,0}$ from Algo. 7.1.)
- Lem. 6 requires that $\widetilde{W}$ is the identity operator on branch $|a\rangle_{\mathtt{a}}$ and is identical to W_0 on branches $|a'\rangle_{\mathtt{a}}$ with $a' \neq a$. In terms of the Claim 12 notation, this is satisfied by $\ddot{C}_2$ (playing the role of $\widetilde{W}$), who is the identity operator on branch $|p_1,p_2\rangle_{\mathtt{p}_1\mathtt{p}_2}$ (playing the role of $|a\rangle_{\mathtt{a}}$) and is identical to $C_{2,0}$ (playing the role of W_0) on branches $|p_1',p_2'\rangle_{\mathtt{p}_1\mathtt{p}_2}$ (playing the role of $|a'\rangle_{\mathtt{a}}$) with $(p_1',p_2') \neq (p_1,p_2)$. (Recall $\ddot{C}_2$ from Algo. 7.1)

– Lem. 6 requires that U_1 should work as identity on register $\mathtt{s}$. In terms of the Claim 12 notation, this is satisfied by $B_{2,1}$ (playing the role of U_1), who works as identity on registers $\mathtt{u}$, $\mathtt{s}$, $\mathtt{t}_1$, $\mathtt{gc}$, and $\mathtt{lc}$ (playing the role of register $\mathtt{s}$). (Recall $B_{2,1}$ from Table 1.)

– Lem. 6 requires that S should act non-trivially *only* on registers $\mathtt{a}$ and $\mathtt{s}$. In terms of the Claim 12 notation, this is satisfied by $U_{gc}\Lambda^{\mathtt{p}_2/\mathtt{m}}U_{gc}^\dagger$ (playing the role of S)—recall from Claim 10 that Λ acts non-trivially on registers $\mathtt{m}$, $\mathtt{u}$, $\mathtt{s}$, $\mathtt{p}_1$, $\mathtt{t}_1$, $\mathtt{lc}$, and $\mathtt{gc}$. Thus, $U_{gc}\Lambda^{\mathtt{p}_2/\mathtt{m}}U_{gc}^\dagger$ acts non-trivially on registers $\mathtt{p}_2$, $\mathtt{u}$, $\mathtt{s}$, $\mathtt{p}_1$, $\mathtt{t}_1$, $\mathtt{lc}$, and $\mathtt{gc}$, which constitute registers $\mathtt{a}$ and $\mathtt{s}$ (see Table 3).

Finally, we apply Lem. 6 (with the notation in Table 3) to the $|\gamma_0^{(t-1)}\rangle$ and $|\gamma_1^{(t-1)}\rangle$ defined in Eq. (8.90) and (8.91) (playing the role of $|\eta_0^{(\mathsf{in})}\rangle$ and $|\eta_1^{(\mathsf{in})}\rangle$ in Lem. 6). This implies the existence of (possibly sub-normalized) pure states $\{|\dot\rho^{(t)}_{p_1',p_2'}\rangle\}_{(p_1',p_2')\in\{0,1\}^\ell\times\{0,1\}^\ell}$ (playing the role of $\{|\rho^{(\mathsf{out})}_{a'}\rangle_{\mathtt{mtso}}\}_{a'\in\{0,1\}^\ell}$ in Lem. 6) such that the following holds

$$|\gamma_0^{(t)}\rangle = |p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\dot\rho^{(t)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)} |p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}B_{2,1}^\dagger C_{2,1}^\dagger|\dot\rho^{(t)}_{p_1',p_2'}\rangle$$
$$|\gamma_1^{(t)}\rangle = |p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\dot\rho^{(t)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)} |p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}|\dot\rho^{(t)}_{p_1',p_2'}\rangle,$$

which are exactly Eq. (8.94) and (8.95) in Claim 12.

This completes the proof of Claim 12.

□

Finishing the Proof for the Induction Step. With Claims 11 and 12 at hand, we now finish the proof for the induction Step.

Proof for Eq. (8.63). We first establish Eq. (8.63):

$$|\phi^{(t)}\rangle = D_2|\gamma_0^{(t)}\rangle \tag{8.96}$$
$$= D_2\left(|p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\dot\rho^{(t)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)} |p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}B_{2,1}^\dagger C_{2,1}^\dagger|\dot\rho^{(t)}_{p_1',p_2'}\rangle\right) \tag{8.97}$$
$$= |p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}D_{2,1}|\dot\rho^{(t)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)} |p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}B_{2,1}^\dagger C_{2,1}^\dagger|\dot\rho^{(t)}_{p_1',p_2'}\rangle \tag{8.98}$$
$$= |p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\rho^{(t)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)} |p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}B_{2,1}^\dagger C_{2,1}^\dagger|\rho^{(t)}_{p_1',p_2'}\rangle, \tag{8.99}$$

where

– Eq. (8.96) follows from Eq. (8.70) in Claim 11 and Eq. (8.92) in Claim 12;

– Eq. (8.97) follows from Eq. (8.94) in Claim 12;

– Eq. (8.98) follows from the definition of D_2 (see Algo. 7.1);

– Eq. (8.99) follows by defining

$$|\rho^{(t)}_{p_1,p_2}\rangle := D_{2,1}|\dot\rho^{(t)}_{p_1,p_2}\rangle \quad\text{and}\quad |\rho^{(t)}_{p_1',p_2'}\rangle := |\dot\rho^{(t)}_{p_1',p_2'}\rangle \quad \big(\forall(p_1',p_2')\in\big(\{0,1\}^\ell\times\{0,1\}^\ell\big)\setminus\{(p_1,p_2)\}\big). \tag{8.100}$$

Note that Eq. (8.99) is exactly Eq. (8.63) in Lem. 18.

Proof for Eq. (8.64). We next establish Eq. (8.64):

$$|\psi^{(t)}\rangle = D_2|\gamma_1^{(t)}\rangle \tag{8.101}$$

$$= D_2\left(|p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\dot{\rho}^{(t)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)}|p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}|\dot{\rho}^{(t)}_{p_1',p_2'}\rangle\right) \tag{8.102}$$

$$= |p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}D_{2,1}|\dot{\rho}^{(t)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)}|p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}|\dot{\rho}^{(t)}_{p_1',p_2'}\rangle \tag{8.103}$$

$$= |p_1\rangle_{\mathtt{p}_1}|p_2\rangle_{\mathtt{p}_2}|\rho^{(t)}_{p_1,p_2}\rangle + \sum_{(p_1',p_2')\neq(p_1,p_2)}|p_1'\rangle_{\mathtt{p}_1}|p_2'\rangle_{\mathtt{p}_2}C_{2,0}|\rho^{(t)}_{p_1',p_2'}\rangle, \tag{8.104}$$

where

- Eq. (8.101) follows from Eq. (8.71) in Claim 11 and Eq. (8.93) in Claim 12;
- Eq. (8.102) follows from Eq. (8.95) in Claim 12;
- Eq. (8.103) follows from the definition of D_2 (see Algo. 7.1);
- Eq. (8.104) follows from *the same definitions* of $|\rho^{(t)}_{p_1,p_2}\rangle$ and $|\rho^{(t)}_{p_1',p_2'}\rangle$ in Expression (8.100).

Note that Eq. (8.104) is exactly Eq. (8.64) in Lem. 18.

This finishes the proof of the induction step of Lem. 18.

Finally, we remark that our proof for the base case in Sec. 8.2.1 and the proof for the induction step in Sec. 8.2.2 together finish the proof of Lem. 18, which in turn finishes the proof of Lem. 16 eventually.

9 Proving Lem. 13 (Full Version)

In this section, we provide the proof of Lem. 13 for the general case (in contrast to the warm-up case $K = 2$ shown in Sec. 8).

Due to a similar argument as we did at the beginning of Sec. 8.1, we claim that: to prove Lem. 13, it suffices to establish the following Lem. 19.

Lemma 19 (Invariance in $H_{k-1}^{(J,\mathsf{pat})}$ and $H_k^{(J,\mathsf{pat})}$). *For a (J,pat) satisfying the requirements in Game 7.1 and a $k\in[K]$, assume that during the execution of $H_{k-1}^{(J,\mathsf{pat})}$ (and $H_k^{(J,\mathsf{pat})}$), the global counter reaches value k for T times in total. For each $t\in[T]$, there exist (possibly sub-normalized) pure states $\{|\rho^{(t)}_{p_1',\ldots,p_k'}\rangle\}_{p_1',\ldots,p_k'\in\{0,1\}^{k\ell}}$ so that the following holds: in hybrid $H_{k-1}^{(J,\mathsf{pat})}$ (resp. $H_k^{(J,\mathsf{pat})}$), when the global counter reaches value k for the t-th time, the overall state can be written as the $|\phi^{(t)}\rangle$ (resp. $|\psi^{(t)}\rangle$) defined as follows:*

$$|\phi^{(t)}\rangle = |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}|\rho^{(t)}_{p_1,\ldots,p_k}\rangle + \sum_{(p_1',\ldots,p_k')\neq(p_1,\ldots,p_k)}|p_1',\ldots,p_k'\rangle_{\mathtt{p}_1\ldots p_k}C_{k,0}B_{k,1}^{\dagger}C_{k,1}^{\dagger}|\rho^{(t)}_{p_1',\ldots,p_k'}\rangle, \tag{9.1}$$

$$|\psi^{(t)}\rangle = |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}|\rho^{(t)}_{p_1,\ldots,p_k}\rangle + \sum_{(p_1',\ldots,p_k')\neq(p_1,\ldots,p_k)}|p_1',\ldots,p_k'\rangle_{\mathtt{p}_1\ldots p_k}C_{k,0}|\rho^{(t)}_{p_1',\ldots,p_k'}\rangle, \tag{9.2}$$

where the summations are taken over all $(p'_1, \ldots, p'_k) \in \{0,1\}^{k\ell} \setminus \{(p_1, \ldots, p_k)\}$ *(abbreviated as* $(p'_1, \ldots, p'_k) \neq (p_1, \ldots, p_k)$ *in the above), the* $(p_1, \ldots, p_k)$ *are defined in* pat*, and the unitaries* $B_{k,1}$*,* $C_{k,0}$*, and* $C_{k,1}$ *are as defined in Algo. 7.1.*

Proving Lem. 19. Similar as the proofs for Lem. 17 and 18, we establish Lem. 19 through mathematical induction on the number $t \in [T]$, indicating the time at which the global counter reaches the value k. Throughout this proof, we will monitor the evolution of the overall states in both $H_{k-1}^{(J,\mathsf{pat})}$ and $H_k^{(J,\mathsf{pat})}$ simultaneously. This is done in Sec. 9.1 and 9.2 respective.

9.1 Base Case ($t = 1$)

We first derive how the overall state evolves in $H_{k-1}^{(J,\mathsf{pat})}$.

This base case corresponds to the very first time the global counter reaches value k. By definition, this is due to the $\mathsf{sq}(k)$ query.

We assume w.l.o.g. that the overall state right before this query is some pure state $|\rho\rangle$. Then, by definition, the $\mathsf{sq}(k)$ query corresponds to the following procedure: $\mathcal{S}$ first applies her local unitary S (followed by the projector $|\downarrow\rangle\langle\downarrow|_{\mathsf{u}}$ which we hide as per Rmk. 12). Next, according to the notation in Game 7.2,

- If $b_k = 0$, then the operator $D_k C_k B_k \, |p_1, \ldots, p_k\rangle\langle p_1, \ldots, p_k|_{\mathsf{p}_1 \ldots \mathsf{p}_k} \, A_k U_{gc}$ will be applied;
- If $b_k = 1$, then the operator $C_{k,0} \, |p_1, \ldots, p_k\rangle\langle p_1, \ldots, p_k|_{\mathsf{p}_1 \ldots \mathsf{p}_k} \, A_k U_{gc}$ will be applied.

In the following, we show the proof for $b_k = 0$ only; the other case (i.e., $b_k = 1$) can be established using the same argument.

It follows from the above discussion that the state $|\phi^{(1)}\rangle$ in $H_1^{(J,\mathsf{pat})}$ can be written as follows:

$$\begin{aligned}
|\phi^{(1)}\rangle &= D_k C_k B_k \, |p_1, \ldots, p_k\rangle\langle p_1, \ldots, p_k|_{\mathsf{p}_1 \ldots \mathsf{p}_k} \, A_k U_{gc} S |\rho\rangle \\
&= D_k C_k B_k |p_1, \ldots, p_k\rangle_{\mathsf{p}_1 \ldots \mathsf{p}_k} |\dot{\rho}_{p_1, \ldots, p_k}\rangle && (9.3) \\
&= |p_1, \ldots, p_k\rangle_{\mathsf{p}_1 \ldots \mathsf{p}_k} D_{k,1} C_{k,1} B_{k,1} |\rho_{p_1, \ldots, p_k}\rangle && (9.4) \\
&= |p_1, \ldots, p_k\rangle_{\mathsf{p}_1 \ldots \mathsf{p}_k} |\rho^{(1)}_{p_1, \ldots, p_k}\rangle, && (9.5)
\end{aligned}$$

where

- Eq. (9.3) follows by defining $|\rho_{p_1, \ldots, p_k}\rangle \coloneqq \langle p_1, \ldots, p_k|_{\mathsf{p}_1 \ldots \mathsf{p}_k} A_k U_{gc} S |\rho\rangle$;
- Eq. (9.4) follows from the definitions of B_k, C_k, and D_k (see Algo. 7.1);
- Eq. (9.5) follows by defining $|\rho^{(1)}_{p_1, \ldots, p_k}\rangle \coloneqq D_{k,1} C_{k,1} B_{k,1} |\rho_{p_1, \ldots, p_k}\rangle$.

It is straightforward that the $|\phi^{(1)}\rangle$ shown in Eq. (9.5) satisfies the format shown in Eq. (9.1) when $t = 1$.

Also, note that the game $H_{k-1}^{(J,\mathsf{pat})}$ and $H_k^{(J,\mathsf{pat})}$ are identical so far and thus $|\phi^{(1)}\rangle = |\psi^{(1)}\rangle$. Therefore, it follows from Eq. (9.5) that

$$|\psi^{(1)}\rangle = |p_1, \ldots, p_k\rangle_{\mathsf{p}_1 \ldots \mathsf{p}_k} |\rho^{(1)}_{p_1, \ldots, p_k}\rangle.$$

Such a $|\psi^{(1)}\rangle$ satisfies the format shown in Eq. (9.2) with $t = 1$ as well.

This finishes the proof for the base case $t = 1$.

9.2 Induction Step ($t \geq 2$)

We assume $|\phi^{(t-1)}\rangle$ and $|\psi^{(t-1)}\rangle$ satisfy the requirements in Lem. 19, and show in the following that Lem. 19 holds when the global counter reaches k for the t-th time.

We establish this claim by considering the following MECE (mutually exclusive and collectively exhaustive) cases:

1. **Case 1:** The t-th arrival at value k happens because of the following operations performed by the simulator $\mathcal{S}$:
   (a) $\mathcal{S}$ first applies her local unitary S and then makes[18] an $\uparrow$-query, which brings the global counter to $k-1$;
   (b) Next, $\mathcal{S}$ could perform any operations, as long as they keep the global counter $\leq k-1$. For instance, $\mathcal{S}$ can choose to further rewind the execution by making another $\uparrow$-query, or $\mathcal{S}$ can choose to make $\uparrow$ and $\downarrow$ queries that cause the global counter to jump between $i-1$ and i as long as $i \leq k-1$. The only restriction is that $\mathcal{S}$ will not bring the global counter back to k (which is captured by the next step).

      Defining Operator Υ: We define the operations performed by $\mathcal{S}$ as an operator Υ.

   (c) After the application of Υ, the simulator $\mathcal{S}$ eventually makes a $\downarrow$-query that bring the global counter back to k (i.e., exactly the t-th arrival at value k).
2. **Case 1:** The t-th arrival at value k happens because of the following operations performed by the simulator $\mathcal{S}$:
   (a) $\mathcal{S}$ first applies her local unitary S and then makes a $\downarrow$-query, which brings the global counter to $k+1$;
   (b) Next, $\mathcal{S}$ could perform any operations, as long as they keep the global counter $\geq k+1$. For instance, $\mathcal{S}$ can choose to make another $\downarrow$-query, or $\mathcal{S}$ can choose to make $\uparrow$ and $\downarrow$ queries that cause the global counter to jump between i and $i+1$ as long as $i \geq k+1$. The only restriction is that $\mathcal{S}$ will not bring the global counter back to k (which is captured by the next step).
   (c) After the application of Λ, the simulator $\mathcal{S}$ eventually makes a $\uparrow$-query that bring the global counter back to k (i.e., exactly the t-th arrival at value k).

   Note that Steps 2a to 2c described above not only transition $|\phi^{(t-1)}\rangle$ to $|\phi^{(t)}\rangle$, but they also transition $|\psi^{(t-1)}\rangle$ to $|\psi^{(t)}\rangle$. This is because these three steps maintain the global counter $\geq k$, and $H_{k-1}^{(J,\mathsf{pat})}$ and $H_{k}^{(J,\mathsf{pat})}$ behave identically by definition until the global counter returns to k (see Game 7.2).

   Defining Operator Λ: We define the operations performed in Steps 2a to 2c as an operator Λ.

In the following, we show the proof for these two cases in Sec. 9.2.1 respectively.

9.2.1 Proving the Induction Step: Case 1

We first describe formally how $|\phi^{(t-1)}\rangle$ (resp. $|\psi^{(t-1)}\rangle$) evolves into $|\phi^{(t)}\rangle$ (resp. $|\psi^{(t)}\rangle$) in Case 1. According to the description in Case 1 and our notation in Game 7.2, it is clear that the states

[18] Recall that this is determined by measuring the $\mathtt{u}$ reigster. And further recall that this measurement is replaced with the projector $|\downarrow\rangle\langle\downarrow|_{\mathtt{u}}$ because we are in the sub-normalized game (see Game 7.2).

$|\phi^{(t)}\rangle$ and $|\psi^{(t)}\rangle$ in this case can be written respectively as:

$$|\phi^{(t)}\rangle = D_k C_k B_k A_k U_{gc} \Upsilon U_{gc}^\dagger A_k^\dagger B_k^\dagger C_k^\dagger D_k^\dagger S |\phi^{(t-1)}\rangle \tag{9.6}$$

$$|\psi^{(t)}\rangle = D_k \ddot{C}_k A_k U_{gc} \Upsilon U_{gc}^\dagger A_k^\dagger \ddot{C}_k^\dagger D_k^\dagger S |\psi^{(t-1)}\rangle, \tag{9.7}$$

where the operator Υ is defined in Step 1b.

We first make a useful claim that specifies the registers on which the operator Υ (defined in Step 1b) acts non-trivially. This claim can be treated as the analogue of Claim 10.

Claim 13 (Non-Trivial Registers for Υ). *The Υ defined in Step 1b acts non-trivially only on registers* $\mathtt{m}$, $\mathtt{u}$, $\mathtt{s}$, $\mathtt{p}_1 \dots \mathtt{p}_{k-1}$, $\mathtt{t}_1 \dots \mathtt{t}_{k-1}$, $\mathtt{lc}$, *and* $\mathtt{gc}$ *(i.e., it works as the identity operator on the other registers).*

Proof of Claim 13. By definition, Υ only contains operations that happen when the global counter is $k-1$ or smaller.

Also, recall that we are talking about hybrids $H_{k-1}^{(J,\mathsf{pat})}$ and $H_k^{(J,\mathsf{pat})}$ now. Thus, all the $\downarrow$ and $\uparrow$ queries happening at global counter $i \le k-1$ will be answered by the "dummy" operators $\ddot{V}$ and $\ddot{V}^\dagger$ respective. (One exception is the $\widetilde{V}$ corresponding $\mathsf{sq}(i)$. But this does not matter in the current proof—Right before the behaviors defined in Case 1 happen, the global counter is already k, which implies that $\mathsf{sq}(i)$ $(\forall i \in [k])$ has already happened and thus cannot be a part of the operator Υ.)

Therefore, according to our notation in Game 7.2, operator Υ can be written as $\Upsilon = \Gamma_z \Gamma_{z-1} \cdots \Gamma_1$ (for some integer z) where each Γ_i $(i \in [z])$ comes from the set of operators $\{S, U_{gc}\} \cup \{A_i\}_{i\in[k-1]} \cup \{\ddot{C}_i\}_{i\in[k-1]} \cup \{D_i\}_{i\in[k-1]}$. It then follows from Table 1 that Υ acts non-trivially only on registers $\mathtt{m}$, $\mathtt{u}$, $\mathtt{s}$, $\mathtt{p}_1 \dots \mathtt{p}_{k-1}$, $\mathtt{t}_1 \dots \mathtt{t}_{k-1}$, $\mathtt{lc}$, and $\mathtt{gc}$.

This completes the proof of Claim 13.

□

High-Level Idea for the Sequel. Recall that our eventual goal is to prove that the states $|\phi^{(t)}\rangle$ and $|\psi^{(t)}\rangle$ are of the format shown in Eq. (9.1) and (9.2) in Lem. 18. At a high level, we prove it by applying Lem. 6 to Eq. (9.6) and (9.7). But we first need to perform some preparation work, putting Eq. (9.6) and (9.7) into a format that is more "compatible" with Lem. 6. In the sequel, we first perform the preparation work in Claims 14 and 15. Then, we show on Page 95 how to use Claims 14 and 15 to complete the proof for the induction step.

The following Claim 14 can be treated as an analogue of Claim 11.

Claim 14. *There exist (possibly sub-normalized) pure states* $\{\dot{\rho}^{(t-1)}_{p'_1,\dots,p'_k}\}_{(p'_1,\dots,p'_k)\in\{0,1\}^{k\ell}}$ *so that the following holds*

$$|\phi^{(t)}\rangle = D_k C_k B_k U_{gc} \Upsilon^{\mathtt{p}_k/\mathtt{m}} U_{gc}^\dagger B_k^\dagger C_k^\dagger \Bigg(|p_1,\dots,p_k\rangle_{\mathtt{p}_1\dots\mathtt{p}_k} |\dot{\rho}^{(t-1)}_{p_1,\dots,p_k}\rangle + \sum_{(p'_1,\dots,p'_k)\neq(p_1,\dots,p_k)} |p'_1,\dots,p'_k\rangle_{\mathtt{p}_1\dots\mathtt{p}_k} C_{k,0} B_{k,1}^\dagger C_{k,1}^\dagger |\dot{\rho}^{(t-1)}_{p'_1,\dots,p'_k}\rangle \Bigg) \tag{9.8}$$

$$|\psi^{(t)}\rangle = D_k \ddot{C}_k U_{gc} \Upsilon^{\mathtt{p}_k/\mathtt{m}} U_{gc}^\dagger \ddot{C}_k^\dagger \Bigg(|p_1,\dots,p_k\rangle_{\mathtt{p}_1\dots\mathtt{p}_k} |\dot{\rho}^{(t-1)}_{p_1,\dots,p_k}\rangle + \sum_{(p'_1,\dots,p'_k)\neq(p_1,\dots,p_k)} |p'_1,\dots,p'_k\rangle_{\mathtt{p}_1\dots\mathtt{p}_k} C_{k,0} |\dot{\rho}^{(t-1)}_{p'_1,\dots,p'_k}\rangle \Bigg) \tag{9.9}$$

where $\Upsilon^{\mathtt{p}_k/\mathtt{m}}$ is identical to the Υ defined in Step 1b, except that it treats register $\mathtt{p}_k$ as $\mathtt{m}$, and the summations are taken over all $(p'_1,\ldots,p'_k) \in \{0,1\}^{k\ell} \setminus \{(p_1,\ldots,p_k)\}$ (abbreviated as $(p'_1,\ldots,p'_k) \neq (p_1,\ldots,p_k)$ in the above).

Proof of Claim 14. First, notice that

$$A_k U_{gc} \Upsilon U_{gc}^\dagger A_k^\dagger = A_k U_{gc} \Upsilon A_k^\dagger U_{gc}^\dagger \tag{9.10}$$
$$= A_k U_{gc} A_k^\dagger \Upsilon^{\mathtt{p}_k/\mathtt{m}} U_{gc}^\dagger \tag{9.11}$$
$$= A_k A_k^\dagger U_{gc} \Upsilon^{\mathtt{p}_k/\mathtt{m}} U_{gc}^\dagger \tag{9.12}$$
$$= U_{gc} \Upsilon^{\mathtt{p}_k/\mathtt{m}} U_{gc}^\dagger, \tag{9.13}$$

where

- Eq. (9.10) and (9.12) follows from the fact that U_{gc} acts non-trivially on different registers from A_k;
- Eq. (9.11) from the fact that Υ acts as the identity operator on $\mathtt{p}_k$ (see Claim 13) and that A_k is nothing but a swap operator between $\mathtt{m}$ and $\mathtt{p}_k$ (see Algo. 6.1).

Proving Eq. (9.8). We now show the derivation for $|\phi^{(t)}\rangle$:

$$|\phi^{(t)}\rangle = D_k C_k B_k A_k U_{gc} \Upsilon U_{gc}^\dagger A_k^\dagger B_k^\dagger C_k^\dagger D_2^\dagger S |\phi^{(t-1)}\rangle \tag{9.14}$$
$$= D_k C_k B_k U_{gc} \Upsilon^{\mathtt{p}_k/\mathtt{m}} U_{gc}^\dagger B_k^\dagger C_k^\dagger D_k^\dagger S |\phi^{(t-1)}\rangle \tag{9.15}$$
$$= D_k C_k B_k U_{gc} \Upsilon^{\mathtt{p}_k/\mathtt{m}} U_{gc}^\dagger B_k^\dagger C_k^\dagger D_k^\dagger S \Bigg(|p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} |\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0} B_{k,1}^\dagger C_{k,1}^\dagger |\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \Bigg) \tag{9.16}$$
$$= D_k C_k B_k U_{gc} \Upsilon^{\mathtt{p}_k/\mathtt{m}} U_{gc}^\dagger B_k^\dagger C_k^\dagger D_k^\dagger \Bigg(|p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} S |\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} S C_{k,0} B_{k,1}^\dagger C_{k,1}^\dagger |\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \Bigg) \tag{9.17}$$
$$= D_k C_k B_k U_{gc} \Upsilon^{\mathtt{p}_k/\mathtt{m}} U_{gc}^\dagger B_k^\dagger C_k^\dagger \Bigg(|p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} D_{k,1}^\dagger S |\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} S C_{k,0} B_{k,1}^\dagger C_{k,1}^\dagger |\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \Bigg) \tag{9.18}$$
$$= D_k C_k B_k U_{gc} \Upsilon^{\mathtt{p}_k/\mathtt{m}} U_{gc}^\dagger B_k^\dagger C_k^\dagger \Bigg(|p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} D_{k,1}^\dagger S |\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0} S^{\mathtt{t}_k/\mathtt{m}} B_{k,1}^\dagger C_{k,1}^\dagger |\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \Bigg) \tag{9.19}$$
$$= D_k C_k B_k U_{gc} \Upsilon^{\mathtt{p}_k/\mathtt{m}} U_{gc}^\dagger B_k^\dagger C_k^\dagger \Bigg(|p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} D_{k,1}^\dagger S |\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0} B_{k,1}^\dagger C_{k,1}^\dagger S^{\mathtt{t}_k/\mathtt{m}} |\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \Bigg) \tag{9.20}$$

$$= D_k C_k B_k U_{gc} \Upsilon^{\mathtt{p_k}/\mathtt{m}} U_{gc}^\dagger B_k^\dagger C_k^\dagger \Bigg(|p_1,\ldots,p_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}} |\dot{\rho}^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}} C_{k,0} B_{k,1}^\dagger C_{k,1}^\dagger |\dot{\rho}^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \Bigg), \quad (9.21)$$

where

- Eq. (9.14) follows from Eq. (9.6);
- Eq. (9.15) follows from Eq. (9.13);
- Eq. (9.16) follows from our induction assumption;
- Eq. (9.17) from from the fact that S acts as the identity operator on registers $\mathtt{p_1}\ldots\mathtt{p_k}$ (see Table 1);
- Eq. (9.18) from the definition of D_k (see Algo. 7.1);
- Eq. (9.19) from the fact that S acts as the identity operator on $\mathtt{t_k}$ (see Table 1) and $C_{k,0}$ is nothing but a swap operator between $\mathtt{t_k}$ and $\mathtt{m}$ (see Algo. 7.1); (Note that $S^{\mathtt{t_k}/\mathtt{m}}$ is defined to be an operator that is identical to S except that it treats $\mathtt{t_k}$ as $\mathtt{m}$.)
- Eq. (9.20) follows from the fact that $S^{\mathtt{t_k}/\mathtt{m}}$ acts non-trivially on different registers from $B_{k,1}$ and $C_{k,1}$ (see Table 1);
- Eq. (9.21) follows by defining

$$\begin{cases} |\dot{\rho}^{(t-1)}_{p_1,\ldots,p_k}\rangle \coloneqq D_{k,1}^\dagger S |\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle \\ |\dot{\rho}^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \coloneqq S^{\mathtt{t_k}/\mathtt{m}} |\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \qquad \forall (p'_1,\ldots,p'_k) \in \big(\{0,1\}^{k\ell} \setminus \{(p_1,\ldots,p_k)\}\big). \end{cases} \quad (9.22)$$

Eq. (9.21) finishes the proof of Eq. (9.8) in Claim 14.

Proving Eq. (9.9). We now show the derivation for $|\psi^{(t)}\rangle$. This is almost identical to the above proof for Eq. (9.8). Nevertheless, we present it for the sake of completeness.

$$|\psi^{(t)}\rangle = D_k \ddot{C}_k A_k U_{gc} \Upsilon U_{gc}^\dagger A_k^\dagger \ddot{C}_k^\dagger D_k^\dagger S |\psi^{(t-1)}\rangle \quad (9.23)$$

$$= D_k \ddot{C}_k U_{gc} \Upsilon^{\mathtt{p_k}/\mathtt{m}} U_{gc}^\dagger \ddot{C}_k^\dagger D_k^\dagger S |\psi^{(t-1)}\rangle \quad (9.24)$$

$$= D_k \ddot{C}_k U_{gc} \Upsilon^{\mathtt{p_k}/\mathtt{m}} U_{gc}^\dagger \ddot{C}_k^\dagger D_k^\dagger S \Bigg(|p_1,\ldots,p_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}} |\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}} C_{k,0} |\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \Bigg) \quad (9.25)$$

$$= D_k \ddot{C}_k U_{gc} \Upsilon^{\mathtt{p_k}/\mathtt{m}} U_{gc}^\dagger \ddot{C}_k^\dagger \Bigg(|p_1,\ldots,p_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}} D_{k,1}^\dagger S |\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}} C_{k,0} S^{\mathtt{t_k}/\mathtt{m}} |\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \Bigg) \quad (9.26)$$

$$= D_k \ddot{C}_k U_{gc} \Upsilon^{\mathtt{p_k}/\mathtt{m}} U_{gc}^\dagger \ddot{C}_k^\dagger \Bigg(|p_1,\ldots,p_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}} |\dot{\rho}^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}} C_{k,0} |\dot{\rho}^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \Bigg), \quad (9.27)$$

where

– Eq. (9.23) follows from Eq. (9.7);

– Eq. (9.24) follows from Eq. (9.13);

– Eq. (9.25) follows from our induction assumption;

– Eq. (9.26) follows from a similar argument as we did to derive Eq. (9.20) from Eq. (9.16);

– Eq. (9.27) follows from *the same definitions* of $|\dot{\rho}^{(t-1)}_{p_1,\ldots,p_k}\rangle$ and $|\dot{\rho}^{(t-1)}_{p'_1,\ldots,p'_k}\rangle$ as shown in Expression (9.22).

Eq. (9.27) finishes the proof of Eq. (9.9) in Claim 14.

This completes the proof of Claim 14.

□

The following Claim 15 can be treated as an analogue of Claim 12.

Claim 15. *Let* $\Upsilon^{\mathtt{p}_k/\mathtt{m}}$ *and* $\{\dot{\rho}^{(t-1)}_{p'_1,\ldots,p'_k}\}_{(p'_1,\ldots,p'_k)\in\{0,1\}^{k\ell}}$ *be as defined in Claim 14. Let*

$$|\gamma_0^{(t-1)}\rangle \coloneqq |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}|\dot{\rho}^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0}B^\dagger_{k,1}C^\dagger_{k,1}|\dot{\rho}^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \tag{9.28}$$

$$|\gamma_1^{(t-1)}\rangle \coloneqq |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}|\dot{\rho}^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0}|\dot{\rho}^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \tag{9.29}$$

$$|\gamma_0^{(t)}\rangle \coloneqq C_kB_kU_{gc}\Upsilon^{\mathtt{p}_k/\mathtt{m}}U^\dagger_{gc}B^\dagger_kC^\dagger_k|\gamma_0^{(t-1)}\rangle \tag{9.30}$$

$$|\gamma_1^{(t)}\rangle \coloneqq \ddot{C}_kU_{gc}\Upsilon^{\mathtt{p}_k/\mathtt{m}}U^\dagger_{gc}\ddot{C}^\dagger_k|\gamma_1^{(t-1)}\rangle. \tag{9.31}$$

Then, there exist (possibly sub-normalized) pure states $\{\dot{\rho}^{(t)}_{p'_1,\ldots,p'_k}\}_{(p'_1,\ldots,p'_k)\in\{0,1\}^{k\ell}}$ *so that the following holds:*

$$|\gamma_0^{(t)}\rangle = |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}|\dot{\rho}^{(t)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0}B^\dagger_{k,1}C^\dagger_{k,1}|\dot{\rho}^{(t)}_{p'_1,\ldots,p'_k}\rangle \tag{9.32}$$

$$|\gamma_1^{(t)}\rangle = |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}|\dot{\rho}^{(t)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0}|\dot{\rho}^{(t)}_{p'_1,\ldots,p'_k}\rangle. \tag{9.33}$$

Proof of Claim 15. This claim follows from an application of Lem. 6, with the notation correspondence listed in Table 4. We provide a detailed explanation below.

First, we argue that the premises in Lem. 6 are satisfied with the notation listed in Table 4:

– Lem. 6 requires that W_1 should act as the identity operator on register $\mathtt{s}$. In terms of the Claim 15 notation, this is satisfied by $C_{k,1}$ (playing the role of W_1) who works as identity on registers $\mathtt{u}$, $\mathtt{s}$, $\mathtt{t}_1\ldots\mathtt{t}_{k-1}$, $\mathtt{gc}$, and $\mathtt{lc}$ (playing the role of registers $\mathtt{s}$). (Recall $C_{k,1}$ from Table 1.)

– Lem. 6 requires that W_0 should be the swap operator between $\mathtt{m}$ and $\mathtt{t}$. In terms of the Claim 15 notation, this is satisfied by $C_{k,0}$ (playing the role of W_0), who is the swap operator between registers $\mathtt{m}$ and $\mathtt{t}_k$ (playing the role of $\mathtt{m}$ and $\mathtt{t}$ respectively). (Recall $C_{k,0}$ from Algo. 7.1.)

– Lem. 6 requires that $\widetilde{W}$ is the identity operator on branch $|a\rangle_{\mathtt{a}}$ and is identical to W_0 on branches $|a'\rangle_{\mathtt{a}}$ with $a'\neq a$. In terms of the Claim 15 notation, this is satisfied by $\ddot{C}_k$ (playing the role of $\widetilde{W}$), who is the identity operator on branch $|p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}$ (playing the role of $|a\rangle_{\mathtt{a}}$) and is identical to $C_{k,0}$ (playing the role of W_0) on branches $|p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}$ (playing the role of $|a'\rangle_{\mathtt{a}}$) with $(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)$. (Recall $\ddot{C}_k$ from Algo. 7.1)

Table 4: Notation Correspondence between Lem. 6 and Claim 15

Registers		Operators		Random Variables	
In Lem. 6	In Claim 15	In Lem. 6	In Claim 15	In Lem. 6	In Claim 15
$\mathtt{a}$	$\mathtt{p}_1 \dots \mathtt{p}_k$	W_1	$C_{k,1}$	$\lvert a\rangle_{\mathtt{a}}$	$\lvert p_1,\dots,p_k\rangle_{\mathtt{p}_1\dots\mathtt{p}_k}$
$\mathtt{m}$	$\mathtt{m}$	W_0	$C_{k,0}$	$\lvert a'\rangle_{\mathtt{a}}$	$\lvert p'_1,\dots,p'_k\rangle_{\mathtt{p}_1\dots\mathtt{p}_k}$
$\mathtt{t}$	$\mathtt{t}_k$	W	C_k	$\lvert\rho_a^{(\mathsf{in})}\rangle_{\mathtt{mtso}}$	$\lvert\dot\rho^{(t-1)}_{p_1,\dots,p_k}\rangle$
$\mathtt{s}$	$\mathtt{u}, \mathtt{s}, \mathtt{t}_1 \dots \mathtt{t}_{k-1}, \mathtt{gc}, \mathtt{lc}$	$\widetilde{W}$	$\ddot{C}_k$	$\lvert\rho_{a'}^{(\mathsf{in})}\rangle_{\mathtt{mtso}}$	$\lvert\dot\rho^{(t-1)}_{p'_1,\dots,p'_k}\rangle$
$\mathtt{o}$	other registers	U_1	$B_{k,1}$	$\lvert\eta_0^{(\mathsf{in})}\rangle$	$\lvert\gamma_0^{(t-1)}\rangle$
		U	B_k	$\lvert\eta_1^{(\mathsf{in})}\rangle$	$\lvert\gamma_1^{(t-1)}\rangle$
		S	$U_{gc}\Upsilon^{\mathtt{p}_k/\mathtt{m}}U_{gc}^\dagger$	$\lvert\rho_a^{(\mathsf{out})}\rangle_{\mathtt{mtso}}$	$\lvert\dot\rho^{(t)}_{p_1,\dots,p_k}\rangle$
				$\lvert\rho_{a'}^{(\mathsf{out})}\rangle_{\mathtt{mtso}}$	$\lvert\dot\rho^{(t)}_{p'_1,\dots,p'_k}\rangle$
				$\lvert\eta_0^{(\mathsf{out})}\rangle$	$\lvert\gamma_0^{(t)}\rangle$
				$\lvert\eta_1^{(\mathsf{out})}\rangle$	$\lvert\gamma_1^{(t)}\rangle$

– Lem. 6 requires that U_1 should work as identity on register $\mathtt{s}$. In terms of the Claim 15 notation, this is satisfied by $B_{k,1}$ (playing the role of U_1), who works as identity on registers $\mathtt{u}$, $\mathtt{s}$, $\mathtt{t}_1 \dots \mathtt{t}_{k-1}$, $\mathtt{gc}$, and $\mathtt{lc}$ (playing the role of register $\mathtt{s}$). (Recall $B_{k,1}$ from Table 1.)

– Lem. 6 requires that S should act non-trivially *only* on registers $\mathtt{a}$ and $\mathtt{s}$. In terms of the Claim 15 notation, this is satisfied by $U_{gc}\Upsilon^{\mathtt{p}_k/\mathtt{m}}U_{gc}^\dagger$ (playing the role of S)—recall from Claim 13 that Υ acts non-trivially on registers $\mathtt{m}$, $\mathtt{u}$, $\mathtt{s}$, $\mathtt{p}_1 \dots \mathtt{p}_{k-1}$, $\mathtt{t}_1 \dots \mathtt{t}_{k-1}$, $\mathtt{lc}$, and $\mathtt{gc}$. Thus, $U_{gc}\Upsilon^{\mathtt{p}_k/\mathtt{m}}U_{gc}^\dagger$ acts non-trivially on registers $\mathtt{p}_k$, $\mathtt{u}$, $\mathtt{s}$, $\mathtt{p}_1 \dots \mathtt{p}_{k-1}$, $\mathtt{t}_1 \dots \mathtt{t}_{k-1}$, $\mathtt{lc}$, and $\mathtt{gc}$, which constitute registers $\mathtt{a}$ and $\mathtt{s}$ (see Table 4).

Finally, we apply Lem. 6 (with the notation in Table 4) to the $\lvert\gamma_0^{(t-1)}\rangle$ and $\lvert\gamma_1^{(t-1)}\rangle$ defined in Eq. (9.28) and (9.29) (playing the role of $\lvert\eta_0^{(\mathsf{in})}\rangle$ and $\lvert\eta_1^{(\mathsf{in})}\rangle$ in Lem. 6). This implies the existence of (possibly sub-normalized) pure states $\{\lvert\dot\rho^{(t)}_{p'_1,\dots,p'_k}\rangle\}_{(p'_1,\dots,p'_k)\in\{0,1\}^{k\ell}}$ (playing the role of $\{\lvert\rho_{a'}^{(\mathsf{out})}\rangle_{\mathtt{mtso}}\}_{a'\in\{0,1\}^\ell}$ in Lem. 6) such that the following holds

$$\lvert\gamma_0^{(t)}\rangle = \lvert p_1,\dots,p_k\rangle_{\mathtt{p}_1\dots\mathtt{p}_k}\lvert\dot\rho^{(t)}_{p_1,\dots,p_k}\rangle + \sum_{(p'_1,\dots,p'_k)\neq(p_1,\dots,p_k)} \lvert p'_1,\dots,p'_k\rangle_{\mathtt{p}_1\dots\mathtt{p}_k} C_{k,0}B_{k,1}^\dagger C_{k,1}^\dagger \lvert\dot\rho^{(t)}_{p'_1,\dots,p'_k}\rangle$$

$$\lvert\gamma_1^{(t)}\rangle = \lvert p_1,\dots,p_k\rangle_{\mathtt{p}_1\dots\mathtt{p}_k}\lvert\dot\rho^{(t)}_{p_1,\dots,p_k}\rangle + \sum_{(p'_1,\dots,p'_k)\neq(p_1,\dots,p_k)} \lvert p'_1,\dots,p'_k\rangle_{\mathtt{p}_1\dots\mathtt{p}_k} C_{k,0}\lvert\dot\rho^{(t)}_{p'_1,\dots,p'_k}\rangle,$$

which are exactly Eq. (9.32) and (9.33) in Claim 15.

This completes the proof of Claim 15.

□

Finishing the Proof for the Induction Step. With Claims 14 and 15 at hand, we now finish the proof for the induction Step.

Proof for Eq. (9.1). We first establish Eq. (9.1):

$$\lvert\phi^{(t)}\rangle = D_k\lvert\gamma_0^{(t)}\rangle \tag{9.34}$$

$$= D_k\left(|p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}|\dot{\rho}^{(t)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0}B^\dagger_{k,1}C^\dagger_{k,1}|\dot{\rho}^{(t)}_{p'_1,\ldots,p'_k}\rangle\right) \tag{9.35}$$

$$= |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} D_{k,1}|\dot{\rho}^{(t)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0}B^\dagger_{k,1}C^\dagger_{k,1}|\dot{\rho}^{(t)}_{p'_1,\ldots,p'_k}\rangle \tag{9.36}$$

$$= |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}|\rho^{(t)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0}B^\dagger_{k,1}C^\dagger_{k,1}|\rho^{(t)}_{p'_1,\ldots,p'_k}\rangle, \tag{9.37}$$

where

- Eq. (9.34) follows from Eq. (9.8) in Claim 14 and Eq. (9.30) in Claim 15;
- Eq. (9.35) follows from Eq. (9.32) in Claim 15;
- Eq. (9.36) follows from the definition of D_k (see Algo. 7.1);
- Eq. (9.37) follows by defining

$$\begin{cases} |\rho^{(t)}_{p_1,\ldots,p_k}\rangle \coloneqq D_{k,1}|\dot{\rho}^{(t)}_{p_1,\ldots,p_k}\rangle \\ |\rho^{(t)}_{p'_1,\ldots,p'_k}\rangle \coloneqq |\dot{\rho}^{(t)}_{p'_1,\ldots,p'_k}\rangle \qquad \forall (p'_1,\ldots,p'_k)\in\left(\{0,1\}^{k\ell}\setminus\{(p_1,\ldots,p_k)\}\right). \end{cases} \tag{9.38}$$

Note that Eq. (9.37) is exactly Eq. (9.1) in Lem. 19.

Proof for Eq. (9.2). Next, we establish Eq. (9.2):

$$|\psi^{(t)}\rangle = D_k|\gamma^{(t)}_1\rangle \tag{9.39}$$

$$= D_k\left(|p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}|\dot{\rho}^{(t)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0}|\dot{\rho}^{(t)}_{p'_1,\ldots,p'_k}\rangle\right) \tag{9.40}$$

$$= |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} D_{k,1}|\dot{\rho}^{(t)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0}|\dot{\rho}^{(t)}_{p'_1,\ldots,p'_k}\rangle \tag{9.41}$$

$$= |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}|\rho^{(t)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k} C_{k,0}|\rho^{(t)}_{p'_1,\ldots,p'_k}\rangle, \tag{9.42}$$

where

- Eq. (9.39) follows from Eq. (9.9) in Claim 14 and Eq. (9.31) in Claim 15;
- Eq. (9.40) follows from Eq. (9.33) in Claim 15;
- Eq. (9.41) follows from the definition of D_k (see Algo. 7.1);
- Eq. (9.42) follows from *the same definitions* of $|\rho^{(t)}_{p_1,\ldots,p_k}\rangle$ and $|\rho^{(t)}_{p'_1,\ldots,p'_k}\rangle$ shown in Expression (9.38).

Note that Eq. (9.42) is exactly Eq. (9.2) in Lem. 19.

This completes the proof of the induction step of Lem. 19 for Case 1.

9.2.2 Proving the Induction Step: Case 2

According to the description of Case 2, it holds that

$$|\phi^{(t)}\rangle = \Lambda|\phi^{(t-1)}\rangle \tag{9.43}$$

$$|\psi^{(t)}\rangle = \Lambda|\psi^{(t-1)}\rangle, \tag{9.44}$$

where recall that the operator Λ was defined toward the end of the description of Case 2.

Structure of Λ. While the exact format of Λ will not be substantial, our proof of Case 2 will rely on certain properties of Λ. We formalize these useful properties in the following Claim 16, which can be treated as an analogue of Claim 9.

Claim 16. *For the operator Λ defined above, there exist two operators Λ_0 and Λ_1 so that*

- *both Λ_1 and Λ_0 act as the identity operator on $\mathtt{p}_1 \dots \mathtt{p}_k$;*
- *Λ_0 acts as the identity operator on $\mathtt{t}_k$;*

and the following holds

$$\Lambda = |p_1,\dots,p_k\rangle\langle p_1,\dots,p_k|_{\mathtt{p}_1\dots\mathtt{p}_k} \otimes \Lambda_1 + \sum_{(p'_1,\dots,p'_k)\neq(p_1,\dots,p_k)} |p'_1,\dots,p'_k\rangle\langle p'_1,\dots,p'_k|_{\mathtt{p}_1\dots\mathtt{p}_k} \otimes \Lambda_0, \tag{9.45}$$

$$\Lambda_0 C_{k,0} = C_{k,0}\Lambda_0^{\mathtt{t}_k/\mathtt{m}}, \tag{9.46}$$

$$\Lambda_0 C_{k,0} B^\dagger_{k,1} C^\dagger_{k,1} = C_{k,0} B^\dagger_{k,1} C^\dagger_{k,1} \Lambda_0^{\mathtt{t}_k/\mathtt{m}}, \tag{9.47}$$

where $\Lambda_0^{\mathtt{t}_k/\mathtt{m}}$ is identical to Λ_0 except that it treats $\mathtt{t}_k$ as $\mathtt{m}$, and the summation is taken over all $(p'_1,\dots,p'_k) \in \{0,1\}^{k\ell} \setminus \{(p_1,\dots,p_k)\}$ (abbreviated as $(p'_1,\dots,p'_k) \neq (p_1,\dots,p_k)$ in the above).

Proof of Claim 16. First, note that Λ can be written as $\Lambda = \Gamma_z\Gamma_{z-1}\cdots\Gamma_1$ (for some integer z), where each Γ_i ($i \in [z]$) comes from the following set

$$\{S, U_{gc}\} \cup \{(A_i,B_i,C_i,D_i)\}_{i\in\{k+1,k+2,\dots,K\}} \cup \{|p_1,\dots,p_i\rangle\langle p_1,\dots,p_i|_{\mathtt{p}_1\dots\mathtt{p}_i}\}_{i\in\{k+1,k+2,\dots,K\}}. \tag{9.48}$$

This is because Λ only corresponds to the operations that happen when the global counter "jumps" between some i and $i+1$ with $i \geq k$ (see Game 7.2). We remark that $\mathcal{S}$ may also apply projectors on $\mathtt{u}$, but we consider it as a part of S as per Rmk. 12.

Therefore, to prove Claim 16, it suffices to show that for each operator Γ in the set shown in Expression (9.48), there exist two operators Γ_0 and Γ_1 so that

- both Γ_1 and Γ_0 act as the identity operator on $\mathtt{p}_1 \dots \mathtt{p}_k$;
- Γ_0 acts as the identity operator on $\mathtt{t}_k$;

and the following holds

$$\Gamma = |p_1,\dots,p_k\rangle\langle p_1,\dots,p_k|_{\mathtt{p}_1\dots\mathtt{p}_k} \otimes \Gamma_1 + \sum_{(p'_1,\dots,p'_k)\neq(p_1,\dots,p_k)} |p'_1,\dots,p'_k\rangle\langle p'_1,\dots,p'_k|_{\mathtt{p}_1\dots\mathtt{p}_k} \otimes \Gamma_0,$$

$$\Gamma_0 C_{k,0} = C_{k,0}\Gamma_0^{\mathtt{t}_k/\mathtt{m}},$$

$$\Gamma_0 C_{k,0} B^\dagger_{k,1} C^\dagger_{k,1} = C_{k,0} B^\dagger_{k,1} C^\dagger_{k,1} \Gamma_0^{\mathtt{t}_k/\mathtt{m}}.$$

In the following, we prove it for each possible Γ.

First, notice that the above is true for $\Gamma = |p_1,\dots,p_i\rangle\langle p_1,\dots,p_i|_{\mathtt{p}_1\dots\mathtt{p}_i}$ for each $i \in \{k+1, k+2,\dots,K\}$, simply because such a Γ can be written in the following format:

$$\Gamma = |p_1,\dots,p_i\rangle\langle p_1,\dots,p_i|_{\mathtt{p}_1\dots\mathtt{p}_i}$$

$$= |p_1,\dots,p_k\rangle\langle p_1,\dots,p_k|_{\mathtt{p_1\dots p_k}} \otimes \Gamma_1 + \sum_{(p'_1,\dots,p'_k)\neq(p_1,\dots,p_k)} |p'_1,\dots,p'_k\rangle\langle p'_1,\dots,p'_k|_{\mathtt{p_1\dots p_k}} \otimes \Gamma_0,$$

with $\Gamma_1 \coloneqq |p_{k+1},p_{k+2}\dots,p_i\rangle\langle p_{k+1},p_{k+2}\dots,p_i|_{\mathtt{p_{k+1}\dots p_i}}$ and $\Gamma_0 \coloneqq I$, and such a Γ_0 vacuously satisfies the requires $\Gamma_0 C_{k,0} = C_{k,0}\Gamma_0^{\mathtt{t_k/m}}$ and $\Gamma_0 C_{k,0} B_{k,1}^\dagger C_{k,1}^\dagger = C_{k,0} B_{k,1}^\dagger C_{k,1}^\dagger \Gamma_0^{\mathtt{t_k/m}}$.

The above is true for $\Gamma = U_{gc}$ as well, because U_{gc} acts non-trivially only on register $\mathtt{gc}$.

The above is also true for $\Gamma = S$ because (1) S does not act on $\mathtt{p_1\dots p_k}$, (2) S does not act on $\mathtt{t_k}$ and $C_{k,0}$ is nothing but a swap operator between $\mathtt{m}$ and $\mathtt{t_k}$, and (3) $S^{\mathtt{t_k/m}}$ acts non-trivially on different registers from $B_{k,1}$ and $C_{k,1}$ (see Table 1).

The above is true for $\Gamma \in \{A_i\}_{i\in\{k+1,k+2,\dots,K\}}$ as well, because for each $i \in \{k+1,k+2,\dots,K\}$, A_i is nothing but the swap operator between registers $\mathtt{p_i}$ and $\mathtt{m}$ (see Algo. 6.1).

The only cases left are $\Gamma \in \{(B_i,C_i,D_i)\}_{i\in\{k+1,k+2,\dots,K\}}$. The proof for these cases are (almost) identical. In the following, we only present the proof for $\Gamma = C_i$ ($\forall i \in \{k+1,k+2,\dots,K\}$) as a representative example:

$$\begin{aligned}
\Gamma = C_i
&= |p_1,\dots,p_i\rangle\langle p_1,\dots,p_i|_{\mathtt{p_1\dots p_i}} \otimes C_{i,1} + \sum_{(p'_1,\dots,p'_i)\neq(p_1,\dots,p_i)} |p'_1,\dots,p'_i\rangle\langle p'_1,\dots,p'_i|_{\mathtt{p_1\dots p_i}} \otimes C_{i,0} \quad (9.49)\\
&= |p_1,\dots,p_k\rangle\langle p_1,\dots,p_k|_{\mathtt{p_1\dots p_k}} \otimes |p_{k+1},\dots,p_i\rangle\langle p_{k+1},\dots,p_i|_{\mathtt{p_{k+1}\dots p_i}} \otimes C_{i,1} +\\
&\quad \sum_{(p'_1,\dots,p'_i)\neq(p_1,\dots,p_i)} |p'_1,\dots,p'_k\rangle\langle p'_1,\dots,p'_k|_{\mathtt{p_1\dots p_k}} \otimes |p'_{k+1},\dots,p'_i\rangle\langle p'_{k+1},\dots,p'_i|_{\mathtt{p_{k+1}\dots p_i}} \otimes C_{i,0}\\
&= |p_1,\dots,p_k\rangle\langle p_1,\dots,p_k|_{\mathtt{p_1\dots p_k}} \otimes |p_{k+1},\dots,p_i\rangle\langle p_{k+1},\dots,p_i|_{\mathtt{p_{k+1}\dots p_i}} \otimes C_{i,1} +\\
&\quad |p_1,\dots,p_k\rangle\langle p_1,\dots,p_k|_{\mathtt{p_1\dots p_k}} \otimes \Big(\sum_{(p'_{k+1},\dots,p'_i)\neq(p_{k+1},\dots,p_i)} |p'_{k+1},\dots,p'_i\rangle\langle p'_{k+1},\dots,p'_i|_{\mathtt{p_{k+1}\dots p_i}} \Big) \otimes C_{i,0} +\\
&\quad \sum_{(p'_1,\dots,p'_k)\neq(p_1,\dots,p_k)} |p'_1,\dots,p'_k\rangle\langle p'_1,\dots,p'_k|_{\mathtt{p_1\dots p_k}} \otimes\\
&\qquad \Big(\sum_{(p'_{k+1},\dots,p'_i)\in\{0,1\}^{(i-k)\cdot\ell}} |p'_{k+1},\dots,p'_i\rangle\langle p'_{k+1},\dots,p'_i|_{\mathtt{p_{k+1}\dots p_i}} \Big) \otimes C_{i,0}\\
&= |p_1,\dots,p_k\rangle\langle p_1,\dots,p_k|_{\mathtt{p_1\dots p_k}} \otimes \Gamma_1 + \sum_{(p'_1,\dots,p'_k)\neq(p_1,\dots,p_k)} |p'_1,\dots,p'_k\rangle\langle p'_1,\dots,p'_k|_{\mathtt{p_1\dots p_k}} \otimes \Gamma_0, \quad (9.50)
\end{aligned}$$

where Eq. (9.49) follows from the definition of C_i (see Algo. 7.1), and Eq. (9.50) follows by defining Γ_0 and Γ_1 as follows

$$\begin{aligned}
\Gamma_1 &\coloneqq |p_{k+1},\dots,p_i\rangle\langle p_{k+1},\dots,p_i|_{\mathtt{p_{k+1}\dots p_i}} \otimes C_{i,1} +\\
&\qquad \Big(\sum_{(p'_{k+1},\dots,p'_i)\neq(p_{k+1},\dots,p_i)} |p'_{k+1},\dots,p'_i\rangle\langle p'_{k+1},\dots,p'_i|_{\mathtt{p_{k+1}\dots p_i}} \Big) \otimes C_{i,0}\\
\Gamma_0 &\coloneqq \Big(\sum_{(p'_{k+1},\dots,p'_i)\in\{0,1\}^{(i-k)\cdot\ell}} |p'_{k+1},\dots,p'_i\rangle\langle p'_{k+1},\dots,p'_i|_{\mathtt{p_{k+1}\dots p_i}} \Big) \otimes C_{i,0}.
\end{aligned}$$

Clearly, such a Γ_0 satisfies the requires $\Gamma_0 C_{k,0} = C_{k,0}\Gamma_0^{\mathtt{t_k/m}}$ and $\Gamma_0 C_{k,0} B_{k,1}^\dagger C_{k,1}^\dagger = C_{k,0} B_{k,1}^\dagger C_{k,1}^\dagger \Gamma_0^{\mathtt{t_k/m}}$, because (1) we are in the case $i \in \{k+1,k+2,\dots,K\}$, (2) $C_{i,0}$ is nothing but a swap operator

between $\mathtt{m}$ and $\mathtt{t_i}$, and (3) $C_{i,0}^{\mathtt{t_k}/\mathtt{m}}$ acts non-trivially on different registers from $B_{k,1}$ and $C_{k,1}$ (see Table 1).

This finishes the proof of Claim 16.

□

Finishing the Proof for Case 2. With Claim 16 in hand, we now show how to finish the proof for Case 2.

Proof of Eq. (9.1). First, we establish Eq. (9.1):

$$|\phi^{(t)}\rangle = \Lambda|\phi^{(t-1)}\rangle \tag{9.51}$$

$$= \Lambda\Bigg(|p_1,\ldots,p_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}}|\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p_1}\ldots p_k} C_{k,0}B^\dagger_{k,1}C^\dagger_{k,1}|\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle\Bigg) \tag{9.52}$$

$$= |p_1,\ldots,p_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}}\Lambda_1|\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p_1}\ldots p_k}\Lambda_0 C_{k,0}B^\dagger_{k,1}C^\dagger_{k,1}|\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \tag{9.53}$$

$$= |p_1,\ldots,p_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}}\Lambda_1|\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p_1}\ldots p_k} C_{k,0}B^\dagger_{k,1}C^\dagger_{k,1}\Lambda_0^{\mathtt{t_k}/\mathtt{m}}|\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \tag{9.54}$$

$$= |p_1,\ldots,p_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}}|\rho^{(t)}_{p_1,\ldots,p_k}\rangle + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p_1}\ldots p_k} C_{k,0}B^\dagger_{k,1}C^\dagger_{k,1}|\rho^{(t)}_{p'_1,\ldots,p'_k}\rangle, \tag{9.55}$$

where

- Eq. (9.51) follows from Eq. (9.43);
- Eq. (9.52) follows from our induction assumption;
- Eq. (9.53) follows from Eq. (9.45) in Claim 16;
- Eq. (9.54) follows from Eq. (9.47) in Claim 16;
- Eq. (9.55) follows by defining

$$\begin{cases} |\rho^{(t)}_{p_1,\ldots,p_k}\rangle \coloneqq \Lambda_1|\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle \\ |\rho^{(t)}_{p'_1,\ldots,p'_k}\rangle \coloneqq \Lambda_0^{\mathtt{t_k}/\mathtt{m}}|\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \qquad \forall (p'_1,\ldots,p'_k)\in\big(\{0,1\}^{k\ell}\setminus\{(p_1,\ldots,p_k)\}\big). \end{cases} \tag{9.56}$$

Clearly, Eq. (9.55) is of the same format as Eq. (9.1) in Lem. 19.

Proof of Eq. (9.2). Next, we present the derivation for Eq. (9.2):

$$|\psi^{(t)}\rangle = \Lambda|\psi^{(t-1)}\rangle \tag{9.57}$$

$$= \Lambda\Big(|p_1,\ldots,p_k\rangle_{\mathtt{p_1}\ldots\mathtt{p_k}}|\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle +$$

$$\sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots p_k} C_{k,0}|\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle\Big) \tag{9.58}$$

$$= |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}\Lambda_1|\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle \; + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots p_k}\Lambda_0 C_{k,0}|\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \tag{9.59}$$

$$= |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}\Lambda_1|\rho^{(t-1)}_{p_1,\ldots,p_k}\rangle \; + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots p_k} C_{k,0}\Lambda_0^{\mathtt{t_k/m}}|\rho^{(t-1)}_{p'_1,\ldots,p'_k}\rangle \tag{9.60}$$

$$= |p_1,\ldots,p_k\rangle_{\mathtt{p}_1\ldots\mathtt{p}_k}|\rho^{(t)}_{p_1,\ldots,p_k}\rangle \; + \sum_{(p'_1,\ldots,p'_k)\neq(p_1,\ldots,p_k)} |p'_1,\ldots,p'_k\rangle_{\mathtt{p}_1\ldots p_k} C_{k,0}|\rho^{(t)}_{p'_1,\ldots,p'_k}\rangle, \tag{9.61}$$

where

- Eq. (9.57) follows from Eq. (9.44);
- Eq. (9.58) follows from our induction assumption;
- Eq. (9.59) follows from Eq. (9.45) in Claim 16;
- Eq. (9.60) follows from Eq. (9.46) in Claim 16;
- Eq. (9.61) follows from *the same definitions of* $|\rho^{(t)}_{p_1,\ldots,p_k}\rangle$ and $|\rho^{(t)}_{p'_1,\ldots,p'_k}\rangle$ as shown in Expression (9.56).

Clearly, Eq. (9.61) is of the same format as Eq. (9.2) in Lem. 19.

This completes the proof of the induction step of Lem. 19 for Case 2.

Finally, we remark that our proof for the base case in Sec. 9.1 and the proof for the induction step in Sec. 9.2 together finish the proof of Lem. 19, which in turn finishes the proof of Lem. 13 eventually.

10 On Expected-Polynomial Time Simulator and Efficient Verifier

In this section, we will fully show how the result Thm. 2 also works with expected polynomial-time simulator and how to make the malicious verifier designed in Sec. 6.1.2 efficient.

We first describe the high-level ideas and intuition.

Expected Polynomial-Time Simulator. To show that our lower bound holds for expected polynomial time simulator, we follow the idea in [CCLY21]. The idea is to design a modified random-terminating malicious verifier V_{exp} based on $\widetilde{V}$ designed in Sec. 6.1.2. If there exists any simulator S_{exp} that runs in expected polynomial time and $S_{\mathsf{exp}}^{V_{\mathsf{exp}}}$ outputs an accepting view, then we can turn S_{exp} into a strict-polynomial time simulator S, which would violate our already proven lower bound.

The new verifier V_{exp} is as follows. V_{exp} operates an honest verifier V and a random-terminating verifier $\widetilde{V}$ designed in Sec. 6.1.2 in superposition, by using a control-qubit in an additional register **Cont**:

$$|\widetilde{\psi}\rangle_{\mathbf{Cont},\mathtt{aux}} = \frac{1}{\sqrt{2}}(|0\rangle_{\mathbf{Cont}} + |1\rangle_{\mathbf{Cont}}) \otimes |\psi_\varepsilon\rangle_{\mathtt{aux}},$$

where $|\psi_\varepsilon\rangle_{\mathtt{aux}}$ is defined as:

$$|\psi_\varepsilon\rangle_{\mathtt{aux}} := \sum_{H\in\mathcal{H}_\varepsilon} \sqrt{D(H)}|H\rangle_{\mathtt{aux}},$$

where $H \in \mathcal{H}_\varepsilon$ is the function family defined in Def. 7 and D is the density function corresponding to $\mathcal{H}_\varepsilon$.

The hope is that if we have any expected-polynomial time simulator S_{exp}, we run it up to a fixed q number of steps and simply stop– the final output view of this "truncated" simulator S with V_{exp} is still an accepting one, with inverse polynomial probability. More importantly, we not only want any accepting view, we also want the random-terminating branch of the verifier V_{exp} to be accepting in this case, because otherwise we would not be able to transform the simulator-verifier interaction $S^{V_{\mathsf{exp}}}$ into the BQP decider we designed in Sec. 6.2. Here comes how we use the control-qubit: at the end of the protocol's execution between S and V_{exp}, we show that the state in register **Cont** has a large enough weight on the $|1\rangle$ component. Therefore, if we measure **Cont** in the end, there is a large probability that we fall into the branch where we use the random-terminating $\widetilde{V}$.

A major technical task we deal with is to ensure that the state in **Cont** is a pure state with a specific format at the end of the protocol execution, for the rest of the arguments in [CCLY21] to go through. To make sure that the final state in the **Cont** register is pure, [CCLY21] uses an "adjusting" unitary when the verifier V_{exp} operates on the $|0\rangle$ (i.e. honest verifier branch) at the end of the protocol execution, to disentangle the $\mathtt{aux}$ register and the **Cont** register. Our adjusting unitary has the same design as in [CCLY21], but the analysis on why this procedure helps us achieve the above goal deviates from the analysis of [CCLY21]. In [CCLY21], the proof considers the verifier to take input a fixed classical randomness, which we don't use in the quantum setting. But we make the observation that the use of a fixed classical randomness is not necessary for the proof. We will prove the corresponding lemma in our scenario and connect it with the rest of the proofs.

Efficient Malicious Verifier. Note that the state $|\psi_\varepsilon\rangle_{\mathtt{aux}}$ is exponentially large, but we will use the same method as in [CCLY21] to make our verifier efficient, by modifying a $2q$-independent hash function to simulate the functins in the family $\mathcal{H}_\varepsilon$.

10.1 Expected Polynomial-Time Simulator

We first cite a lemma from [CCLY21]. This lemma helps us make sure that the qubit in register **Cont** remains as a pure state for the rest of the arguments to go through.

To achieve this, we must let the verifier V_{exp} apply an "adjusting" unitary when operating under the $|0\rangle$ (i.e. honest verifier) branch, The following unitary will be used at the end of the honest verifier brancha and performs the following task: for any K-round prover message $\mathbf{p}$, it adjusts the $\mathtt{aux}$ register to have support only on the H's that satisfies $H(\mathbf{p}) = \mathbf{1}$.

Lemma 20 (Lemma 3.8 [CCLY21]). *For any $\mathbf{p} = (p_1, ..., p_k) \in \mathcal{M}^k$, let $S_{\mathbf{p}} \subseteq \mathcal{H}_\varepsilon : \mathcal{M}^{\leq K} \to \{0,1\}$ be the subset consisting of all H such that $H(p_1, ..., p_i) = 1$ for all $i \in [K]$. There exists a unitary $U_{\mathbf{p}}$ such that*

$$U_{\mathbf{p}} \sum_{H\in\mathcal{H}_\varepsilon} \sqrt{D(H)}|H\rangle_{\mathtt{aux}} = \sum_{H\in S_{\mathbf{p}}} \sqrt{\frac{D(H)}{\varepsilon^K}}|H\rangle_{\mathtt{aux}}.$$

The proof is the same as in [CCLY21] and we omit it here.

Next, we describe how the new malicious verifier V_{exp} works.

V_{exp} uses the control-qubit **Cont** to control whether to apply an honest verifier unitary V_{hon} or a random-terminating verifier unitary $\widetilde{V}$ defined in Sec. 6.1.2.

$$\begin{aligned}&V_{\mathsf{exp}}(|0\rangle_{\mathbf{Cont}}|\mathbf{other}_0\rangle_{\mathbf{other}}+|1\rangle_{\mathbf{Cont}}|\mathbf{other}_1\rangle_{\mathbf{other}})\\=&|0\rangle_{\mathbf{Cont}}V_{\mathsf{hon}}|\mathbf{other}_0\rangle_{\mathbf{other}}+|1\rangle_{\mathbf{Cont}}\widetilde{V}|\mathbf{other}_1\rangle_{\mathbf{other}}\end{aligned}$$

where the **other** register refers to all the other registers the verifier will ever act on.

Additionally, the honest unitary V_{hon} applies the unitary $U_{\mathbf{p}}$ from the Lem. 20 to register **Cont** at the end of the protocol. More specifically, V_{hon} operates as follows:

Honest V_{hon}**:** it non-trivially acts on registers $\mathtt{ins}, \mathtt{gc}, \mathtt{lc}, \mathtt{p}_1, \cdots, \mathtt{p}_K, \mathtt{v}_1, \cdots, \mathtt{v}_K, \mathtt{t}_1, \cdots, \mathtt{t}_K, \mathtt{m}, \mathtt{w}, \mathtt{aux}$ defined in Sec. 6.1.1:

1. It reads the value j in $\mathtt{gc}$ and increments it to $i = j + 1 \mod C$ [19]. It swaps $\mathtt{m}$ and $\mathtt{p_i}$.
2. If the value in the global counter $\mathtt{gc}$ is $i < K$, it then applies the honest verifier's unitary at round i to registers $\mathtt{ins}, \mathtt{p}_1, \cdots, \mathtt{p}_K, \mathtt{v}_1, \cdots, \mathtt{v}_K, \mathtt{m}, \mathtt{w}$. Note that register $\mathtt{aux}$ is not used by the honest verifier because it only contains the function used for determining random termination.
3. If the value in the global counter $\mathtt{gc}$ is K , then it first applies the honest verifier's unitary in the corresponding round, then applies the adjusting unitary $U_{\mathbf{p}}$ to $\mathtt{p}_1, \cdots, \mathtt{p}_K$ and $\mathtt{aux}$:

$$|p_1, \cdots, p_K, H\rangle \to U_{\mathbf{p}}|p_1, \cdots, p_K, H\rangle$$

Next, we show that by running the above new malicious verifier V_{exp} with an honest prover P on some instance $x \in L$, the final state in register **Cont** will be negligibly close to a pure state that contains $\frac{1}{\varepsilon^K+1}$ fraction of weight on the $|0\rangle$ component and $\frac{\varepsilon^K}{\varepsilon^K+1}$ fraction of weight on the $|1\rangle$ component.

Lemma 21. *For any $x \in L \cap \{0,1\}^\lambda$ and $w \in R_L(x)$, suppose that we run $\langle P(w), V_{\mathsf{exp}}(|\widetilde{\psi}_\varepsilon\rangle)\rangle(x)$ and measure the final output register and obtain outcome 1. Then the resulting state in* **Cont** *(tracing out other registers) is negligibly close to $|\phi_\varepsilon\rangle_{\mathbf{Cont}} := \sqrt{\frac{1}{1+\varepsilon^K}}|0\rangle_{\mathbf{Cont}} + \sqrt{\frac{\varepsilon^K}{1+\varepsilon^K}}|1\rangle_{\mathbf{Cont}}$.*

Proof. We adapt the proof in [CCLY21] to be of use in our setting. By the completeness of the honestly executed protocol Π, the completeness error is $\mathsf{negl}(\lambda)$.

Let $|\eta\rangle$ be the final state of the internal register of V_{exp} after executing $\langle P(w), V_{\mathsf{exp}}(|\widetilde{\psi}_\varepsilon\rangle)\rangle(x)$. We additionally note that the final output register that stores V_{exp}'s decision bit is called $\mathtt{d}$ (defined in Sec. 5.1.2). We denote Acc as a final measurement on the $\mathtt{d}$ register in the computational basis.

For $\beta \in \{0,1\}$, let $|\eta_\beta\rangle$ be the final state of the internal register of V_{exp} after executing $\langle P(w), V_{\mathsf{exp}}(|\widetilde{\psi}_\varepsilon\rangle)\rangle(x)$. where $|\widetilde{\psi}_\varepsilon^{(\beta)}\rangle_{\mathbf{Cont},\mathtt{aux}} := |\beta\rangle_{\mathbf{Cont}} \otimes |\psi_\varepsilon\rangle_{\mathtt{aux}}$. Since V_{exp} only uses **Cont** as a control register and $|\widetilde{\psi}_\varepsilon\rangle_{\mathbf{Cont},\mathtt{aux}} = \frac{1}{\sqrt{2}}\left(|\widetilde{\psi}_\varepsilon^{(0)}\rangle_{\mathbf{Cont},\mathtt{aux}} + |\widetilde{\psi}_\varepsilon^{(1)}\rangle_{\mathbf{Cont},\mathtt{aux}}\right)$, it is easy to see that we have

$$|\eta\rangle = \frac{1}{\sqrt{2}}\left(|\eta_0\rangle + |\eta_1\rangle\right). \tag{10.1}$$

In the following, when we consider summations over H (they are over all $H \in \mathcal{H}_\varepsilon$, respectively, unless otherwise specified) and prover messages $\mathbf{p} = (p_1, \cdots, p_K)$.

[19] $C = 2^\lambda$ defined in in Sec. 6.1.1.

By the definition of V_{exp} and Lemma 20, we have

$$|\eta_0\rangle = U_{\mathbf{p}}|0\rangle_{\mathbf{Cont}}|x\rangle_{\mathtt{ins}}|K\rangle_{\mathtt{gc}} \otimes \left(\sum_{\mathbf{p}} \alpha_{\mathbf{p}} \left(\sum_{H} \sqrt{D(H)}|H\rangle_{\mathtt{aux}}\right) |\mathbf{p}\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_K}|\mathbf{other}_{\mathbf{p}}\rangle_{\mathtt{other}}|b_{\mathbf{p}}\rangle_{\mathtt{d}}\right)$$
$$= |0\rangle_{\mathbf{Cont}}|x\rangle_{\mathtt{ins}}|K\rangle_{\mathtt{gc}} \otimes \left(\sum_{\mathbf{p}} \alpha_{\mathbf{p}} \left(\sum_{H\in S_{\mathbf{p}}} \sqrt{\frac{D(H)}{\varepsilon^K}}|H\rangle_{\mathtt{aux}}\right) |\mathbf{p}\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_K}|\mathbf{other}_{\mathbf{p}}\rangle_{\mathtt{other}}|b_{\mathbf{p}}\rangle_{\mathtt{d}}\right)$$

where $\mathbf{p}$ is prover's messages and the register $\mathtt{other}$ refers to the concatenation of the registers $\mathtt{w}$, $\mathtt{m}$, $\mathtt{lc}$, $\mathtt{v}_1 \cdots \mathtt{v}_K$, and $\mathtt{t}_1, \cdots, \mathtt{t}_K$. The $|\mathbf{other}_{\mathbf{p}}\rangle_{\mathtt{other}}$ is some sub-normalized state. The verifier's final output decision state $|b_{\mathbf{p}}\rangle$ in register $\mathtt{d}$ is dependent on the contents in register $\mathtt{other}$.

Since V_{hon} is an honest verifier's unitary except applying $U_{\mathbf{p}}$ at the very end: the registers $\mathtt{p}_1 \cdots \mathtt{p}_k$, $\mathtt{other}$, and $\mathtt{d}$ will evolve to exactly what they will be at the end of an honestly executed protocol before $U_{\mathbf{p}}$ is applied. After we apply $U_{\mathbf{p}}$, the only change is the weight on the $\mathtt{aux}$ register for each $\mathbf{p}$, according to Lem. 20. Also note that since P is an honest prover, the register $\mathtt{lc}$ and $\mathtt{gc}$ will always have the same value throughout the protocol and the registers $\mathtt{t}_1, \cdots, \mathtt{t}_K$ are not used. While for each $\mathbf{p}$, the corresponding $\mathtt{other}$ register may hold a mixed state, the analysis assuming $\mathtt{other}$ holding a pure state is without loss of generality.

By the completeness of Π, we have $\Pr[\mathsf{Acc}(|\eta_0\rangle) = 1] = 1 - \mathsf{negl}(\lambda)$. This implies

$$|\eta_0\rangle \approx |0\rangle_{\mathbf{Cont}}|x\rangle_{\mathtt{ins}}|K\rangle_{\mathtt{gc}} \otimes \left(\sum_{\mathbf{p}} \alpha_{\mathbf{p},1} \left(\sum_{H\in S_{\mathbf{p}}} \sqrt{\frac{D(H)}{\varepsilon^k}}|H\rangle_{\mathtt{aux}}\right) |\mathbf{p}\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_k}|\mathbf{other}_{\mathbf{p},1}\rangle_{\mathtt{other}}\right) |1\rangle_{\mathtt{d}} \tag{10.2}$$

where $\approx$ means that the trace distance between both sides is $\mathsf{negl}(\lambda)$. The register $\mathtt{p}_1, \cdots, \mathtt{p}_k, \mathtt{other}$ now contains a state such that the final output $b_{\mathbf{p}} = 1$.

On the other hand, by the definition of V_{exp}, the value in $\mathtt{d}$ of $|\eta_1\rangle$ can be 1 only if $H \in S_{\mathbf{p}}$ for the transcript $\mathbf{p}$. Therefore we have

$$|\eta_1\rangle = |1\rangle_{\mathbf{Cont}}|x\rangle_{\mathtt{ins}}|K\rangle_{\mathtt{gc}} \otimes \begin{pmatrix} \sum_{\mathbf{p}} \alpha_{\mathbf{p},1} \sum_{H\in S_{\mathbf{p}}} \left(\sqrt{D(H)}|H\rangle_{\mathtt{aux}} \otimes |\mathbf{p}\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_k}|\mathbf{other}_{\mathbf{p},1}\rangle_{\mathtt{other}} \otimes |1\rangle_{\mathtt{d}}\right) \\ + |\mathsf{garbage}\rangle_{\mathtt{aux},\mathtt{p}_1,\ldots,\mathtt{p}_k,\mathtt{other}} \otimes |0\rangle_{\mathtt{d}} \end{pmatrix}$$

for some (sub-normalized) state $|\mathsf{garbage}\rangle_{\mathtt{aux},\mathtt{p}_1,\ldots,\mathtt{p}_k,\mathtt{other}} \otimes |0\rangle_{\mathtt{d}}$. By our design of the random-terminating adversary $\widetilde{V}$, conditioned on that $H \in S_{\mathbf{p}}$, $\widetilde{V}$ performs the same operations as the honest verifier does on the registers $\mathtt{p}_1, \cdots, \mathtt{p}_k, \mathtt{other}$. $\widetilde{V}$ can also easily uncompute the registers $\mathtt{t}_1, \cdots, \mathtt{t}_K$ used to store aborting information for each round, which are not used in $|\eta_0\rangle$. Therefore, if we have $b = 1$ in register $\mathtt{d}$, the state over the registers $\mathtt{p}_1, \cdots, \mathtt{p}_k, \mathtt{other}$ is exactly the same as the state over registers $\mathtt{p}_1, \cdots, \mathtt{p}_k, \mathtt{other}$ in $|\eta_0\rangle$ (conditioned on $b = 1$ in $\mathtt{d}$).

By a similar argument to that for $|\eta_0\rangle$, we have

$$|\eta_1\rangle \approx |1\rangle_{\mathbf{Cont}}|x\rangle_{\mathtt{ins}}|K\rangle_{\mathtt{gc}} \otimes \begin{pmatrix} \sum_{\mathbf{p}} \alpha_{\mathbf{p},1} \sum_{H\in S_{\mathbf{p}}} \left(\sqrt{D(H)}|H\rangle_{\mathtt{aux}}\right) \otimes |\mathbf{p}\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_k}|\mathbf{other}_{\mathbf{p},1}\rangle_{\mathtt{other}} \otimes |1\rangle_{\mathtt{d}} \\ + |\mathsf{garbage}\rangle_{\mathtt{aux},\mathtt{p}_1,\ldots,\mathtt{p}_k,\mathtt{other}}|0\rangle_{\mathtt{d}} \end{pmatrix} \tag{10.3}$$

By eq. (10.1) to (10.3), we have

$$(|1\rangle\langle 1|)_{\mathtt{d}}|\eta\rangle \approx \frac{1}{\sqrt{2}} \left(\sqrt{\frac{1}{\varepsilon^K}}|0\rangle_{\mathbf{Cont}} + |1\rangle_{\mathbf{Cont}}\right) \otimes |x\rangle_{\mathtt{ins}}|K\rangle_{\mathtt{gc}}$$

$$\otimes \sum_{\mathbf{p}} \alpha_{\mathbf{p},1} \sum_{H \in S_{\mathbf{p}}} \left(\sqrt{D(H)} |H\rangle_{\mathtt{aux}} \right) |\mathbf{p}\rangle_{\mathtt{p}_1,\ldots,\mathtt{p}_k} |\mathbf{other}_{\mathbf{p},1}\rangle_{\mathtt{other}} \otimes |1\rangle_{\mathtt{d}}.$$

We omit the identity operator on registers other than $\mathtt{d}$ and $(|1\rangle\langle 1|)_{\mathtt{d}}$ simply means the projection onto states whose values in $\mathtt{d}$ is 0.

By normalization, we can see that the final state in **Cont** conditioned on the measurement outcome of $\mathtt{d}$ is 1 is negligibly close to $|\phi_\varepsilon\rangle_{\mathbf{Cont}}$.

□

The rest of the proof follows from [CCLY21] Section 3.2. We describe the ideas here. Suppose that there is a quantum black-box simulator S_{exp} for the protocol Π whose expected number of queries is at most $q/2 = \mathsf{poly}(\lambda)$ that works for all possibly inefficient verifiers, then we can truncate it into a strict polynomial time simulator S that makes only q queries based on the following arguments:

1. The probability that V_{exp} accepts and the number of S_{exp}'s queries is at most q is at least $1/4 - \mathsf{negl}(\lambda)$ (See [CCLY21] Lemma 3.10).
2. The final state in **Cont** after the execution of $S_{\mathsf{exp}}^{V_{\mathsf{exp}}(|\widetilde{\psi}_\varepsilon\rangle)}$ conditioned on the above event is negligibly close in trace distance to $|\phi_\varepsilon\rangle$ defined in Lem. 21 (See [CCLY21] Lemma 3.11).

In the end, we can argue that the truncated simulator S interacting with $V_{\mathsf{exp}}(|\widetilde{\psi}_\varepsilon\rangle)$ will lead to an accepting view with probability $\frac{\varepsilon^k}{4} - \mathsf{negl}(\lambda)$ on any $x \in L \cap \{0,1\}^\lambda$.

We refer the readers to [CCLY21] Section 3.2 for the full proof.

10.2 Expected Polynomial-Time Simulator with Efficient Malicious Verifier

Now we describe how to make the above malicious verifier V_{exp} efficient.

The following lemma states that we can replace the inefficient function (represented by a truth table) $H \in \mathcal{H}_\varepsilon$ used for random-termination in the previous sections by a function with an efficient description. Moreover, there exists an efficient implementation of the unitary in Lem. 20.

Lemma 22 ([CCLY21] Lemma 3.13). *Let $\varepsilon \in [0,1]$ be a rational number expressed as $\varepsilon = \frac{B}{A}$ for some $A, B \in \mathbb{N}$ such that $\log A = \mathsf{poly}(\lambda)$ and $\log B = \mathsf{poly}(\lambda)$.[20] For any $Q = \mathsf{poly}(\lambda)$, there exists a family $\widetilde{\mathcal{H}}_\varepsilon = \{\widetilde{\mathcal{H}}_\kappa : \mathcal{M}^{\le k} \to \{0,1\}\}_{\kappa \in \mathcal{K}}$ of classical polynomial-time computable functions that satisfies the following properties.*

1. *For any algorithm $\mathcal{A}$ that makes at most q quantum queries and any quantum input ρ, we have*

$$\Pr_{H \leftarrow \mathcal{H}_\varepsilon}\left[\mathcal{A}^H(\rho) = 1\right] = \Pr_{\kappa \leftarrow \mathcal{K}}\left[\mathcal{A}^{\widetilde{\mathcal{H}}_\kappa}(\rho) = 1\right].$$

2. *For any $\mathbf{p} = (p_1, \ldots, p_k) \in \mathcal{M}^k$, let $S_{\mathbf{p}} \subseteq \mathcal{K}$ be the subset consisting of all κ such that $\widetilde{H}_\kappa(p_1, \ldots, p_i) = 1$ for all $i \in [k]$. There exists a unitary $U_{\mathbf{p}}^{(q)}$ such that*

$$U_{\mathbf{p}}^{(q)} \sqrt{\frac{1}{|\mathcal{K}|}} \sum_{\kappa \in \mathcal{K}} |\kappa\rangle = \sqrt{\frac{1}{|S_{\mathbf{p}}|}} \sum_{\kappa \in S_{\mathbf{p}}} |\kappa\rangle.$$

$U_{\mathbf{p}}$ can be implemented by a quantum circuit of size $\mathsf{poly}(\lambda)$.

[20] Note that ε is also a function of λ, but we omit to explicitly write the dependence on ε for simplicity.

The proof follows from [CCLY21]. Therefore we present the statement and intuition and refer the readers to [CCLY21] for the full proof.

The idea of is to construct a family of $2q$-wise independent hash function that have output distribution the same as $\mathcal{H}_\varepsilon$, from a family of $2q$-wise independent hash functions that have almost uniform-random output distributions. We first make use of a $2q$-independent hash function H'. We take the key κ to be k values of random $\{a_i\}_{i\in[k]}, a_i \leftarrow [A]$. When computing $\widetilde{H}_\kappa(p_1, \cdots, p_i)$: we add a_i to the outcome of $H'(p_1, \cdots, p_i)$ and check if the sum satisfies $H'(p_1, \cdots, p_i) + a_i \mod A \leq B$; output 1 if yes and 0 otherwise.

References

ACL21. Prabhanjan Ananth, Kai-Min Chung, and Rolando L. La Placa. On the concurrent composition of quantum zero-knowledge. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 346–374, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. 3

Bar01. Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science*, pages 106–115, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. 3

BB84. Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, pages 175–179, 1984. 2

BBBV97. Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997. 32

BCKM21. James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. One-way functions imply secure computation in a quantum world. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 467–496, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. 2

BCY91. Gilles Brassard, Claude Crépeau, and Moti Yung. Constant-round perfect zero-knowledge computationally convincing protocols. *Theoretical Computer Science*, 84(1):23–52, 1991. 1

BG01. Boaz Barak and Oded Goldreich. Universal arguments and their applications. Cryptology ePrint Archive, Report 2001/105, 2001. https://eprint.iacr.org/2001/105. 3

BG20. Anne Broadbent and Alex B. Grilo. QMA-hardness of consistency of local density matrices with applications to quantum zero-knowledge. In *61st Annual Symposium on Foundations of Computer Science*, pages 196–205, Durham, NC, USA, November 16–19, 2020. IEEE Computer Society Press. 2, 3

BGG+90. Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56, Santa Barbara, CA, USA, August 21–25, 1990. Springer, Heidelberg, Germany. 1

BJSW16. Anne Broadbent, Zhengfeng Ji, Fang Song, and John Watrous. Zero-knowledge proof systems for QMA. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 31–40, New Brunswick, NJ, USA, October 9–11, 2016. IEEE Computer Society Press. 2, 3

BJY97. Mihir Bellare, Markus Jakobsson, and Moti Yung. Round-optimal zero-knowledge arguments based on any one-way function. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 280–305, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany. 1

BL02. Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *34th Annual ACM Symposium on Theory of Computing*, pages 484–493, Montréal, Québec, Canada, May 19–21, 2002. ACM Press. 1, 2, 3, 4, 5, 6, 9, 10, 37

BMO90. Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. The (true) complexity of statistical zero knowledge. In *22nd Annual ACM Symposium on Theory of Computing*, pages 494–502, Baltimore, MD, USA, May 14–16, 1990. ACM Press. 1

BP12. Nir Bitansky and Omer Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *53rd Annual Symposium on Foundations of Computer Science*, pages 223–232, New Brunswick, NJ, USA, October 20–23, 2012. IEEE Computer Society Press. 3

BQSY24. John Bostanci, Luowen Qian, Nicholas Spooner, and Henry Yuen. An efficient quantum parallel repetition theorem and applications, 2024. Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024. 2

BS20. Nir Bitansky and Omri Shmueli. Post-quantum zero knowledge in constant rounds. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd Annual ACM Symposium on Theory of Computing*, pages 269–279, Chicago, IL, USA, June 22–26, 2020. ACM Press. 2, 3

CCLY21. Nai-Hui Chia, Kai-Min Chung, Qipeng Liu, and Takashi Yamakawa. On the impossibility of post-quantum black-box zero-knowledge in constant round. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 59–67. IEEE, 2021. ii, 2, 3, 4, 6, 7, 9, 10, 18, 19, 20, 29, 37, 44, 45, 100, 101, 102, 104, 105

CCLY22. Nai-Hui Chia, Kai-Min Chung, Xiao Liang, and Takashi Yamakawa. Post-quantum simulatable extraction with minimal assumptions: Black-box and constant-round. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 533–563, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. 2, 3

CCY21. Nai-Hui Chia, Kai-Min Chung, and Takashi Yamakawa. A black-box approach to post-quantum zero-knowledge in constant rounds. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 315–345, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. 2, 3

CM21. Orestis Chardouvelis and Giulio Malavolta. The round complexity of quantum zero-knowledge. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part I*, volume 13042 of *Lecture Notes in Computer Science*, pages 121–148, Raleigh, NC, USA, November 8–11, 2021. Springer, Heidelberg, Germany. 2

CPS13. Kai-Min Chung, Rafael Pass, and Karn Seth. Non-black-box simulation from one-way functions and applications to resettable security. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 231–240, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 3

DFM20. Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 602–631, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. 6, 7, 19, 20

DFMS19. Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 356–383, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 6, 19

DGJ+20. Yfke Dulek, Alex B. Grilo, Stacey Jeffery, Christian Majenz, and Christian Schaffner. Secure multi-party quantum computation with a dishonest majority. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 729–758, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. 3

DNS12. Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. Actively secure two-party evaluation of any quantum operation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 794–811, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. 3

FGJ18. Nils Fleischhacker, Vipul Goyal, and Abhishek Jain. On the existence of three round zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 3–33, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. 3

FS90. Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. 1

GIK+23. Riddhi Ghosal, Yuval Ishai, Alexis Korb, Eyal Kushilevitz, Paul Lou, and Amit Sahai. Hard languages in NP ∩ conp and NIZK proofs from unstructured hardness. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1243–1256. ACM, 2023. 1

GK96a. Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, June 1996. 1

GK96b. Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996. 1

GLSV21. Alex B. Grilo, Huijia Lin, Fang Song, and Vinod Vaikuntanathan. Oblivious transfer is in MiniQCrypt. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 531–561, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany. 2, 3

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, RI, USA, May 6–8, 1985. ACM Press. 1

GMW86. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 174–187, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. 1

GOSV14. Vipul Goyal, Rafail Ostrovsky, Alessandra Scafuro, and Ivan Visconti. Black-box non-black-box zero knowledge. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 515–524, New York, NY, USA, May 31 – June 3, 2014. ACM Press. 3

Goy13. Vipul Goyal. Non-black-box simulation in the fully concurrent setting. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 221–230, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 3

GSY19. Alex Bredariol Grilo, William Slofstra, and Henry Yuen. Perfect zero knowledge for quantum multiprover interactive proofs. In David Zuckerman, editor, *60th Annual Symposium on Foundations of Computer Science*, pages 611–635, Baltimore, MD, USA, November 9–12, 2019. IEEE Computer Society Press. 1

HPV20. Carmit Hazay, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Which languages have 4-round fully black-box zero-knowledge arguments from one-way functions? In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 599–619, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. 3

HRS16. Andreas Hülsing, Joost Rijneveld, and Fang Song. Mitigating multi-target attacks in hash-based signatures. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 387–416, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany. 19

HSS11. Sean Hallgren, Adam Smith, and Fang Song. Classical cryptographic protocols in a quantum world. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 411–428, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany. 30

IY88. Russell Impagliazzo and Moti Yung. Direct minimum-knowledge computations. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 40–51, Santa Barbara, CA, USA, August 16–20, 1988. Springer, Heidelberg, Germany. 1

JKMR09. Rahul Jain, Alexandra Kolla, Gatis Midrijanis, and Ben W Reichardt. On parallel composition of zero-knowledge proofs with black-box quantum simulators. *Quantum Information & Computation*, 9(5):513–532, 2009. 2

Kat08. Jonathan Katz. Which languages have 4-round zero-knowledge proofs? In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 73–88, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany. 3

KKMV09. Julia Kempe, Hirotada Kobayashi, Keiji Matsumoto, and Thomas Vidick. Using entanglement in quantum multi-prover interactive proofs. *Comput. Complex.*, 18(2):273–307, 2009. 2

KRR17. Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of Fiat-Shamir for proofs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 224–251, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 3

KW00. Alexei Kitaev and John Watrous. Parallelization, amplification, and exponential time simulation of quantum interactive proof systems. In *32nd Annual ACM Symposium on Theory of Computing*, pages 608–617, Portland, OR, USA, May 21–23, 2000. ACM Press. 2

Lin03. Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *35th Annual ACM Symposium on Theory of Computing*, pages 683–692, San Diego, CA, USA, June 9–11, 2003. ACM Press. 3

LMS21. Alex Lombardi, Fermi Ma, and Nicholas Spooner. Post-quantum zero knowledge, revisited (or: How to do quantum rewinding undetectably). https://arxiv.org/abs/2111.12257, 2021. 4

LMS22. Alex Lombardi, Fermi Ma, and Nicholas Spooner. Post-quantum zero knowledge, revisited or: How to do quantum rewinding undetectably. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 851–859. IEEE, 2022. 3

Pas04. Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 232–241, Chicago, IL, USA, June 13–16, 2004. ACM Press. 3

PPS15. Omkant Pandey, Manoj Prabhakaran, and Amit Sahai. Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for NP. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 638–667, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. 3

RTV04. Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 3

SV97a. Amit Sahai and Salil P. Vadhan. A complete promise problem for statistical zero-knowledge. In *38th Annual Symposium on Foundations of Computer Science*, pages 448–457, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press. 1

SV97b. Amit Sahai and Salil P. Vadhan. Manipulating statistical difference. In Panos M. Pardalos, Sanguthevar Rajasekaran, and José Rolim, editors, *Randomization Methods in Algorithm Design, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, December 12-14, 1997*, volume 43 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 251–270. DIMACS/AMS, 1997. 1

Yan22. Jun Yan. General properties of quantum bit commitments (extended abstract). In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 628–657. Springer, 2022. 2

YZ21. Takashi Yamakawa and Mark Zhandry. Classical vs quantum random oracles. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 568–597, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany. 6

Zha12. Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 758–775, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. 19