MIP* = RE

Zhengfeng Ji[*1], Anand Natarajan[†2,3], Thomas Vidick[‡3], John Wright[§2,3,4], and Henry Yuen[¶5]

[1]*Centre for Quantum Software and Information, University of Technology Sydney*
[2]*Institute for Quantum Information and Matter, California Institute of Technology*
[3]*Department of Computing and Mathematical Sciences, California Institute of Technology*
[4]*Department of Computer Science, University of Texas at Austin*
[5]*Department of Computer Science and Department of Mathematics, University of Toronto*

July 17, 2020

Abstract

We show that the class MIP* of languages that can be decided by a classical verifier interacting with multiple all-powerful quantum provers sharing entanglement is equal to the class RE of recursively enumerable languages. Our proof builds upon the quantum low-degree test of (Natarajan and Vidick, FOCS 2018) by integrating recent developments from (Natarajan and Wright, FOCS 2019) and combining them with the recursive compression framework of (Fitzsimons et al., STOC 2019).

An immediate byproduct of our result is that there is an efficient reduction from the Halting Problem to the problem of deciding whether a two-player nonlocal game has entangled value 1 or at most $\frac{1}{2}$. Using a known connection, undecidability of the entangled value implies a negative answer to Tsirelson's problem: we show, by providing an explicit example, that the closure C_{qa} of the set of quantum tensor product correlations is strictly included in the set C_{qc} of quantum commuting correlations. Following work of (Fritz, Rev. Math. Phys. 2012) and (Junge et al., J. Math. Phys. 2011) our results provide a refutation of Connes' embedding conjecture from the theory of von Neumann algebras.

---

[*]Email: zhengfeng.ji@uts.edu.au
[†]Email: anandn@caltech.edu
[‡]Email: vidick@caltech.edu
[§]Email: wright@cs.utexas.edu
[¶]Email: hyuen@cs.toronto.edu

Contents

1 Introduction

For integer $n, k \geq 2$ define the quantum (spatial) correlation set $C_{qs}(n,k)$ as the subset of $\mathbb{R}^{n^2 k^2}$ that contains all tuples (p_{abxy}) representing families of bipartite distributions that can be locally generated in non-relativistic quantum mechanics. Formally, $(p_{abxy}) \in C_{qs}(n,k)$ if and only if there exist separable Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$, for every $x \in \{1,\ldots,n\}$ (resp. $y \in \{1,\ldots,n\}$), a collection of projections $\{A_a^x\}_{a \in \{1,\ldots,k\}}$ on $\mathcal{H}_A$ (resp. $\{B_b^y\}_{b \in \{1,\ldots,k\}}$ on $\mathcal{H}_B$) that sum to identity, and a state (unit vector) $\psi \in \mathcal{H}_A \otimes \mathcal{H}_B$ such that

$$\forall x,y \in \{1,2,\ldots,n\}, \quad \forall a,b \in \{1,2,\ldots,k\}, \qquad p_{abxy} = \psi^* (A_a^x \otimes B_b^y) \psi. \tag{1}$$

Note that due to the normalization conditions on ψ and on $\{A_a^x\}$ and $\{B_b^y\}$, for each x, y, (p_{abxy}) is a probability distribution on $\{1,2,\ldots,k\}^2$. By taking direct sums it is easy to see that the set $C_{qs}(n,k)$ is convex. Let $C_{qa}(n,k)$ denote its closure (it is known that $C_{qs}(n,k) \neq C_{qa}(n,k)$, see [Slo19a]).

Our main result is that the family of sets $\{C_{qa}(n,k)\}_{n,k \in \mathbb{N}}$ is extraordinarily complex, in the following computational sense. For any $0 < \varepsilon < 1$ define the *ε-weak membership problem* for C_{qa} as the problem of deciding, given $n, k \in \mathbb{N}$ and a point $p = (p_{abxy}) \in \mathbb{R}^{n^2 k^2}$, whether p lies in $C_{qa}(n,k)$ or is ε-far from it in ℓ_1 distance, promised that one is the case. Then we show that for any given $0 < \varepsilon < 1$ the ε-weak membership problem for C_{qa} cannot be solved by a Turing machine that halts with the correct answer on every input.

We show this by directly reducing the Halting problem to the weak membership problem for C_{qa}: we show that for all $0 < \varepsilon < 1$ and any Turing machine $\mathcal{M}$ one can efficiently compute integers $n, k \in \mathbb{N}$ and a linear functional $\ell_{\mathcal{M}}$ on $\mathbb{R}^{n^2 k^2}$ such that, whenever $\mathcal{M}$ halts it holds that

$$\sup_{p \in C_{qa}(n,k)} |\ell_{\mathcal{M}}(p)| = 1, \tag{2}$$

whereas if $\mathcal{M}$ does not halt then

$$\sup_{p \in C_{qa}(n,k)} |\ell_{\mathcal{M}}(p)| \leq 1 - \varepsilon. \tag{3}$$

By standard results in convex optimization, this implies the aforementioned claim on the undecidability of the ε-weak membership problem for C_{qa} (for any $0 < \varepsilon < 1$).

Our result has interesting consequences for long-standing conjectures in quantum information theory and the theory of von Neumann algebras. Through a connection that follows from the work of Navascues, Pironio, and Acin [NPA08] the undecidability result implies a negative answer to Tsirelson's problem [Tsi06]. Let $C_{qc}(n,k)$ denote the set of quantum commuting correlations, which is the set of tuples (p_{abxy}) arising from operators $\{A_a^x\}$ and $\{B_b^y\}$ acting on a single Hilbert space $\mathcal{H}$ and a state $\psi \in \mathcal{H}$ such that

$$\forall x,y \in \{1,\ldots,n\}, \forall a,b \in \{1,\ldots,k\}, \qquad p_{abxy} = \psi^* (A_a^x B_b^y) \psi \quad \text{and} \quad [A_a^x, B_b^y] = 0. \tag{4}$$

Then *Tsirelson's problem* asks if, for all n, k, the sets $C_{qa}(n,k)$ and $C_{qc}(n,k)$ are equal. Using results from [NPA08] we give integer n, k and an explicit linear function ℓ on $\mathbb{R}^{n^2 k^2}$ such that

$$\sup_{p \in C_{qc}(n,k)} |\ell(p)| = 1, \qquad \text{but} \qquad \sup_{p \in C_{qa}(n,k)} |\ell(p)| \leq \frac{1}{2},$$

which implies that $C_{qa}(n,k) \neq C_{qc}(n,k)$. By an implication of Fritz [Fri12] and Junge et al. [JNP$^+$11] we further obtain that Connes' Embedding Conjecture [Con76] is false; in other words, there exist type II_1

von Neumann factors that do not embed in an ultrapower of the hyperfinite II_1 factor. We explain these connections in more detail in Section 1.3 below.

Our approach to constructing such linear functionals on correlation sets goes through the theory of interactive proofs from complexity theory. To explain this connection we first review the concept of interactive proofs. The reader familiar with interactive proofs may skip the next section to arrive directly at a formal statement of our main complexity-theoretic result in Section 1.2.

1.1 Interactive proof systems

An *interactive proof system* is an abstraction that generalizes the familiar notion of *proof*. Intuitively, given a formal statement z (for example, "this graph admits a proper 3-coloring"), a proof π for z is information that enables one to check the validity of z more efficiently than without access to the proof (in this example, π could be an explicit assignment of colors to each vertex of the graph).

Complexity theory formalizes the notion of proof in a way that emphasizes the role played by the verification procedure. To explain this, first recall that in complexity theory a *language L* is a subset of $\{0,1\}^*$, the set of all bit strings of any length, that intuitively represents all problem instances to which the answer should be "yes". For example, the language $L = 3\text{-COLORING}$ contains all strings z such that z is the description (according to some pre-specified encoding scheme) of a 3-colorable graph G. We say that a language L admits efficiently verifiable proofs if there exists an algorithm V (formally, a polynomial-time Turing machine) that satisfies the following two properties: (i) for any $z \in L$ there is a string π such that $V(z, \pi)$ returns 1 (we say that V "accepts"), and (ii) for any $z \notin L$ there is no string π such that $V(z, \pi)$ accepts. Property (i) is generally referred to as the *completeness* property, and (ii) is the *soundness*. The set of all languages L with both these completeness and soundness properties is denoted by the complexity class NP.

Research in complexity and cryptography in the 1980s and 1990s led to a significant generalization of the notion of "efficiently verifiable proof". The first modification is to allow *randomized* verification procedures by relaxing (i) and (ii) to *high probability* statements: every $z \in L$ should have a proof π that is accepted *with probability at least c* (the completeness parameter), and for no $z \notin L$ should there be a proof π that is accepted *with probability larger than s* (the soundness parameter). A common setting is to take $c = \frac{2}{3}$ and $s = \frac{1}{3}$; standard amplification techniques reveal that the exact values do not significantly affect the class of languages that admit such proofs, provided that they are chosen within reasonable bounds.

The second modification is to allow *interactive* verification. Informally, this means that instead of receiving a proof string π in its entirety and making a decision based on it, the verification algorithm (called the "verifier") instead communicates with another algorithm called a "prover", and based on the communication decides whether $z \in L$. There are no restrictions on the computational power of the prover, whereas the verifier is required to run in polynomial time.[1]

To understand how randomization and interaction can help for proof checking, consider the following example of an interactive proof for the language GRAPH NON-ISOMORPHISM, which contains all pairs of graphs (G_0, G_1) such that G_0 and G_1 are *not* isomorphic.[2] It is not known if GRAPH NON-ISOMORPHISM $\in$ NP, because it is not clear how to give an efficiently verifiable proof string that two graphs G_0 and G_1 are

---

[1] The reader may find the following mental model useful: in an interactive proof, an all-powerful prover is trying to convince a skeptical, but computationally limited, verifier that a string z (known to both) lies in the set L, even when it may be that in fact $z \notin L$. By interactively interrogating the prover, the verifier can reject false claims, i.e. determine with high statistical confidence whether $z \in L$ or not. Importantly, the verifier is allowed to probabilistically and adaptively choose its messages to the prover.

[2] Here and in the rest of the section, we implicitly assume that graphs and tuples of graphs have a canonical encoding as binary strings.

5

not isomorphic. (A proof of isomorphism is, of course, trivial: given a bijection from the vertices of G_0 to those of G_1 it is straightforward to verify that the bijection induces an isomorphism.) However, consider the following *randomized*, *interactive* verification procedure. Suppose the input to the verifier and prover is a pair of n-vertex graphs (G_0, G_1) (if the graphs do not have the same number of vertices, they are trivially non-isomorphic and the verification procedure can automatically accept). The verifier first selects a uniformly random $b \in \{0, 1\}$ and a uniformly random permutation σ of $\{1, \ldots, n\}$ and sends the graph $H = \sigma(G_b)$ to the prover. The prover is then supposed to respond with a bit $b' \in \{0, 1\}$; if $b' = b$ the verifier accepts and if $b' \neq b$ it rejects.

Clearly, if G_0 and G_1 are not isomorphic then there exists a prover strategy to compute b from H with probability 1: using its unlimited computational power, the prover can determine whether H is isomorphic to G_0 or to G_1. However, if G_0 and G_1 *are* isomorphic then the distribution of H is uniform over the isomorphism class of G_0, which is the same as the isomorphism class of G_1, and the prover (despite having unlimited computational power) cannot distinguish between whether the verifier generated H using G_0 or G_1. Thus the probability that *any* prover can correctly guess $b' = b$ is exactly $\frac{1}{2}$. As a result, we have shown that the graph non-isomorphism problem has an interactive proof system with completeness $c = 1$ and soundness $s = \frac{1}{2}$. Note how little "information" is communicated by the prover: a single bit! The extreme succinctness of the "proof" comes from the fact that whether G_0 is isomorphic to G_1 determines whether a prover can reliably compute, given the data available to it (which is G_0, G_1, and H), the correct bit b.

We denote by IP the class of languages that admit randomized interactive proof systems such as the one just described. The class IP is easily seen to contain NP, but it is thought to be a much larger class: one of the famous results of complexity theory is that IP is exactly the same as PSPACE [LFKN90, Sha90], the class of languages decidable by Turing machines using polynomial space.[3] Thus a polynomial-time verifier, when augmented with the ability to interrogate an all-powerful prover and use randomization, can solve computational problems that are (believed to be) vastly more difficult than those that can be checked using static, deterministic proofs (i.e. NP problems).

Multiprover interactive proofs. We now discuss a generalization of interactive proofs called *multiprover interactive proofs*. Here, a polynomial-time verifier can interact with *two* (or more) provers to decide whether a given instance z is in a language L or not. In this setting, after the verifier and all the provers receive the common input z, the provers are not allowed to communicate with each other, and the verifier "cross-interrogates" the provers in order to decide if $z \in L$. The provers may coordinate a joint strategy ahead of time, but once the protocol begins the provers can only interact with the verifier. As we will see, the extra condition that the provers cannot communicate with each other is a powerful constraint that can be leveraged by the verifier.

Consider the computational problem of deciding membership in a *promise* language called GAP-MAXCUT. A promise language L is specified by two disjoint subsets $L_{yes}, L_{no} \subseteq \{0, 1\}^*$, and the task is to decide whether a given instance z is in L_{yes} or L_{no}, promised that $z \in L_{yes} \cup L_{no}$. In a proof system for a promise language, the completeness case consists of accepting with probability at least c if $z \in L_{yes}$, and the soundness case consists of accepting with probability at most s if $z \in L_{no}$. If $z \notin L_{yes} \cup L_{no}$, then there are no constraints on the behavior of the verifier.

The promise language GAP-MAXCUT is defined as follows: GAP-MAXCUT$_{yes}$ (resp. GAP-MAXCUT$_{no}$)

---

[3]The reason PSPACE is considered a "difficult" class of problems is because many computational problems believed to require super-polynomial or exponential time (such as 3-COLORING or deciding whether a quantified Boolean formula is true) can be solved using a polynomial amount of space.

is the set of all graphs G with a *cut* (i.e. a bipartition of the vertices) such that at least 90% of edges cross the cut (resp. at most 60% of edges cross the cut).[4] For simplicity, we also assume that all graphs in GAP-MAXCUT$_{yes}$ $\cup$ GAP-MAXCUT$_{no}$ are regular, i.e. the degree is a constant across all vertices in the graph.

The GAP-MAXCUT problem clearly lies in NP, since given a candidate cut it is easy to count the number of edges that cross it and verify that it is at least 90% of the total number of edges. Observe that the length of the proof and the time required to verify it are linear in the size of the graph (the number of vertices and edges). *Finding* the proof is of course much harder, but we are only concerned with the complexity of the verification procedure.

Now consider the following simple *two-prover interactive proof system* for GAP-MAXCUT. Given a graph G, the verification procedure first samples a uniformly random edge $e = \{u, v\}$ in G. It then sends a uniformly random $x \in \{u, v\}$ to the first prover, and a uniformly random $y \in \{u, v\}$ to the second prover. Each prover sees its respective question only and is expected to respond with a single bit, $a, b \in \{0, 1\}$ respectively. The verification procedure accepts if and only if $a = b$ if $x = y$, and $a \neq b$ if $x \neq y$.

We claim that the verification procedure described in the preceding paragraph is a *multiprover interactive proof system* for the language GAP-MAXCUT, with completeness $c = 0.95$ and soundness $s = 0.9$, in the following sense. First, whenever $G \in$ GAP-MAXCUT$_{yes}$ then there is a successful strategy for the provers: specifically, the provers can fix an optimal bipartition and consistently answer "0" when asked about a vertex from one side of the partition, and "1" when asked about a vertex from the other side; assuming there exists a cut that is crossed by at least 90% of the edges, this strategy succeeds with probability at least $\frac{1}{2} + \frac{1}{2}0.9$, where the first factor $\frac{1}{2}$ arises from the case when both provers are sent the same vertex, in which case they always succeed.

Conversely, suppose given a strategy for the provers that is accepted with probability $p = \frac{1}{2} + \frac{1}{2}(1 - \delta)$ when the verification procedure is executed on a (regular) n-vertex graph G. We then claim that G has a cut crossed by at least a $1 - 2\delta$ fraction of all edges. To show this, we leverage the non-communication assumption on the provers. Since either prover's question is always a single vertex, their strategy can be represented by a function from the vertices of G to answers in $\{0, 1\}$. Any such function specifies a bipartition of G. While the provers' bipartitions need not be identical, the fact that they succeed with high probability, for the case when they are sent the same vertex, implies that they must be consistent with high probability. Finally, the fact that they also succeed with high probability when sent opposite endpoints of a randomly chosen edge implies that either prover's bipartition must be cut by a large number of edges. Taking the contrapositive establishes the soundness property.

We denote by MIP the class of languages that have multiprover interactive proof systems such as the one described in the preceding paragraph. Note that, in comparison to the NP verification procedure for GAP-MAXCUT considered earlier, the interactive, two-prover verification is much more efficient in terms of the effort required for the verifier. Assuming the graph is provided in a convenient format,[5] it is possible to sample a random edge and verify the provers' answers in time and space that scales *logarithmically* with the size of the graph. This exponential improvement in the efficiency of the verification procedure serves as the starting point for another celebrated result from complexity theory: MIP is exactly the same as the class NEXP [BFL91], which are problems that admit *exponential-time checkable proofs*.[6] The class NEXP

---

[4]The specific numbers 90% and 60% are not too important; the only thing that really matters is that the first one is strictly less than 100% and the second strictly larger than 50%, as otherwise the problem becomes much easier.

[5]For example, the graph can be specified via a circuit that takes as input an edge index — using some arbitrary ordering — and returns labels for the two endpoints of the edge.

[6]An example of such a problem is the language SUCCINCT-3-COLORING, which contains descriptions of polynomial-size circuits C that specify a 3-colorable graph G_C on *exponentially many* vertices.

contains PSPACE, but is believed to be much larger; this suggests that the ability to interrogate more than one prover enables a polynomial-time verifier to verify much more complex statements.

Nonlocal games. In this paper we will only be concerned with multiprover interactive proof systems that consist of a single round of communication with two provers: the verifier first sends its questions to each of the provers, the provers respond with their answers, and the verifier decides whether to accept or reject. The class of problems that admit such interactive proofs is denoted $\mathsf{MIP}(2,1)$, and it is known that $\mathsf{MIP} = \mathsf{MIP}(2,1)$ [FL92]. Such proof systems have a convenient reformulation using the language of *nonlocal games*, that we now explain.

In a nonlocal game, we say that a verifier interacts with multiple non-communicating *players* (instead of provers — there is no formal difference between the two terms). An n-question, k-answer nonlocal game $\mathfrak{G}$ is specified by two procedures: a *question sampling* procedure that samples a pair of questions $(x, y) \in \{1, \ldots, n\}^2$ for the players according to a distribution μ (known to the verifier and the players), and a *decision* procedure that takes as input the players' questions and their respective answers $a, b \in \{1, \ldots, k\}$ and evaluates a predicate $D(x, y, a, b) \in \{0, 1\}$ to determine the verifier's acceptance or rejection. In classical complexity theory, the main quantity associated with a nonlocal game $\mathfrak{G}$ is its *classical value*, which is the maximum success probability that two cooperating but non-communicating players have in the game. Formally, the classical value is defined as

$$\mathrm{val}(\mathfrak{G}) = \sup_{p \in C_c(n,k)} \sum_{x,y} \mu(x,y) \sum_{a,b} D(x,y,a,b) p_{abxy} \,, \tag{5}$$

where the set $C_c(n, k)$ is the set of *classical correlations*, which are tuples (p_{abxy}) such that there exists a set Λ with probability measure ν and for every $\lambda \in \Lambda$ functions $A^\lambda, B^\lambda : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, k\}$ such that

$$\forall x, y \in \{1, 2, \ldots, n\}, \quad \forall a, b \in \{1, 2, \ldots, k\}, \quad p_{abxy} = \Pr_{\lambda \sim \nu} \left(A^\lambda(x) = a \wedge B^\lambda(y) = b \right).$$

This definition captures the intuitive notion that a classical strategy for the players is specified by (i) a distribution ν on Λ that represents some probabilistic information shared by the players that is independent of the verifier's questions, and (ii) two functions A^λ, B^λ that represent each players' "local strategy" for answering given their shared randomness λ and question x or y. [7] Note that due to the shared randomness λ, the set $C_c(n, k)$ is a (closed) convex subset of $[0, 1]^{n^2 k^2}$.

To make the connection with interactive proof systems, observe that the assertion that $L \in \mathsf{MIP}(2, 1)$ precisely amounts to the specification of an efficient mapping[8] from problem instances z to games $\mathfrak{G}_z$ such that whenever $z \in L$ then $\mathrm{val}(\mathfrak{G}_z) \geq \frac{2}{3}$, whereas if $z \notin L$ then $\mathrm{val}(\mathfrak{G}_z) \leq \frac{1}{3}$. Thus the complexity of the optimization problem (5) captures the complexity of the decision problem L. The aforementioned characterization of MIP as the class NEXP by [BFL91] shows that in general this optimization problem is very difficult: it is as hard as deciding any language in NEXP.

---

[7]For the functional analyst we briefly note that if we define a tensor

$$L = \sum_{x,y,a,b} \mu(x,y) D(x,y,a,b) e_{xa} \otimes e_{yb} \in \mathbb{R}^{nk} \otimes \mathbb{R}^{nk}$$

then $\mathrm{val}(\mathfrak{G}) = \|L\|_{\ell_1^n(\ell_\infty)^k \otimes_\epsilon \ell_1^n(\ell_\infty)^k}$, with $\otimes_\epsilon$ denoting the injective tensor norm of Banach spaces. (For more connections between interactive proofs, nonlocal games and tensor norms we refer to the survey [PV16].)

[8]Here by "efficient" we mean that there should be a polynomial-time Turing machine that on input z returns (i) a polynomial-size randomized circuit that samples from μ, and (ii) a polynomial-size circuit that evaluates the predicate D.

1.2 Statement of result

We now introduce the main complexity class that is the focus of this paper: MIP^*, the "entangled-prover" analogue of the class MIP considered earlier. Informally the class MIP^*, first introduced in [CHTW04], contains all languages that can be decided by a classical polynomial-time verifier interacting with multiple *quantum* provers sharing *entanglement*. We focus on the class $\mathsf{MIP}^*(2,1)$, which corresponds to the setting of one-round protocols with two provers. Equivalently, a language L is in $\mathsf{MIP}^*(2,1)$ if and only if there is an efficient mapping from instances $z \in \{0,1\}^*$ to nonlocal games $\mathfrak{G}_z$ such that if $z \in L$, then $\mathrm{val}^*(\mathfrak{G}_z) \geq 2/3$ and otherwise $\mathrm{val}^*(\mathfrak{G}_z) \leq 1/3$. Here, for an n-question, k-answer game $\mathfrak{G}$, we let $\mathrm{val}^*(\mathfrak{G})$ denote its *entangled value*, which is defined as

$$\mathrm{val}^*(\mathfrak{G}) = \sup_{p \in C_{qs}(n,k)} \sum_{x,y} \mu(x,y) \sum_{a,b} D(x,y,a,b) p_{abxy} \,, \tag{6}$$

where the set $C_{qs}(n,k)$ is the quantum spatial correlation set introduced in (1). In other words, the entangled value is the supremum of the success probabilities achieved by players that use quantum spatial strategies (i.e., perform local measurements on a shared entangled state). Note that (6) can be equivalently defined as taking the supremum over the set $C_{qa}(n,k)$, the closure of $C_{qs}(n,k)$.

Since $C_c(n,k) \subseteq C_{qs}(n,k)$, we have that $\mathrm{val}(\mathfrak{G}) \leq \mathrm{val}^*(\mathfrak{G})$; in other words, using quantum spatial strategies can do at least as well as classical strategies in a nonlocal game.

The consideration of quantum strategies and the set $C_{qs}(n,k)$ for the definition of MIP^* is motivated by a long line of works in the foundations of quantum mechanics around the topic of *Bell inequalities*, that are linear functionals which separate the sets $C_c(n,k)$ and $C_{qs}(n,k)$. The simplest such functional is the *CHSH inequality* [CHSH69], that shows $C_c(2,2) \subsetneq C_{qs}(2,2)$. The CHSH inequality can be reformulated as a game $\mathfrak{G}$ such that $\mathrm{val}^*(\mathfrak{G}) > \mathrm{val}(\mathfrak{G})$. This game is very simple: it is defined by setting $\mu(x,y) = \frac{1}{4}$ for all $x,y \in \{0,1\}$ and $D(x,y,a,b) = 1$ if and only if $a \oplus b = x \wedge y$. It can be shown that $\mathrm{val}(\mathfrak{G}) = \frac{3}{4}$ and $\mathrm{val}^*(\mathfrak{G}) = \frac{1}{2} + \frac{1}{2\sqrt{2}} > \frac{3}{4}$. The study of Bell inequalities is a large area of research not only in foundations, where they are a tool to study the nonlocal properties of entanglement, but also in quantum cryptography, where they form the basis for cryptographic protocols for e.g. quantum key distribution [Eke91].

The introduction of entanglement in the setting of interactive proofs has interesting consequences for complexity theory; indeed it is not *a priori* clear how the class MIP^* compares to MIP. Take a language $L \in \mathsf{MIP}(2,1)$, and let z be an instance. Then the associated game $\mathfrak{G}_z$ is such that $\mathrm{val}(\mathfrak{G}_z) \geq \frac{2}{3}$ if $z \in L$, and $\mathrm{val}(\mathfrak{G}_z) \leq \frac{1}{3}$ otherwise. The fact that in general $\mathrm{val}^*(\mathfrak{G}_z) \geq \mathrm{val}(\mathfrak{G}_z)$ (and that as demonstrated by the CHSH game inequality can be strict) cuts both ways. On the one hand, the soundness property can be affected, so that instances $z \notin L$ could have $\mathrm{val}^*(\mathfrak{G}_z) = 1$, meaning that we would not be able to establish that $L \in \mathsf{MIP}^*$. On the other hand, a language $L \in \mathsf{MIP}^*(2,1)$ may not necessarily be in MIP, because for $z \in L$ the fact that $\mathrm{val}^*(\mathfrak{G}_z) \geq \frac{2}{3}$ does not automatically imply $\mathrm{val}(\mathfrak{G}_z) > \frac{1}{3}$ (in other words, the game $\mathfrak{G}_z$ may require the players to use a quantum strategy in order to win with probability greater than $1/3$). Just as the complexity of the class MIP is characterized by the complexity of approximating the classical value of nonlocal games (the optimization problem in (5)), the complexity of MIP^* is intimately related to the complexity of approximating the entangled value of games (the optimization problem in (6)).

In [IV12] the first non-trivial lower bound on MIP^* was shown, establishing that $\mathsf{MIP} = \mathsf{NEXP} \subseteq \mathsf{MIP}^*$. (Earlier results [KKM$^+$11, IKM09] gave more limited hardness results, for approximating the entangled value up to inverse polynomial precision.) This was proved by arguing that for the specific games constructed by [BFL91] that show $\mathsf{NEXP} \subseteq \mathsf{MIP}$, the classical and entangled values are approximately the same. In other words, the classical soundness and completeness properties of the proof system of [BFL91]

are maintained in the presence of shared entanglement between the provers. Following [IV12] a sequence of works [Vid16, Ji16, NV18b, Ji17, NV18a, FJVY19] established progressively stronger lower bounds on the complexity of approximating the entangled value of nonlocal games, culminating in [NW19] which showed that approximating the entangled value is at least as hard as NEEXP, the collection of languages decidable in non-deterministic *doubly exponential* time. This proves that $\mathsf{NEEXP} \subseteq \mathsf{MIP}^*$, and since it is known that $\mathsf{NEXP} \subsetneq \mathsf{NEEXP}$ it follows that $\mathsf{MIP} \neq \mathsf{MIP}^*$.

In contrast to these increasingly strong lower bounds the only upper bound known on MIP^* is the trivial inclusion $\mathsf{MIP}^* \subseteq \mathsf{RE}$, the class of recursively enumerable languages, i.e. languages L such that there exists a Turing machine $\mathcal{M}$ such that $x \in L$ if and only if $\mathcal{M}$ halts and accepts on input x. This inclusion follows since the supremum in (6) can be approximated from below by performing an exhaustive search in increasing dimension and with increasing accuracy. We note that, in addition to containing all decidable languages, this class also contains undecidable problems such as the Halting problem, which is to decide whether a given Turing machine eventually halts.

Our main result is a proof of the reverse inclusion: $\mathsf{RE} \subseteq \mathsf{MIP}^*$. Combined with the preceding observation it follows that

$$\mathsf{MIP}^* = \mathsf{RE} \,,$$

which is a full characterization of the power of entangled-prover interactive proofs. In particular for any $0 < \varepsilon < 1$, it is an undecidable problem to determine whether a given nonlocal game has entangled value 1 or at most $1 - \varepsilon$ (promised that one is the case).

Proof summary. The proof of the inclusion $\mathsf{RE} \subseteq \mathsf{MIP}^*$ is obtained by designing an entangled-prover interactive proof for the Halting problem, which is complete for the class RE. Specifically, we design an efficient transformation that maps any Turing machine $\mathcal{M}$ to a nonlocal game $\mathfrak{G}_{\mathcal{M}}$ such that, if $\mathcal{M}$ halts (when run on an empty input tape) then there is a quantum strategy for the provers that succeeds with probability 1 in $\mathfrak{G}_{\mathcal{M}}$ (i.e. $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M}}) = 1$), whereas if $\mathcal{M}$ does not halt then no quantum strategy can succeed with probability larger than $\frac{1}{2}$ in the game (i.e. $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M}}) \leq \frac{1}{2}$).

A very rough sketch of this construction is as follows (we give a detailed overview in Section 2). Given an infinite family of games $\{\mathfrak{G}_n\}_{n \in \mathbb{N}}$, we say that the family is *uniformly generated* if there is a polynomial-time Turing machine that on input n returns a description of the game $\mathfrak{G}_n$. Given a game $\mathfrak{G}$ and $p \in [0,1]$ let $\mathscr{E}(\mathfrak{G}, p)$ denote the minimum local dimension of an entangled state shared by the players in order for them to succeed in $\mathfrak{G}$ with probability at least p.

We proceed in two steps. First, we design a *compression* procedure for a specific class of nonlocal games that we call *normal form*. Given as input a uniformly generated family $\{\mathfrak{G}_n\}_{n \in \mathbb{N}}$ of normal form games, the compression procedure returns another uniformly generated family $\{\mathfrak{G}'_n\}_{n \in \mathbb{N}}$ of normal form games with the following properties: (i) for all n, if $\mathrm{val}^*(\mathfrak{G}_{2^n}) = 1$ then $\mathrm{val}^*(\mathfrak{G}'_n) = 1$, and (ii) for all n, if $\mathrm{val}^*(\mathfrak{G}_{2^n}) \leq \frac{1}{2}$ then $\mathrm{val}^*(\mathfrak{G}'_n) \leq \frac{1}{2}$ and moreover

$$\mathscr{E}(\mathfrak{G}'_n, \tfrac{1}{2}) \geq \max\left\{ \mathscr{E}\left(\mathfrak{G}_{2^n}, \tfrac{1}{2}\right), 2^{2^{\Omega(n)}} \right\} \,.$$

The construction of this compression procedure is our main contribution. Informally, it combines the recursive compression technique developed in [Ji17, FJVY19] with the so-called "introspection" technique of [NW19] that was used to prove $\mathsf{NEEXP} \subseteq \mathsf{MIP}^*$. The introspection technique itself relies heavily on the quantum low-degree test of [NV18a] to robustly self-test certain distributions that arise from constructions of *classical* probabilistically checkable proofs. The quantum low-degree test and the introspection technique allow us to avoid the shrinking gap limitation of the results from [FJVY19].

10

In the second step, we use the compression procedure in an iterated fashion to construct an interactive proof system for the Halting problem. Fix a Turing machine $\mathcal{M}$ and consider the following family of nonlocal games $\{\mathfrak{G}_{\mathcal{M},n}^{(0)}\}_{n\in\mathbb{N}}$: for all $n \in \mathbb{N}$, if $\mathcal{M}$ halts in at most n steps (when run on an empty input tape), then $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M},n}^{(0)}) = 1$, and otherwise $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M},n}^{(0)}) \leq \frac{1}{2}$.[9]

Constructing such a family of games is trivial; furthermore, they can be made in the "normal form" required by the compression procedure. However, consider applying the compression procedure to obtain a family of normal form games $\{\mathfrak{G}_{\mathcal{M},n}^{(1)}\}_{n\in\mathbb{N}}$. Then for all $n \in \mathbb{N}$, it holds that if $\mathcal{M}$ halts in at most 2^n steps then $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M},n}^{(1)}) = 1$, and otherwise $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M},n}^{(1)}) \leq \frac{1}{2}$, and furthermore any strategy that achieves a value of at least $\frac{1}{2}$ requires an entangled state of dimension at least $2^{2^{\Omega(n)}}$.

Intuitively, one would expect that iterating this procedure and "taking the limit" gives a family of games $\{\mathfrak{G}_{\mathcal{M},n}^{(\infty)}\}_{n\in\mathbb{N}}$ such that if $\mathcal{M}$ halts then $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M},n}^{(\infty)}) = 1$ for all $n \in \mathbb{N}$, whereas if $\mathcal{M}$ does not halt then no finite-dimensional strategy can succeed with probability larger than $\frac{1}{2}$ in $\mathfrak{G}_{\mathcal{M},n}^{(\infty)}$, for all $n \in \mathbb{N}$; in particular $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M},n}^{(\infty)}) \leq \frac{1}{2}$. Formally, we do not take such a limit but instead define directly the family of games $\{\mathfrak{G}_{\mathcal{M},n}^{(\infty)}\}_{n\in\mathbb{N}}$ as a *fixed point* of the Turing machine that implements the compression procedure. The game $\mathfrak{G}_{\mathcal{M}}$ can then be taken as $\mathfrak{G}_{\mathcal{M},1}^{(\infty)}$. We describe this in more detail in Section 2.

1.3 Consequences

Our result is motivated by a connection with Tsirelson's problem from quantum information theory, itself related to Connes' Embedding Conjecture in the theory of von Neumann algebras [Con76]. In a celebrated sequence of papers, Tsirelson [Tsi93] initiated the systematic study of quantum correlation sets. Recall the definition of the set of quantum spatial correlations

$$C_{qs}(n,k) = \left\{ (p_{abxy}) \mid p_{abxy} = \langle\psi|A_a^x \otimes B_b^y|\psi\rangle, \; |\psi\rangle \in \mathcal{H}_{\mathrm{A}} \otimes \mathcal{H}_{\mathrm{B}}, \; \forall xy, \; \{A_a^x\}_a, \{B_b^y\}_b \text{ POVM} \right\}, \quad (7)$$

where here $|\psi\rangle$ ranges over all unit norm vectors $|\psi\rangle \in \mathcal{H}_{\mathrm{A}} \otimes \mathcal{H}_{\mathrm{B}}$ with $\mathcal{H}_{\mathrm{A}}$ and $\mathcal{H}_{\mathrm{B}}$ arbitrary (separable) Hilbert spaces, and a POVM is defined as a collection of positive semidefinite operators that sum to identity. (From now on we use the Dirac ket notation $|\psi\rangle$ for states.) Recall the closure $C_{qa}(n,k)$ of $C_{qs}(n,k)$.

Tsirelson observed that there is a natural alternative definition to the quantum spatial correlation set, called the *quantum commuting correlation set* and defined as

$$C_{qc}(n,k) = \left\{ (p_{abxy}) \mid p_{abxy} = \langle\psi|A_a^x B_b^y|\psi\rangle \right\}, \quad (8)$$

where $|\psi\rangle \in \mathcal{H}$ is a quantum state, $\{A_a^x\}$ and $\{B_b^y\}$ are POVMs for all x,y, and $[A_a^x, B_b^y] = 0$ for all a,b,x,y. Note the key difference with spatial correlations is that in (8) all operators act on the same (separable) Hilbert space. The requirement that operators associated with different inputs (questions) x,y commute is arguably a minimal requirement within the context of quantum mechanics for there to not exist any causal connection between outputs (answers) a,b obtained in response to the respective input.

The set $C_{qc}(n,k)$ is closed and convex, and it is easy to see that $C_{qa}(n,k) \subseteq C_{qc}(n,k)$ for all $n,k \geq 1$. When Tsirelson initially introduced these sets he claimed that equality holds. However, it was later pointed out that this is not obviously true. The question of equality between C_{qc} and C_{qa} (for all n,k) is now known as *Tsirelson's problem* [Tsi06]. Let $C_q(n,k)$ denote the same as $C_{qs}(n,k)$ except that both $\mathcal{H}_{\mathrm{A}}$ and $\mathcal{H}_{\mathrm{B}}$

---

[9]There is nothing special about the choice of $\frac{1}{2}$; this can be set to any constant that is less than 1.

in (7) are restricted to finite-dimensional spaces. Then more generally one can consider the following chain of inclusions

$$C_q(n,k) \subseteq C_{qs}(n,k) \subseteq C_{qa}(n,k) \subseteq C_{qc}(n,k) \,, \tag{9}$$

for all $n, k \in \mathbb{N}$, and ask which (if any) of these inclusions are strict. We let $C_q, C_{qs}, C_{qa}, C_{qc}$ denote the union of $C_q(n,k), C_{qs}(n,k), C_{qa}(n,k), C_{qc}(n,k)$, respectively, over all integers $n, k \in \mathbb{N}$. In a breakthrough work, Slofstra established the first separation between these four correlation sets by proving that $C_{qs} \neq C_{qc}$ [Slo19b]; he later proved the stronger statement that $C_{qs} \neq C_{qa}$ [Slo19a]. As a consequence of the technique used to demonstrate the separation Slofstra also obtains the complexity-theoretic statement that the problem of determining whether an element p lies in C_{qc}, even promised that if it does, then it also lies in C_{qa}, is undecidable. Interestingly, this is shown by reduction from the *complement* of the halting problem; for our result we reduce from the halting problem (see Section 1.4 for further discussion of this point). Since his work, simpler proofs of Slofstra's results have been found [DPP19, MR18, Col19]. In [CS18], Coladangelo and Stark showed that $C_q \neq C_{qs}$ by exhibiting a 5-input, 3-output correlation that can be attained using infinite-dimensional spatial strategies (i.e. infinite-dimensional Hilbert spaces, a state and POVMs satisfying (7)) but cannot be attained via finite-dimensional strategies.

As already noted in [FNT14] (and further elaborated on by [FJVY19]), the undecidability of MIP*$(2,1)$ implies the separation $C_{qa} \neq C_{qc}$.[10] This follows from the observation that if $C_{qa} = C_{qc}$, then there exists an algorithm that can correctly determine if a nonlocal game $\mathfrak{G}$ satisfies val*$(\mathfrak{G}) = 1$ or val*$(\mathfrak{G}) \leq \frac{1}{2}$ and always halts: this algorithm interleaves a hierarchy of semidefinite programs providing outer approximations to the set C_{qc} [NPA08, DLTW08] with a simple exhaustive search procedure providing inner approximations to C_q. Our result that RE $\subseteq$ MIP*$(2,1)$ implies that no such algorithm exists, thus resolving Tsirelson's problem in the negative.

We furthermore exhibit an explicit nonlocal game $\mathfrak{G}$ such that val*$(\mathfrak{G}) <$ val$^{\mathrm{co}}(\mathfrak{G}) = 1$, where val$^{\mathrm{co}}(\mathfrak{G})$ is defined as val*$(\mathfrak{G})$ except that the supremum is taken over the set $C_{qc}(n,k)$ in (8). This in turn yields an explicit correlation that is in the set C_{qc} but not in C_{qa}. This game closely resembles the game $\mathfrak{G}_{\mathcal{M}}$ described in the sketch of the proof that RE $\subseteq$ MIP*, where $\mathcal{M}$ is the Turing machine that runs the hierarchy of semidefinite programs on the game $\mathfrak{G}_{\mathcal{M}}$ and halts if it certifies that val$^{\mathrm{co}}(\mathfrak{G}_{\mathcal{M}}) < 1$. It is in principle possible to determine an upper bound on the parameters n, k for our separating correlation from the proof. While we do not provide such a bound, there is no step in the proof that requires it to be astronomical; e.g. we believe (without proof) that 10^{20} is a clear upper bound.

Connes' Embedding Conjecture. Connes' Embedding Conjecture (CEC) [Con76] is a conjecture in the theory of von Neumann algebras. Briefly, CEC posits that every type II$_1$ von Neumann factor embeds into an ultrapower of the hyperfinite II$_1$ factor. We refer to [Oza13] for a precise formulation of the conjecture and connections to other conjectures in operator algebras, such as Kirchberg's QWEP conjecture. In independent work Fritz [Fri12] and Junge et al. [JNP+11] showed that a positive answer to CEC would imply a positive resolution of Tsirelson's problem, i.e. that $C_{qa}(n,k) = C_{qc}(n,k)$ for all n, k. (This was later promoted to an equivalence by Ozawa [Oza13].) Since our result disproves this equality for some n, k it also implies that CEC does not hold. We note that using the constructive aspect of our result it may be possible to give an explicit description of a factor that does not embed into an ultrapower of the hyperfinite II$_1$ factor, but we do not give such a construction.

---

[10]Technically [FNT14] make the observation for the commuting-prover analogue MIP$^{\mathrm{co}}(2,1)$, discussed further in Section 1.4, but the reasoning is the same.

Entanglement tests. As a step towards showing our result for any integer $n \geq 1$ we construct a game $\mathfrak{G}_n$, with question and answer length polynomial in the size of the smallest Turing machine $\mathcal{M}_n$ that halts (on the empty tape) in exactly n steps (i.e. the *Kolmogorov complexity* of n), such that $\mathrm{val}^*(\mathfrak{G}_n) = 1$ yet any quantum strategy that succeeds in $\mathfrak{G}_n$ with probability larger than $\frac{1}{2}$ must use an entangled state whose Schmidt rank is at least $2^{\Omega(n)}$. This is by far the most efficient entanglement test that we are aware of.

Prover and round reduction for MIP^* **protocols.** Let $\mathsf{MIP}^*(k, r)$ denote the collection of languages decidable by MIP^* protocols with $k \geq 2$ provers and r rounds. Prior to our work it was known how to perform *round reduction* for MIP^* protocols, at the cost of adding provers; it was shown by [Ji17, FJVY19] that $\mathsf{MIP}^*(k, r) \subseteq \mathsf{MIP}^*(k + 15, 1)$ for all k, r. However, it was an open question whether the complexity of the class MIP^* increases if we add more provers. Our main complexity-theoretic result implies that $\mathsf{MIP}^* = \mathsf{MIP}^*(2, 1)$. This follows from the following chain of inclusions: for all polynomially-bounded functions k, r,

$$\mathsf{MIP}^*(2, 1) \subseteq \mathsf{MIP}^*(k, r) \subseteq \mathsf{RE} \subseteq \mathsf{MIP}^*(2, 1) .$$

The first inclusion follows since the verifier in an MIP^* protocol can always ignore extra provers and rounds; the second inclusion follows from a simple exhaustive-search procedure that enumerates over strategies for a given $\mathsf{MIP}^*(k, r)$ protocol; the third result is proven in this paper.[11]

However, this method of reducing provers and rounds in a given MIP^* protocol is indirect; it involves first converting a given MIP^* protocol into a Turing machine that accepts if and only if the MIP^* protocol has value larger than $\frac{1}{2}$, and then constructing an $\mathsf{MIP}^*(2, 1)$ protocol to decide whether the Turing machine halts. In particular this transformation does not generally preserve the complexity of the provers and verifier in the original protocols. We leave it as an open question to find a more direct method for reducing the number of provers in an MIP^* protocol.

1.4 Open questions

We mention several questions left open by our work.

Explicit constructions of counter-examples to Connes' Embedding Conjecture. We provide an explicit counter-example to Tsirelson's problem in the form of a game whose entangled value differs from its commuting-operator value. Through the aforementioned connection with Connes' embedding conjecture [Fri12, JNP$^+$11, Oza13], the counter-example may lead to the construction of interesting objects in other areas of mathematics. A first question is whether it can lead to an explicit description of a type II_1 factor that does not satisfy the Connes embedding property. Such a construction could be obtained along the lines of [KPS18], using the fact that our game $\mathfrak{G}$ such that $\mathrm{val}^*(\mathfrak{G}) < \mathrm{val}^{\mathrm{co}}(\mathfrak{G}) = 1$ has the property of being *synchronous*, i.e. perfect strategies in the game are required to return the same answer when both parties are provided the same question.

Going further, one may ask if the example can eventually lead to a construction of a group that is not sofic, or even not hyperlinear (see e.g. [CLP15] for the connection). Many other formulations of CEC are known, and we leave the discussion of additional potential applications of our results to a future version of the paper.

---

[11] In fact, we note that the second term $\mathsf{MIP}^*(k, r)$ can be replaced by $\mathsf{QMIP}^*(k, r)$, which is the analogous class with a quantum verifier and quantum messages, since the first inclusion is trivial and the second remains true. As a result, we obtain that $\mathsf{QMIP}^* = \mathsf{MIP}^*(2, 1)$ as well.

The complexity of variants of MIP^*. Our result characterizes the complexity class MIP^* as the set of recursively enumerable languages. One can also consider the complexity class MIP^{co}, which stands for *multiprover interactive proofs in the commuting-operator model*. For the sake of the discussion we consider only two-prover one-round protocols; a language L is in $\mathsf{MIP}^{co}(2,1)$ if there exists an efficient reduction that maps $z \in \{0,1\}^*$ to a nonlocal game $\mathfrak{G}_z$ such that if $z \in L$ then $\mathrm{val}^{co}(\mathfrak{G}_z) \geq \frac{2}{3}$, and otherwise $\mathrm{val}^{co}(\mathfrak{G}_x) \leq \frac{1}{3}$.

The semidefinite programming hierarchy of [NPA08, DLTW08] can be used to show that $\mathsf{MIP}^{co}(2,1)$ is contained in the *complement* of RE, denoted as coRE: to certify that $z \notin L$ it suffices to run the hierarchy until it obtains a certificate that $\mathrm{val}^{co}(\mathfrak{G}_z) < \frac{2}{3}$. Since it is known that RE $\neq$ coRE,[12] this implies that $\mathsf{MIP}^*(2,1) \neq \mathsf{MIP}^{co}(2,1)$.

It is thus plausible that $\mathsf{MIP}^{co} = \mathsf{coRE}$,[13] which would provide a very pleasing "dual" complexity characterization to $\mathsf{MIP}^* = \mathsf{RE}$. One possible route to proving this would be to adapt our gap-preserving compression framework to the commuting-operator setting by showing that each of the steps (question reduction, answer reduction, and parallel repetition) remain sound against commuting-operator strategies. Using the connection established in [FNT14], this would imply that the operator norm over the maximal C^* algebra $C^*(F_2 * F_2)$, where F_2 is the free group on two elements, is uncomputable.

Another interesting open question concerns the *zero gap* variants of MIP^* and MIP^{co}, which we denote by MIP_0^* and $\mathsf{MIP}^{co}{}_0$, respectively. These classes capture the complexity of deciding whether a nonlocal game $\mathfrak{G}$ has entangled value (or commuting-operator value respectively) *exactly* equal to 1. In [Slo19a], Slofstra shows that there is an efficient reduction from Turing machines $\mathcal{M}$ to nonlocal games $\mathfrak{G}_{\mathcal{M}}$ such that $\mathcal{M}$ does not halt if and only if $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M}}) = \mathrm{val}^{co}(\mathfrak{G}_{\mathcal{M}}) = 1$. This implies that $\mathsf{coRE} = \mathsf{MIP}^{co}{}_0(2,1)$ and furthermore $\mathsf{coRE} \subseteq \mathsf{MIP}_0^*$. However, since $\mathsf{RE} \subseteq \mathsf{MIP}^* \subseteq \mathsf{MIP}_0^*$, this implies that MIP_0^* is *strictly* bigger than either RE and coRE. Thus it is plausible that the complexity landscape of nonlocal games looks like the following: $\mathsf{MIP}^{co} = \mathsf{MIP}^{co}{}_0 = \mathsf{coRE}$, but $\mathsf{RE} = \mathsf{MIP}^* \neq \mathsf{MIP}_0^*$. Such statements about the complexity of MIP^* versus MIP^{co}, in both the gapped and zero-gap cases, may reveal additional insights into the difference between the tensor product and commuting-operator models of correlations.

---

[12] RE $\neq$ coRE follows from the fact that RE $\cap$ coRE is the set of decidable languages and RE contains undecidable languages.

[13] We note that the "co" modifier on both sides of the equation $\mathsf{MIP}^{co} = \mathsf{coRE}$ refer to different things!

2 Proof Overview

In this section we give an overview of the proof of the inclusion $\mathrm{RE} \subseteq \mathrm{MIP}^*$. Since all interactive proof systems considered in the paper involve a single-round interaction between a classical verifier and two quantum provers sharing entanglement we generally use the language of nonlocal games to describe such proof systems, and often refer to the provers as "players". In a nonlocal game $\mathfrak{G}$ (or simply "game" for short), the verifier can be described as the combination of two procedures: a *question sampling* procedure that samples a pair of questions (x, y) for the players according to a distribution μ (known to the verifier and the players), and a *decision* procedure (also known to all parties) that takes as input the players' questions and their respective answers a, b and evaluates a predicate $D(x, y, a, b) \in \{0, 1\}$ to determine the verifier's acceptance or rejection. Given a description of a nonlocal game $\mathfrak{G}$, recall that $\mathrm{val}^*(\mathfrak{G})$ denotes the *entangled value* of the game, which is defined as the supremum (6) of the players' success probability in the game over all finite-dimensional tensor product strategies. (We refer to Section 5 for definitions regarding nonlocal games.)

Our results establish the existence of transformations on *families* of nonlocal games $\{\mathfrak{G}_n\}_{n \in \mathbb{N}}$ having certain properties. In order to keep track of efficiency (and ultimately, computability) properties it is important to have a way to specify such families in a uniform manner. Towards this we introduce the following formalism. A *uniformly generated family of games* is specified through a pair of Turing machines $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ that satisfy certain conditions, in which case the pair is called a *normal form verifier*. The Turing machine $\mathcal{S}$ (called a *sampler*) takes as input an index $n \in \mathbb{N}$ and returns the description of a procedure that can be used to sample questions (x, y) in the game (this procedure itself obeys a certain format associated with "conditionally linear" distributions, defined below). The Turing machine $\mathcal{D}$ (called a *decider*) takes as input an index n, questions (x, y), and answers (a, b), and returns a single-bit decision. For the sake of this proof overview we assume that the sampling and decision procedures run in time polynomial in the index n; we refer to the running time of these procedures as the *complexity* of the verifier. Given a normal form verifier $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ we associate to it an infinite family of nonlocal games $\{\mathfrak{G}_n = \mathcal{V}_n\}$ indexed by positive integers in the natural way.

The main technical result of this paper is a *gap-preserving compression* transformation on normal form verifiers. The following theorem presents an informal summary of the properties of this transformation. Recall that for a game $\mathfrak{G}$ and probability $0 \leq p \leq 1$, $\mathscr{E}(\mathfrak{G}, p)$ denotes the minimum local dimension of an entangled state shared by the players in order for them to succeed in $\mathfrak{G}$ with probability at least p.

Theorem 2.1 (Gap-preserving compression of normal form verifiers, informal). *There exists a polynomial-time Turing machine* Compress *that, when given as input the description of a normal form verifier* $\mathcal{V} = (\mathcal{S}, \mathcal{D})$, *outputs the description of another normal form verifier* $\mathcal{V}' = (\mathcal{S}', \mathcal{D}')$ *that satisfies the following properties: for all* $n \in \mathbb{N}$, *letting* $N = 2^n$,

1. *(**Completeness**) If* $\mathrm{val}^*(\mathcal{V}_N) = 1$ *then* $\mathrm{val}^*(\mathcal{V}'_n) = 1$.

2. *(**Soundness**) If* $\mathrm{val}^*(\mathcal{V}_N) \leq \frac{1}{2}$ *then* $\mathrm{val}^*(\mathcal{V}'_n) \leq \frac{1}{2}$.

3. *(**Entanglement lower bound**)* $\mathscr{E}(\mathcal{V}'_n, \frac{1}{2}) \geq \max\{\mathscr{E}(\mathcal{V}_N, \frac{1}{2}), 2^{2^{\Omega(n)}}\}$.

The formal version of this theorem is stated in Section 12 as Theorem 12.2. The terminology *compression* is motivated by the fact, implicit in the informal statement of the theorem, that the time complexity of the verifier's sampling and decision procedures in the game $\mathcal{V}'_n$, which is polynomial in n, is exponentially smaller than the time complexity of the verifier in the game $\mathcal{V}_N$, which is polynomial in N and thus exponential in n.

Before giving an overview of the proof of Theorem 2.1 we sketch how the existence of a Turing machine Compress with the properties stated in the theorem implies the inclusion $\mathsf{RE} \subseteq \mathsf{MIP}^*$. Recall that the complexity class RE consists of all languages L such that there is a Turing machine $\mathcal{M}$ that accepts instances x in L, and does not accept instances x that are not in L (but is not required to terminate on such instances). To show $\mathsf{RE} \subseteq \mathsf{MIP}^*$ we give an MIP^* protocol for the Halting Problem, which is a complete problem for RE. The Halting Problem is the language that consists of all Turing machine descriptions $\mathcal{M}$ such that $\mathcal{M}$ halts when run on an empty input tape. (For the purposes of this overview, we blur the distinction between a Turing machine and its description as a string of bits.) We give a procedure that given a Turing machine $\mathcal{M}$ as input returns the description of a normal form verifier $\mathcal{V}^{\mathcal{M}} = (\mathcal{S}^{\mathcal{M}}, \mathcal{D}^{\mathcal{M}})$ with the following properties. First, if $\mathcal{M}$ does eventually halt on an empty input tape, then it holds that for all $n \in \mathbb{N}$, $\mathrm{val}^*(\mathcal{V}_n^{\mathcal{M}}) = 1$. Second, if $\mathcal{M}$ does not halt then for all $n \in \mathbb{N}$, $\mathrm{val}^*(\mathcal{V}_n^{\mathcal{M}}) \leq \frac{1}{2}$.

We describe the procedure that achieves this. Informally, the procedure returns the specification of a verifier $\mathcal{V}^{\mathcal{M}} = (\mathcal{S}^{\mathcal{M}}, \mathcal{D}^{\mathcal{M}})$ such that $\mathcal{D}^{\mathcal{M}}$ proceeds as follows: on input (n, x, y, a, b) it first executes the Turing machine $\mathcal{M}$ for n steps. If $\mathcal{M}$ halts, then $\mathcal{D}^{\mathcal{M}}$ accepts. Otherwise, $\mathcal{D}^{\mathcal{M}}$ computes the description of the compressed verifier $\mathcal{V}' = (\mathcal{S}', \mathcal{D}')$ that is the output of Compress on input $\mathcal{V}^{\mathcal{M}}$, then executes the decision procedure $\mathcal{D}'(n, x, y, a, b)$ and accepts if and only if $\mathcal{D}'$ accepts.[14]

To show that this procedure achieves the claimed transformation, consider two cases. First, observe that if $\mathcal{M}$ eventually halts in some number of time steps T, then by definition $\mathrm{val}^*(\mathcal{V}_n^{\mathcal{M}}) = 1$ for all $n \geq T$. Using Theorem 2.1 along with an inductive argument it then follows that $\mathrm{val}^*(\mathcal{V}_n^{\mathcal{M}}) = 1$ for all $n \geq 1$. Second, if $\mathcal{M}$ never halts, then observe that for any $n \geq 1$ Theorem 2.1 implies two separate lower bounds on the amount of entanglement required to win the game $\mathcal{V}_n^{\mathcal{M}}$ with probability at least $\frac{1}{2}$: the dimension is (a) at least $2^{2^{\Omega(n)}}$, and (b) at least the dimension needed to win the game $\mathcal{V}_{2^n}^{\mathcal{M}}$ with probability at least $\frac{1}{2}$. Applying an inductive argument it follows that an *infinite* amount of entanglement is needed to win the game $\mathcal{V}_n$ with any probability greater than $\frac{1}{2}$. Thus, a sequence of finite-dimension strategies for $\mathcal{V}_n$ cannot lead to a limiting value larger than $\frac{1}{2}$, and $\mathrm{val}^*(\mathcal{V}_n^{\mathcal{M}}) \leq \frac{1}{2}$.

We continue with an overview of the ideas behind the proof of Theorem 2.1.

Compression by introspection. To start, it is useful to review the protocol introduced in [NW19] to show the inclusion $\mathsf{NEEXP} \subseteq \mathsf{MIP}^*$. Fix an NEEXP-complete language L. The MIP^* protocol for NEXP from [NV18b], when scaled up to decide languages from NEEXP, yields a family of nonlocal games $\{\mathfrak{G}_z\}$ that are indexed by instances $z \in \{0, 1\}^*$. The family of games decides L in the sense that for all z, the game $\mathfrak{G}_z$ has entangled value 1 if $z \in L$, and has entangled value at most $\frac{1}{2}$ if $z \notin L$. Furthermore, if $n = |z|$ is the length of z, the verifier of the game $\mathfrak{G}_z$ has complexity $\mathrm{poly}(N) = \exp(|z|)$ (recall that we use this terminology to refer to an upper bound on the running time of the verifier's sampling and decision procedure). Thus, this family of games does not by itself yield an MIP^* protocol for L. To overcome this the main contribution in [NW19] is the design of an efficient compression procedure $\mathsf{Compress}^{\mathsf{NW}}$ that applies specifically to the family of games $\{\mathfrak{G}_z\}$. When given as input the description of $\mathfrak{G}_z$, $\mathsf{Compress}^{\mathsf{NW}}$ returns the description of a game $\mathfrak{G}_z'$ such that if $\mathrm{val}^*(\mathfrak{G}_z) = 1$, then $\mathrm{val}^*(\mathfrak{G}_z') = 1$, and if $\mathrm{val}^*(\mathfrak{G}_z) \leq \frac{1}{2}$, then $\mathrm{val}^*(\mathfrak{G}_z') \leq \frac{1}{2}$. Furthermore, the complexity of the verifier for $\mathfrak{G}_z'$ is $\mathrm{poly}(n)$. Thus the family of games $\{\mathfrak{G}_z'\}$ decides L and this shows that $\mathsf{NEEXP} \subseteq \mathsf{MIP}^*$, which is the best lower bound known on MIP^* prior to our work.

Presented in this way, it is natural to suggest iterating the procedure $\mathsf{Compress}^{\mathsf{NW}}$ to achieve e.g. the inclusion $\mathsf{NEEEXP} \subseteq \mathsf{MIP}^*$. To explain the difficulty in doing so, we give a little more detail on the

---

[14]The fact that the decider $\mathcal{D}^{\mathcal{M}}$ can invoke the Compress procedure on itself follows from a well-known result in computability theory known as *Kleene's recursion theorem* (also called *Roger's fixed point theorem*) [Kle54, Rog87].

compression procedure. It consists of two main steps: starting from $\mathfrak{G}_z$, perform (1) *question reduction*, and (2) *answer reduction*. The goal of (1) is to reduce the length of the questions generated by the verifier in $\mathfrak{G}_z$ from $\mathrm{poly}(N)$ to $\mathrm{poly}(n)$. The goal of (2) is to achieve the same with respect to the length of answers expected from the players. Furthermore, the complexity of the verifier of the resulting game $\mathfrak{G}'_z$ should be reduced from $\mathrm{poly}(N)$ to $\mathrm{poly}(n)$.

Part (1) is achieved through a technique referred to as "introspection" where, rather than sampling questions (x, y) of length $\mathrm{poly}(N)$ as in the game $\mathfrak{G}_z$, the verifier instead executes a carefully crafted nonlocal game with the players that (a) requires questions of length $\mathrm{poly}(n)$, (b) checks that the players share $\mathrm{poly}(N)$ EPR pairs, and (c) checks that the players measure the EPR pairs in such a way as to *sample for themselves* a question pair (x, y) such that one player gets x and the other player gets y. In other words, the players are essentially forced to introspectively ask themselves the questions of $\mathfrak{G}_z$.

After question reduction, the players still respond with $\mathrm{poly}(N)$-length answers, which the verifier has to check satisfies the decision predicate of the original game $\mathfrak{G}_z$. The goal of Part (2) is to enable the decision procedure to implement the verification procedure while not requiring the entire full-length answers from the players. In the answer reduction scheme of [NW19] this is achieved by having the verifier run a *probabilistically checkable proof* (PCP) with the players so that they succinctly *prove* that first, they have introspected questions (x, y) from the correct distribution, and second, that they are able to generate $\mathrm{poly}(N)$-length answers (a, b) that would satisfy the decision predicate of the original game $\mathfrak{G}_z$ when executed on (x, y) and (a, b). Since the questions and answers in the PCP are of length $\mathrm{poly}(n)$, this achieves the desired answer length reduction.

Iterating this scheme presents a number of immediate difficulties that have to do with the fact that the sampling and decision procedures of the verifier in $\mathfrak{G}'_z$ do not have such a nice form as those in $\mathfrak{G}_z$. First of all, the compression procedure of [NW19] can only "introspect" a very specific question distribution, which is a variant of the plane-point distribution used by the verifier from [NV18b].[15] However, the resulting question distribution of the question-reduced verifier, which is used to check the introspection, has a much more complex structure. A similar issue arises with the modifications required to perform answer reduction. In the PCP employed to achieve this the question distribution appears to be much more complex than the plane-point distribution (this is in large part due to the need for a specially tailored PCP procedure that encodes separately different chunks of the witness, corresponding to answers from different players). As a result it is entirely unclear at first whether the question distribution used by the verifier in $\mathfrak{G}'_z$ can be "introspected" for a second time.

To overcome these difficulties we identify a natural class of question distributions, called *conditionally linear distributions*, that generalize the classic plane-point distribution. We show that conditionally linear distributions can be "introspected" using conditionally linear distributions only, enabling recursive introspection. (In particular, they are a rich enough class to capture the types of question distributions produced by the compression scheme of [NW19].) We define normal form verifiers by restricting their sampling procedure to generate conditionally linear question distributions, and this allows us to obtain the compression procedure on normal form verifiers described in Theorem 2.1.

Conceptually, the identification of a natural class of distributions that is "closed under introspection" is a key step that enables the introspection technique to be applied recursively. (As we will see later, other closure properties of conditionally linear distributions, such as taking direct products, play an important role as well.) Since conditionally linear distributions are central to our construction we describe them next.

---

[15]This distribution returns a pair $(x, y = p)$ where p is the description a uniformly random affine plane in $\mathbb{F}^m$, for some given finite field $\mathbb{F}$ and integer $m \geq 2$, and x a uniformly random point in p.

Conditionally linear distributions. Fix a vector space V that is identified with $\mathbb{F}^m$, for a finite field $\mathbb{F}$ and integer m. Informally (see Definition 4.1 for a precise definition), a function L on V is *conditionally linear* (CL for short) if it can be evaluated by a procedure that takes the following form: (i) read a substring $z^{(1)}$ of z; (ii) evaluate a linear function L_1 on $z^{(1)}$; (iii) repeat steps (i) and (ii) with the remaining coordinates $z \backslash z^{(1)}$, such that the next steps are allowed to depend in an arbitrary way on $L_1(z^{(1)})$ but not directly on $z^{(1)}$ itself. What distinguishes a function of this form from an arbitrary function is that we restrict the number of iterations of (i)—(ii) to a constant number, typically 2–8. (One may also think of CL functions as "adaptively linear" functions, where the number of "levels" of adaptivity is the number of iterations of (i)—(ii).)

A distribution μ over pairs $(x, y) \in V \times V$ is called conditionally linear if it is the image under a pair of conditionally linear functions $L^A, L^B : V \to V$ of the uniform distribution on V, i.e. $(x, y) \sim (L^A(z), L^B(z))$ for uniformly random $z \in V$. An important example of a CL distribution is the plane-point distribution. Set $V = V_1 \oplus V_2 \oplus V_3$, where for $i \in \{1, 2, 3\}$, $V_i = \mathbb{F}^m$. Set L^B to be the projection on V_1. Define L^A as follows. Let $z \in V$. First, read the components z_2 and z_3 of z that lie in V_2 and V_3 respectively and set L_1^A to be the identity function on $V_2 \oplus V_3$. Second, conditioned on the observed value (z_2, z_3), let L_2^A be the linear function on V_1 that projects orthogonally to $\mathrm{Span}\{z_1, z_2\}$, seen as an (at most) 2-dimensional subspace of V_1. Finally, let $L^A(z) = L_1^A(z) + L_2^A(z_1) \in V$. It is not hard to see (and shown formally in Section 7.1.2) that the distribution of $(L^A(z), L^B(z))$, for z uniform in V, is identical (up to relabeling) to the distribution $(\mathrm{PL}, \mathrm{PT})$ where PL is a uniformly random subspace of $\mathbb{F}^m$ of dimension at most 2, and PT a uniformly random point in PL.

Our main result about CL distributions, presented in Section 8, is that any CL distribution μ, associated with a pair of CL functions (L^A, L^B) over a linear space $V = F^m$, can be "introspected" using a CL distribution that is "exponentially smaller" than the initial distribution. Slightly more formally, to any CL distribution μ we associate a two-player game $\mathfrak{G}_\mu$ (called the "introspection game") in which questions from the verifier are sampled from a CL distribution μ' over $\mathbb{F}^{m'}$ for some $m' = \mathrm{poly}\log(m)$ and such that in any successful strategy for the game $\mathfrak{G}_\mu$, when the players are queried on a special question labeled INTRO, they must respond with a pair (x, y) that is approximately distributed according to μ. (The game allows us to do more: it allows us to conclude *how* the players obtained (x, y) — by measuring shared EPR pairs in a specific basis — and this will be important when using the game as part of a larger protocol that involves other checks.) Crucially for us, the distribution μ' only depends on a size parameter associated with (L^A, L^B) (essentially, the integer m together with the number of "levels" of adaptivity of L^A and L^B), but not on any other structural property of (L^A, L^B). Only the decision predicate for the introspection game $\mathfrak{G}_\mu$ depends on the entire description of (L^A, L^B).

We say a few words about the design of μ' and the associated introspection game, which borrow heavily from [NW19]. Building on the "quantum low-degree test" introduced in [NV18a] it is already known how a verifier can force a pair of players to measure m EPR pairs in either the computational or Hadamard basis and report the (necessarily identical) outcome z obtained, all the while using questions of length polylogarithmic in m only. The added difficulty in our situation is to ensure that a player obtains, and returns, precisely the information about z that is contained in $L^A(z)$ (resp. $L^B(z)$), and not more. A simple example is the plane-point distribution described earlier: there, the idea to ensure that e.g. the first player only obtains the first component, z_1, of z, the verifier demands that the player measures their qubits associated with spaces V_2 and V_3 in the Hadamard, instead of computational, basis; due to the uncertainty principle this has the effect of "erasing" the outcome in the computational basis. The case of the player receiving a "plane" question is a little more complex, but it was shown possible in [NW19].

We can now describe how samplers of normal form verifiers are defined: these are Turing machines $\mathcal{S}$ that specify an infinite family of CL distributions $\{\mu_n\}$ such that, when given index n, the sampler $\mathcal{S}$

computes the CL functions $(L^{A,n}, L^{B,n})$ associated with μ_n and also computes various parameters of the CL functions. (See Definition 4.12 for a formal definition of samplers.) Thus, the question distributions of a normal form verifier $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ are the CL distributions corresponding to $\mathcal{S}$.

Question reduction. Just like the compression procedure of [NW19], the compression procedure Compress of Theorem 2.1 begins with performing question reduction on the input game. Given a normal form verifier $\mathcal{V} = (\mathcal{S}, \mathcal{D})$, the procedure Compress first computes a normal form verifier $\mathcal{V}^{\text{INTRO}} = (\mathcal{S}^{\text{INTRO}}, \mathcal{D}^{\text{INTRO}})$ where for all $n \in \mathbb{N}$, the game $\mathcal{V}_n^{\text{INTRO}}$ consists of playing the original game $\mathcal{V}_N$ where $N = 2^n$, except that instead of sampling the questions according to the CL distribution μ_N specified by the sampler $\mathcal{S}_N$, the verifier executes the introspection game $\mathfrak{G}_{\mu_N}$ described in the previous subsection. Thus, in the game $\mathcal{V}_n^{\text{INTRO}}$, when both players receive the question labeled INTRO they are expected to sample (x, y) respectively according to μ_N, and respond with the sampled question together with answers a, b respectively. The decider $\mathcal{D}^{\text{INTRO}}$ on index n evaluates $\mathcal{D}(N, x, y, a, b)$ and accepts if and only if $\mathcal{D}$ accepts. As a result the time complexity of decider $\mathcal{D}^{\text{INTRO}}$ on index n remains that of $\mathcal{D}$, i.e. $\text{poly}(N)$. However, the length of questions asked in $\mathcal{V}_n^{\text{INTRO}}$ and the complexity of the sampler $\mathcal{S}^{\text{INTRO}}$ are exponentially reduced, to $\text{poly}(n)$.

For convenience we refer to the questions asked by the verifier in the "question-reduced" game $\mathcal{V}_n^{\text{INTRO}}$ as "small questions," and the questions that are introspected by the players in $\mathcal{V}_n^{\text{INTRO}}$ (equivalently, the questions asked in the original game $\mathcal{V}_N$) as "big questions."

Answer reduction. Having reduced the complexity of the question sampling, the next step in the compression procedure Compress is to reduce the complexity of decider $\mathcal{D}^{\text{INTRO}}$ from $\text{poly}(N)$ to $\text{poly}(n)$ (which necessarily implies reducing the answer length to $\text{poly}(n)$). To achieve this the compression procedure computes a normal form verifier $\mathcal{V}^{\text{AR}} = (\mathcal{S}^{\text{AR}}, \mathcal{D}^{\text{AR}})$ from $\mathcal{V}^{\text{INTRO}}$ such that both the sampler and decider complexity in $\mathcal{V}^{\text{AR}}$ are $\text{poly}(n)$ (here, AR stands for "answer reduction").

Similarly to the answer reduction performed in [NW19], at a high level this is achieved by composing the game $\mathcal{V}_n^{\text{INTRO}}$ with a probabilistically checkable proof (PCP). In our context a PCP is a proof encoding that allows a verifier to check whether, given Turing machine $\mathcal{A}$ and time bound T provided as input, there exists some input x that $\mathcal{A}$ accepts in time T. The PCP proof can be computed from $\mathcal{A}$, T, and the accepting input (if it exists) and has length polynomial in T and the description length $|\mathcal{A}|$ of $\mathcal{A}$. Crucially, the verifier can check a purported proof while only reading a constant number of symbols of it, each of length $\text{polylog}(T, |\mathcal{A}|)$, and executing a verification procedure that runs in time $\text{polylog}(T, |\mathcal{A}|)$.

We use PCPs for answer reduction as follows. The verifier in the game $\mathcal{V}_n^{\text{AR}}$ samples questions as $\mathcal{V}_n^{\text{INTRO}}$ would and sends them to the players. Instead of receiving the introspected questions and answers (x, y, a, b) for the original game $\mathcal{V}_N$ and running the decision procedure $\mathcal{D}(N, x, y, a, b)$, the verifier instead asks the players to compute a PCP Π for the statement that the original decider $\mathcal{D}$ accepts the input (N, x, y, a, b) in time $T = \text{poly}(N)$. The verifier then samples additional questions for the players that ask them to return specific entries of the proof Π. Finally, upon receipt of the players' answers, the verifier executes the PCP verification procedure. Because of the efficiency of the PCP, both the sampling of the additional questions and the decision procedure can be executed in time $\text{poly}(n)$.[16]

This very rough sketch presents some immediate difficulties. A first difficulty is that in general no player by themselves has access to the entire input (N, x, y, a, b) to $\mathcal{D}$, so no player can compute the entire proof Π. We discuss this issue in the next paragraph. A second difficulty is that a black-box application of an existing PCP, as done in [NW19], results in a question distribution for $\mathcal{V}_n^{\text{AR}}$ (i.e. the sampling of the proof locations

---

[16]This idea is inspired by the technique of composition in the PCP literature, in which the complexity of a verification procedure can be reduced by composing a proof system (often a PCP itself) with another PCP.

to be queried) that is rather complex — and in particular, it may no longer fall within the framework of CL distributions for which we can do introspection. To avoid this, we design a bespoke PCP based on the classical MIP for NEXP (in particular, we borrow and adapt techniques from [BSS05, BSGH+06]). Two essential properties for us are that (i) the PCP proof is a collection of several low-degree polynomials, two of which are low-degree encodings of each player's big answer in the game $\mathcal{V}_n^{\mathrm{INTRO}}$, and (ii) verifying the proof only requires (a) running low-degree tests, (b) querying all polynomials at a uniformly random point, and (c) performing simple consistency checks. Property (i) allows us to eliminate the extra layer of encoding in [NW19], who had to consider a PCP of proximity for a circuit applied to the low-degree encodings of the players' big answers. Property (ii) allows us to ensure that the question distribution employed by $\mathcal{V}_n^{\mathrm{AR}}$ remains conditionally linear.

Oracularization. The preceding paragraph raises a non-trivial difficulty. In order for the players to compute a proof for the claim that $\mathcal{D}(N, x, y, a, b) = 1$ they need to know the entire input (x, y, a, b). However, in general a player only has access to their own question and answer: one player only knows (x, a) and the other player knows (y, b). The standard way of circumventing this difficulty is to consider an "oracularized" version of the game, where one player gets both questions (x, y) and is able to determine both answers (a, b), while the other player only gets one of the questions at random, and is only asked for one of the answers, that is then checked for consistency with the first player's answer.

While this technique works well for games with classical players, when the players are allowed to use quantum strategies using entanglement oracularization does not, in general, preserve the completeness property of the game. To ensure that completeness is preserved we need an additional property of a completeness-achieving strategy for the original game: that there exists a *commuting and consistent strategy* on all pairs of questions (x, y) that are asked in the game with positive probability. Here commuting means that the measurement $\{A_a^x\}_a$ performed by the player receiving x commutes with the measurement $\{B_b^y\}_b$ performed by the player receiving y.[17] Consistent means that if both players perform measurements associated with the same question they obtain the same answer. If both properties hold then in the oracularized game when one player receives a pair (x, y) and the other player receives the question x (say), the first player can simultaneously measure both $\{A_a^x\}_a$ and $\{B_b^y\}_b$ on their own space to obtain a pair of answers (a, b), and the second player can measure $\{A_a^x\}_a$ to obtain a consistent answer a.

For answer reduction to be possible it is thus applied to the oracularized version of the introspection game $\mathcal{V}_n^{\mathrm{INTRO}}$. This in turn requires us to ensure that the introspection game $\mathcal{V}_n^{\mathrm{INTRO}}$ has a commuting and consistent strategy achieving value 1 whenever it is the case that $\mathrm{val}^*(\mathcal{V}_n^{\mathrm{INTRO}}) = 1$. For this property to hold we verify that it holds for the initial game that is used to seed the compression procedure (this is true because we can start with an MIP* protocol for NEXP for which there exists a perfect classical strategy) and we also ensure that each of the transformations of the compression protocol (question reduction, answer reduction, and parallel repetition described next) maintains it.

Parallel repetition. The combined steps of question reduction (via introspection) and answer reduction (via PCP composition) result in a game $\mathcal{V}_n^{\mathrm{AR}}$ such that the complexity of the verifier is $\mathrm{poly}(n)$. Furthermore, if the original game $\mathcal{V}_N$ has value 1, then $\mathcal{V}_n^{\mathrm{AR}}$ also has value 1. Unfortunately the sequence of transformations incurs a loss in the soundness parameters: if $\mathrm{val}^*(\mathcal{V}_N) \leq \frac{1}{2}$, then we can only establish that $\mathrm{val}^*(\mathcal{V}_n^{\mathrm{AR}}) \leq 1 - C$ for some positive constant $C < \frac{1}{2}$ (we call C the *soundness gap*). Such a loss would

---

[17]We stress that the commuting property only applies to question pairs that occur with positive probability, and does not mean that *all* pairs of measurement operators are required to commute; indeed this would imply that the strategy is effectively classical.

prevent us from recursively applying the compression procedure Compress an arbitrary number of times, which is needed to obtain the desired complexity results for MIP*.

To overcome this we need a final transformation to restore the soundness gap of the games after answer reduction to a constant larger than $\frac{1}{2}$. To achieve this we use the technique of parallel repetition. The parallel repetition of a game $\mathfrak{G}$ is another nonlocal game $\mathfrak{G}^k$, for some number of repetitions k, which consists of playing k independent and simultaneous instances of $\mathfrak{G}$ and accepting if and only if all k instances accept. Intuitively, parallel repetition is meant to decrease the value of a game $\mathfrak{G}$ exponentially fast in k, provided val$^*(\mathfrak{G}) < 1$ to begin with. However, it is an open question of whether this is generally true for the entangled value val*.

Nevertheless, some variants of parallel repetition are known to achieve exponential amplification. We use a variant called "anchored parallel repetition" and introduced in [BVY17]. This allows us to devise a transformation that efficiently amplifies the soundness gap to a constant. The resulting game $\mathcal{V}_n^{\text{REP}}$ has the property that if val$^*(\mathcal{V}_n^{\text{AR}}) = 1$, then val$^*(\mathcal{V}_n^{\text{REP}}) = 1$ (and moreover this is achieved using a commuting and consistent strategy), whereas if val$^*(\mathcal{V}_n^{\text{AR}}) \leq 1 - C$ for some universal constant $C > 0$ then val$^*(\mathcal{V}_n^{\text{REP}}) \leq \frac{1}{2}$. Furthermore, we have the additional property, essential for us, that good strategies in the game $\mathcal{V}_n^{\text{REP}}$ require as much entanglement as good strategies in the game $\mathcal{V}_n^{\text{AR}}$ (which in turn require as much entanglement as good strategies in the game $\mathcal{V}_N$). The complexity of the verifier in $\mathcal{V}_n^{\text{REP}}$ remains poly(n).

The anchored parallel repetition procedure, when applied to a normal form verifier, also yields a normal form verifier: this is because the direct product of CL distributions is still conditionally linear.

Putting it all together.   This completes the overview of the transformations performed by the compression procedure Compress of Theorem 2.1. To summarize, given an input normal form verifier $\mathcal{V}$, question reduction is applied to obtain $\mathcal{V}^{\text{INTRO}}$, answer reduction is applied to the oracularized version of $\mathcal{V}^{\text{INTRO}}$ to obtain $\mathcal{V}^{\text{AR}}$, and anchored parallel repetition is applied to obtain $\mathcal{V}^{\text{REP}}$, which is returned by the compression procedure. Each of these transformations preserves completeness (including the commuting and consistent properties of a value-1 strategy) as well as the entanglement requirements of each game; moreover, the overall transformation preserves soundness.

3 Preliminaries

Notation. We use Σ to denote a finite alphabet. $\mathbb{N}$ is the set of positive integers. For $w \in \{0,1\}$, $\overline{w}$ denotes $1 - w$. For $w \in \{A, B\}$, $\overline{w} = B$ if $w = A$ and $\overline{w} = A$ otherwise. (For notational convenience we often implicitly make the identifications $1 \leftrightarrow A$ and $2 \leftrightarrow B$.) We use $\mathbb{F}$ to denote a finite field. We write $M_n(\mathbb{F})$ to denote the set of $n \times n$ matrices over $\mathbb{F}$. We write I to denote the identity operator on a vector space. We write $\mathrm{Tr}(\cdot)$ for the matrix trace. We write $\mathcal{H}$ to denote a separable Hilbert space. For a linear operator T, $\|T\|$ denotes the operator norm.

Asymptotics. All logarithms are base 2. We use the notation $O(\cdot)$, $\mathrm{poly}(\cdot)$, and $\mathrm{polylog}(\cdot)$ in the following way. For $f, g : \mathbb{N} \to \mathbb{R}_+$ we write $f(n) = O(g(n))$ (omitting the integer n when it is clear from context) to mean that there exists a constant $C > 0$ such that for all $n \in \mathbb{N}$, $f(n) \leq Cg(n)$. When we write $f(a_1, \ldots, a_k) = \mathrm{poly}(a_1, \ldots, a_k)$, this indicates that there exists a universal constant $C > 0$ (which may vary each time the notation is used in the paper) such that $f(a_1, \ldots, a_k) \leq C(a_1 \cdots a_k)^C$ for all positive $a_1, \ldots, a_k$. Similarly, when we write $f(a_1, \ldots, a_k) = \mathrm{polylog}(a_1, \ldots, a_k)$, there exists a universal constant C such that $f(a_1, \ldots, a_k) \leq C \prod_{i=1}^k \log^C(1 + a_i)$ for all positive $a_1, \ldots, a_k$. Finally, we write $\log(a_1, \ldots, a_k)$ as short hand for $\prod_{i=1}^k \log(1 + a_i)$.[18]

3.1 Turing machines

Turing machines are a model of computation introduced in [Tur37]. Turing machines play a central role in our modeling of verifiers for nonlocal games. For an in-depth discussion of Turing machines, we refer the reader to Papadimitriou's textbook [Pap94]. Here we establish notation used throughout the paper.

All Turing machines considered in the paper are deterministic and use the binary alphabet. The tapes of a Turing machine are infinite one-dimensional arrays of cells that are indexed by natural numbers. A k-*input* Turing machine $\mathcal{M}$ has k input tapes, one work tape, and one output tape. Each cell of a tape has symbols taken either from the set $\{0,1\}$ or the blank symbol $\sqcup$. At the start of the execution of a Turing machine, the work and output tapes are initialized to have all blank symbols. A Turing machine *halts* when it enters a designated halt state. The *output* of a Turing machine, when it halts, is the binary string that occupies the longest initial stretch of the output tape that does not have a blank symbol. If there are only blank symbols on the output tape, then by convention we say that the Turing machine's output is 0.

Every k-input Turing machine $\mathcal{M}$ computes a (partial) function $f : (\{0,1\}^*)^k \to \{0,1\}^*$ where the function is only defined on subset $S \subseteq (\{0,1\}^*)^k$ of inputs x on which $\mathcal{M}$ halts. We use $\mathcal{M}(x_1, x_2, \ldots, x_k)$ to denote the output of a k-input Turing machine $\mathcal{M}$ when $x_i \in \{0,1\}^*$ is written on the i-th input tape for $i \in \{1, 2, \ldots, k\}$. If $\mathcal{M}$ does not halt on an input x, then we define $\mathcal{M}(x)$ to be $\bot$. A Turing machine that halts on all inputs computes a *total* function.

We often leave the number of input tapes of a Turing machine implicit. The *time complexity* of a Turing machine $\mathcal{M}$ on input $x = (x_1, x_2, \ldots, x_k)$, denoted by $\mathsf{TIME}_{\mathcal{M},x}$, is the number of time steps that $\mathcal{M}$ takes on input x before it enters its designated halt state; if $\mathcal{M}$ never halts on input x, then we define $\mathsf{TIME}_{\mathcal{M},x} = \infty$.

The finite number of states and the transition rules of a Turing machine $\mathcal{M}$ can be encoded as a bit string $\overline{\mathcal{M}} \in \{0,1\}^*$, called the *description of $\mathcal{M}$*. For every integer $k \in \mathbb{N}$ and every string $\alpha \in \{0,1\}^*$, the k-input Turing machine described by α is denoted $[\alpha]_k$. We assume without loss of generality that for

---

[18]The additional 1 in the argument of the $\log(\cdot)$ is to ensure that this quantity is strictly positive.

all $k \in \mathbb{N}$, every bit string represents some k-input Turing machine and every k-input Turing machine is represented by infinitely many different bit strings.

Throughout the paper we frequently construct Turing machines that run or simulate other Turing machines. Implicitly we assume that this simulation can be done efficiently, as given by the following result [HS66].

Theorem 3.1 (Efficient universal Turing machine). *For all $k \in \mathbb{N}$, there exists a 2-input Turing machine $\mathcal{U}_k$ such that for every $x, \alpha \in \{0,1\}^*$, $\mathcal{U}_k(\alpha, x) = [\alpha]_k(x)$. Moreover, if $[\alpha]_k$ halts on input x in T steps then $\mathcal{U}_k(\alpha, x)$ halts within $CT \log T$ steps, where C is a constant depending only on k and the number of states of $[\alpha]_k$.*

Remark 3.2. *Although the inputs and outputs of a Turing machine are strictly speaking binary strings, we oftentimes slightly abuse notation and specify Turing machines that treat their inputs and outputs as objects with more structure, such as finite field elements, integers, symbols from a larger alphabet, and so on. In this case we implicitly assume that the Turing machine specification uses a consistent convention to represent these structured objects as binary strings. Conventions for objects such as integers are straightforward. For representations of finite field elements, we refer the reader to Section 3.3.2. We also sometimes pass tuples as inputs to a single tape; here again we assume the binary encoding of inputs chosen such that the Turing machine can separate the different components of the tuple: specifically, we may precede each element of the tuple by its length in unary followed by a single "0".*

3.2 Linear spaces

Linear spaces considered in the paper generally take the form $V = \mathbb{F}^n$ for a finite field $\mathbb{F}$ and integer $n \geq 1$. In particular, when we write "let V be a linear space", unless explicitly stated otherwise we always mean a space of the form $\mathbb{F}^n$. Let $E = \{e_1, e_2, \ldots, e_n\}$ denote the standard basis of V, where for $i \in \{1, 2, \ldots, n\}$,

$$e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$$

has a 1 in the i-th coordinate and 0's elsewhere. We write $\mathrm{End}(V)$ to denote the set of linear transformations from V to itself.

Definition 3.3 (Register subspace). A *register subspace S* of V is a subspace that is the span of a subset of the standard basis of V.[19] We often represent such a subspace as an indicator vector $u \in \{0,1\}^s$, where $s = \dim(V)$, such that if $\{e_1, \ldots, e_s\}$ is the standard basis of V then $S = \mathrm{span}\{e_i | u_i = 1\}$.

Definition 3.4. Let $E = \{e_i\}$ be the standard basis of $V = \mathbb{F}^n$. For two vectors $u = \sum_{i=1}^n u_i e_i$, $v = \sum_{i=1}^n v_i e_i$ in V, define the *dot product*

$$u \cdot v = \sum_{i=1}^n u_i v_i \in \mathbb{F} \,.$$

Let S be a subspace of V. The *subspace orthogonal to S in V* is

$$S^\perp = \{u \in V : u \cdot v = 0 \text{ for all } v \in S\} \,.$$

Although the notation $S^\perp$ does not explicitly refer to V, the ambient space will always be clear from context.

---

[19] The use of the term "register" is meant to create an analogy for how the space of multiple qubits is often partitioned into "registers" containing a few qubits each.

We note that over finite fields, the notion of orthogonality does not possess all of the same intuitive properties of orthogonality over fields such as $\mathbb{R}$ or $\mathbb{C}$; for example, a non-zero subspace S may be orthogonal to itself (e.g. $\mathrm{span}\{(1,1)\}$ over $\mathbb{F}_2$). However, the following remains true over all fields.

Lemma 3.5. *Suppose S is a subspace of V. Then*

$$(S^\perp)^\perp = S .$$

Furthermore, $\dim(S) + \dim(S^\perp) = \dim(V)$.

Proof. The "furthermore" part follows from the fact that vectors in $S^\perp$ are the solution to a feasible linear system of equations with $\dim(S)$ linearly independent rows; this implies that the solution space has dimension exactly $\dim(V) - \dim(S)$. Next, we argue that $S \subseteq (S^\perp)^\perp$. Let $u \in S$. Since all vectors $v \in S^\perp$ are orthogonal to every vector in S, in particular u, this implies that $u \in (S^\perp)^\perp$. By dimension counting, it follows that $(S^\perp)^\perp = S$. $\qquad\square$

Definition 3.6. Given a linear space V, two subspaces S and T of V are said to form a pair of *complementary subspaces* of V if

$$S \cap T = \{0\}, \quad S + T = V .$$

In this case, we write $V = S \oplus T$. Any $x \in V$ can be written as $x = x^S + x^T$ for $x^S \in S$ and $x^T \in T$ in a unique way. We refer to x^S (resp. x^T) as the *projection of x onto S parallel to T* (resp. *onto T parallel to S*). We call the unique linear map $L : V \to V$ that maps $x \mapsto x^S$ the *projector onto S parallel to T*.

A given subspace may have many different complementary subspaces: consider the example of $S = \mathrm{span}\{(1,1)\}$ in $\mathbb{F}_2^2$. Different complementary subspaces include $T = \mathrm{span}\{(1,0)\}$ and $T' = \mathrm{span}\{(0,1)\}$. It is convenient to define the notion of a *canonical complement* of a subspace S, given a basis for S.

Definition 3.7. Let E be the standard basis of linear space $V = \mathbb{F}^n$. Let $F = \{v_1, v_2, \ldots, v_m\} \subset V$ be a set of m linearly independent vectors in V. The *canonical complement $F^\perp$ of F* is the set of $n - m$ independent vectors defined as follows. Write $v_i = \sum_{j=1}^n a_{i,j} e_j$. Using a canonical algorithm for Gaussian elimination that works over arbitrary fields, transform the $m \times n$ matrix $(a_{i,j})$ to reduced row echelon form $(b_{i,j})$. Let J be the set of m column indices of the leading 1 entry in each row of $(b_{i,j})$. The canonical complement is defined as $F^\perp = \{e_j : j \notin J\}$.

Remark 3.8. *Let E be the standard basis of V. Suppose subspace S is a register subspace of V spanned by $E_0 \subseteq E$. Then it is not hard to verify that the canonical complement of S is the span of $E \setminus E_0$ and coincides with $S^\perp$.*

Lemma 3.9. *Let S be the span of linearly independent vectors $F = \{v_1, \ldots, v_m\} \subseteq V$ and let $F^\perp$ be the canonical complement of F as defined in Definition 3.7. Let $T = \mathrm{span}(F^\perp)$. Then*

$$S \cap T = \{0\} , \qquad S + T = V .$$

Proof. Let $A = (a_{i,j})$ be the $m \times n$ matrix over $\mathbb{F}$ associated with the v_i as in Definition 3.7. Write $A = UB$ where U is invertible and B is in reduced row echelon form. Let J be as in Definition 3.7. Then the columns of A indexed by J are linearly independent and span $\mathbb{F}^m$. This means that for any vector $u \in V$ there is $v \in S$ such that $u_j = v_j$ for all $j \in J$. Then $u = v + w$ for some $w \in T$. This shows $S + T = V$. Counting dimensions shows that necessarily $S \cap T = \{0\}$. $\qquad\square$

Definition 3.10. Let $F \subseteq V$ be a set of linearly independent vectors. Let $F^\perp$ be the canonical complement of F. Define the *canonical linear map* $L \in \text{End}(V)$ *with kernel basis* F as the projector onto T parallel to S, where $S = \text{span}(F)$ and $T = \text{span}(F^\perp)$. When the basis F for S is clear from context, we refer to this map as the *canonical linear map with kernel* S.

Definition 3.11. Let $L \in \text{End}(V)$ be a linear map, and let F be a basis for $\ker(L)^\perp$. Define $L^\perp : V \to V$ as the canonical linear map with kernel basis F.

Lemma 3.12. *Let* $L \in \text{End}(V)$ *be a linear map and* F *a basis for* $\ker(L)^\perp$. *Let* $L^\perp \in \text{End}(V)$ *be the linear map defined in Definition 3.11. Then* $\ker(L^\perp) = \ker(L)^\perp$.

Proof. Let $F^\perp$ be the canonical complement of F. By definition, $L^\perp$ is the projector onto $\text{span}(F^\perp)$ parallel to $\text{span}(F) = \ker(L)^\perp$. By Lemma 3.9, $\text{span}(F^\perp)$ and $\text{span}(F)$ are complementary subspaces, and the projector onto $\text{span}(F^\perp)$ parallel to $\ker(L)^\perp$ must map all vectors in $\ker(L)^\perp$ to 0. Furthermore, if the projector maps a vector v to 0, it must be that $v \in \ker(L)^\perp$. $\qquad\square$

3.3 Finite fields

Let p be a prime and $q = p^k$ be a prime power. We denote the finite fields of p and q elements by $\mathbb{F}_p$ and $\mathbb{F}_q$ respectively. The prime p is the characteristic of field $\mathbb{F}_q$, and field $\mathbb{F}_p$ is the *prime subfield* of $\mathbb{F}_q$. We sometimes omit the subscript and simply use $\mathbb{F}$ to denote the finite field when the size of the field is implicit from context. For general background on finite fields, and explicit algorithms for elementary arithmetic operations, we refer to [MP13].

3.3.1 Subfields and bases

Let q be a prime power and k an integer. The field $\mathbb{F}_q$ is a subfield of $\mathbb{F}_{q^k}$ and $\mathbb{F}_{q^k}$ is a linear space of dimension k over $\mathbb{F}_q$. Let $\{e_i\}_{i=1}^k$ be a basis of $\mathbb{F}_{q^k}$ as a linear space over $\mathbb{F}_q$. Introduce a bijection $\kappa_q : \mathbb{F}_{q^k} \to \mathbb{F}_q^k$ between $\mathbb{F}_{q^k}$ and $\mathbb{F}_q^k$ defined with respect to the basis $\{e_i\}_{i=1}^k$ by

$$\kappa_q : a \mapsto (a_i)_{i=1}^k$$

where $a = \sum_{i=1}^k a_i e_i$. This map satisfies several nice properties. First, the map is $\mathbb{F}_q$-linear and, in particular, addition in $\mathbb{F}_{q^k}$ naturally corresponds to vector addition in $\mathbb{F}_q^k$. Namely, for all $a, b \in \mathbb{F}_{q^k}$,

$$\kappa_q(a + b) = \kappa_q(a) + \kappa_q(b) \ .$$

Second, multiplication by a field element in $\mathbb{F}_{q^k}$ corresponds to a linear map on $\mathbb{F}_q^k$. For all $a \in \mathbb{F}_{q^k}$, there exists a matrix $K_a \in \text{M}_k(\mathbb{F}_q)$ such that for all $b \in \mathbb{F}_{q^k}$,

$$\kappa_q(ab) = K_a \kappa_q(b) \ .$$

The matrix K_a is called the *multiplication table of* a *with respect to basis* $\{e_i\}_{i=1}^k$.

We extend the map κ_q to vectors, matrices and sets over $\mathbb{F}_{q^k}$. For $v = (v_1, v_2, \ldots, v_n) \in \mathbb{F}_{q^k}^n$, define

$$\kappa_q(v) = \left(\kappa_q(v_i)\right)_{i=1}^n \in \mathbb{F}_q^{kn} \ .$$

25

Similarly, for matrix $M = (M_{i,j}) \in \mathrm{M}_{m,n}(\mathbb{F}_{q^k})$, define

$$\chi_q(M) = (K_{M_{i,j}}) \in \mathrm{M}_{mk,nk}(\mathbb{F}_q) ,$$

the block matrix whose (i,j)-th block is the multiplication table $K_{M_{i,j}}$ of $M_{i,j}$ with respect to basis $\{e_i\}_{i=1}^k$. For a set S of vectors in $\mathbb{F}_{q^k}^n$, define

$$\kappa_q(S) = \{\kappa_q(v) : v \in S\} .$$

We omit the subscript and write κ and χ for κ_q and χ_q respectively when q equals to p, the characteristic of the field.

The *trace of* $\mathbb{F}_{q^k}$ *over* $\mathbb{F}_q$ is defined as

$$\mathrm{tr}_{q^k \to q} : a \mapsto \mathrm{Tr}(K_a) \tag{10}$$

for $a \in \mathbb{F}_{q^k}$, where $\mathrm{Tr}(K_a)$ is the trace of the multiplication table of a with respect to the basis $\{e_i\}$. By definition, the trace is an $\mathbb{F}_q$-linear map from $\mathbb{F}_{q^k}$ to $\mathbb{F}_q$. An equivalent definition of the trace is

$$\mathrm{tr}_{q^k \to q}(a) = \sum_{j=0}^{k-1} a^{q^j} .$$

A *dual basis* $\{e_1', e_2', \ldots, e_k'\}$ of $\{e_1, e_2, \ldots, e_k\}$ is a basis such that $\mathrm{tr}_{q^k \to q}(e_i e_j') = \delta_{i,j}$ for all $i, j \in \{1, 2, \ldots, k\}$. A *self-dual basis* is one that is equal to its dual. If for some $\alpha \in \mathbb{F}_{q^k}$ the set $\{\alpha^{q^j}\}_{j=0}^{k-1}$ forms a basis of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$, the basis is called a *normal basis*.

We record some convenient facts about the maps $\kappa(\cdot)$ and $\chi(\cdot)$ for self-dual bases.

Lemma 3.13. *Let q be a prime power, k an integer and $\{e_i\}$ a self-dual basis for $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$. The map $\kappa_q(\cdot)$ corresponding to $\{e_i\}$ satisfies the following properties:*

1. *For all $x \in \mathbb{F}_{q^k}$, $\kappa_q(x) = (\mathrm{tr}_{q^k \to q}(xe_1), \ldots, \mathrm{tr}_{q^k \to q}(xe_k))$.*

2. *For all $x, y \in \mathbb{F}_{q^k}$, $\mathrm{tr}_{q^k \to q}(xy) = \kappa_q(x) \cdot \kappa_q(y)$.*

3. *For all $M \in \mathrm{M}_{m,n}(\mathbb{F}_{q^k})$ and $v \in \mathbb{F}_{q^k}^n$, we have $\chi_q(M)\kappa_q(v) = \kappa_q(Mv)$.*

Proof. The properties follow from the definition of the map $\kappa_q(\cdot)$ and the fact that $\{e_i\}$ is a self-dual basis. $\square$

For $z \in \mathbb{F}^n$ and V, W a pair of complementary subspaces, recall from Definition 3.6 the notation z^V for the projection of z onto V and parallel to W.

Lemma 3.14. *Let $\kappa_q(\cdot)$ denote the map corresponding to a self-dual basis $\{e_i\}$ for $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$. Let V be a subspace of $\mathbb{F}_{q^k}^n$ with linearly independent basis $\{b_1, \ldots, b_t\} \subseteq \mathbb{F}_{q^k}^n$. Then the following hold:*

1. *$\kappa_q(V)$ is a subspace of $\mathbb{F}_q^{nk}$.*

2. *$\{\kappa_q(e_i b_j)\}_{i,j}$ is a linearly independent basis of $\kappa_q(V)$ over $\mathbb{F}_q$.*

3. Let V, W be complementary subspaces of $\mathbb{F}_{q^k}^n$. Then $V' = \kappa_q(V)$ and $W' = \kappa_q(W)$ are complementary subspaces of $\mathbb{F}_q^{kn}$, and furthermore for all vectors $z \in \mathbb{F}_{q^k}^n$, we have $\kappa_q(z^V) = \kappa_q(z)^{V'}$ and $\kappa_q(z^W) = \kappa_q(z)^{W'}$.

Proof. For the first item, we first verify that $\kappa_q(V)$ is a subspace. Since V is a subspace, it contains $0 \in \mathbb{F}_{q^k}^n$, and therefore $\kappa_q(0) = 0$ is also in $\kappa_q(V)$. Let $u', v' \in \kappa_q(V)$. Using that κ_q is a bijection there exist $u, v \in \mathbb{F}_{q^k}^n$ such that $u' = \kappa_q(u)$ and $v' = \kappa_q(v)$. Therefore

$$u' + v' = \kappa_q(u) + \kappa_q(v) = \kappa_q(u + v) \in \kappa_q(V) ,$$

where the inclusion follows because V is a subspace and thus contains $u + v$. Finally, for all $x' \in \mathbb{F}_q$, for all $v \in V$, we have that $x' \kappa_q(v) = \kappa_q(x'v) \in \kappa_q(V)$ where we used that V is closed under scalar multiplication by $\mathbb{F}_{q^k}$ and thus by $\mathbb{F}_q$ (since $\mathbb{F}_q$ is a subfield of $\mathbb{F}_{q^k}$). Thus $\kappa_q(V)$ is closed under scalar multiplication by $\mathbb{F}_q$.

For the second item, note that an element $v \in V$ can be expressed uniquely as $v = \sum_{i=1}^t v_i b_i$ for $v_i \in \mathbb{F}_{q^k}$. The element v_i can further be written as $\sum_j v_{i,j} e_j$ where $v_{i,j} \in \mathbb{F}_q$. Thus v is a linear combination of the vectors $\{e_j b_i\}$, and therefore $\kappa_q(v)$ is a linear combination of the vectors $\{\kappa_q(e_j b_i)\}$. To establish that the vectors $\{\kappa_q(e_j b_i)\}$ are linearly independent, suppose towards contradiction that they are not. Then there would exist $\alpha_{i,j} \in \mathbb{F}_q$ such that at least one $\alpha_{i,j}$ is nonzero and

$$0 = \sum_{i,j} \alpha_{i,j} \kappa_q(e_j b_i)$$
$$= \kappa_q \left(\sum_i \left(\sum_j \alpha_{i,j} e_j \right) b_i \right)$$
$$= \kappa_q \left(\sum_i \beta_i b_i \right) ,$$

where we define $\beta_i = \sum_j \alpha_{i,j} e_j$. Since at least one $\alpha_{i,j} \neq 0$ and the $\{e_j\}$ are linearly independent over $\mathbb{F}_q$, there exists i such that $\beta_i \neq 0$, which means that there is a non-trivial linear combination of the basis elements b_i that equals 0 under $\kappa_q(\cdot)$. Since $\kappa_q(\cdot)$ is injective, we get a contradiction with linear independence of the $\{b_i\}$.

For the third item, we observe that $\kappa_q(V)$ and $\kappa_q(W)$ must be complementary because $\kappa_q(\cdot)$ is a linear map as well as a bijection. Let $\{v_1, \ldots, v_m\}$ and $\{v_{m+1}, \ldots, v_n\}$ denote bases for V and W, respectively. Thus the set $\{v_1, \ldots, v_n\}$ forms a basis for $\mathbb{F}_{q^k}^n$, and from the previous item, the set $\{\kappa_q(e_j v_i)\}_{i,j}$ is a basis for $\mathbb{F}_q^{kn}$. Furthermore, the sets $\{\kappa_q(e_j v_i)\}_{j, i=1,\ldots m}$ and $\{\kappa_q(e_j v_i)\}_{j, i=m+1,\ldots n}$ are bases for $\kappa_q(V)$ and $\kappa_q(W)$, respectively.

There is a unique choice of coefficients $\alpha_{i,j} \in \mathbb{F}_q$ such that $\kappa_q(z) = \sum_{i,j} \alpha_{i,j} \kappa_q(e_j v_i)$. But then

$$\kappa_q(z) = \kappa_q \left(\sum_i \left(\sum_j \alpha_{i,j} e_j \right) v_i \right)$$
$$= \kappa_q \left(\sum_i \alpha_i v_i \right) ,$$

where we define $\alpha_i = \sum_j \alpha_{i,j} e_j$. Since $\kappa_q(\cdot)$ is a bijection, this implies that $z = \sum_i \alpha_i v_i$, and therefore $z^V = \sum_{i=1}^m \alpha_i v_i$ (and similarly $z^W = \sum_{i=m+1}^n \alpha_i v_i$). This implies that

$$\kappa_q(z)^{V'} = \sum_{i=1}^m \sum_j \alpha_{i,j} \kappa_q(e_j v_i) = \kappa_q(z^V) \,,$$

and similarly $\kappa_q(z)^{W'} = \kappa_q(z^W)$. This completes the proof of the lemma. $\qquad\square$

3.3.2 Bit string representations

As mentioned in Remark 3.2, we sometimes treat the inputs and outputs of Turing machines as representing elements of a finite field, or a vector space over a finite field. We discuss some important details about bit representations of finite field elements and arithmetic over finite fields.

In the paper we only consider fields $\mathbb{F}_{2^k}$ where k is odd.

Definition 3.15. A field size q is called an *admissible* field size if $q = 2^k$ for odd k.

Elements of $\mathbb{F}_2$ are naturally represented using bits. To represent elements of $\mathbb{F}_{2^k}$ as binary strings we require the specification of a basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$. Given a basis $\{e_i\}_{i=1}^k$ of $\mathbb{F}_{2^k}$, every element $a \in \mathbb{F}_{2^k}$ has a unique expansion $a = \sum_{i=1}^k a_i e_i$ and can be represented as the k-bit string corresponding to $\kappa(a) \in \mathbb{F}_2^k$. Note that we omitted the subscript 2 of κ as it maps to the linear space over the prime subfield $\mathbb{F}_2$. Thus the *binary representation* of $a \in \mathbb{F}_{2^k}$ is defined as the natural binary representation of $\kappa(a) \in \mathbb{F}_2^k$ (which in turn is the $\mathbb{F}_2$-*representation of a*). Throughout the paper we freely associate between the binary representation of a field element $a \in \mathbb{F}_{2^k}$ and its $\mathbb{F}_2$-representation, although—technically speaking—these are distinct objects.

Given the representations $\kappa(a), \kappa(b)$ of $a, b \in \mathbb{F}_{2^k}$, to compute the binary representation of $a + b$ it suffices to compute the addition bit-wise, modulo 2. Computing the multiplication of elements a, b requires the specification of the multiplication tables $\{K_{e_i} \in M_k(\mathbb{F}_2)\}_{i=1}^k$ for the basis $\{e_i\}$. Given representations $\kappa(a) = (a_i)_{i=1}^k$, $\kappa(b) = (b_i)_{i=1}^k$ for $a, b \in \mathbb{F}_{2^k}$ respectively, the representation $\kappa(ab)$ of the product ab is computed as

$$\kappa(ab) = \sum_{i=1}^k a_i \, \kappa(e_i b) = \sum_{i=1}^k a_i \big(K_{e_i} \kappa(b) \big) \,. \tag{11}$$

Thus, using our representation for field elements, efficiently performing finite field arithmetic in $\mathbb{F}_{2^k}$ reduces to having access to the multiplication table of some basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$.

The following fact provides an efficient deterministic algorithm for computing a self-dual normal basis for $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$ and the corresponding multiplication tables for any odd k.

Lemma 3.16. *There exists a deterministic algorithm that given an odd integer $k > 0$, outputs a self-dual normal basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$ and the multiplication tables of the basis in $\mathrm{poly}(k)$ time.*

Proof. The algorithm of Shoup [Sho90, Theorem 3.2] shows that for prime p, an irreducible polynomial in $\mathbb{F}_p[X]$ of degree k can be computed in time $\mathrm{poly}(p, k)$. Then, the algorithm of Lenstra [LJ91, Theorem 1.1] shows that given such an irreducible polynomial, the multiplication table of a normal basis of $\mathbb{F}_{p^k}$ over $\mathbb{F}_p$ can be computed in $\mathrm{poly}(k, \log p)$ time. Finally, the algorithm of Wang [Wan89] shows that for odd k and a multiplication table K of a normal basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$, a multiplication table K' for a self-dual normal basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$ can be computed in $\mathrm{poly}(k)$ time. Putting these three algorithms together yields the claimed statement. $\qquad\square$

Lemma 3.17. *Let k be an odd integer and $\{e_i\}_{i=1}^k$ be a self-dual normal basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$. Then $\mathrm{tr}(e_i) = 1$ for all i, and furthermore the representation $\kappa(1)$ of the unit $1 \in \mathbb{F}_{2^k}$ is the all ones vector in $\mathbb{F}_2^k$.*

Proof. Since $\{e_i\}$ is a normal basis, $e_i = \alpha^{2^i}$ for some element $\alpha \in \mathbb{F}_{2^k}$. Furthermore, for every element $b \in \mathbb{F}_{2^k}$, we have that $\mathrm{tr}(b^2) = \mathrm{tr}(b)$. This is because

$$\mathrm{tr}(b^2) = \sum_{i=0}^{k-1} b^{2^{i+1}} = \sum_{i=0}^{k-1} b^{2^i} = \mathrm{tr}(b) \,,$$

where we use that $b^{2^k} = b$ for all $b \in \mathbb{F}_{2^k}$. Since $e_{i+1} = e_i^2$, we get that $\mathrm{tr}(e_i) = \mathrm{tr}(e_j)$ for all i, j. It cannot be the case that $\mathrm{tr}(e_i) = 0$ for all i. Suppose that this were the case. This would imply that $\mathrm{tr}(b) = 0$ for all $b \in \mathbb{F}_{2^k}$. But then for all $j \in \{1, \ldots, k\}$ and for some $b \neq 0$, we would also have that $b_j = \mathrm{tr}(be_j) = 0$ where $b = \sum_j b_j e_j$ with $b_j \in \mathbb{F}_2$. This implies that b is the all zero element of $\mathbb{F}_{2^k}$, which is a contradiction. Thus $\mathrm{tr}(e_i) = 1$ for all $i = 1, 2, \ldots, k$.

The "furthermore" part follows from the expansion

$$1 = \sum_{i=1}^k \mathrm{tr}(1 \cdot e_i) \, e_i = \sum_{i=1}^k e_i \,. \qquad \square$$

Lemma 3.18. *For any odd integer k, let $\{e_i\}_{i=1}^k$ denote the self-dual normal basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$ that is returned by the algorithm specified in Lemma 3.16 on input k. Then the following can be computed in time $\mathrm{poly}(k)$ on input k:*

1. *The representation $\kappa(a + b)$ of the sum $a + b$ given the representations $\kappa(a)$ and $\kappa(b)$ of $a, b \in \mathbb{F}_{2^k}$.*

2. *The representation $\kappa(ab)$ of the product ab given the representations $\kappa(a)$ and $\kappa(b)$ of $a, b \in \mathbb{F}_{2^k}$.*

3. *The multiplication table $K_a \in \mathrm{M}_k(\mathbb{F}_2)$ given the representation $\kappa(a)$ of $a \in \mathbb{F}_{2^k}$.*

4. *The representation $\kappa(a^{-1})$ of the multiplicative inverse of $a \in \mathbb{F}_{2^k}$, given the representation $\kappa(a)$.*

5. *The trace $\mathrm{tr}(a)$ given the multiplication table K_a of $a \in \mathbb{F}_{2^k}$.*

Furthermore, for all integers n, the representations of projections $\kappa(x^S)$ and $\kappa(x^T)$ of $x \in \mathbb{F}_{2^k}^n$ for complementary subspaces S, T of $\mathbb{F}_{2^k}^n$ can be computed in $\mathrm{poly}(k, n)$ time, given the representations $\kappa(x)$, $\{\kappa(v_1), \kappa(v_2), \ldots, \kappa(v_m)\}$ and $\{\kappa(w_1), \kappa(w_2), \ldots, \kappa(w_{n-m})\}$ where $\{v_i\}$ and $\{w_j\}$ are bases for S and T respectively.

Proof. Given an odd integer k as input, by Lemma 3.16 it is possible to compute the self-dual basis $\{e_i\}_{i=1}^k$ together with the multiplication tables K_{e_i} for $i = 1, 2, \ldots, k$. Addition is performed component-wise, and multiplication is done using Eq. (11). For the multiplication table K_a it suffices to compute the k products $\kappa(ae_i)$ for $i \in \{1, \ldots, k\}$. To compute inverses, observe that $\kappa(1) = \kappa(aa^{-1}) = K_a \kappa(a^{-1})$. The matrices K_a are invertible over $\mathbb{F}_2$, so therefore $\kappa(a^{-1}) = K_a^{-1} \kappa(1)$; moreover, $\kappa(1)$ can be computed by Lemma 3.17. Inverting the matrix can be done in $\mathrm{poly}(k)$ time via Gaussian elimination. The trace of an element $a \in \mathbb{F}_{2^k}$ is by definition the trace of the multiplication table K_a.

For the "Furthermore" part, we observe that since $\{v_1, v_2, \ldots, v_m\} \cup \{w_1, w_2, \ldots, w_{n-m}\}$ forms a basis for $\mathbb{F}_{2^k}^n$, there is a unique way to write x as a $\mathbb{F}_{2^k}$ linear combination of $\{v_i\}$ and $\{w_j\}$. Via Gaussian elimination over $\mathbb{F}_{2^k}$, the $\mathbb{F}_2$-representation of the coefficients of this linear combination can be computed in $\mathrm{poly}(n, k)$ time. Here we use that addition, multiplication and division over $\mathbb{F}_{2^k}$ can be performed in time $\mathrm{poly}(k)$ using the previous items of the Lemma. $\qquad \square$

Remark 3.19. *Throughout this paper, whenever we refer to Turing machines that perform computations with elements of a field $\mathbb{F}_q$ for an admissible field size $q = 2^k$, we mean that that the Turing machines are representing elements of $\mathbb{F}_q$ as vectors in $\{0,1\}^k$ using the basis specified by the algorithm of Lemma 3.16 and performing arithmetic as described in Lemma 3.18.*

3.4 Low-degree encoding

In this section, we introduce a standard error-correcting code in the literature known as the *low-degree code*. Given a finite set S and a "message" string $a \in \mathbb{F}_q^S$, the low-degree code encodes a as a low-degree multivariate polynomial $g : \mathbb{F}_q^m \to \mathbb{F}_q$. This polynomial is constructed so that the coordinates of a are directly embedded into specific coordinates of g, meaning that for each $s \in S$, there is a $\pi(s) \in \mathbb{F}_q^m$ such that $g(s) = a_s$. As a result, g can be defined by polynomial interpolation though the points $\{\pi(s)\}_{s \in S}$. In general, polynomial interpolation can produce polynomials with high degree, and so to ensure that g is low-degree, we restrict $\pi(\cdot)$ so that it only ever maps elements $s \in S$ to elements of the set H^m, where H is a subset of $\mathbb{F}_q$ generally selected to have size much smaller than q. This is an error-correcting code because two different strings $a, a' \in \mathbb{F}_q^S$ will be mapped to two different low-degree polynomials g, g', and by the Schwartz-Zippel lemma (Lemma 3.20 below), two non-equal low-degree polynomials will disagree on most points in their domain. In our application, we will always take S to be either $S = \{0, \ldots, n-1\}$ or $S = \{1, \ldots, n\}$, for some integer n.

Let $h, m \geq 0$ be integers, and let $q = 2^k$ be a power of 2 such that $h \leq q$. Let H be a subset of $\mathbb{F}_q$ of size h. Given a point $x \in H^m$, we define the *indicator polynomial* $\text{ind}_{H,m,x} : \mathbb{F}_q^m \to \mathbb{F}_q$ as follows:

$$\text{ind}_{H,m,x}(y) = \frac{\prod_{i=1}^m \prod_{a \in H : a \neq x_i}(y_i - a)}{\prod_{i=1}^m \prod_{a \in H : a \neq x_i}(x_i - a)} \, .$$

This is a degree-$m(h-1)$ polynomial and has the property that for any $y \in H^m$, $\text{ind}_{H,m,x}(y) = 1$ if $y = x$ and 0 otherwise.

Let S be a finite set such that $h^m \geq |S|$, and let $\pi : S \to H^m$ be an injection. We define the function $\text{ind}_{H,m,\pi} : \mathbb{F}_q^m \to \mathbb{F}_q^S$ as follows: given $y \in \mathbb{F}_q^m$, $z = \text{ind}_{H,m,\pi}(y)$ is the element of $\mathbb{F}_q^S$ such that for each $s \in S$,

$$z_s = \text{ind}_{H,m,\pi(s)}(y) \, .$$

Supposing that $y \in H^m$, if $y = \pi(s)$ for some $s \in S$, then z is equal to e_s, the standard basis vector corresponding to s, and otherwise $z = 0$.

Let a be a point in $\mathbb{F}_q^S$. Then the *low-degree encoding of a* is the polynomial $g_{a,\pi} : \mathbb{F}_q^m \to \mathbb{F}_q$ defined as

$$g_{a,\pi}(x) = a \cdot \text{ind}_{H,m,\pi}(x) = \sum_{s \in S} a_s \cdot \text{ind}_{H,m,\pi(s)}(x) \, . \tag{12}$$

This is a degree-$m(h-1)$ polynomial and has the property that for any $s \in S$,

$$g_{a,\pi}(\pi(s)) = a_s \, .$$

In addition, we define the *low-degree decoding* Dec_π to be the function which maps functions $g : \mathbb{F}_q^m \to \mathbb{F}_q$ to strings $a \in \mathbb{F}_q^S$ defined as follows: $a = \text{Dec}_\pi(g)$ is the string in $\mathbb{F}_q^S$ such that for each $s \in S$,

$$a_s = g(\pi(s)) \, .$$

By construction, $\text{Dec}_\pi(g_{a,\pi}) = a$.

The set of low-degree encodings of strings $a \in \mathbb{F}_q^S$ forms an error-correcting code known as the *low-degree code*. The following lemma gives a lower bound on the distance of this code.

Lemma 3.20 (Schwartz-Zippel lemma [Sch80, Zip79]). *Let* $f, g : \mathbb{F}_q^m \to \mathbb{F}_q$ *be two unequal degree-d polynomials. Then*

$$\Pr_{x \sim \mathbb{F}_q^m}[f(x) = g(x)] \leq d/q .$$

3.4.1 A canonical injection

The low-degree code affords us with the ability to pick a variety of parameters. In this section, we will describe a canonical way to choose the parameters h, H, and π once the parameters m, $q = 2^k$, and S have been chosen, where $S = \{0, \ldots, n-1\}$ for some integer n. This is similar to the canonical choice of parameters in [NW19, Section 3.4].

Let k be an odd integer and $\mathrm{basis}(k) = \{e_i\}_{i=1}^k$ be the self-dual normal basis of $\mathbb{F}_{2^k}$ over $\mathbb{F}_2$ produced by the algorithm from Lemma 3.16 on input k. In addition, for $\ell \in \{0, \ldots, k\}$, we write $H(k, \ell)$ for the subspace of $\mathbb{F}_{2^k}$ spanned by $\{e_i\}_{i=1}^\ell$ over $\mathbb{F}_2$, i.e.

$$H(k, \ell) = \{x_1 e_1 + x_2 e_2 + \cdots + x_\ell e_\ell \mid x_i \in \mathbb{F}_2\} .$$

Definition 3.21 (Binary representation). Given an integer n and another integer c between 0 and $2^n - 1$, we write $\mathrm{binary}_n(c)$ for the n-digit binary representation of c. In addition, given a string $x \in \{0,1\}^n$, we write $\mathrm{number}_n(x)$ for the integer between 0 and $2^n - 1$ encoded by x. As a result, binary_n and number_n are each others' inverses.

Definition 3.22 (Canonical injection). Let m be an integer and $q = 2^k$ be a power of 2 for odd k. Let n be an integer such that $n \leq q^m$. Let

$$b := b(n) = \begin{cases} 0 & \text{if } n = 1, \\ \lfloor \log_2(n-1) \rfloor + 1 & \text{otherwise,} \end{cases}$$

and let $\ell = \ell(n, m) = \lceil b(n)/m \rceil$. The *canonical subspace* is defined to be the set $H = H_{\mathrm{canon},m,k,n} := H(k, \ell)$. It has size $h_{\mathrm{canon},m,k,n} := 2^\ell$. Next, define the function $\mathrm{coeff} : \{0,1\}^{m\ell} \to H^m$ given by

$$\mathrm{coeff}(a_1, \ldots, a_{m\ell}) = (\kappa^{-1}(a_1, \ldots, a_\ell), \kappa^{-1}(a_{\ell+1}, \ldots, a_{2\ell}), \ldots, \kappa^{-1}(a_{(m-1)\cdot\ell+1}, \ldots, a_{m\cdot\ell}))$$

$$= (a_1 e_1 + \cdots + a_\ell e_\ell, a_{\ell+1} e_1 + \cdots + a_{2\ell} e_\ell, \ldots, a_{(m-1)\cdot\ell+1} e_1 + \cdots + a_{m\cdot\ell} e_\ell) .$$

Then the *canonical injection* is the map $\pi = \pi_{\mathrm{canon},m,k,n} : \{0, 1, \ldots, n-1\} \to H^m$ given by

$$\pi(c) := \mathrm{coeff}(\mathrm{binary}_{m\ell}(c)) .$$

We note that it is a bijection when $n = 2^{s \cdot m}$ for some integer $s \in \{0, \ldots, k\}$.

Now we observe that the canonical injection can be computed efficiently.

Lemma 3.23 (Runtime of the canonical injection). *Let m be an integer and k be an odd integer, and let $q := 2^k$. Let n be an integer such that $n \leq q^m$, and set $\pi := \pi_{\mathrm{canon},m,k,n}$. Then the following can be computed in time* $\mathrm{poly}(m, k)$.

1. *The representation $\kappa(\pi(c))$ given m, k, and a number $c \in \{0, 1, \ldots, n-1\}$.*

2. *The inverse $\pi^{-1}(a)$ given m, k, and the representation $\kappa(a)$ of $a \in H_{\mathrm{canon},m,k,n}^m$.*

Proof. The key step in computing π is computing $\mathrm{coeff}(a_1,\ldots,a_{m\ell})$. This step involves first rearranging the a_i's, a $\mathrm{poly}(m,k)$-time task, and then applying the κ^{-1} map, which is trivial, as elements of $\mathbb{F}_q$ are already represented on a Turing machine via their coefficients in the basis $\mathrm{basis}(k)$. The remaining tasks are computing $b(n)$, $\ell(n,m)$, and subtracting 1 from c, and these are all $\mathrm{poly}(m,k)$-time tasks. Computing the inverse follows similarly. $\qquad\square$

Having defined the canonical injection, we can now define the canonical low-degree encoding of a string.

Definition 3.24 (Canonical low-degree encoding)**.** Let m be an integer and k be an odd integer, and let $q = 2^k$. Let n be an integer such that $n \leq q^m$. Let $a \in \mathbb{F}_q^S$ for $S = \{0,\ldots,n-1\}$. Then the *canonical low-degree encoding of a* is the polynomial $g_{\mathrm{canon},a,m,k,n} : \mathbb{F}_q^m \to \mathbb{F}_q$ equal to $g_{a,\pi_{\mathrm{canon},m,k,n}}$.

We will need a further, technical property of the canonical injection, which is that its inverse can be computed by a low-degree polynomial. This is provided by the following definition.

Definition 3.25 (Canonical semi-inverse)**.** Let m be an integer and k be an odd integer, and let $q := 2^k$. Let n be an integer such that $n \leq q^m$, and set $b := b(n)$ and $\ell := \ell(n,m)$. Let $\{e_i\}_{i=1}^k = \mathrm{basis}(k)$ and $H = H_{\mathrm{canon},m,k,n}$. For each $i \in \{1,\ldots,\ell\}$, define the function $\iota : \mathbb{F}_q \to \mathbb{F}_q^\ell$ whose i-th coordinate is given by

$$\iota_i(y) = \sum_{x \in H \,:\, \mathrm{tr}(e_i \cdot x)=1} \mathrm{ind}_{H,1,x}(y).$$

Next, define the function $\gamma : \mathbb{F}_q^m \to \mathbb{F}_q^{m\ell}$ given by $\gamma(x_1,\ldots,x_m) = (\iota(x_1),\ldots,\iota(x_m))$. Then the *canonical semi-inverse* is the function $v_{\mathrm{canon},m,k,n} : \mathbb{F}_q^m \to \mathbb{F}_q^b$ given by

$$v_{\mathrm{canon},m,k,n}(x) := (\gamma_{mk-b+1}(x),\ldots,\gamma_{mk}(x)) \,.$$

(Here, we write $\gamma_i : \mathbb{F}_q^m \to \mathbb{F}_q$ for the i-th component of γ.)

The canonical injection $\pi = \pi_{\mathrm{canon},m,k,n}$ maps integers from $\{0,1,\ldots,n-1\}$ into the set $H^m \subseteq \mathbb{F}_q^m$, where $H = H_{\mathrm{canon},m,k,n}$. The purpose of the canonical semi-inverse $v = v_{\mathrm{canon},m,k,n}$ is to invert this map: given $x \in \mathbb{F}_q^m$, if $x = \pi(c)$ for some $c \in \{0,1,\ldots,n-1\}$, then $v(x)$ is the $(b = b(n))$-digit binary representation of c, i.e. $\mathrm{binary}_b(c)$, as an element of $\mathbb{F}_q^b$. Otherwise, $v(x)$ is some element of $\mathbb{F}_q^b$, though we won't care which one. The crucial property we need is that v can achieve these properties while still being a low-degree multivariate polynomial. This is formalized in the following proposition.

Proposition 3.26 (Properties of the canonical semi-inverse)**.** *Let m be an integer and k be an odd integer, and let $q := 2^k$. Let n be an integer such that $n \leq q^m$, let $b = b(n)$, and let $h := h_{\mathrm{canon},m,k,n}$. Then the following statements are true.*

1. *Each coordinate of $v := v_{\mathrm{canon},m,k,n}$ is computed a degree-$(h-1)$ polynomial.*

2. *Writing $\pi := \pi_{\mathrm{canon},m,k,n}$, we have that for any $c \in \{0,1,\ldots,n-1\}$, $v(\pi(c)) = \mathrm{binary}_b(c)$.*

3. *For any $x \in \mathbb{F}_q^m$, the value $v(x)$ can be computed in time $\mathrm{poly}(m,h,k)$.*

Proof. The first item follows from the fact that each coordinate of v is expressed as a sum of polynomials of the form $\mathrm{ind}_{H,x}(y)$, which are degree-$(h-1)$ by definition. As for the second, let $y = b_1 e_1 + \cdots + b_\ell e_\ell \in H$. Then for each $i \in [\ell]$,

$$\iota_i(y) = \sum_{x \in H : \mathrm{tr}(e_i \cdot x)=1} \mathrm{ind}_{H,1,x}(y) = \sum_{x \in H : \mathrm{tr}(e_i \cdot x)=1} \mathbf{1}[x = y] = b_i.$$

32

As a result, $\iota(b) = (b_1, \ldots, b_\ell)$. Let $c \in [n]$, and write $a = \text{binary}_{m\ell}(c)$. Then

$$
\begin{aligned}
\gamma(\pi(c)) = \gamma(\text{coeff}(a)) &= \gamma(\kappa^{-1}(a_1, \ldots, a_\ell), \ldots, \kappa^{-1}(a_{(m-1)\cdot\ell+1}, \ldots, a_{m\cdot\ell})) \\
&= (\iota(\kappa^{-1}(a_1, \ldots, a_\ell)), \ldots, \iota(\kappa^{-1}(a_{(m-1)\cdot\ell+1}, \ldots, a_{m\cdot\ell}))) \\
&= (a_1, \ldots, a_\ell, \ldots, a_{(m-1)\cdot\ell+1}, \ldots, a_{m\cdot\ell}) = a = \text{binary}_{m\ell}(c).
\end{aligned}
$$

The claim follows, because ν simply truncates γ to its last b binary digits, producing $\text{binary}_b(c)$.

Computing $\text{ind}_{H,1,x}(y)$ requires taking $O(h)$ sums and products in $\mathbb{F}_q$, each of which takes time $\text{poly}(k)$, for a total cost of $O(h) \cdot \text{poly}(k)$. Computing $\iota_i(y)$ requires computing $O(h)$ $\text{ind}_{H,1,x}(y)$'s and summing them together, which takes a total of $O(h^2) \cdot \text{poly}(k)$ time. Computing $\iota(y)$ requires computing ℓ $\iota_i(y)$'s, and computing $\nu(c)$ requires computing $O(m)$ $\iota(y)$'s, for a total time complexity of $O(m \cdot \ell \cdot h^2 \cdot \text{poly}(k)) = \text{poly}(m, h, k)$. $\qquad\square$

3.5 Linear spaces and registers

For a set V, we write $\mathbb{C}^V$ for the complex vector space of dimension $|V|$. The space $\mathbb{C}^V$ is endowed with a canonical orthonormal basis $\{|x\rangle\}_{x \in V}$. By "a quantum state on V" we mean a unit vector

$$
|\psi\rangle_V \in \mathbb{C}^V .
$$

If $V = \bigoplus_{i=1}^k V_k$ is the direct sum of subspaces V_k over $\mathbb{F}$, then $\mathbb{C}^V$ can be identified with $\bigotimes_{i=1}^k \mathbb{C}^{V_i}$. As a special case, if $\{e_i\}$ is a basis of V the decomposition $V = \bigoplus_{i=1}^k (\mathbb{F}e_i)$ yields the tensor product decomposition $\mathbb{C}^V = \bigotimes_{i=1}^k \mathbb{C}^{|\mathbb{F}|}$. We sometimes refer to the spaces $\mathbb{C}^{|\mathbb{F}|}$ as the "qudits" of $\mathbb{C}^V$ (or of a state on it).

Definition 3.27. For linear space V over finite field $\mathbb{F}$, define the EPR state on $\mathbb{C}^V \otimes \mathbb{C}^V$ by

$$
|\text{EPR}\rangle_V = \frac{1}{\sqrt{|V|}} \sum_{x \in V} |x\rangle \otimes |x\rangle .
$$

We also write $|\text{EPR}\rangle_{\mathbb{F}_q}$ as $|\text{EPR}_q\rangle$ and $|\text{EPR}_2\rangle$ as $|\text{EPR}\rangle$.

3.6 Measurements and observables

Quantum measurements are modeled as positive operator-valued measures (POVMs). A POVM consists of a set of positive semidefinite operators $\{M_a\}_{a \in S}$ indexed by outcomes $a \in S$ that satisfy the condition $\sum_a M_a = I$. We sometimes use the same letter M to refer to the collection of operators defining the POVM. The probability that the measurement returns outcome a on state $|\psi\rangle$ is given by

$$
\Pr(a) = \langle \psi | M_a | \psi \rangle .
$$

A POVM $M = \{M_a\}$ is said to be *projective* if each operator M_a is a projector ($M_a^2 = M_a$). An *observable* is a unitary matrix. A *binary observable* is an observable O such that $O^2 = I$, i.e. O has eigenvalues in $\{-1, 1\}$.

We follow the convention that subscripts of the measurement index the outcome and superscripts of the measurement are used to index different measurements. For example, we use $\{M_{a,b}^x\}$ to represent a

measurement indexed by x whose outcome consists of two parts a and b. In this case, by slightly abusing the notation, we use $\{M_a^x\}$ and $\{M_b^x\}$ to denote

$$M_a^x = \sum_b M_{a,b}^x, \quad M_b^x = \sum_a M_{a,b}^x.$$

For any x, $\{M_a^x\}$ and $\{M_b^x\}$ are POVMs sometimes referred to as the "marginals" of $\{M_{a,b}^x\}$.

Definition 3.28. Let $\{M_a^x\}_{a \in A}$ be a family of POVMs indexed by $x \in \mathcal{X}$. Let $f : A \to B$ be an arbitrary function. We write $\{M_{[f(\cdot)=b]}^x\}$ for the POVM derived from $\{M_a^x\}$ by applying the function f before returning the outcome. More precisely,

$$M_{[f(\cdot)=b]}^x = \sum_{a:f(a)=b} M_a^x.$$

If b is not in the image of f, then we define $M_{[f(\cdot)=b]}^x$ to be 0.

3.7 Generalized Pauli observables

For prime number p, the generalized Pauli operators over $\mathbb{F}_p$ are a collection of observables indexed by a basis setting X or Z and an element a or b of $\mathbb{F}_p$, with eigenvalues that are p-th roots of unity. They are given by

$$\sigma^X(a) = \sum_{j \in \mathbb{F}_p} |j + a\rangle\langle j| \quad \text{and} \quad \sigma^Z(b) = \sum_{j \in \mathbb{F}_p} \omega^{bj} |j\rangle\langle j|, \tag{13}$$

where $\omega = e^{\frac{2\pi i}{p}}$, and addition and multiplication are over $\mathbb{F}_p$. These observables obey the "twisted commutation" relations

$$\forall a, b \in \mathbb{F}_p, \quad \sigma^X(a)\,\sigma^Z(b) = \omega^{-ab}\,\sigma^Z(b)\,\sigma^X(a). \tag{14}$$

Similarly, over a field $\mathbb{F}_q$ we can consider a set of generalized Pauli operators, indexed by a basis setting X or Z and an element of $\mathbb{F}_q$ and with eigenvalues that are p-th roots of unity. For $a, b \in \mathbb{F}_q$ they are given by

$$\tau^X(a) = \sum_{j \in \mathbb{F}_q} |j + a\rangle\langle j| \quad \text{and} \quad \tau^Z(b) = \sum_{j \in \mathbb{F}_q} \omega^{\mathrm{tr}(bj)} |j\rangle\langle j|,$$

where addition and multiplication are over $\mathbb{F}_q$. For all $W \in \{X, Z\}$, $a \in \mathbb{F}_q$, and $b \in \mathbb{F}_p$, powers of these observables obey the relation

$$\left(\tau^W(a)\right)^b = \tau^W(ab).$$

In particular, since $pa = 0$ for any $a \in \mathbb{F}_q$ we get that that $(\tau^W(a))^p = I$ for any $a \in \mathbb{F}_q$. The observables obey analogous "twisted commutation" relations to (14),

$$\forall a, b \in \mathbb{F}_q, \quad \tau^X(a)\,\tau^Z(b) = \omega^{-\mathrm{tr}(ab)}\,\tau^Z(b)\,\tau^X(a). \tag{15}$$

It is clear from the definition that all of the τ^X operators commute with each other, and similarly all the τ^Z operators with each other. Thus, it is meaningful to speak of a common eigenbasis for all τ^X operators, and a common eigenbasis for all τ^Z operators. The common eigenbasis for the τ^Z operators is the computational basis. To map this basis to the common eigenbasis of the τ^X operators, one can apply the Fourier transform

$$F = \frac{1}{\sqrt{q}} \sum_{a,b \in \mathbb{F}_q} \omega^{-\mathrm{tr}(ab)} |a\rangle\langle b|. \tag{16}$$

Explicitly, the eigenbases consist of the vectors $|e_W\rangle$ labeled by an element $e \in \mathbb{F}_q$ and $W \in \{X, Z\}$, given by

$$|e_X\rangle = \frac{1}{\sqrt{q}} \sum_j \omega^{-\text{tr}(ej)} |j\rangle , \qquad |e_Z\rangle = |e\rangle .$$

We denote the POVM whose elements are projectors onto basis vectors of the eigenbasis associated with the observables τ^W by $\{\tau_e^W\}_e$. Then for all $W \in \{W, Z\}$ and $a \in \mathbb{F}_q$, the observables $\tau^W(a)$ can be written as

$$\tau^W(a) = \sum_{b \in \mathbb{F}_q} \omega^{\text{tr}(ab)} \tau_b^W . \tag{17}$$

This relation can be inverted as

$$\mathop{\mathbb{E}}_{a \in \mathbb{F}_q} \omega^{-\text{tr}(ab)} \tau^W(a) = \sum_{b' \in \mathbb{F}_q} \mathop{\mathbb{E}}_{a \in \mathbb{F}_q} \omega^{\text{tr}(a(b'-b))} \tau_{b'}^W = \tau_b^W , \tag{18}$$

where the second step follows from Lemma 8.3.

For systems with many qudits, we will consider tensor products of the operators τ^W. Slightly abusing notation, for $W \in \{X, Z\}$ and $a \in \mathbb{F}_q^n$ we denote by $\tau^W(a)$ the tensor product $\tau^W(a_1) \otimes \ldots \otimes \tau^W(a_n)$. These obey the twisted commutation relations

$$\forall a, b \in \mathbb{F}_q^n, \qquad \tau^X(a)\, \tau^Z(b) = \omega^{-\text{tr}(a \cdot b)}\, \tau^Z(b)\, \tau^X(a) ,$$

where $a \cdot b = \sum_{i=1}^n a_i b_i \in \mathbb{F}_q$. For $W \in \{X, Z\}$ and $e \in \mathbb{F}_q^n$ define the eigenstates

$$|e_W\rangle = |(e_1)_W\rangle \otimes \ldots \otimes |(e_n)_W\rangle ,$$

and associated rank-1 projectors τ_e^W.

Since we only consider finite fields $\mathbb{F}_q$ such that $q = 2^k$ the maximally entangled state $|\text{EPR}_q\rangle$ and the corresponding qudit Pauli observables/projectors are isomorphic to a tensor product of maximally entangled states $|\text{EPR}_2\rangle$ and *qubit* Pauli observables/projectors respectively; this is shown in the next lemma. This is used to argue that the Pauli basis test (described in Section 7.3) gives a self-test for Pauli observables and maximally entangled states over qubits.

Lemma 3.29. *For all admissible field sizes $q = 2^k$ and integers Γ, there exists an isomorphism $\phi : (\mathbb{C}^q)^{\otimes \Gamma} \to (\mathbb{C}^2)^{\otimes \Gamma k}$ such that*

$$\phi \otimes \phi \, |\text{EPR}_q\rangle^{\otimes \Gamma} = |\text{EPR}_2\rangle^{\otimes \Gamma k} , \tag{19}$$

and for all $W \in \{X, Z\}$ and for all $u \in \mathbb{F}_q^\Gamma$

$$\tau_u^W = \phi^\dagger \left(\bigotimes_{i=1}^\Gamma \bigotimes_{j=1}^k \sigma_{u_{ij}}^W \right) \phi . \tag{20}$$

Here, the $(u_{ij})_{i,j}$ denotes a vector of $\mathbb{F}_2$ values such that $u_i = \sum_j u_{ij} e_j$ for all $i \in \{1, \ldots, \Gamma\}$ with $\{e_1, \ldots, e_k\}$ being the self-dual basis of $\mathbb{F}_q$ over $\mathbb{F}_2$ specified by Lemma 3.16. For $i \in \{1, \ldots, \Gamma\}$ and $j \in \{1, \ldots, q\}$ the (i, j)-th factor $\sigma_{u_{ij}}^W$ denotes the projector $\frac{1}{2}\left(I + (-1)^{u_{ij}} \sigma^W(1)\right)$ acting on the s-th qubit of $|\text{EPR}_2\rangle^{\otimes \Gamma k}$, where $s = (i-1)k + j$.

Proof. Since $q = 2^k$ is an admissible field size, there exists a self-dual basis $\{e_1, \ldots, e_k\}$ of $\mathbb{F}_q$ over $\mathbb{F}_2$. Define the isometry $\theta : \mathbb{C}^q \to (\mathbb{C}^2)^{\otimes k}$ as $\theta \, |a\rangle = |a_1 a_2 \cdots a_k\rangle$ where $\kappa(a) = (a_1, a_2, \ldots, a_k) \in \mathbb{F}_2^k$ is the bijection introduced in Section 3.3 corresponding to the basis $\{e_1, \ldots, e_k\}$.

Let $a \in \mathbb{F}_q$, and let $\kappa(a) = (a_1, \ldots, a_k) \in \mathbb{F}_2^k$. Then from (18), we get

$$
\begin{aligned}
\tau_a^W &= \mathbb{E}_{b \in \mathbb{F}_q} (-1)^{\mathrm{tr}(ab)} \, \tau^W(b) \\
&= \mathbb{E}_{b \in \mathbb{F}_q} (-1)^{\mathrm{tr}(\sum_j a_j e_j b)} \, \tau^W(b) \\
&= \mathbb{E}_{b \in \mathbb{F}_q} (-1)^{\sum_j a_j b_j} \, \tau^W \left(\sum_j b_j e_j \right),
\end{aligned}
\tag{21}
$$

where $b = \sum_j b_j e_j$; since the basis $\{e_j\}$ is self-dual, we have that $b_j = \mathrm{tr}(b e_j)$. From (21) we get that

$$
\begin{aligned}
\tau_a^W &= \mathbb{E}_{b_1, \ldots, b_k \in \mathbb{F}_2} \prod_{j=1}^k (-1)^{a_j b_j} \tau^W(b_j e_j) \\
&= \prod_{j=1}^k \mathbb{E}_{b_j \in \mathbb{F}_2} (-1)^{a_j b_j} \tau^W(b_j e_j).
\end{aligned}
\tag{22}
$$

Next we claim that for all $c \in \mathbb{F}_2$, we have $\tau^W(ce_j) = \theta^\dagger \sigma^{W,j}(c)\theta$ where $\sigma^{W,j}(c) = I$ if $c = 0$, and otherwise is the Pauli W observable acting on the j-th qubit of $(\mathbb{C}^2)^{\otimes k}$. This can be verified by comparing the actions of both operators on the basis states of $\mathbb{C}^q$.

Thus we obtain that the right-hand side of (22) is equal to

$$
\prod_{j=1}^k \mathbb{E}_{b_j \in \mathbb{F}_2} (-1)^{a_j b_j} \left[\theta^\dagger \sigma^{W,j}(b_j)\theta \right] = \theta^\dagger \bigotimes_{j=1}^k \left(\mathbb{E}_{b_j \in \mathbb{F}_2} (-1)^{a_j b_j} \sigma^W(b_j) \right) \theta
\tag{23}
$$

$$
= \theta^\dagger \left(\bigotimes_{j=1}^k \sigma_{b_j}^W \right) \theta.
\tag{24}
$$

Define $\phi = \theta^{\otimes \Gamma}$. The projector τ_u^W can be decomposed as the tensor product $\bigotimes_{i=1}^\Gamma \tau_{u_i}^W$ where $\tau_{u_i}^W$ acts on the i-th factor of $(\mathbb{C}^q)^{\otimes \Gamma}$. Express each u_i as $\sum_j u_{ij} e_j$ where $u_{ij} \in \mathbb{F}_2$. Then from Equation (24) we get that

$$
\tau_u^W = \bigotimes_{i=1}^\Gamma \tau_{u_i}^W = \phi^\dagger \left(\bigotimes_{i=1}^\Gamma \bigotimes_{j=1}^k \sigma_{u_{ij}}^W \right) \phi,
\tag{25}
$$

which establishes Equation (20). We observe that (19) follows immediately from (20). $\qquad\square$

4 Conditionally Linear Functions, Distributions, and Samplers

4.1 Conditionally linear functions and distributions

We first introduce *conditionally linear functions*, which are used to specify the question distribution for games considered in the paper in a way that the question distribution can be "introspected", as described in Section 8. Intuitively, a conditionally linear function takes as input an element $x \in V = \mathbb{F}^n$ for some $n \geq 0$, and applies linear maps L_j sequentially on x^{V_j} where $V_1, V_2, \ldots$ are a sequence of complementary register subspaces such that both the linear maps L_j and the subspace V_j may depend on the values taken by previous linear maps $L_1(x^{V_1})$, $L_2(x^{V_2})$, etc.

In the remainder of the section we use V to denote the linear space $\mathbb{F}^n$ for some integer $n \geq 0$. For ease of notation we extensively use the subscript range notation. For example, if $V_1, V_2, \ldots, V_\ell$ are fixed subspaces of V and $k \in \{1, 2, \ldots, \ell\}$ we write

$$V_{<k} = \bigoplus_{j : 1 \leq j < k} V_j, \quad V_{>k} = \bigoplus_{j : \ell \geq j > k} V_j,$$

and it is understood that $V_{\leq k}$ and $V_{\geq k}$ are identical to $V_{<k+1}$ and $V_{>k-1}$, respectively. Moreover, if V' is a register subspace of V, $F : V' \to V'$ a linear map, and $x \in V$, we write x^F to denote $F(x^{V'})$. For example, in the following definition x^{L_1} is used as shorthand notation for $L_1(x^{V_1})$.

Definition 4.1. Let V be $\mathbb{F}^n$ for some $n \geq 0$. For all integers $\ell \geq 0$ the collection of ℓ-*level conditionally linear functions* (implicitly, *on V*) is defined inductively as follows.

1. There is a single 0-level conditionally linear function, which is the 0 function on V.

2. Let $\ell \geq 1$ and suppose the collection of $(\ell - 1)$-level conditionally linear functions has been defined. The collection of ℓ-level conditionally linear functions on V consists of all functions L on V that can be expressed in the following form. There exist complementary register subspaces V_1 and $V_{>1}$ of V, a linear function L_1 on V_1, and for all $v \in L_1(V_1)$, an $(\ell - 1)$-level conditionally linear function $L_{>1, v}$ on $V_{>1}$, such that for all $x \in V$,

$$L(x) = x^{L_1} + L_{>1, x^{L_1}}(x^{V_{>1}}).$$

Remark 4.2. *Note that for any integer $\ell \geq 1$ the collection of ℓ-level CL functions trivially contains the collection of $(\ell - 1)$-level CL functions: for this it suffices to note that the 0 function, which is a 0-level CL function, is also a 1-level CL function by setting $V_1 = V$, $V_{>1} = \{0\}$, $L_1(x) = 0$ for all $x \in V$, and $L_{>1, x^{L_1}}$ is the 0 map for all $x \in V$.*

Definition 4.3. Let $L, R : V \to V$ be conditionally linear functions. The *conditionally linear distribution* $\mu_{L,R}$ corresponding to (L, R) is defined as the distribution over pairs $(L(x), R(x)) \in V \times V$ for x drawn uniformly at random from V.

Throughout the paper we abbreviate "conditionally linear functions" and "conditionally linear distributions" as *CL functions* and *CL distributions*, respectively.

The following lemma elucidates structural properties of ℓ-level CL functions. Recall that using our shorthand notation, $x^{L_{<k}}$ and x^{L_k} in the lemma denote $L_{<k}(x)$ and $L_k(x^{V_{k,u}})$ where $u = L_{<k}(x)$.

Lemma 4.4. *Let $\ell \geq 1$ and $V = \mathbb{F}^n$ for some integer $n \geq 0$. A function $L : V \to V$ is an ℓ-level CL function if and only if the following collection of functions and subspaces exists:*

(i) For each $k \in \{1, 2, \dots, \ell\}$, a function $L_{\leq k} : V \to V$ called the k-th marginal of L;

(ii) For each $k \in \{1, 2, \dots, \ell\}$ and $u \in L_{<k}(V)$, a register subspace $V_{k,u}$ of V called the k-th factor space with prefix u;

(iii) For each $k \in \{1, 2, \dots, \ell\}$ and $u \in L_{<k}(V)$, a linear map $L_{k,u} : V_{k,u} \to V_{k,u}$ called the k-th linear map of L with prefix u;

such that the following conditions hold for all $k \in \{1, 2, \dots, \ell\}$.

1. $L_{\leq k}$ is a k-level CL function on V;

2. $V = \bigoplus_{i=1}^{\ell} V_{i, x^{L_{<i}}}$ for all $x \in V$;

3. $L_{\leq k}(x) = \sum_{i=1}^{k} x^{L_i}$ for all $x \in V$, where L_i is shorthand notation for $L_{i, x^{L_{<i}}}$;

4. $L = L_{\leq \ell}$.

As in Item 3, we sometimes use V_k and L_k to denote $V_{k,u}$ and $L_{k,u}$ respectively, leaving the prefix u implicit.

Proof. We first prove the "if" direction: if there exist spaces and functions satisfying the conditions in the lemma, the fact that L is an ℓ-level CL function follows from Items 1 and 4 of the lemma statement.

We now prove the "only if" direction. Given a CL function L on V, we construct the k-th family of subspaces and functions for all $k \in \{1, \dots, \ell\}$ by induction on the level ℓ. First consider the base case $\ell = 1$. Since $L_{<1} = 0$, we omit the mentioning of the prefix $u \in L_{<1}(V)$. Define $L_{\leq 1} = L$, the factor space $V_1 = V$, and the linear map $L_1 = L$. It is straightforward to verify that the conditions in the lemma hold for these choices of linear maps and spaces.

Now, assume that the lemma holds for CL functions of level at most $\ell - 1$, and we prove the lemma for ℓ-level CL functions. By definition, an ℓ-level CL function L can be written as

$$L(x) = x^{L_1} + L_{>1, x^{L_1}}(x^{V_{>1}})$$

for some linear map $L_1 : V_1 \to V_1$ and a family of $(\ell - 1)$-level CL functions

$$\left\{ L_{>1, v} : V_{>1} \to V_{>1} \right\}_{v \in L_1(V_1)}$$

where V_1 and $V_{>1}$ are complementary register subspaces of V. Next, using the inductive hypothesis on the $(\ell - 1)$-level CL function $L_{>1, v}$ we get that for all $v \in L_1(V_1)$ and all $k \in \{1, 2, \dots, \ell - 1\}$ there exist k-th marginal functions $L'_{v, \leq k} : V_{>1} \to V_{>1}$, k-th factor spaces $V'_{v, k, u}$, and k-th linear maps $L'_{v, k, u}$ of $L_{>1, v}$ with prefix $u \in L'_{v, <k}(V_{>1})$ such that the conditions of the lemma for $L_{>1, v}$ hold.

Define the marginal functions $L_{\leq k} : V \to V$, factor spaces $V_{k,u}$ and linear maps $L_{k,u}$ for L as follows.

(i) Define $L_{\leq 1} = L_1$ and the first factor space to be V_1;

(ii) For all $k \in \{2, 3, \dots, \ell\}$, define

$$L_{\leq k} : x \mapsto x^{L_1} + L'_{x^{L_1}, <k}(x^{V_{>1}}) \quad \text{for } x \in V; \qquad (26)$$

(iii) For all $k \in \{2, 3, \dots, \ell\}$ and $u \in L_{<k}(V)$, define $V_{k,u} = V'_{v, k-1, w}$ and $L_{k,u} = L'_{v, k-1, w}$ where $v = u^{V_1}$ and $w = u^{V_{>1}}$.

We verify that the conditions of the lemma are satisfied. Since $L'_{v,<k}$ is by assumption a $(k-1)$-level CL function on $V_{>1}$, we get that $L_{\leq k}$ is a k-level CL function on V from Eq. (26), establishing Item 1. By the induction hypothesis, we have for all $v \in L_1(V_1)$ and $y \in V_{>1}$,

$$V_{>1} = \bigoplus_{i=1}^{\ell-1} V'_{v,i,y^{L'_{v,<i}}}, \tag{27}$$

which implies that for all $x \in V$ and $v = x^{L_1}$,

$$V = V_1 \oplus \left(\bigoplus_{i=1}^{\ell-1} V'_{v,i,x^{L'_{v,<i}}} \right) = V_1 \oplus \left(\bigoplus_{i=2}^{\ell} V_{i,x^{L_{<i}}} \right) = \bigoplus_{i=1}^{\ell} V_{i,x^{L_{<i}}} .$$

The first equality follows from Eq. (27) while the second and third equalities follow from the definition of $V_{k,u}$. This establishes Item 2.

Next, we have that for all $x \in V$, $v = x^{L_1}$, and $k \in \{1, 2, \ldots, \ell\}$,

$$L_{\leq k}(x) = v + L'_{v,<k}(x^{V_{>1}}) \tag{28}$$

$$= v + \sum_{i=1}^{k-1} x^{L'_{v,i}} = \sum_{i=1}^{k} x^{L_i}, \tag{29}$$

where $L'_{v,i}$ is the i-th linear map of $L_{>1,v}$ with prefix $L'_{v,<i}(x)$ and L_i is the i-th linear map of L with prefix $L_{<i}(x)$. The second line follows from the inductive hypothesis applied to $L'_{v,<k}$ and the third line follows from the definition of L_i. Line (29) implies Item 3 of the lemma.

Finally, Item 4 follows from (28) where we set $k = \ell$ and observe that $L'_{x^{L_1},\leq\ell-1}$ is equal to $L_{>1,x^{L_1}}$ by the inductive hypothesis. This shows that $L_{\leq k}$, $V_{k,u}$, and $L_{k,u}$ satisfy the conditions of the lemma and completes the induction. $\square$

We note that the marginal functions, factor spaces, and linear maps of a given CL function L may not be unique; for example, consider the identity function on a linear space $V = \mathbb{F}^n$. This is clearly a 1-level CL function, but it can also be viewed as a k-level CL function for $k \in \{2, \ldots, n\}$ with an arbitrary partition of V into factor spaces.

Lemma 4.5. *Let $\ell, k \geq 0$ be integers and $U = \mathbb{F}^n$, $V = \mathbb{F}^m$ be linear spaces. Suppose L is a k-level CL function on U and R_u is an ℓ-level CL function on V for each $u \in L(U)$. Then the* concatenation T *of L and $\{R_u\}_u$ defined as*

$$T(x) = L(x^U) + R_{L(x^U)}(x^V)$$

is a $(k+\ell)$-level conditionally linear function on $U \oplus V$.

Proof. We prove the claim by induction on k. The case $k = 0$ follows from the Definition 4.1. Assume that the lemma holds for L being at most $(k-1)$-level. By Definition 4.1, there are complementary register subspaces U_1 and $U_{>1}$ of U, a linear function L_1 on U_1, and a family of $(k-1)$-level CL functions $L_{>1,v}$ for $v \in L_1(U_1)$ such that

$$L(x^U) = x^{L_1} + L_{>1,x^{L_1}}(x^{U_{>1}}).$$

For all x^{L_1}, define function $T_{>1,x^{L_1}}$ on $U_{>1} \oplus V$ as

$$T_{>1,x^{L_1}}(x^{U_{>1}\oplus V}) = L_{>1,x^{L_1}}(x^{U_{>1}}) + R_{x^{L_1}+x^{L_{>1}}}(x^V),$$

39

the concatenation of $L_{>1,x^{L_1}}$ and $\{R_{L(x^U)}\}_{x^{L_{>1}}}$ where $L_{>1}$ is the shorthand notation of $L_{>1,x^{L_1}}$. By the induction hypothesis, $T_{>1,x^{L_1}}$ is $(k+\ell-1)$-level conditionally linear. The lemma follows from Definition 4.1. $\qquad\square$

Lemma 4.6 (Direct sums of CL functions). *Let $V^{(1)}, V^{(2)}, \ldots, V^{(m)}$ be register subspaces of V such that $V = \bigoplus_{j=1}^m V^{(j)}$. Suppose that, for each $j \in \{1, 2, \ldots, m\}$, $L^{(j)}$ is an ℓ_j-level conditionally linear function on $V^{(j)}$. Then the direct sum $L = \bigoplus_{j=1}^m L^{(j)}$ is an ℓ-level CL function over V for $\ell = \max_j\{\ell_j\}$, where L is defined by*

$$L(x) = \sum_{j=1}^m L^{(j)}(x^{(j)})$$

for all $x = \sum_{j=1}^m x^{(j)} \in \bigoplus_{j=1}^m V^{(j)}$.

Proof. It is easy to see that an ℓ-level CL function is also k-level conditionally linear for all $k \geq \ell$. Hence, it suffices to prove the claim where $\ell_j = \ell$ for $j = 1, 2, \ldots, m$.

We prove the theorem by an induction on ℓ. For $\ell = 1$, the functions $L^{(j)}$ are linear and the claim follows by the fact that the direct sum of linear maps is linear.

Assume now the theorem holds for conditionally linear functions of level at most $\ell - 1$ and $L^{(j)}$ are ℓ-level conditionally linear functions for $j = 1, 2, , \ldots, m$. By definition, $L^{(j)}$ is the concatenation of conditionally linear functions $L_1^{(j)}$ on $V_1^{(j)}$ and $\{L_{>1,v_j}^{(j)}\}_{v_j}$ on $V_{>1}^{(j)}$ of levels 1, and $\ell - 1$ respectively. Furthermore,

$$L(x) = \sum_{j=1}^m L^{(j)}(x^{(j)}) = \sum_{j=1}^m \left(v_j + L_{>1,v_j}^{(j)}\left((x^{(j)})^{V_{>1}^{(j)}}\right)\right),$$

where $v_j = L_1^{(j)}\left((x^{(j)})^{V_1^{(j)}}\right)$. By the induction hypothesis,

$$L_1(x^{V_1}) = \sum_{j=1}^m L_1^{(j)}\left((x^{(j)})^{V_1^{(j)}}\right), \quad L_{>1,v}(x^{V_{>1}}) = \sum_{j=1}^m L_{>1,v_j}^{(j)}\left((x^{(j)})^{V_{>1}^{(j)}}\right)$$

are 1-level and $(\ell - 1)$-level conditionally linear respectively for $v = \sum_j v_j$, $V_1 = \bigoplus_{j=1}^m V_1^{(j)}$, and $V_{>1} = \bigoplus_{j=1}^m V_{>1}^{(j)}$. This proves that L is ℓ-level conditionally linear. $\qquad\square$

Lemma 4.7. *For each $i \in \{1, \ldots, m\}$ let $L^{(i)}, R^{(i)} : V^{(i)} \to V^{(i)}$ be ℓ_i-level conditionally linear functions and let $L, R : V \to V$ be the direct sums $L = \bigoplus_i L_i$ and $R = \bigoplus_i R_i$, respectively, as defined in Lemma 4.6. Then the conditionally linear distribution $\mu_{L,R}$ is the product distribution $\prod_{i=1}^m \mu_{L^{(i)},R^{(i)}}$ over $V \times V$.*

Proof. The distribution $\mu_{L,R}$ is the distribution over pairs $(L(x), R(x))$ where x is sampled uniformly from $V \times V$. By Lemma 4.6, this is equivalent to the distribution over pairs $\left((L_i(x^{V_i}))_{i=1}^m, (R_i(x^{V_i}))_{i=1}^m\right)$ where x is chosen uniformly at random from V. This distribution is exactly the product of the distributions $\mu_{L^{(i)},R^{(i)}}$ for $i = 1, 2, \ldots, m$. $\qquad\square$

CL functions used in the paper are frequently defined over a "large" field $\mathbb{F}_{2^t}$ (e.g., the CL functions used in the low degree tests of Section 7). However, the introspection protocol in Section 8 handles CL functions defined over $\mathbb{F}_2$. The following definition and lemma show that CL functions over prime power fields can be viewed as CL functions over the prime field via a "downsizing" operation.

Definition 4.8 (Downsizing CL functions). Let $V = \mathbb{F}_q^n$ be a linear space for a prime power $q = p^t$. Let $L : V \to V$ be a function. Let $\kappa(\cdot)$ denote the downsize map from Section 3.3 corresponding to the basis $\{e_1, \ldots, e_t\}$ of $\mathbb{F}_q$ over $\mathbb{F}_p$ specified by Lemma 3.16. In particular, κ is linear over $\mathbb{F}_p$, and by Lemma 3.14, the set $\kappa(V)$ is the linear space $\mathbb{F}_p^{nt}$. Define the *downsized function* $L^\kappa : \kappa(V) \to \kappa(V)$ by $L^\kappa = \kappa \circ L \circ \kappa^{-1}$.

Lemma 4.9. *Let $V = \mathbb{F}_q^n$ for a prime power $q = p^t$. Let $L : V \to V$ be an ℓ-level CL function over V for some integer $\ell \geq 0$. Let $L_{\leq j}$, $V_{j,u}$, and $L_{j,u}$ denote the j-th marginal functions, factor spaces, and linear maps corresponding to L as guaranteed by Lemma 4.4. Then $L^\kappa : \kappa(V) \to \kappa(V)$ is an ℓ-level CL function on $V^\kappa = \kappa(V) = \mathbb{F}_p^{nt}$ with marginal functions $L_{\leq j}^\kappa$, factor spaces $V_{j,v}^\kappa$, and linear maps $L_{j,v}^\kappa$ that satisfy the following for all $j \in \{1, \ldots, \ell\}$.*

1. *The j-th marginal function $L_{\leq j}^\kappa$ of L^κ is equal to $\kappa \circ L_{\leq j} \circ \kappa^{-1}$.*

2. *For all $u \in L_{<j}(V)$, the j-th factor space $V_{j,\kappa(u)}^\kappa$ and the j-th linear map $L_{j,\kappa(u)}^\kappa$ of L^κ are equal to $\kappa(V_{j,u})$ and $\kappa \circ L_{j,\kappa(u)} \circ \kappa^{-1}$ respectively.*

Proof. We prove the lemma by induction on ℓ. Let $L : V \to V$ be an ℓ-level CL function. For the base case $\ell = 1$, observe that since κ is a linear bijection between $\mathbb{F}_q$ and $\mathbb{F}_p^t$ as linear spaces over $\mathbb{F}_p$, the function L^κ is linear, and thus a 1-level CL function over $\kappa(V) = \mathbb{F}_p^{nt}$. Furthermore, the first marginal function $L_{\leq 1}^\kappa = L^\kappa = \kappa \circ L_{\leq 1} \circ \kappa^{-1}$; the factor space $V_1^\kappa = \kappa(V_1) = \kappa(V)$, and $L_1^\kappa = L = \kappa \circ L_1 \circ \kappa^{-1}$.

Assume that the statement of the lemma holds for some $\ell - 1 \geq 1$. Let $L_{\leq j}$, $V_{j,u}$, and $L_{j,u}$ denote the marginal functions, factor spaces, and linear maps corresponding to L as guaranteed by Lemma 4.4. Recursively define the following functions and spaces, for $j \in \{1, \ldots, \ell\}$.

1. $L_{\leq j}^\kappa = \kappa \circ L_{\leq j} \circ \kappa^{-1}$.

2. For all $u \in L_{<j}(V)$, set $V_{j,\kappa(u)}^\kappa = \kappa(V_{j,u})$ and set $L_{j,\kappa(u)}^\kappa = \kappa \circ L_{j,\kappa(u)} \circ \kappa^{-1}$.

We argue that $\{L_{\leq j}^\kappa\}$, $\{V_{j,v}^\kappa\}$, and $\{L_{j,v}^\kappa\}$ satisfy the conditions of Lemma 4.4 for the function L^κ, which implies that L^κ is an ℓ-level CL function over $\kappa(V)$.

We first establish Item 4 of Lemma 4.4. Since $L_{\leq \ell} = L$, this implies

$$L^\kappa = \kappa \circ L \circ \kappa^{-1} = \kappa \circ L_{\leq \ell} \circ \kappa^{-1} = L_{\leq \ell}^\kappa,$$

as desired. Next, for all $j \in \{1, 2, \ldots, \ell\}$, for all $y \in \kappa(V)$ with $y = \kappa(x)$ for some $x \in V$, letting $u_j = L_{<j}(x)$, we have

$$\kappa(V) = \kappa\left(\bigoplus_{j=1}^{\ell} V_{j, x^{L_{<j}}}\right) = \bigoplus_{j=1}^{\ell} \kappa\left(V_{j, x^{L_{<j}}}\right) = \bigoplus_{j=1}^{\ell} V_{j, \kappa(x^{L_{<j}})}^\kappa.$$

The first equality follows from Item 2 of Lemma 4.4, the second equality follows from Lemma 3.14, and the third equality follows by definition. Since $\kappa(x^{L_{<j}}) = L_{<j}^\kappa \circ \kappa(x) = L_{<j}^\kappa(y)$, this establishes Item 2 of Lemma 4.4.

Next, we have for all $j \in \{1, 2, \ldots, \ell\}$ and all $y \in \kappa(V)$ with $y = \kappa(x)$ for some $x \in V$,

$$L_{\leq j}^\kappa(y) = \left(\kappa \circ L_{\leq j} \circ \kappa^{-1}\right)(y) = \left(\kappa \circ L_{\leq j}\right)(x) = \sum_{i=1}^{j} \kappa\left(x^{L_{i, x^{L_{<i}}}}\right)$$

$$= \sum_{i=1}^{j} L_{i, v_i}^\kappa\left(\kappa(x^{V_{i, x^{L_{<i}}}})\right) = \sum_{i=1}^{j} L_{i, v_i}^\kappa\left(y^{V_{i, v_i}^\kappa}\right)$$

41

where $v_i = L^{\kappa}_{<i}(y) = \kappa(x^{L_{<i}})$. The first equality follows from definition of $L^{\kappa}_{\leq j}$, the second equality follows from $y = \kappa(x)$, the third equality follows from Item 3 of Lemma 4.4 applied to $L_{\leq j}$, the fourth equality follows from the definition of the linear map $L^{\kappa}_{i,v}$, and the fifth equality follows from Lemma 3.14. This establishes Item 3 of Lemma 4.4 for $L^{\kappa}_{\leq j}$.

Finally, since $L_{\leq j}$ is a j-level CL function over V, using the inductive hypothesis we have that $L^{\kappa}_{\leq j}$ is a j-level CL function over $\kappa(V)$ when $j \in \{1, 2, \ldots, \ell - 1\}$. It remains to establish that $L^{\kappa}_{\leq \ell}$ is an ℓ-level CL function. Since L is an ℓ-level CL function, there exists register subspaces $V_1, V_{>1}$ such that $V = V_1 \oplus V_{>1}$, a linear map $L_1 : V_1 \to V_1$ and a collection of $(\ell - 1)$-level CL functions $\{L_{>1,v} : V_{>1} \to V_{>1}\}_{v \in L_1(V_1)}$ such that $L(x) = x^{L_1} + L_{>1, x^{L_1}}(x^{V_{>1}})$ for all $x \in V$. Observe that L^{κ}_1 is a 1-level CL function on $V^{\kappa}_1 = \kappa(V_1)$, and for $v' = \kappa(v) \in L^{\kappa}_1(V^{\kappa}_1)$, the inductive hypothesis implies the function $L^{\kappa}_{>1,v'}$ is an $(\ell - 1)$-level CL function on $V^{\kappa}_{>1} = \kappa(V_{>1})$. Furthermore, since $L^{\kappa}_{\leq \ell} = L^{\kappa}$, we have that for all $y \in \kappa(V)$ with $y = \kappa(x)$ for some $x \in V$,

$$L^{\kappa}_{\leq \ell}(y) = L^{\kappa}(y) = L^{\kappa}_1(y) + L^{\kappa}_{>1, L^{\kappa}_1(y)}(y^{V^{\kappa}_{>1}})$$

which implies that $L^{\kappa}_{\leq \ell}$ is an ℓ-level CL function over $\kappa(V) = \kappa(V_1) \oplus \kappa(V_{>1})$. This establishes Item 1 of Lemma 4.4, and completes the induction. $\qquad\square$

Lemma 4.10. *Let $V = \mathbb{F}^n_q$ for some integer n and prime power $q = p^t$. Let $L, R : V \to V$ be CL functions. Let $L^{\kappa}, R^{\kappa} : \kappa(V) \to \kappa(V)$ be the associated downsized CL functions, as defined in Definition 4.8. Then the distribution $\mu_{L^{\kappa}, R^{\kappa}}$ over $\kappa(V) \times \kappa(V)$ defined in Definition 4.3 is identical to the distribution of $(x, y) \in \kappa(V) \times \kappa(V)$ obtained by first sampling (x', y') according to $\mu_{L,R}$ and then returning $(\kappa(x'), \kappa(y'))$.*

Proof. The fact that L^{κ}, R^{κ} are well-defined CL functions follows from Lemma 4.9. The lemma is immediate from the definition of $\mu_{L^{\kappa}, R^{\kappa}}$ and the fact that κ is a bijection. $\qquad\square$

4.2 Conditionally linear samplers

Samplers are Turing machines that perform computations corresponding to CL functions defined in Section 4.1. The inputs and outputs of the sampler are binary strings that are interpreted as representing data of different types (integers, bits, vectors in $\mathbb{F}^s_q$, etc.). See Section 3.3.2 and in particular Remark 3.19 for an in-depth discussion of representing structured objects on a Turing machine.

Definition 4.11. A function $q : \mathbb{N} \to \mathbb{N}$ is an *admissible field size function* if for all $n \in \mathbb{N}$, $q(n)$ is an admissible field size as defined in Definition 3.15.

Definition 4.12 (Conditionally linear samplers). Let $q : \mathbb{N} \to \mathbb{N}$ be an admissible field size function, and let $s : \mathbb{N} \to \mathbb{N}$ be a function. A 6-input Turing machine $\mathcal{S}$ is a *ℓ-level conditionally linear sampler with field size $q(n)$ and dimension $s(n)$* if for all $n \in \mathbb{N}$, letting $q = q(n)$ and $s = s(n)$, there exist ℓ-level CL functions $L^{\mathrm{A},n}, L^{\mathrm{B},n} : \mathbb{F}^s_q \to \mathbb{F}^s_q$ with marginal functions $\{L^{w,n}_{\leq j}\}$ and factor spaces $\{V^{w,n}_{j,u}\}$ for $w \in \{\mathrm{A}, \mathrm{B}\}$ satisfying the conditions of Lemma 4.4, such that for all $w \in \{\mathrm{A}, \mathrm{B}\}, j \in \{1, \ldots, \ell\}, z \in \mathbb{F}^s_q$:

- On input $(n, \mathrm{DIMENSION})$, the sampler $\mathcal{S}$ outputs the dimension $s(n)$.

- On input $(n, w, \mathrm{MARGINAL}, j, z)$, the sampler $\mathcal{S}$ outputs the binary representation of $L^{w,n}_{\leq j}(z)$,

- On input $(n, w, \mathrm{LINEAR}, j, u, y)$, the sampler $\mathcal{S}$ outputs the binary representation of $L^{w,n}_{j,u}(y)$, where u is interpreted as an element of $V^{w,n}_{<j}$,

- On input $(n, w, \text{FACTOR}, j, u)$, the sampler $\mathcal{S}$ outputs the j-th factor space $V_{j,u}^{w,n}$ of $L^{w,n}$ with prefix $u \in L_{<j}^{w,n}(V)$, represented as an indicator vector in $\{0,1\}^s$.

We call $\mathbb{F}_{q(n)}^{s(n)}$ the *ambient space of* $\mathcal{S}$. We call the CL functions $L^{w,n}$ for $w \in \{A, B\}$ the *CL functions of $\mathcal{S}$ on index n*. The *time complexity* of $\mathcal{S}$, denoted as $\text{TIME}_{\mathcal{S}}(n)$, is the number of steps before $\mathcal{S}$ halts for index n. The *randomness complexity* of $\mathcal{S}$, denoted by $\text{RAND}_{\mathcal{S}}(n)$, is defined to be $s(n) \log q(n)$.

Remark 4.13. *Conditionally linear samplers are defined to have 6-input tapes, but depending on the input, not all input tapes are read. For example, if the second input tape has the input* DIMENSION, *then the remaining input tapes are ignored. Thus for notational convenience we write samplers with different numbers of arguments, depending on the type of argument it gets. The number of arguments is always at most 6, however.*

The following definition shows how samplers naturally correspond to conditionally linear distributions.

Definition 4.14 (Distribution of a sampler). Let $\mathcal{S}$ be a sampler with field size $q(n)$, dimension $s(n)$. For each $n \in \mathbb{N}$, let $L^{A,n}, L^{B,n}$ denote the CL functions of $\mathcal{S}$ on index n. Let $\mu_{\mathcal{S},n}$ denote the CL distribution $\mu_{L^{A,n}, L^{B,n}}$ corresponding to $(L^{A,n}, L^{B,n})$, as defined in Definition 4.3. We call $\mu_{\mathcal{S},n}$ the *distribution of sampler $\mathcal{S}$ on index n*.

The following provides a definition of a "downsized" sampler that can be obtained from any sampler $\mathcal{S}$ over an admissible field $\mathbb{F}_q$.

Definition 4.15 (Downsized sampler). Let $q : \mathbb{N} \to \mathbb{N}$ be an admissible field size function. Let $\mathcal{S}$ be an ℓ-level sampler with field size $q(n)$ and dimension $s(n)$. Define $\kappa(\mathcal{S})$ as the following Turing machine. For all $n \in \mathbb{N}$, $w \in \{A, B\}$, $j \in \{1, \ldots, \ell\}$, and $z \in \mathbb{F}_2^{s \log q}$ where $q = q(n)$ and $s = s(n)$:

- On input $(n, \text{DIMENSION})$, the sampler returns the output of $\mathcal{S}(n, \text{DIMENSION})$ multiplied by $\log q$.

- On input $(n, w, \text{MARGINAL}, j, z)$, the sampler $\kappa(\mathcal{S})$ returns the output of $\mathcal{S}(n, w, \text{MARGINAL}, j, z)$.

- On input $(n, w, \text{LINEAR}, j, u', y)$, the sampler $\kappa(\mathcal{S})$ computes u such that $u' = \kappa(u)$ and returns the output of $\mathcal{S}(n, w, \text{LINEAR}, j, u, y)$.

- On input $(n, w, \text{FACTOR}, j, u')$, the sampler $\kappa(\mathcal{S})$ computes u such that $u' = \kappa(u)$ and the indicator vector

$$C = \mathcal{S}(n, w, \text{FACTOR}, j, u) \in \{0,1\}^s ,$$

and returns the expanded indicator vector $(D_1, D_2, \ldots, D_s) \in (\{0,1\}^{\log q})^s$ where D_i is the all ones vector in $\{0,1\}^{\log q}$ if $C_i = 1$ and D_i is the all zeroes vector otherwise.

The next lemma establishes that $\kappa(\mathcal{S})$ is a well-defined CL sampler, in the sense that it can be derived from a family of CL functions as in Definition 4.12.

Lemma 4.16. *Let $\ell \geq 1$ be such that $\mathcal{S}$ is an ℓ-level CL sampler, and let $q(n)$ and $s(n)$ be as in Definition 4.15. Then the Turing machine $\kappa(\mathcal{S})$ is an ℓ-level CL sampler with field size 2, dimension $s'(n) = s(n) \log q(n)$, and randomness and time complexities*

$$\text{RAND}_{\kappa(\mathcal{S})}(n) = \text{RAND}_{\mathcal{S}}(n) , \quad \text{TIME}_{\kappa(\mathcal{S})}(n) = O\big(\text{TIME}_{\mathcal{S}}(n) \log q(n)\big) .$$

Furthermore, for every integer $n \in \mathbb{N}$ the CL functions of $\kappa(\mathcal{S})$ on index n are $(L^{A,n})^\kappa$ and $(L^{B,n})^\kappa$, where $L^{A,n}, L^{B,n}$ are the CL functions of $\mathcal{S}$ on index n.

Proof. To show that $\kappa(\mathcal{S})$ is an ℓ-level CL sampler we first show the "Furthermore" part, i.e. verify that for any integer $n \geq 1$ the CL functions $(L^{\mathrm{A},n})^\kappa$ and $(L^{\mathrm{B},n})^\kappa$ are its associated CL functions on index n, as defined in Definition 4.12.

Observe that for $z \in V$, the binary representation of z as an element of $\{0,1\}^{s \log q}$ passed as input to $\mathcal{S}$ is, by definition (see Section 3.3.2), identical to the binary representation of $\kappa(z)$. Using the definition $(L^{w,n})^\kappa = \kappa \circ L^{w,n} \circ \kappa^{-1}$ for $w \in \{\mathrm{A}, \mathrm{B}\}$ this justifies that $\kappa(\mathcal{S})$ returns the correct output when executed on inputs of the form $(n, \mathrm{DIMENSION})$, $(n, w, \mathrm{MARGINAL}, j, z)$ and $(n, w, \mathrm{LINEAR}, j, u', y)$.

Next, if T is a register subspace of $\mathbb{F}_q^s$ with indicator vector $C \in \{0,1\}^s$, then $\kappa(T)$ is a register subspace of $\mathbb{F}_2^{s \log q}$ with indicator vector D defined from C as in Definition 4.15. Thus the output of $\kappa(\mathcal{S})$ on input $(n, w, \mathrm{FACTOR}, j, u')$ is equal to the indicator vector of $\kappa(V_{j,u}^{w,n})$, which is the j-th factor space of $L^{w,n}$ with prefix $u' = \kappa(u)$.

The time and randomness complexities of $\kappa(\mathcal{S})$ are the same as with the sampler $\mathcal{S}$, except it takes $O(\log q(n))$ times longer to output the factor space indicator vectors. $\qquad\square$

5 Nonlocal Games

We introduce definitions associated with nonlocal games and strategies that will be used throughout.

5.1 Games and strategies

Definition 5.1 (Two-player one-round games)**.** A *two-player one-round game* $\mathfrak{G}$ is specified by a tuple $(\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}, \mu, D)$ where

1. $\mathcal{X}$ and $\mathcal{Y}$ are finite sets (called the *question alphabets*),

2. $\mathcal{A}$ and $\mathcal{B}$ are finite sets (called the *answer alphabets*),

3. μ is a probability distribution over $\mathcal{X} \times \mathcal{Y}$ (called the *question distribution*), and

4. $D : \mathcal{X} \times \mathcal{Y} \times \mathcal{A} \times \mathcal{B} \to \{0, 1\}$ is a function (called the *decision predicate*).

Definition 5.2 (Tensor product strategies)**.** A *tensor product strategy* $\mathscr{S}$ for a game $\mathfrak{G} = (\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}, \mu, D)$ is a tuple $(|\psi\rangle, A, B)$ where

- $|\psi\rangle$ is a pure quantum state in $\mathcal{H}_A \otimes \mathcal{H}_B$ for finite dimensional complex Hilbert spaces $\mathcal{H}_A, \mathcal{H}_B$,

- A is a set $\{A^x\}$ such that for every $x \in \mathcal{X}$, $A^x = \{A^x_a\}_{a \in \mathcal{A}}$ is a POVM over $\mathcal{H}_A$, and

- B is a set $\{B^y\}$ such that for every $y \in \mathcal{Y}$, $B^y = \{B^y_b\}_{b \in \mathcal{B}}$ is a POVM over $\mathcal{H}_B$.

Definition 5.3 (Tensor product value)**.** The *tensor product value* of a tensor product strategy $\mathscr{S} = (|\psi\rangle, A, B)$ with respect to a game $\mathfrak{G} = (\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}, \mu, D)$ is defined as

$$\mathrm{val}^*(\mathfrak{G}, \mathscr{S}) = \sum_{x, y, a, b} \mu(x, y)\, D(x, y, a, b)\, \langle \psi | A^x_a \otimes B^y_b\, | \psi \rangle\, .$$

For $v \in [0, 1]$ we say that the strategy $\mathscr{S}$ *passes (or wins)* $\mathfrak{G}$ *with probability* v if $\mathrm{val}^*(\mathfrak{G}, \mathscr{S}) \geq v$. The *tensor product value* of $\mathfrak{G}$ is defined as

$$\mathrm{val}^*(\mathfrak{G}) = \sup_{\mathscr{S}} \mathrm{val}^*(\mathfrak{G}, \mathscr{S})\, ,$$

where the supremum is taken over all tensor product strategies $\mathscr{S}$ for $\mathfrak{G}$.

Remark 5.4. *Unless specified otherwise, all strategies considered in this paper are tensor product strategies, and we simply call them* strategies. *Similarly, we refer to* $\mathrm{val}^*(\mathfrak{G})$ *as the* value *of the game* $\mathfrak{G}$.

Definition 5.5 (Projective strategies)**.** We say that a strategy $\mathscr{S} = (|\psi\rangle, A, B)$ is *projective* if all the measurements $\{A^x_a\}_a$ and $\{B^y_b\}_b$ are projective.

Remark 5.6. *A game* $\mathfrak{G} = (\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}, \mu, D)$ *is* symmetric *if the question and answer alphabets are the same for both players (i.e.* $\mathcal{X} = \mathcal{Y}$ *and* $\mathcal{A} = \mathcal{B}$*), the distribution* μ *is symmetric (i.e.* $\mu(x, y) = \mu(y, x)$*), and the decision predicate D treats both players symmetrically (i.e. for all* x, y, a, b*,* $D(x, y, a, b) = D(y, x, b, a)$*). Furthermore, we call a strategy* $\mathscr{S} = (|\psi\rangle, A, B)$ symmetric *if* $|\psi\rangle$ *is a state in* $\mathcal{H} \otimes \mathcal{H}$*, for some Hilbert space* $\mathcal{H}$*, that is invariant under permutation of the two factors, and the measurement operators of both players are identical. We specify symmetric games* $\mathfrak{G}$ *and symmetric strategies* $\mathscr{S}$ *using a more compact notation: we write* $\mathfrak{G} = (\mathcal{X}, \mathcal{A}, \mu, D)$ *and* $\mathscr{S} = (|\psi\rangle, M)$ *where M denotes the set of measurement operators for both players.*

Lemma 5.7. *Let* $\mathfrak{G} = (\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}, \mu, D)$ *be a symmetric game such that* $\mathrm{val}^*(\mathfrak{G}) = 1 - \varepsilon$ *for some* $\varepsilon \geq 0$. *Then there exists a symmetric and projective strategy* $\mathscr{S} = (|\psi\rangle, M)$ *such that* $\mathrm{val}^*(\mathfrak{G}, \mathscr{S}) \geq 1 - 2\varepsilon$.

Proof. By definition there exists a strategy $\mathscr{S}' = (|\psi'\rangle, A, B)$ such that $\mathrm{val}^*(\mathfrak{G}, \mathscr{S}') \geq 1 - 2\varepsilon$. Enlarging one player's space if necessary, assume without loss of generality that $|\psi'\rangle \in \mathbb{C}^d_{A'} \otimes \mathbb{C}^d_{B'}$ for some integer d and that for every x and y, A^x and B^y is a projective measurement. Let

$$
|\psi\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle_A |1\rangle_B |\psi'\rangle_{A'B'} + |1\rangle_A |0\rangle_B |\psi'_\tau\rangle_{A'B'} \right) \in (\mathbb{C}^2_A \otimes \mathbb{C}^d_{A'}) \otimes (\mathbb{C}^2_B \otimes \mathbb{C}^d_{B'}) \,,
$$

where $|\psi'_\tau\rangle$ is obtained from $|\psi\rangle$ by permuting the two players' registers. Observe that $|\psi\rangle$ is invariant under permutation of AA' and BB'. Let $w \in \{A, B\}$. For any question $x \in \mathcal{X} = \mathcal{Y}$, let M^x be the measurement obtained by first measuring the qubit in register w and depending on the outcome, applying the measurement A^x on player w's d-dimensional register w' to obtain an outcome a. Using that by assumption the decision predicate D for $\mathfrak{G}$ is symmetric, it is not hard to verify that $\mathrm{val}^*(\mathfrak{G}, \mathscr{S}) = \mathrm{val}^*(\mathfrak{G}, \mathscr{S}')$. $\qquad\square$

Definition 5.8. Let $\mathfrak{G} = (\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}, \mu, D)$ be a game, and let $\mathscr{S} = (|\psi\rangle, A, B)$ be a strategy for $\mathfrak{G}$ such that the spaces $\mathcal{H}_A \simeq \mathcal{H}_B$ canonically. Let $S \subseteq \mathcal{X} \times \mathcal{Y}$ denote the support of the question distribution μ, i.e. the set of (x, y) such that $\mu(x, y) > 0$. We say that $\mathscr{S}$ is a *commuting strategy for* $\mathfrak{G}$ if for all question pairs $(x, y) \in S$, we have $[A^x_a, B^y_b] = 0$ for all $a \in \mathcal{A}, b \in \mathcal{B}$, where $[A, B] = AB - BA$ denotes the commutator.

Definition 5.9 (Consistent measurements). Let $\mathcal{A}$ be a finite set, let $|\psi\rangle \in \mathcal{H} \otimes \mathcal{H}$ a state, and $\{M_a\}_{a \in \mathcal{A}}$ a projective measurement on $\mathcal{H}$. We say that $\{M_a\}_{a \in \mathcal{A}}$ *is consistent on* $|\psi\rangle$ if and only if

$$
\forall a \in \mathcal{A}, \quad M_a \otimes I_B |\psi\rangle = I_A \otimes M_a |\psi\rangle \,.
$$

Definition 5.10 (Consistent strategies). Let $\mathscr{S} = (|\psi\rangle, A, B)$ be a projective strategy with state $|\psi\rangle \in \mathcal{H} \otimes \mathcal{H}$, for some Hilbert space $\mathcal{H}$, which is defined on question alphabets $\mathcal{X}$ and $\mathcal{Y}$ and answer alphabets $\mathcal{A}$ and $\mathcal{B}$, respectively. We say that the strategy $\mathscr{S}$ is *consistent* if for all $x \in \mathcal{X}$, the measurement $\{A^x_a\}_{a \in \mathcal{A}}$ is consistent on $|\psi\rangle$ and if for all $y \in \mathcal{Y}$, the measurement $\{B^y_b\}_{b \in \mathcal{B}}$ is consistent on $|\psi\rangle$.

Definition 5.11. We say that a game $\mathfrak{G}$ has a *PCC strategy* if it has a strategy $\mathscr{S}$ that is projective, consistent, and commuting for $\mathfrak{G}$. Additionally, we say that a game $\mathfrak{G}$ has an *SPCC strategy* if it has a symmetric PCC strategy.

Definition 5.12 (Entanglement requirements of a game). For all games $\mathfrak{G}$ and $\nu \in [0, 1]$, let $\mathscr{E}(\mathfrak{G}, \nu)$ denote the minimum integer d such that there exists a finite dimensional tensor product strategy $\mathscr{S}$ that achieves success probability at least ν in the game $\mathfrak{G}$ with a state $|\psi\rangle$ whose Schmidt rank is at most d. If there is no finite dimensional strategy that achieves success probability ν, then define $\mathscr{E}(\mathfrak{G}, \nu)$ to be ∞.

5.2 Distance measures

We introduce several distance measures that are used throughout.

Definition 5.13 (Distance between states). Let $\{|\psi_n\rangle\}_{n \in \mathbb{N}}$ and $\{|\psi'_n\rangle\}_{n \in \mathbb{N}}$ be two families of states in the same space $\mathcal{H}$. For some function $\delta : \mathbb{N} \to [0, 1]$ we say that $\{|\psi_n\rangle\}$ and $\{|\psi'_n\rangle\}$ are δ-close, denoted as $|\psi\rangle \approx_\delta |\psi'\rangle$, if $\||\psi_n\rangle - |\psi'_n\rangle\|^2 = O(\delta(n))$. (For convenience we generally leave the dependence of the states and δ on the indexing parameter n implicit.)

Definition 5.14 (Consistency between POVMs). Let $\mathcal{X}$ be a finite set and μ a distribution on $\mathcal{X}$. Let $|\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$ be a quantum state, and for all $x \in \mathcal{X}$, $\{A_a^x\}$ and $\{B_a^x\}$ POVMs. We write

$$A_a^x \otimes I_{\mathrm{B}} \simeq_\delta I_{\mathrm{A}} \otimes B_a^x$$

on state $|\psi\rangle$ and distribution μ if

$$\mathop{\mathbb{E}}_{x \sim \mu} \sum_{a \neq b} \langle \psi | A_a^x \otimes B_b^x | \psi \rangle \leq O(\delta) \ .$$

In this case, we say that $\{A_a^x\}$ and $\{B_a^x\}$ are δ-consistent on $|\psi\rangle$.

Note that a consistent measurement according to Definition 5.9 is 0-consistent with itself, under the singleton distribution, according to Definition 5.14 (and vice-versa).

Definition 5.15 (Distance between POVMs). Let $\mathcal{X}$ be a finite set and μ a distribution on $\mathcal{X}$. Let $|\psi\rangle \in \mathcal{H}$ be a quantum state, and for all $x \in \mathcal{X}$, $\{M_a^x\}$ and $\{N_a^x\}$ two POVM on $\mathcal{H}$. We say that $\{M_a^x\}$ and $\{N_a^x\}$ are δ-close on state $|\psi\rangle$ and under distribution μ if

$$\mathop{\mathbb{E}}_{x \sim \mu} \sum_a \| (M_a^x - N_a^x) |\psi\rangle \|^2 \leq \delta \ ,$$

and we write $M_a^x \approx_\delta N_a^x$ to denote this when the state $|\psi\rangle$ and distribution μ are clear from context. This distance is referred to as the *state-dependent* distance.

Definition 5.16 (Distance between strategies). Let $\mathfrak{G} = (\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}, \mu, D)$ be a nonlocal game and let $\mathscr{S} = (\psi, A, B)$, $\mathscr{S}' = (\psi', A', B')$ be partial strategies for $\mathfrak{G}$. For $\delta \in [0,1]$ we say that $\mathscr{S}$ is δ-close to $\mathscr{S}'$ if the following conditions hold.

1. The states $|\psi\rangle, |\psi'\rangle$ are states in the same Hilbert space $\mathcal{H}_A \otimes \mathcal{H}_B$ and are δ-close.

2. For all $x \in \mathcal{X}, y \in \mathcal{Y}$, we have $A_a^x \approx_\delta (A')_a^x$ and $B_b^y \approx_\delta (B')_b^y$, with the approximations holding under the distribution μ, and on either $|\psi\rangle$ or $|\psi'\rangle$.

We record several useful facts about the consistency measure and the state-dependent distance without proof. Readers are referred to Sections 4.4 and 4.5 in [NW19] for additional discussion and proofs.

Fact 5.17 (Fact 4.13 and Fact 4.14 in [NW19]). *For POVMs $\{A_a^x\}$ and $\{B_a^x\}$, the following hold.*

1. *If $A_a^x \otimes I_{\mathrm{B}} \simeq_\delta I_{\mathrm{A}} \otimes B_a^x$ then $A_a^x \otimes I_{\mathrm{B}} \approx_\delta I_{\mathrm{A}} \otimes B_a^x$.*

2. *If $A_a^x \otimes I_{\mathrm{B}} \approx_\delta I_{\mathrm{A}} \otimes B_a^x$ and $\{A_a^x\}$ and $\{B_a^x\}$ are projective measurements, then $A_a^x \otimes I_{\mathrm{B}} \simeq_\delta I_{\mathrm{A}} \otimes B_a^x$.*

3. *If $A_a^x \otimes I_{\mathrm{B}} \approx_\delta I_{\mathrm{A}} \otimes B_a^x$ and either $\{A_a^x\}$ or $\{B_a^x\}$ is a projective measurement, then $A_a^x \otimes I_{\mathrm{B}} \simeq_{\delta^{1/2}} I_{\mathrm{A}} \otimes B_a^x$.*

Fact 5.18 (Fact 4.20 in [NW19]). *Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be finite sets, and let D be a distribution over question pairs (x, y). Let $\{A_{a,b}^x\}$ and $\{B_{a,b}^x\}$ be POVMs whose outcomes range over the product set $\mathcal{A} \times \mathcal{B}$. Suppose a set of operators $\{C_{a,c}^y\}$, whose outcomes range over the product set $\mathcal{A} \times \mathcal{C}$, satisfies the condition $\sum_{a,c} (C_{a,c}^y)^\dagger C_{a,c}^y \leq I$ for all y. If $A_{a,b}^x \approx_\delta B_{a,b}^x$ on average over x sampled from the corresponding marginal of distribution D, then $C_{a,c}^y A_{a,b}^x \approx_\delta C_{a,c}^y B_{a,b}^x$ on average over (x, y) sampled from D.*

Proof. Fix questions x, y and answers $a \in \mathcal{A}, b \in \mathcal{B}$. We have then that

$$\sum_c \left\| (C_{a,c}^y A_{a,b}^x - C_{a,c}^y B_{a,b}^x) |\psi\rangle \right\|^2 = \sum_c \langle \psi | (A_{a,b}^x - B_{a,b}^x)^\dagger (C_{a,c}^y)^\dagger (C_{a,c}^y)(A_{a,b}^x - B_{a,b}^x) |\psi\rangle \tag{30}$$

$$\leq \langle \psi | (A_{a,b}^x - B_{a,b}^x)^\dagger (A_{a,b}^x - B_{a,b}^x) |\psi\rangle \tag{31}$$

$$= \left\| (A_{a,b}^x - B_{a,b}^x) |\psi\rangle \right\|^2 \tag{32}$$

where the inequality follows from the fact that $\sum_c (C_{a,c}^y)^\dagger C_{a,c}^y \leq \sum_{a,c} (C_{a,c}^y)^\dagger C_{a,c}^y \leq I$. Thus we obtain the desired conclusion

$$\mathop{\mathbb{E}}_{(x,y) \sim D} \sum_{a,b,c} \left\| (C_{a,c}^y A_{a,b}^x - C_{a,c}^y B_{a,b}^x) |\psi\rangle \right\|^2 \leq \mathop{\mathbb{E}}_{(x,y) \sim D} \sum_{a,b} \left\| (A_{a,b}^x - B_{a,b}^x) |\psi\rangle \right\|^2 \leq \delta. \tag{33}$$

$\square$

Fact 5.19 (Triangle inequality, Fact 4.28 in [NW19]). *If $A_a^x \approx_\delta B_a^x$ and $B_a^x \approx_\epsilon C_a^x$, then $A_a^x \approx_{\delta+\epsilon} C_a^x$.*

Fact 5.20 (Triangle inequality for "$\simeq$", Fact 4.29 in [NW19]). *If $A_a^x \otimes I_B \simeq_\delta I_A \otimes B_a^x$, $C_a^x \otimes I_B \simeq_\delta I_A \otimes B_a^x$, and $C_a^x \otimes I_B \simeq_\delta I_A \otimes D_a^x$, then $A_a^x \otimes I_B \simeq_\delta I_A \otimes D_a^x$.*

Fact 5.21 (Data processing, Fact 4.26 in [NW19]). *Suppose $A_a^x \otimes I_B \simeq_\delta I_A \otimes B_a^x$. Then $A_{[f(\cdot)=b]}^x \otimes I_B \simeq_\delta I_A \otimes B_{[f(\cdot)=b]}^x$.*

The state-dependent distance is the right tool for reasoning about the closeness of measurement operators in a strategy. The following lemma ensures that, when two families of measurements are close on a state, changing from one family of measurement to the other only introduces a small error to the value of the strategy.

Lemma 5.22. *Let $\{A_{a,b}^x\}$, $\{B_{a,b,c}^x\}$, $\{C_{a,c}^x\}$ be POVMs. Suppose $\{B_{a,b,c}^x\}$ is projective, and*

$$A_{a,b}^x \otimes I_B \approx_\delta I_A \otimes B_{a,b}^x \,,$$
$$C_{a,c}^x \otimes I_B \approx_\delta I_A \otimes B_{a,c}^x \,.$$

Then the following approximate commutation relation holds:

$$[A_{a,b}^x, C_{a,c}^x] \otimes I_B \approx_\delta 0 \,.$$

Proof. Applying Fact 5.18 to $C_{a,c}^x \otimes I_B \approx_\delta I_A \otimes B_{a,c}^x$ and $\{A_{a,b}^x \otimes I_B\}$, we have

$$A_{a,b}^x C_{a,c}^x \otimes I_B \approx_\delta A_{a,b}^x \otimes B_{a,c}^x. \tag{34}$$

Similarly, applying Fact 5.18 to $A_{a,b}^x \otimes I_B \approx_\delta I_A \otimes B_{a,b}^x$ and $\{I_A \otimes B_{a,c}^x\}$, and using the fact that $\{B_{a,b,c}^x\}$ is projective, we have

$$A_{a,b}^x \otimes B_{a,c}^x \approx_\delta I_A \otimes B_{a,c}^x B_{a,b}^x$$
$$= I_A \otimes B_{a,b,c}^x. \tag{35}$$

Combining Equations (34) and (35), we have

$$A_{a,b}^x C_{a,c}^x \otimes I_B \approx_\delta I_A \otimes B_{a,b,c}^x. \tag{36}$$

A similar argument gives

$$C_{a,c}^x A_{a,b}^x \otimes I_B \approx_\delta I_A \otimes B_{a,b,c}^x. \tag{37}$$

The claim follows from Equations (36) and (37). $\square$

The following lemma is a slightly modified version of [NW19, Fact 4.34].

Lemma 5.23. *Let $k \geq 0$ be a constant and $\varepsilon > 0$. Let $\mathcal{X}$ be a finite set and μ a distribution over $\mathcal{X}$. For each $1 \leq i \leq k$ let $\mathcal{G}_i$ be a set of functions $g_i : \mathcal{Y} \to \mathcal{R}_i$ and for each $x \in \mathcal{X}$ let $\{G_g^{i,x}\}_{g \in \mathcal{G}_i}$ be a projective measurement. Suppose that for all $i \in \{1, \ldots, k\}$, $\mathcal{G}_i$ satisfies the following property: for any two $g_i \neq g_i' \in \mathcal{G}_i$, the probability that $g_i(y) = g_i'(y)$ over a uniformly random $y \in \mathcal{Y}$ is at most ε.*

Let $\{A_{g_1, g_2, \ldots, g_k}^x\}$ be a projective measurement with outcomes $(g_1, \ldots, g_k) \in \mathcal{G}_1 \times \cdots \times \mathcal{G}_k$. For each $1 \leq i \leq k$, suppose that on average over $x \sim \mu$ and $y \in \mathcal{Y}$ sampled uniformly at random,

$$A_{[\text{eval}_y(\cdot)_i = a_i]}^x \otimes I_\text{B} \simeq_\delta I_\text{A} \otimes G_{[\text{eval}_y(\cdot) = a_i]}^{i, x} . \tag{38}$$

Define the POVM family $\{C_{g_1, g_2, \ldots, g_k}^x\}$, for $x \in \mathcal{X}$, by

$$C_{g_1, g_2, \ldots, g_k}^x = G_{g_k}^{k, x} \cdots G_{g_2}^{2, x} G_{g_1}^{1, x} G_{g_2}^{2, x} \cdots G_{g_k}^{k, x} .$$

Then on average over $x \sim \mu$ and $y \in \mathcal{Y}$ sampled uniformly at random,

$$A_{[\text{eval}_y(\cdot) = (a_1, a_2, \ldots, a_k)]}^x \otimes I_\text{B} \simeq_{(\delta + \varepsilon)^{1/2}} I_\text{A} \otimes C_{[\text{eval}_y(\cdot) = (a_1, a_2, \ldots, a_k)]}^x . \tag{39}$$

Proof. The proof is identical to the one given in [NW19, Fact 4.34], with the only modification needed to insert the dependence on x for all measurements considered. $\square$

5.3 Self-testing

Definition 5.24 (Partial strategies). A strategy $\mathscr{S} = (|\psi\rangle, A, B)$ is a *partial strategy* for a game $\mathfrak{G}$ if A and B only specify POVM A^x and B^y for a subset of the questions x, y in $\mathfrak{G}$ (called the *question set of the strategy $\mathscr{S}$*). A strategy $\mathscr{S}' = (|\psi\rangle, A', B')$ *extends* $\mathscr{S}$ if $(A')^x = A^x$ and $(B')^y = B^y$ for every x, y in the question set of $\mathscr{S}$. A *full strategy* $\mathscr{S}$ for a game $\mathfrak{G}$ is one whose question set is the entire question alphabet of $\mathfrak{G}$.

Definition 5.25 (Self-testing). Let $\mathscr{S} = (|\psi\rangle, A, B)$ be a partial strategy for a game $\mathfrak{G} = (\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}, \mu, D)$ and $\delta : [0, 1] \to \mathbb{R}_+$. We say that $\mathfrak{G}$ is a *self-test for $\mathscr{S}$ with robustness* $\delta(\varepsilon)$ if the following hold:

- (Completeness) There exists a (full) strategy $\mathscr{S}_{\text{FULL}}$ that extends $\mathscr{S}$ such that $\text{val}^*(\mathfrak{G}, \mathscr{S}_{\text{FULL}}) = 1$.

- (Soundness) Let $\hat{\mathscr{S}} = (\hat{\psi}, \hat{A}, \hat{B})$ be a strategy that wins $\mathfrak{G}$ with probability $1 - \varepsilon$, for some $\varepsilon \geq 0$. Then there exists a local isometry $\phi = \phi_A \otimes \phi_B$ and a state $|\text{AUX}\rangle$ such that

$$\|\phi(|\hat{\psi}\rangle) - |\psi\rangle |\text{AUX}\rangle\|^2 \leq \delta(\varepsilon) .$$

Furthermore, letting $\tilde{A}_a^x = \phi_A \hat{A}_a^x \phi_A^*$ and $\tilde{B}_b^y = \phi_B \hat{B}_b^y \phi_B^*$, we have

$$\tilde{A}_a^x \otimes I_\text{B} \approx_{\delta(\varepsilon)} (A_a^x \otimes I_{\text{AUX}}) \otimes I_\text{B}$$

on state $|\psi\rangle |\text{AUX}\rangle$, where x is drawn from the marginal distribution of μ on one of the players. A similar relation holds for operators $\tilde{B}_b^y$ and B_b^y.

5.4 Normal form verifiers

We introduce a normal form for verifiers in nonlocal games. The normal form uses Turing machines to specify the two actions performed by the verifier in a game: the generation of questions and the verification of answers. For the generation of questions, we use the formalism of samplers introduced in Section 4.2. The normal form for verifiers gives a uniform method to specify an infinite family of nonlocal games.

Definition 5.26 (Decider). A *decider* is a 5-input Turing machine $\mathcal{D}$ that on all inputs of the form (n, x, y, a, b) where n is an integer and $x, y, a, b \in \{0, 1\}^*$, $\mathcal{D}$ halts and returns a single bit. Let $\mathsf{TIME}_{\mathcal{D}}(n)$ denote the time complexity of $\mathcal{D}$ on inputs of the form $(n, \ldots)$. When the decider $\mathcal{D}$ outputs 0 we say that it *rejects*, otherwise we say that it *accepts*. Furthermore, we call the input n to a decider the *index*.

Definition 5.27. A *normal form verifier* is a pair $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ where $\mathcal{S}$ is a sampler with field size $q(n) = 2$ and $\mathcal{D}$ is a decider. The description length of $\mathcal{V}$ is defined to be $|\mathcal{V}| = |\mathcal{S}| + |\mathcal{D}|$, the sum of the description lengths of $\mathcal{S}$ and $\mathcal{D}$.

Normal form verifiers specify an infinite family of nonlocal games indexed by natural numbers in the following way.

Definition 5.28. Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier. For $n \in \mathbb{N}$, we define the following nonlocal game $\mathcal{V}_n$ to be the *n-th game corresponding to the verifier* $\mathcal{V}$. The question sets $\mathcal{X}$ and $\mathcal{Y}$ are $\{0, 1\}^{\mathsf{RAND}_{\mathcal{S}}(n)}$. The answer sets $\mathcal{A}$ and $\mathcal{B}$ are $\{0, 1\}^{\mathsf{TIME}_{\mathcal{D}}(n)}$. The question distribution is the distribution $\mu_{\mathcal{S}, n}$ specified in Definition 4.14. The decision predicate is the function computed by $\mathcal{D}(n, \cdot, \cdot, \cdot, \cdot)$, when the last four inputs are restricted to $\mathcal{X} \times \mathcal{Y} \times \mathcal{A} \times \mathcal{B}$. The value of the game is denoted by $\mathrm{val}^*(\mathcal{V}_n)$.

We note that the game $\mathcal{V}_n$ is well-defined since for a normal form verifier the distribution $\mu_{\mathcal{S}, n}$ is supported on $\{0, 1\}^{\mathsf{RAND}_{\mathcal{S}}(n)} \times \{0, 1\}^{\mathsf{RAND}_{\mathcal{S}}(n)}$ and a normal form decider always halts with a single-bit output.

Definition 5.29 (Verifier with commuting strategy). Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier. For $v : \mathbb{N} \to [0, 1]$ say that $\mathcal{V}$ has a *value-v commuting strategy* if for all $n \in \mathbb{N}$, the game $\mathcal{V}_n$ has a value-$v(n)$ commuting strategy.

6 Types

We augment the definition of conditionally linear functions with a construct we call *types*. A type t is an element of a *type set* $\mathcal{T}$, and a $\mathcal{T}$-typed family of conditionally linear functions is a collection $\{L_t\}_{t\in\mathcal{T}}$ containing a CL function L_t for each type $t \in \mathcal{T}$. The utility of this definition is that it allows us to define another object, namely conditionally linear distributions parameterized by an undirected graph $G = (\mathcal{T}, E)$ on the set of types known as a *type graph*. Given two $\mathcal{T}$-typed families of conditionally linear functions $\{L_u\}_{u\in\mathcal{T}}, \{R_v\}_{v\in\mathcal{T}}$, the $(\mathcal{T}, G)$-typed conditionally linear distribution corresponding to them is the distribution which samples a pair of types (u, v) uniformly at random from the edges of G (with each endpoint having equal probability as being chosen for u or v, respectively) and then samples (x, y) from μ_{L_u, R_v}. The output is the pair $((u, x), (v, y))$.

The normal form verifiers we present in the paper frequently use typed CL distributions to sample their questions, rather than untyped CL distributions. Types allow us to model the parts of their question distributions which are unstructured and unsuitable for being sampled from CL distributions. A common use of types is to allow the verifier to use previously defined games as subroutines. Here, the type helps indicate which subroutine the verifier selects, and an edge in the type graph between two different types allows us to introduce a test that cross-checks the results of one subroutine with the results of another.

Finally, we show how to convert any typed CL distribution into an equivalent (in the precise sense defined below) untyped CL distribution with two additional levels, a technique we call *detyping*. This entails showing how to "simulate" the *graph distribution* of $G = (\mathcal{T}, E)$, i.e. the uniform distribution on its edges, using an untyped CL distribution. The simulation we give is based on rejection sampling and is only approximate: its quality degrades exponentially with the number of types in $\mathcal{T}$. As a result, we will ensure throughout the paper that all type sets we consider are of a small, in fact generally constant, size.

This section is organized as follows. In Section 6.1 we define typed variants of CL distributions, samplers, deciders, and verifiers. In Section 6.2 we define a CL distribution which samples from the graph distribution of a given graph $G = (\mathcal{T}, E)$. In Section 6.3 we define a canonical way to detype typed samplers, deciders, and verifiers using the graph sampler from Section 6.2. We then prove the main result of the section, Lemma 6.18, which relates the value of the detyped normal form verifier to the value of the original typed verifier.

6.1 Typed samplers, deciders, and verifiers

Definition 6.1 (Typed conditionally linear functions). Let $\mathcal{T}$ be a finite set and V be $\mathbb{F}^n$ for some integer $n \geq 0$. A $\mathcal{T}$-*typed family of ℓ-level conditionally linear functions* (implicitly, *on V*) is a collection $\{L_t\}_{t\in\mathcal{T}}$ such that, for each $t \in \mathcal{T}$, L_t is an ℓ-level conditionally linear function on V.

Definition 6.2 (Graph distribution). Let $G = (U, E)$ be an undirected graph with vertex set U and edge set E. Edges in E are written as multisets $\{u, v\}$ of two vertices; the case $u = v$ represents a self-loop. Suppose there are m edges, k of which are self-loops. Then the *graph distribution μ_G of G* is the distribution over $U \times U$ such that for every $(u, v) \in U \times U$,

$$\mu_G(u, v) = \begin{cases} 1/(2m - k) & \text{if } \{u, v\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

This is identical to the uniform distribution over pairs $(u, v) \in U \times U$ such that $\{u, v\} \in E$.

Definition 6.3 (Typed conditionally linear distributions). Let $\mathcal{T}$ be a type set and $L = \{L_u\}_{u \in \mathcal{T}}$, $R = \{R_v\}_{v \in \mathcal{T}}$ be $\mathcal{T}$-typed families of conditionally linear functions on V. Let $G = (\mathcal{T}, E)$ be a graph with vertex set $\mathcal{T}$. The $(\mathcal{T}, G)$-*typed conditionally linear distribution* $\mu_{L,R}^G$ corresponding to (L, R) is the distribution over pairs $((u, x), (v, y))$, where (u, v) is drawn from μ_G and (x, y) is drawn from μ_{L_u, R_v}.

Definition 6.4 (Typed conditionally linear samplers). Let $q : \mathbb{N} \to \mathbb{N}$ be an admissible field size function and $s : \mathbb{N} \to \mathbb{N}$ be a function. Let $\mathcal{T}$ be a finite type set. A 6-input Turing machine $\mathcal{S}$ is a $\mathcal{T}$-*typed, ℓ-level conditionally linear sampler with field size $q(n)$ and dimension $s(n)$* if for all $n \in \mathbb{N}$, letting $q = q(n)$ and $s = s(n)$, there exist $\mathcal{T}$-typed families of ℓ-level conditionally linear functions $\{L_t^{A,n}\}_{t \in \mathcal{T}}$ and $\{L_t^{B,n}\}_{t \in \mathcal{T}}$ on $V = \mathbb{F}_q^s$ where $t \in \mathcal{T}, w \in \{A, B\}$, the conditionally linear function $L_t^{w,n}$ has marginal functions $\{L_{t,\leq j}^{w,n}\}$ and factor spaces $\{V_{t,j,u}^{w,n}\}$ satisfying the conditions of Lemma 4.4, and for all $t \in \mathcal{T}$, $w \in \{A, B\}$, $j \in \{1, \ldots, \ell\}$, and $z \in V$:

- On input $(n, \text{DIMENSION})$, the sampler $\mathcal{S}$ returns the dimension $s(n)$.

- On input $(n, w, \text{MARGINAL}, j, z, t)$, the sampler $\mathcal{S}$ returns the binary representation of $L_{t,\leq j}^{w,n}(z)$.

- On input $(n, w, \text{LINEAR}, j, u, y, t)$, the sampler $\mathcal{S}$ outputs the binary representation of $L_{t,j,u}^{w,n}(y)$,

- On input $(n, w, \text{FACTOR}, j, u, t)$, the sampler $\mathcal{S}$ returns the factor space $V_{t,j,u}^{w,n}$ of $L_t^{w,n}$, represented as an indicator vector in $\{0, 1\}^s$.

We call $\mathbb{F}_{q(n)}^{s(n)}$ the *ambient space* of $\mathcal{S}$. We call $\{L_t^{A,n}\}, \{L_t^{B,n}\}$ the *CL functions of $\mathcal{S}$ on index n*. The *time complexity* of $\mathcal{S}$, denoted $\text{TIME}_{\mathcal{S}}(n)$, is the number of steps before $\mathcal{S}$ halts for index n. The *randomness complexity* of $\mathcal{S}$, denoted by $\text{RAND}_{\mathcal{S}}(n)$, is defined as the quantity $s(n) \log q(n)$.

We assume that types $t \in \mathcal{T}$ are represented using binary strings of length at most $\lceil \log |\mathcal{T}| \rceil$; if a type t is given as input to the sampler $\mathcal{S}$ and is not an element of $\mathcal{T}$, then the sampler returns 0. Furthermore, as described in Remark 4.13 for un-typed samplers, we write typed samplers with different numbers of arguments depending on the input.

Definition 6.5 (Distribution of a typed sampler). Let $\mathcal{S}$ be a $\mathcal{T}$-typed sampler and $G = (\mathcal{T}, E)$ be a graph. Let $L^w = \{L_t^w\}_t$ for $w \in \{A, B\}$ be the CL functions of $\mathcal{S}$ on index n. The *distribution of sampler $\mathcal{S}$ with graph G on index n*, denoted $\mu_{\mathcal{S},n}^G$, is the $(\mathcal{T}, G)$-typed conditionally linear distribution corresponding to (L^A, L^B).

Definition 6.6 (Downsizing typed CL samplers). Let $\mathcal{S}$ be a $(\mathcal{T}, G)$-typed sampler. The downsized $(\mathcal{T}, G)$-typed sampler $\kappa(\mathcal{S})$ is defined as in Definition 4.15 with the only difference that the type t is included as part of the input to the sampler, as in Definition 6.4. (The type set $\mathcal{T}$ and type graph G themselves are unchanged.)

Lemma 6.7. *Let $\mathcal{S}$ be a $(\mathcal{T}, G)$-typed ℓ-level CL sampler, for some finite set $\mathcal{T}$, type graph G, and integer $\ell \geq 0$. Let $q(n)$ and $s(n)$ be as in Definition 6.4. Then $\kappa(\mathcal{S})$ defined in Definition 6.6 is a $(\mathcal{T}, G)$-typed ℓ-level CL sampler with field size 2, dimension $s(n) \log q(n)$, and randomness and time complexities*

$$\text{RAND}_{\kappa(\mathcal{S})}(n) = \text{RAND}_{\mathcal{S}}(n), \quad \text{TIME}_{\kappa(\mathcal{S})}(n) = O(\text{TIME}_{\mathcal{S}}(n) \log q(n)).$$

Furthermore, for every integer $n \geq 1$, the CL functions of $\kappa(\mathcal{S})$ on index n are $\{(L_t^{w,n})^\kappa\}_{w \in \{A,B\}, t \in \mathcal{T}}$, as defined in Definition 4.8.

Proof. The proof is analogous to the proof of Lemma 4.16, and we omit it. □

Definition 6.8 (Typed decider). A *typed decider* is a 7-input Turing machine $\mathcal{D}$ that on all inputs of the form $(n, \mathsf{u}, x, \mathsf{v}, y, a, b)$ where n is an integer and $\mathsf{u}, x, \mathsf{v}, y, a, b \in \{0,1\}^*$, $\mathcal{D}$ halts and returns a single bit. When $\mathcal{D}$ returns 0 we say that it *rejects*, otherwise we say that it *accepts*. We use $\mathsf{TIME}_{\mathcal{D}}(n)$ to denote the time complexity of $\mathcal{D}$ on inputs of the form $(n, \ldots)$.

Definition 6.9. Let $\mathcal{T}$ be a finite set and let $G = (\mathcal{T}, E)$ be a graph. A $(\mathcal{T}, G)$-*typed normal form verifier* is a pair $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ where $\mathcal{S}$ is a $\mathcal{T}$-typed sampler with field size $q(n) = 2$ and $\mathcal{D}$ is a typed decider.

Definition 6.10. Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a $(\mathcal{T}, G)$-typed normal form verifier. For $n \in \mathbb{N}$, we define the following nonlocal game $\mathcal{V}_n$ to be the *n-th game corresponding to the verifier* $\mathcal{V}$. The question sets $\mathcal{X}$ and $\mathcal{Y}$ are $\mathcal{T} \times \{0,1\}^{\mathsf{RAND}_{\mathcal{S}}(n)}$. The answer sets $\mathcal{A}$ and $\mathcal{B}$ are $\{0,1\}^{\mathsf{TIME}_{\mathcal{D}}(n)}$. The question distribution is the distribution $\mu_{\mathcal{S},n}^G$ specified in Definition 6.5. The decision predicate is the function computed by $\mathcal{D}(n, \cdot, \cdot, \cdot, \cdot)$, when the last four inputs are restricted to $\mathcal{X} \times \mathcal{Y} \times \mathcal{A} \times \mathcal{B}$. The value of the game is denoted by $\mathrm{val}^*(\mathcal{V}_n)$.

For $w \in \{\mathrm{A}, \mathrm{B}\}$ and a question (u, x) to player w we refer to u as the *question type* and x as the *question content*.

6.2 Graph distributions

We describe a construction of conditionally linear distributions which sample from the graph distribution (see Definition 6.2) of a graph $G = (U, E)$. We begin with a technical definition, followed by the definition of the conditionally linear distribution.

Definition 6.11 (Neighbor indicator). Given a graph $G = (U, E)$, the *neighbor indicator* of a vertex $u \in U$ is the vector $\mathrm{neigh}_G(u) \in \mathbb{F}_2^U$ in which, for all $v \in U$,

$$\mathrm{neigh}_G(u)_v = \begin{cases} 1 & \text{if } \{u, v\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

In addition, the $\mathbb{F}_2$-*encoding* of a vertex $u \in U$ is the vector $\mathrm{enc}_G(u) \in \mathbb{F}_2^U \times \mathbb{F}_2^U$ given by $\mathrm{enc}_G(u) = (e_u, \mathrm{neigh}_G(u))$, where e_u is the standard basis vector with a 1 in the u-th position and 0's everywhere else.

Definition 6.12 (Graph sampler). Let $G = (U, E)$ be a graph with n vertices. Then the *conditionally linear functions corresponding to G* are the pair of functions $L_G^{\mathrm{A}}, L_G^{\mathrm{B}}$ on linear space V_G specified in Fig. 1 where $V_G = V_{\mathrm{VA}} \oplus V_{\mathrm{NA}} \oplus V_{\mathrm{VB}} \oplus V_{\mathrm{NB}}$.

These conditionally linear functions do not simulate the graph distribution in the sense of sampling directly from it. The following proposition, however, does show a sense in which these functions simulate the graph distribution, namely via rejection sampling.

Proposition 6.13 (Simulating the graph distribution). *Let $G = (U, E)$ be a graph with n vertices and m edges, k of which are self-loops. Let $L_G^{\mathrm{A}}, L_G^{\mathrm{B}}$ be the conditionally linear functions corresponding to G (see Figure 1 for the definition and associated notation). Let $(x, y) \sim \mu_{L_G^{\mathrm{A}}, L_G^{\mathrm{B}}}$. Consider the event $\mathcal{E}_G$ that there exists $u, v \in U$ such that the following two statements are true.*

(i) $x^{V_{\mathrm{VA}} \oplus V_{\mathrm{NA}}} = \mathrm{enc}_G(u)$ *and* $y^{V_{\mathrm{VB}} \oplus V_{\mathrm{NB}}} = \mathrm{enc}_G(v)$,

53

Subspaces			
V_{VA}	V_{NA}	V_{VB}	V_{NB}
$\mathbb{F}_2^U$	$\mathbb{F}_2^U$	$\mathbb{F}_2^U$	$\mathbb{F}_2^U$

Conditionally linear function L_G^{A}

1st factor subspace 1st linear function	$V_{\mathrm{VA}} \oplus V_{\mathrm{NA}}$ Identity function
2nd factor subspace 2nd linear functions	$V_{\mathrm{VB}} \oplus V_{\mathrm{NB}}$ For all $x \in V_{\mathrm{VA}} \oplus V_{\mathrm{NA}}$, suppose there exists a $u \in U$ such that $x = \mathrm{enc}_G(u)$. Then for all $y \in V_{\mathrm{VB}} \oplus V_{\mathrm{NB}}$, $L_{G,2,x}^{\mathrm{A}}$ zeroes out all entries of y except for $(y^{V_{\mathrm{NB}}})_u$. Otherwise, $L_{G,2,x}^{\mathrm{A}} = 0$.

Conditionally linear function L_G^{B}

1st factor subspace 1st linear function	$V_{\mathrm{VB}} \oplus V_{\mathrm{NB}}$ Identity function
2nd factor subspace 2nd linear functions	$V_{\mathrm{VA}} \oplus V_{\mathrm{NA}}$ Similarly defined as those for L_G^{A} by swapping V_{VA} and V_{NA} with V_{VB} and V_{NB} respectively.

Figure 1: Specification of the conditionally linear functions corresponding to G.

(ii) $(x^{V_{\mathrm{NB}}})_u = (y^{V_{\mathrm{NA}}})_v = 1$.

Then

1. $\mathrm{Pr}_{x,y}(\mathcal{E}_G) = (2m - k)/16^n$.

2. *Conditioned on $\mathcal{E}_G$, (u,v) are distributed as the graph distribution of G (see Definition 6.2).*

Note that $\mathcal{E}_G$ occurs if and only if both $x^{V_{\mathrm{NB}}}$ and $y^{V_{\mathrm{NA}}}$ are nonzero. In particular, if x and y are sampled from $\mu_{L_G^{\mathrm{A}}, L_G^{\mathrm{B}}}$ and given to the respective players, then at least one of them knows when the event $\mathcal{E}_G$ does not occur.

Proof. Let z be drawn uniformly at random from $V_{\mathrm{VA}} \oplus V_{\mathrm{NA}} \oplus V_{\mathrm{VB}} \oplus V_{\mathrm{NB}}$, and let $x = L_G^{\mathrm{A}}(z)$ and $y = L_G^{\mathrm{B}}(z)$. Then with probability $n^2/16^n$, there exist $u, v \in U$ such that

$$x^{V_{\mathrm{VA}} \oplus V_{\mathrm{NA}}} = \mathrm{enc}_G(u) \quad \text{and} \quad y^{V_{\mathrm{VB}} \oplus V_{\mathrm{NB}}} = \mathrm{enc}_G(v) \ .$$

Conditioned on this occurring, u and v are distributed as independent, uniformly random vertices in U. If we further condition on $\{u, v\} \in E$, which occurs with probability $(2m - k)/n^2$, then by definition, (u, v) is distributed as the graph distribution of G. But this event is exactly the event that $\mathcal{E}_G$ holds on (x, y), establishing the proposition. $\qquad\square$

6.3 Detyping typed verifiers

We give a canonical method for taking a typed normal form verifier and producing an untyped normal form verifier which simulates it. Throughout this section, $\mathcal{T}$ denotes a finite set, $G = (\mathcal{T}, E)$ denotes a graph, and L_G^A, L_G^B denote the conditionally linear functions corresponding to G acting on the vector space $V_G = V_{VA} \oplus V_{NA} \oplus V_{VB} \oplus V_{NB}$ of dimension $4 \cdot |\mathcal{T}|$ over $\mathbb{F}_2$, as in Definition 6.12.

Definition 6.14 (Detyped CL functions). Let $L^A = \{L_t^A\}, L^B = \{L_t^B\}$ be $\mathcal{T}$-typed families of ℓ-level conditionally linear functions on V. We define the *detyped CL functions corresponding to* (L^A, L^B) *on* G to be the pair of $(\ell + 2)$-level CL functions $(R^A, R^B) = \mathrm{detype}_G(L^A, L^B)$ on linear space $V_{DETYPE} = V_G \oplus V$ as follows. For $w \in \{A, B\}$, and $z \in L_G^w(V_G)$, define the family of ℓ-level CL functions $\{L_z^w\}$ on V as

$$L_z^w = \begin{cases} 0 & \text{if } z^{V_{N\overline{w}}} = 0, \\ L_t^w & \text{otherwise, for } z^{V_{Vw}} = e_t. \end{cases}$$

We note that when $z^{V_{N\overline{w}}}$ is nonzero, it is always the case that $z^{V_{Vw}} = e_t$ for some type t, by Definition 6.12. For $w \in \{A, B\}$, R^w is the concatenation of L_G^w and $\{L_z^w\}_z$ (cf. Lemma 4.5).

Definition 6.15 (Detyped samplers). Let $\mathcal{S}$ be a $\mathcal{T}$-typed sampler. For each $n \in \mathbb{N}$, let $\{L_t^{A,n}\}, \{L_t^{B,n}\}$ be the CL functions of $\mathcal{S}$ with graph G on index n, and set $(R^{A,n}, R^{B,n}) = \mathrm{detype}_G(L^{A,n}, L^{B,n})$. Then the *detyped sampler* $\mathrm{detype}_G(\mathcal{S})$ is the (standard) sampler whose CL functions on index n are $R^{A,n}, R^{B,n}$. Its dimension function is $s_{DETYPE}(n) = 4|\mathcal{T}| + s(n)$.

Definition 6.16 (Detyped deciders). Let $\mathcal{D}$ be a typed decider. We define the *detyped decider* $\mathrm{detype}_G(\mathcal{D})$ to be the (standard) decider that behaves as follows: on input (n, x, y, a, b), it attempts to parse $x = (x', x'')$, $y = (y', y'') \in V_G \times \{0, 1\}^*$ (using a canonical scheme for representing pairs of strings). If it cannot, it accepts. Otherwise, suppose that there exists $\{u, v\} \in E_G$ such that, using notation from Definition 6.11,

$$x' = (e_u, \mathrm{neigh}_G(u), 0, e_u), \quad y' = (0, e_v, e_v, \mathrm{neigh}_G(v)) \in V_{VA} \oplus V_{NA} \oplus V_{VB} \oplus V_{NB}.$$

Then it returns the output of $\mathcal{D}$ on input $(n, u, x'', v, y'', a, b)$. Otherwise, it accepts.

Definition 6.17 (Detyped verifiers). Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a $(\mathcal{T}, G)$-typed normal form verifier. We define the *detyped verifier*, denoted by $\mathrm{detype}(\mathcal{V})$, to be the (standard) normal form verifier $(\mathrm{detype}_G(\mathcal{S}), \mathrm{detype}_G(\mathcal{D}))$.

Lemma 6.18 (Typed verifiers to detyped verifiers). *Let* $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ *be a* $(\mathcal{T}, G)$*-typed normal form verifier. The detyped verifier* $\mathrm{detype}(\mathcal{V}) = (\mathrm{detype}_G(\mathcal{S}), \mathrm{detype}_G(\mathcal{D}))$ *satisfies the following properties: for all* $n \in \mathbb{N}$,

1. (**Completeness**) *If* $\mathcal{V}_n$ *has a value-1 PCC strategy, then* $\mathrm{detype}(\mathcal{V})_n$ *has a value-1 PCC strategy.*

2. (**Soundness**) *If* $\mathrm{val}^*(\mathrm{detype}(\mathcal{V})_n) \geq 1 - \varepsilon$, *then* $\mathrm{val}^*(\mathcal{V}_n) \geq 1 - 16^{|\mathcal{T}|} \cdot \varepsilon$. *Furthermore,*

$$\mathscr{E}(\mathrm{detype}(\mathcal{V})_n, 1 - \varepsilon) \geq \mathscr{E}(\mathcal{V}_n, 1 - 16^{|\mathcal{T}|} \cdot \varepsilon).$$

3. (**Sampler parameters**) *If* $\mathcal{S}$ *is an* ℓ*-level sampler, then* $\mathrm{detype}_G(\mathcal{S})$ *is an* $(\ell + 2)$*-level sampler. The randomness and time complexities of* $\mathrm{detype}_G(\mathcal{S})$ *satisfy the following:*

$$\mathsf{RAND}_{\mathrm{detype}_G(\mathcal{S})}(n) = 4 \cdot |\mathcal{T}| + \mathsf{RAND}_{\mathcal{S}}(n)$$
$$\mathsf{TIME}_{\mathrm{detype}_G(\mathcal{S})}(n) = \mathrm{poly}(|\mathcal{T}|, \mathsf{TIME}_{\mathcal{S}}(n)).$$

4. **(*Decider complexity*)** The decider $\mathrm{detype}_G(\mathcal{D})$ has time complexity $\mathrm{poly}(|\mathcal{T}|, \mathrm{TIME}_\mathcal{D}(n))$.

5. **(*Efficient computability*)** The descriptions of $\mathrm{detype}_G(\mathcal{S})$ and $\mathrm{detype}_G(\mathcal{D})$ are polynomial time computable from the description of G and the descriptions of $\mathcal{S}$ and $\mathcal{D}$, respectively.

Proof. Throughout this proof, we fix an index n. Let $s = s(n)$ be the dimension of $\mathcal{S}$. The ambient space of $\mathcal{S}$ is $V = \mathbb{F}_2^s$ and the ambient space of $\mathrm{detype}_G(\mathcal{S})$ is $V_{\mathrm{DETYPE}} = V_G \oplus V$. Let $\mathsf{u}, \mathsf{v} \in \mathcal{T}$. For this proof, we introduce the notation

$$\mathrm{view}^\mathrm{A}(\mathsf{u}) = (e_\mathsf{u}, \mathrm{neigh}_G(\mathsf{u}), 0, e_\mathsf{u}), \ \mathrm{view}^\mathrm{B}(\mathsf{v}) = (0, e_\mathsf{v}, e_\mathsf{v}, \mathrm{neigh}_G(\mathsf{v})) \in V_{\mathrm{VA}} \oplus V_{\mathrm{NA}} \oplus V_{\mathrm{VB}} \oplus V_{\mathrm{NB}} .$$

Supposing that players A and B receive x and y in V_{DETYPE}, and supposing that (x, y) satisfies event $\mathcal{E}_G$ from Proposition 6.13, then $x^{V_G} = \mathrm{view}^\mathrm{A}(\mathsf{u})$ and $y^{V_G} = \mathrm{view}^\mathrm{B}(\mathsf{v})$ for some $\{\mathsf{u}, \mathsf{v}\} \in E$.

Completeness. Let $\mathscr{S} = (|\psi\rangle, A, B)$ be a value-1 PCC strategy for $\mathcal{V}_n$. We construct a PCC strategy $\mathscr{S}^{\mathrm{DETYPE}}$ for $\mathrm{detype}(\mathcal{V})_n$ with value 1. This strategy also uses the state $|\psi\rangle$. When a player receives a question, they perform measurements described as follows.

Player A: given $x \in V_{\mathrm{DETYPE}}$ the player checks if for some $\mathsf{u} \in \mathcal{T}$, $x^{V_G} = \mathrm{view}^\mathrm{A}(\mathsf{u})$. If so, they perform the measurement

$$\left\{ A_a^{(\mathsf{u}, x^V)} \right\}$$

to obtain an outcome a, which they use as their answer. If not, they reply with the empty string. (This entails performing the measurement whose POVM element corresponding to the empty string is the identity matrix.)

Player B: given $y \in V_{\mathrm{DETYPE}}$, the player checks if for some $\mathsf{v} \in \mathcal{T}$, $y^{V_G} = \mathrm{view}^\mathrm{B}(\mathsf{v})$. If so, they perform the measurement

$$\left\{ B_b^{(\mathsf{v}, y^V)} \right\}$$

to obtain an outcome b, which they use as their answer. If not, they reply with the empty string.

This strategy is projective and consistent because the only measurements it uses are those in $\mathscr{S}$ and "trivial" measurements containing the identity matrix. Suppose the players receive questions x and y such that both $x^{V_G} = \mathrm{view}^\mathrm{A}(\mathsf{u})$ and $y^{V_G} = \mathrm{view}^\mathrm{B}(\mathsf{v})$. In this case, the questions (u, x^V) and (v, y^V) are in the support of the question distribution of $\mathcal{V}$. As a result, the players succeed with probability 1 on these questions, and their measurements always commute. For the remaining pairs of questions, the decider $\mathrm{detype}_G(\mathcal{D})$ always accepts, and the measurements always commute by virtue of the fact that at least one is trivial, i.e. containing the identity matrix as a POVM element.

Soundness. Let $\mathscr{S} = (|\psi\rangle, A, B)$ be a strategy for $\mathrm{detype}(\mathcal{V})_n$ with value $1 - \varepsilon$. Suppose G has m edges, k of which are self-loops. For any (x, y) drawn from $\mu_{\mathrm{detype}_G(\mathcal{S}), n}$, the decider $\mathrm{detype}_G(\mathcal{D})$ automatically accepts unless (x^{V_G}, y^{V_G}) satisfies event $\mathcal{E}_G$ from Proposition 6.13, which occurs with probability $(2m - k)/16^{|\mathcal{T}|}$. When this happens, x^{V_G} and y^{V_G} are distributed as $\mathrm{view}^\mathrm{A}(\mathsf{u})$ and $\mathrm{view}^\mathrm{B}(\mathsf{v})$, where (u, v) are distributed as the graph distribution on G. As a result, conditioned on $\mathcal{E}_G$, the probability that $\mathscr{S}$ succeeds on $\mathrm{detype}(\mathcal{V})_n$ is equal to the probability that the strategy $\mathscr{S}' = (|\psi\rangle, A', B')$ succeeds on $\mathcal{V}_n$, where

$$(A')_a^{\mathsf{u}, x^V} = A_a^{(\mathrm{view}^\mathrm{A}(\mathsf{u}), x^V)} , \quad (B')_b^{\mathsf{v}, y^V} = B_b^{(\mathrm{view}^\mathrm{B}(\mathsf{v}), y^V)} .$$

This means that

$$\mathrm{val}^*(\mathrm{detype}(\mathcal{V})_n, \mathscr{S}) = \left(1 - \frac{2m-k}{16^{|\mathcal{T}|}}\right) + \frac{2m-k}{16^{|\mathcal{T}|}} \cdot \mathrm{val}^*(\mathcal{V}_n, \mathscr{S}')$$

$$\leq \left(1 - \frac{1}{16^{|\mathcal{T}|}}\right) + \frac{1}{16^{|\mathcal{T}|}} \cdot \mathrm{val}^*(\mathcal{V}_n, \mathscr{S}') \,.$$

Thus, $\mathscr{S}'$ has value at least $1 - 16^{|\mathcal{T}|} \cdot \epsilon$. This proves the first statement in the soundness. As for the second, $\mathscr{S}$ and $\mathscr{S}'$ use the same state $|\psi\rangle$, and therefore both strategies have the same Schmidt rank, which by definition is at least $\mathscr{E}(\mathcal{V}_n, 1 - 16^{|\mathcal{T}|} \cdot \varepsilon)$.

Complexity. Definition 6.14 implies that $\mathrm{detype}_G(\mathcal{S})$ is an $(\ell+2)$-level sampler by Lemma 4.5 and that V_{DETYPE} has dimension $4|\mathcal{T}| + s$. From this, we conclude that

$$\mathrm{RAND}_{\mathrm{detype}_G(\mathcal{S})}(n) = 4 \cdot |\mathcal{T}| + \mathrm{RAND}_{\mathcal{S}}(n) \,.$$

The claimed time bounds of $\mathrm{detype}_G(\mathcal{S})$ and $\mathrm{detype}_G(\mathcal{D})$ follow from the fact that these perform simple, $\mathrm{poly}(|\mathcal{T}|)$-time computations followed by running $\mathcal{S}$ and $\mathcal{D}$ as subroutines. $\qquad\square$

7 Classical and Quantum Low-degree Tests

In this section we introduce the classical and quantum low-degree tests. The classical low-degree test, first introduced in [BFL91, AS98], has for almost three decades played a central role in the area of probabilistically checkable proofs (PCPs) and hardness of approximation, and is used as a building block in many MIP and MIP* protocols. The quantum low-degree test was introduced more recently in [NV18a], but it has already led to significant improvements in the power of MIP* protocols [NV18a, NW19]. The protocol in this work combines both of these tests, using the quantum low-degree test for question reduction and the classical low-degree test for answer reduction. Section 7.1 below introduces the classical low-degree test, and Section 7.3 does the same for the quantum low-degree test. Prior to doing this, we introduce the *Magic Square game* in Section 7.2, a key subroutine in the quantum low-degree test.

7.1 The classical low-degree test

We begin with a generalization of the classical low-degree test known as the "simultaneous plane-point low-degree test". We sometimes refer to this as the "classical low-degree test" for short. The low-degree test is used as a subroutine in the Pauli Basis test (see Section 7.3) as well as the answer-reduction normal form verifier (see Section 10). We describe the test as a nonlocal game in Section 7.1.1. In Section 7.1.2, we show how to generate questions for the low-degree test using a CL distribution.

7.1.1 The game

The game $\mathfrak{G}^{\mathrm{LD}}$ is parametrized by a tuple $\mathsf{ldparams} = (q, m, d, k)$ where $m, d, k \in \mathbb{N}$ are integers and $q \in \mathbb{N}$ is an admissible field size. The test is intended to check that the players' responses are consistent with k functions $(f_1, f_2, \ldots, f_k)$ such that each function $f_i : \mathbb{F}_q^m \to \mathbb{F}_q$ is a total degree-d polynomial. We sometimes write $\mathfrak{G}^{\mathrm{LD}}_{\mathsf{ldparams}}$ to emphasize the dependence of the classical low-degree test on the parameter tuple $\mathsf{ldparams}$.

Definition 7.1 (Plane encoding). The *plane* $\mathbf{p}$ in the linear space $\mathbb{F}_q^m$ specified by the triple $v = (v_0, v_1, v_2) \in (\mathbb{F}_q^m)^3$ is the subset

$$\left\{ v_0 + \lambda_1 v_1 + \lambda_2 v_2 : \lambda_1, \lambda_2 \in \mathbb{F}_q \right\} \subseteq \mathbb{F}_q^m . \tag{40}$$

The first entry v_0 of the triple is called the *intercept* of the plane $\mathbf{p}$, and the second and third entries v_1, v_2 are called its *directions*. Note that different triples (v_0, v_1, v_2) can specify the same plane. The plane specified by the triple $v = (v_0, v_1, v_2)$ is denoted as $\mathbf{p}(v)$. The collection of planes in $\mathbb{F}_q^m$ is denoted $\mathrm{Pl}(\mathbb{F}_q^m)$.

Definition 7.2 (Plane-point distribution). The *plane-point distribution* on $\mathbb{F}_q^m$ is the distribution over $(\mathbf{p}, x)$ where $\mathbf{p} = \mathbf{p}(v_0, v_1, v_2)$ is the plane associated with a uniformly random triple of points $(v_0, v_1, v_2) \in (\mathbb{F}_q^m)^3$ and x is chosen uniformly at random from $\mathbf{p}$.

The game $\mathfrak{G}^{\mathrm{LD}}$ is symmetric, so both players have the same question and answer alphabets. The question alphabet is

$$\mathcal{X} = \left(\{\text{POINT}\} \times \mathbb{F}_q^m \right) \cup \left(\{\text{PLANE}\} \times (\mathbb{F}_q^m)^3 \right) .$$

In other words, the questions in the game $\mathfrak{G}^{\mathrm{LD}}$ are pairs (t, x), where the first component t indicates the *type* of the question, and the second component x consists of the *content* of the question.

The distribution μ_{LD} over questions $((\mathsf{t}_A, x_A), (\mathsf{t}_B, x_B))$ for game $\mathfrak{G}^{\mathrm{LD}}$ is the following. First, sample a triple of points $v = (v_0, v_1, v_2) \in (\mathbb{F}_q^m)^3$ uniformly at random. Next, pick a uniformly random point u from

the plane $\mathbf{p}(v)$. For each $w \in \{A, B\}$, with probability $1/2$ (t_w, x_w) is chosen to be (PLANE, v) and with probability $1/2$ it is (POINT, u).

The decision procedure $\mathcal{D}^{\text{LD}}$ for the game $\mathfrak{G}^{\text{LD}}$ is presented in Figure 2. The table at the top specifies a parsing scheme for the questions and answers, depending on the type of question. For example, when a player receives a question with type POINT, the question content x, a bit string of length $m \log q$, should be interpreted by the decision procedure and the players as an element of the vector space $\mathbb{F}_q^m$, as indicated in Section 3.3.2. Similarly the answer to a question with type POINT is expected to be a bit string of length $k \log q$, and is interpreted as an element of $\mathbb{F}_q^k$. For questions with type PLANE, the question content is a $3m \log q$-bit string, which can be parsed as a triple $v = (v_0, v_1, v_2) \in (\mathbb{F}_q^m)^3$, which in turn can be interpreted as a specification for a plane $\mathbf{p}(v)$ in $\mathbb{F}_q^m$. The answer is interpreted as the description of k degree-d bivariate polynomials defined on the plane $\mathbf{p}(v)$. If the answers returned by the players do not fit this format the decision procedure rejects.

Type	Question Content	Answer Format
POINT	$x \in \mathbb{F}_q^m$	Element of $\mathbb{F}_q^k$
PLANE	$v = (v_0, v_1, v_2) \in (\mathbb{F}_q^m)^3$	Bivariate polynomial $f : \mathbf{p}(v) \to \mathbb{F}_q^k$ of degree d

Input to $\mathcal{D}^{\text{LD}}$: $(t_A, x_A, t_B, x_B, a_A, a_B)$. In all cases where no action is indicated, accept. For $w \in \{A, B\}$,

1. (**Consistency test**) If $t_A = t_B$, accept iff $a_A = a_B$.

2. (**Low degree test**) If $t_w = \text{PLANE}$ and $t_{\overline{w}} = \text{POINT}$, parse a_w as a degree-d polynomial $f : \mathbf{p}(v) \to \mathbb{F}_q^k$, and accept iff $f(x_{\overline{w}}) = a_{\overline{w}}$.

Figure 2: The decision procedure $\mathcal{D}^{\text{LD}}$ for the simultaneous low-degree test, parameterized by the parameter tuple $\mathsf{ldparams} = (q, m, d, k)$.

We define a special class of measurements that are relevant to the soundness properties of the low-degree test.

Definition 7.3 (Low-degree polynomial measurements). Define $\text{PolyMeas}(m, d, q)$ to be the set of POVM measurements whose outcomes correspond to degree-d polynomials of m variables over $\mathbb{F}_q$. More generally, for an integer k and tuples $m = (m_1, m_2, \ldots, m_k)$, $d = (d_1, d_2, \ldots, d_k)$ and $q = (q_1, q_2, \ldots, q_k)$, we let $\text{PolyMeas}(m, d, q, k)$ be the set of measurements $G = \{G_{g_1, g_2, \ldots, g_k}\}$ such that for $i \in \{1, 2, \ldots, k\}$, g_i is a degree-d_i polynomial $g_i : \mathbb{F}_{q_i}^{m_i} \to \mathbb{F}_{q_i}$.

Quantum soundness of the classical low-degree test was established in [NV18a] for the case of $k = 1$. It was later extended to the case of general k in [NW19, Theorem 4.43] via a standard reduction. We quote this result below, adapted to our notation.

Lemma 7.4 (Quantum soundness of the simultaneous classical low-degree test). *There exists a function* $\delta_{\text{LD}}(\varepsilon, q, m, d, k) = a(\varepsilon + d(m + k)/q^c)^b$ *for universal constants* $a \geq 1$ *and* $0 < b, c \leq 1$ *such that for every* q, m, d, k, δ_{LD} *is a concave function of* ε, *and the following holds. For all* $\varepsilon > 0$ *and parameter tuple*

59

ldparams $= (q,m,d,k)$, *for all projective strategies (ψ, A, B) that succeed with probability at least $1 - \varepsilon$ in the game $\mathfrak{G}^{\mathrm{LD}}_{\mathsf{ldparams}}$ there exists measurements*

$$G^w \in \mathrm{PolyMeas}(m,d,q,k)$$

on $\mathcal{H}_w$, for $w \in \{\mathrm{A},\mathrm{B}\}$, such that

$$A^{\mathrm{POINT},x}_{b_1, b_2, \dots, b_k} \otimes I_{\mathrm{B}} \simeq_{\delta_{\mathrm{LD}}} I_{\mathrm{A}} \otimes G^{\mathrm{B}}_{[\mathrm{eval}_x(\cdot) = (b_1, b_2, \dots, b_k)]} ,$$

$$G^{\mathrm{A}}_{[\mathrm{eval}_x(\cdot) = (b_1, b_2, \dots, b_k)]} \otimes I_{\mathrm{B}} \simeq_{\delta_{\mathrm{LD}}} I_{\mathrm{A}} \otimes B^{\mathrm{POINT},x}_{b_1, b_2, \dots, b_k} ,$$

$$G^{\mathrm{A}}_{g_1, g_2, \dots, g_k} \otimes I_{\mathrm{B}} \simeq_{\delta_{\mathrm{LD}}} I_{\mathrm{A}} \otimes G^{\mathrm{B}}_{g_1, g_2, \dots, g_k} ,$$

where $\delta_{\mathrm{LD}} = \delta_{\mathrm{LD}}(\varepsilon, q, m, d, k)$, $\mathrm{eval}_x(g_1, g_2, \dots, g_k) = \big(g_1(x), g_2(x), \dots, g_k(x)\big)$ and the approximation holds under the uniform distribution over $x \in \mathbb{F}_q^m$.

Remark 7.5. *Although the decision procedure expects questions and answers that are binary strings, for convenience we index measurements using more structured objects such as vectors or polynomials over $\mathbb{F}_q$, where we implicitly assume a consistent and canonical encoding scheme for these objects as binary strings (as discussed in Section 3.3.2).*

7.1.2 Conditionally linear functions for the plane-point distribution

We introduce CL functions L^{PL}, L^{PT} whose corresponding CL distribution $\mu_{L^{\mathrm{PL}}, L^{\mathrm{PT}}}$ implements the plane-point distribution introduced in Definition 7.2. The functions are parametrized by a field size q, a dimension m, and three disjoint m-dimensional register subspaces $V_{\mathrm{X}}, V_{\mathrm{V1}}, V_{\mathrm{V2}}$ of some ambient space. The register V_{X} is called the *point register* and $V_{\mathrm{V1}}, V_{\mathrm{V2}}$ are called the *direction registers*, respectively. We let V denote the direct sum $V_{\mathrm{X}} \oplus V_{\mathrm{V1}} \oplus V_{\mathrm{V2}}$. The details of the functions are specified in Figure 3.

We explain how to interpret the figure. The first part of the specification identifies m-dimensional vector spaces over $\mathbb{F}_q$ labeled V_{X}, V_{V1}, and V_{V2}. The next part of Figure 3 defines the CL functions L^{PT} and L^{PL} by specifying their factor spaces as well as the associated linear maps. For example, the CL function L^{PT} is a 1-level CL function (i.e. a linear function) that maps every $x \in V$ to its projection $x^{V_{\mathrm{X}}}$ to subspace V_{X}. The CL function L^{PL} is a 2-level CL function that is the concatenation of the identity function on $V_{\mathrm{V1}} \oplus V_{\mathrm{V2}}$ (a 1-level CL function) with a family of linear maps $\{L^{\mathrm{PL}}_v\}_v$ that act on the subspace V_{X}, indexed by $v \in V_{\mathrm{V1}} \oplus V_{\mathrm{V2}}$. We use the convention that the CL functions L^{PT} and L^{PL} are implicitly defined to be 0 on subspaces that are complementary to their factor spaces.

Consider a pair $(x, y) \in V \times V$ sampled from the CL distribution $\mu_{L^{\mathrm{PL}}, L^{\mathrm{PT}}}$. Parse x as $(v_0, v_1, v_2) \in \left(\mathbb{F}_q^m\right)^3$ and parse $y^{V_{\mathrm{X}}}$ as $w \in \mathbb{F}_q^m$. Observe that the joint distribution of (v, w) is different from the marginal distribution of $(x_{\mathrm{A}}, x_{\mathrm{B}})$ sampled from μ_{LD}, conditioned on $\mathsf{t}_{\mathrm{A}} = \mathrm{PLANE}$ and $\mathsf{t}_{\mathrm{B}} = \mathrm{POINT}$. This is because x_{A} is a uniformly random triple of points $(v_0, v_1, v_2) \in (\mathbb{F}_q^m)^3$, and x_{B} is a uniformly random point $w \in \mathbf{p}(v_0, v_1, v_2)$. On the other hand, by definition, we have that $v_0 = L^{\mathrm{PL}}_{v_1, v_2}(w)$. Nevertheless, we show in the following lemma that this syntactic difference does not change the soundness properties of the game.

Let $\mathsf{ldparams} = (q, m, d, k)$ denote a parameter tuple. Let $\tilde{\mathfrak{G}}^{\mathrm{LD}}_{\mathsf{ldparams}}$ denote the classical low-degree test where the distribution $\tilde{\mu}_{\mathrm{LD}}$ is the following distribution over tuples $(\mathsf{t}_{\mathrm{A}}, x_{\mathrm{A}}, \mathsf{t}_{\mathrm{B}}, x_{\mathrm{B}})$: $(\mathsf{t}_{\mathrm{A}}, \mathsf{t}_{\mathrm{B}})$ is sampled uniformly from $\{\mathrm{POINT}, \mathrm{PLANE}\} \times \{\mathrm{POINT}, \mathrm{PLANE}\}$. Next, sample a uniformly random vector $z \in V = V_{\mathrm{X}} \oplus V_{\mathrm{V1}} \oplus V_{\mathrm{V2}}$. Finally, for $w \in \{\mathrm{A}, \mathrm{B}\}$, define

$$x_w = \begin{cases} L^{\mathrm{PL}}(z) & \text{if } \mathsf{t}_w = \mathrm{PLANE}, \\ L^{\mathrm{PT}}(z) & \text{if } \mathsf{t}_w = \mathrm{POINT}. \end{cases}$$

60

Subspaces over $\mathbb{F}_q$			
Subspace	V_{X}	V_{V1}	V_{V2}
Dimension	m	m	m

Conditionally linear function L^{PT}	
1st factor subspace	V
1st linear function	Projector onto V_{X}

Conditionally linear function L^{PL}	
1st factor subspace	$V_{\mathrm{V1}} \oplus V_{\mathrm{V2}}$
1st linear function	Identity function on $V_{\mathrm{V1}} \oplus V_{\mathrm{V2}}$
2nd factor subspace	V_{X}
2nd linear function	For all $v \in V_{\mathrm{V1}} \oplus V_{\mathrm{V2}}$, the linear map L_v^{PL} is the canonical linear map with kernel basis $\{v_1, v_2\}$ (see Definition 3.10), where v_i is the projection of v onto $V_{\mathrm{V}i}$, naturally identified with an element of V_{X} using that both spaces are canonically isomorphic to $\mathbb{F}_q^m$.

Figure 3: Specification of the CL functions used in the plane-point sampler, parametrized by field size q and dimension m.

Lemma 7.6. *Let* $\mathsf{ldparams} = (q, m, d, k)$ *denote a parameter tuple. Then Lemma 7.4 applies to the game* $\tilde{\mathfrak{G}}^{\mathrm{LD}}_{\mathsf{ldparams}}$.

Proof. For notational clarity, we omit mention of $\mathsf{ldparams}$. Fix $\varepsilon > 0$. Let $\mathscr{S} = (\psi, A, B)$ be a PCC strategy for $\tilde{\mathfrak{G}}^{\mathrm{LD}}$ that succeeds with probability $1 - \varepsilon$. We show that there exists a strategy $\mathscr{S}' = (\psi, C, D)$ that succeeds with the same probability $1 - \varepsilon$ in $\mathfrak{G}^{\mathrm{LD}}$. Consider the following strategy for $\mathfrak{G}^{\mathrm{LD}}$: suppose a player receives a question (PLANE, v) such that $v = (v_0, v_1, v_2) \in (\mathbb{F}_q^m)^3$. First, the player computes $u = L_{v_1, v_2}^{\mathrm{PL}}(v_0)$. It then performs the same measurement as a player in the game $\tilde{\mathfrak{G}}^{\mathrm{LD}}$ would when it receives question $(\mathrm{PLANE}, u, v_1, v_2)$ and returns the outcome (which is a list of k bivariate polynomials defined on $\mathbf{p}(u, v_1, v_2)$). If the player receives (POINT, w) for some $w \in \mathbf{p}(v_0, v_1, v_2)$, it performs the same measurement as a player in the game $\tilde{\mathfrak{G}}^{\mathrm{LD}}$ would when it receives question (POINT, w), and returns the outcome (which is a list of k values in $\mathbb{F}_q$).

If both players receive questions of the same type (either PLANE or POINT type) in the game $\mathfrak{G}^{\mathrm{LD}}$, then the measurements performed are exactly the same as those performed in the game $\tilde{\mathfrak{G}}^{\mathrm{LD}}$ when both players receive the same type. This is because the marginal distributions of μ_{LD} and $\tilde{\mu}_{\mathrm{LD}}$ are identical when the question type is POINT, and the marginal distribution of PLANE-type questions in $\tilde{\mu}_{\mathrm{LD}}$ is the same as that of μ_{LD} after the CL function L^{PL} is applied to content of the PLANE-type question.

Suppose that in the game $\mathfrak{G}^{\mathrm{LD}}$ one player receives a question with type PLANE, and the other player receives a question with type POINT. We argue that the distribution on (u, v_1, v_2, w) obtained by sampling $((\mathrm{PLANE}, v), (\mathrm{POINT}, w))$ from μ_{LD} and setting $u = L_{v_1, v_2}^{\mathrm{PL}}(v_0)$ as above is the same as the CL distribution

61

$\mu_{L^{\text{PL}}, L^{\text{PT}}}$.

To see this, observe as we already did earlier that the marginal distribution of (v_1, v_2, w) is uniform over $(\mathbb{F}_q^m)^3$. Furthermore, observe that $u = L_{v_1, v_2}^{\text{PL}}(w)$: since $w \in \mathbf{p}(v_0, v_1, v_2)$ (by definition of μ_{LD}), we have that

$$u = L_{v_1, v_2}^{\text{PL}}(v_0) = L_{v_1, v_2}^{\text{PL}}(w),$$

where the second equality follows from the fact that by construction $\ker(L_{v_1, v_2}^{\text{PL}}) = \text{span}\{v_1, v_2\}$. Thus the measurements performed using strategy $\mathscr{S}'$ in $\mathfrak{G}^{\text{LD}}$ are the same as those using the strategy $\mathscr{S}$ in the game $\tilde{\mathfrak{G}}^{\text{LD}}$ when there is a PLANE player and a POINT player.

Thus, the success probability of $\mathscr{S}'$ in $\mathfrak{G}^{\text{LD}}$ is exactly the same as the success probability of $\mathscr{S}$ in $\tilde{\mathfrak{G}}^{\text{LD}}$. $\qquad\square$

7.1.3 Complexity of the classical low-degree test.

The CL functions and decision procedure of the low-degree test are incorporated as subroutines in some of the normal form verifiers constructed in subsequent sections. The next lemma establishes the time complexity of these procedures as a function of the parameter tuple $\text{ldparams} = (q, m, d, k)$. The lemma also establishes the time complexity of computing the description of the decision procedure $\mathcal{D}^{\text{LD}}$ as a Turing machine, given the parameter tuple ldparams as input.

The CL functions L^{PT} and L^{PL} are additionally parametrized by three m-dimensional register subspaces V_X, V_{V1}, V_{V2} of some larger ambient space V. We can treat the CL functions as acting on the linear space $(\mathbb{F}_q^m)^3$ that decomposes into a direct sum of V_X, V_{V1}, followed by V_{V2}.

Lemma 7.7 (Complexity of the classical low-degree test). *Let* $\text{ldparams} = (q, m, d, k)$ *denote a parameter tuple.*

1. *The time complexity of the decision procedure $\mathcal{D}^{\text{LD}}$ parametrized by ldparams is $\text{poly}(m, d, k, \log q)$.*

2. *The time complexity of evaluating marginals of the CL functions L^{PT} and L^{PL} at a given input point is $\text{poly}(m, \log q)$.*

3. *The Turing machine description of the decision procedure $\mathcal{D}^{\text{LD}}$ parametrized by ldparams can be computed from ldparams in $\text{polylog}(q, m, d, k)$ time.*

Proof. Finite field arithmetic over $\mathbb{F}_q$ can be performed in time $\text{polylog}\,q$, by Lemma 3.18. The most expensive step in $\mathcal{D}^{\text{LD}}$ is to evaluate a bivariate polynomial $f : \mathbf{p} \to \mathbb{F}_q^k$ at a point in $\mathbb{F}_q^m$, which takes time $\text{poly}(m, d, k, \log q)$. The function L^{PT} is a projection onto V_X, which takes time $\text{poly}(m, \log q)$ to compute. The function L^{PL} requires computing a canonical linear map, which requires performing Gaussian elimination and can be done in time $\text{poly}(m, \log q)$.

The Turing-machine description of the decision procedure $\mathcal{D}^{\text{LD}}$ can be uniformly computed from the integers (q, m, d, k) expressed in binary; the complexity of computing the description comes from describing the parameter tuple ldparams, which takes time that is at most polynomial in the bit length of (q, m, d, k). $\qquad\square$

7.2 The Magic Square game

We recall the *Magic Square game* of Mermin and Peres [Mer90, Per90, Ara02]. The Magic Square game is a simple self-test for EPR pairs (it tests for two of them). In addition, it allows one to test that a pair of observables *anticommutes*. Here we use it as a building block to construct the quantum low-degree test.

There are several formulations of the Magic Square game; here we present it as a *binary constraint satisfaction* game [CM14]. In this formulation of the game (denoted by $\mathfrak{G}_{MS}$) there are 6 linear equations defined over 9 variables that take values in $\mathbb{F}_2$. The variables correspond to the cells of a 3×3 grid, as depicted in Figure 4. Five of the equations correspond to the constraint that the sum of the variables in each row and the first two columns must be equal to 0, and the last equation requires that the sum of the variables in the last column must be equal to 1.

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

Figure 4: The Magic Square game

The question set $\mathcal{T}^{MS}$ of the Magic Square game is the following:

$$\mathcal{T}^{MC} = \{\text{CONSTRAINT}_i : i = 1, 2, \ldots, 6\},$$
$$\mathcal{T}^{MV} = \{\text{VARIABLE}_j : j = 1, 2, \ldots, 9\},$$
$$\mathcal{T}^{MS} = \mathcal{T}^{MC} \cup \mathcal{T}^{MV}.$$

The questions CONSTRAINT_i for $i \in \{1, 2, 3\}$ correspond to the three row constraints, the questions CONSTRAINT_4, CONSTRAINT_5 correspond to the first two column constraints, and question CONSTRAINT_6 corresponds to the third column constraint.

In the Magic Square game, the verifier first samples a constraint $\text{CONSTRAINT}_i \in \mathcal{T}^{MC}$ uniformly at random, and then samples VARIABLE_j, one of the three variables in the row or column corresponding to CONSTRAINT_i, uniformly at random. One player is randomly assigned to be the CONSTRAINT player, and the other is assigned to be the VARIABLE player. The CONSTRAINT player is sent the question CONSTRAINT_i and is expected to respond with three bits $(\beta_{v_1}, \beta_{v_2}, \beta_{v_3}) \in \mathbb{F}_2^3$, where (v_1, v_2, v_3) are the indices of the three variables corresponding to CONSTRAINT_i. The VARIABLE player is given question VARIABLE_j and is expected to respond with a single bit $\gamma \in \mathbb{F}_2$. The players win if the CONSTRAINT player's answers satisfy the equation associated with CONSTRAINT_i, and $\gamma = \beta_j$. More precisely, the verifier samples an edge of the type graph (see Section 6) G^{MS} in Fig. 5, sends one endpoint to a random player, and the other endpoint to the other player.

The following theorem records the self-testing (also known as *rigidity*) properties of the Magic Square game. Although we do not explicitly refer to the theorem, its self-testing properties are crucial to the Pauli basis test. In particular, it is used to enforce anticommutation relations between certain pairs of operators.

Theorem 7.8 (Rigidity of the Magic Square game). *Let $\mathscr{S} = (|\psi\rangle, M)$ be the partial strategy where $|\psi\rangle = |\text{EPR}_2\rangle^{\otimes 2}$ and for all $b \in \{0, 1\}$,*

$$M_b^{\text{VARIABLE}_1} = \sigma_b^X \otimes I \qquad and \qquad M_b^{\text{VARIABLE}_5} = \sigma_b^Z \otimes I,$$

where σ_b^X, σ_b^Z are the X and Z Pauli projectors over qubits. Then $\mathfrak{G}_{MS}$ is a self-test for $\mathscr{S}$ with robustness $O(\sqrt{\varepsilon})$.

Proof. See [WBMS16]. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ □
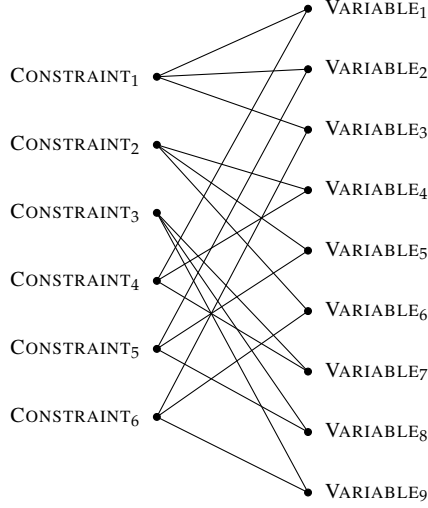
Figure 5: Type graph G^{MS} for the Magic Square game.

We will need the following theorem, which shows that any pair of anticommuting observables can be used to form a value-1 strategy for the Magic Square game.

Theorem 7.9. *Let $A = \{A_b\}_{b \in \mathbb{F}_2}$ and $B = \{B_b\}_{b \in \mathbb{F}_2}$ be two-outcome projective measurements acting on $(\mathbb{C}^q)^{\otimes n}$ which are consistent on $|\mathrm{EPR}_q\rangle^{\otimes n}$, and let $\mathcal{O}_A = A_0 - A_1$ and $\mathcal{O}_B = B_0 - B_1$ be the corresponding observables. Suppose that $\mathcal{O}_A \mathcal{O}_B = -\mathcal{O}_B \mathcal{O}_A$. Then there exists a symmetric strategy $\mathscr{S} = (\psi, M)$ for the Magic Square game with the following properties.*

1. *$\mathscr{S}$ is an SPCC strategy of value 1.*

2. *The state $|\psi\rangle$ has the form $|\psi\rangle = |\mathrm{EPR}_q\rangle^{\otimes n} \otimes |\mathrm{EPR}_2\rangle$.*

3. *For $b \in \{0, 1\}$, we have $M_b^{\mathrm{VARIABLE}_1} = A_b \otimes I$ and $M_b^{\mathrm{VARIABLE}_5} = B_b \otimes I$.*

Proof. The strategy $\mathscr{S}$ is based on the canonical two-qubit strategy for the Magic Square game as described in, for example, [Ara02]. The state is $|\psi\rangle = |\mathrm{EPR}_q\rangle^{\otimes n} \otimes |\mathrm{EPR}_2\rangle$. We specify the measurements of $\mathscr{S}$ in Figure 6 as an *operator solution* for the Magic Square game, meant to be read as follows: each cell contains a two-outcome projective measurement $\{E_0, E_1\}$ on $(\mathbb{C}^q)^{\otimes n} \otimes \mathbb{C}^2$ written as its ± 1-*valued observable* $E_0 - E_1$. When Player A or B receives the question $\mathrm{VARIABLE}_j$ for $j \in \{1, \dots, 9\}$, they measure their share of $|\psi\rangle$ using the measurement specified by the cell corresponding to $\mathrm{VARIABLE}_j$ and receive a single-bit measurement. When they receive the question $\mathrm{CONSTRAINT}_i$ for $i \in \{1, 2, \dots, 6\}$, they simultaneously perform the three measurements in the corresponding row or column on $|\psi\rangle$ to obtain three bits. For example, if Player A receives question $\mathrm{VARIABLE}_1$, they measure $|\psi\rangle$ using the measurement $\{A_0 \otimes I, A_1 \otimes I\}$ corresponding to the observable $\mathcal{O}_A \otimes I$ (where the first operator acts on $|\mathrm{EPR}_q\rangle^{\otimes n}$ and the second acts on $|\mathrm{EPR}_2\rangle$). Similarly, on question $\mathrm{VARIABLE}_5$, they use the measurement $\{B_0 \otimes I, B_1 \otimes I\}$. This establishes Item 3 of the theorem.

First, we show that this gives a well-defined strategy. The $\mathrm{VARIABLE}_j$ measurements are well-defined because each cell contains a ± 1-valued observable. This is obvious for all $j \neq 9$; when $j = 9$, the bottom-right cell contains $\mathcal{O}_A \mathcal{O}_B \otimes \sigma^Z \sigma^X$. Because $\mathcal{O}_A$ and $\mathcal{O}_B$ anti-commute,

$$\mathcal{O}_A \mathcal{O}_B \otimes \sigma^Z \sigma^X = -\mathcal{O}_A \mathcal{O}_B \otimes \sigma^X \sigma^Z = \mathcal{O}_B \mathcal{O}_A \otimes \sigma^X \sigma^Z = (\mathcal{O}_A \mathcal{O}_B \otimes \sigma^Z \sigma^X)^\dagger. \tag{41}$$

64

$\mathcal{O}_A \otimes I$	$I \otimes \sigma^X$	$\mathcal{O}_A \otimes \sigma^X$
$I \otimes \sigma^Z$	$\mathcal{O}_B \otimes I$	$\mathcal{O}_B \otimes \sigma^Z$
$\mathcal{O}_A \otimes \sigma^Z$	$\mathcal{O}_B \otimes \sigma^X$	$\mathcal{O}_A \mathcal{O}_B \otimes \sigma^Z \sigma^X$

Figure 6: Observables for Magic Square strategy

As a result, this matrix is Hermitian. In addition,

$$(\mathcal{O}_A \mathcal{O}_B \otimes \sigma^Z \sigma^X)^2 = (\mathcal{O}_A \mathcal{O}_B \otimes \sigma^Z \sigma^X) \cdot (\mathcal{O}_B \mathcal{O}_A \otimes \sigma^X \sigma^Z) = (\mathcal{O}_A \mathcal{O}_B \cdot \mathcal{O}_B \mathcal{O}_A) \otimes (\sigma^Z \sigma^X \cdot \sigma^X \sigma^Z) = I,$$

where the first step uses Equation (41) and the final step uses the fact that $\mathcal{O}_A, \mathcal{O}_B, \sigma^X, \sigma^Z$ are ± 1-valued observables and hence square to the identity. As a result, this matrix is Hermitian and squares to the identity. Therefore, it is a ± 1-valued observable.

As for the CONSTRAINT$_i$ measurements, we must show that the three measurements in each row and column are simultaneously measurable. This is equivalent to the three ± 1-valued observables being simultaneously diagonalizable, which is equivalent to them being pairwise commuting. This can be easily verified for the cases of $i = 1, 2, 4, 5$ (i.e. the first two rows and columns). In the case of $i = 3$, commutativity of $\mathcal{O}_A \otimes \sigma^Z$ and $\mathcal{O}_B \otimes \sigma^X$ follows from Equation (41). Since these two matrices commute, they also commute with their product $(\mathcal{O}_A \otimes \sigma^Z)(\mathcal{O}_B \otimes \sigma^X) = \mathcal{O}_A \mathcal{O}_B \otimes \sigma^Z \sigma^X$. The case of $i = 6$ is similar.

By construction, $\mathscr{S}$ is symmetric, and we have already shown that it is projective. It remains to show that it is commuting, consistent, and value 1. To show that it is commuting, it suffices to show that the measurement for each cell is simultaneously measurable with all three measurements in its row or column, which was already proved above. Now we show consistency. By linearity, because A and B are consistent on $|\text{EPR}_q\rangle^{\otimes n}$, so too are $\mathcal{O}_A$ and $\mathcal{O}_B$. We claim that this implies the observable in each cell of Figure 6 is consistent on $|\psi\rangle$. To see why, consider the $j = 9$ case:

$$\begin{aligned}
(\mathcal{O}_A \mathcal{O}_B \otimes \sigma^Z \sigma^X)_\text{A} \otimes I_\text{B} \cdot |\text{EPR}_q\rangle^{\otimes n} \otimes |\text{EPR}_2\rangle &= (\mathcal{O}_A \mathcal{O}_B \otimes \sigma^Z)_\text{A} \otimes (I \otimes \sigma^X)_\text{B} \cdot |\text{EPR}_q\rangle^{\otimes n} \otimes |\text{EPR}_2\rangle \\
&= (\mathcal{O}_A \mathcal{O}_B \otimes I)_\text{A} \otimes (I \otimes \sigma^X \sigma^Z)_\text{B} \cdot |\text{EPR}_q\rangle^{\otimes n} \otimes |\text{EPR}_2\rangle \\
&= (\mathcal{O}_A \otimes I)_\text{A} \otimes (\mathcal{O}_B \otimes \sigma^X \sigma^Z)_\text{B} \cdot |\text{EPR}_q\rangle^{\otimes n} \otimes |\text{EPR}_2\rangle \\
&= I_\text{A} \otimes (\mathcal{O}_B \mathcal{O}_A \otimes \sigma^X \sigma^Z)_\text{B} \cdot |\text{EPR}_q\rangle^{\otimes n} \otimes |\text{EPR}_2\rangle \\
&= I_\text{A} \otimes (\mathcal{O}_A \mathcal{O}_B \otimes \sigma^Z \sigma^X)_\text{B} \cdot |\text{EPR}_q\rangle^{\otimes n} \otimes |\text{EPR}_2\rangle.
\end{aligned}$$

The first two steps use the consistency of σ^X, σ^Z on $|\text{EPR}_2\rangle$, the next two use the consistency of $\mathcal{O}_B, \mathcal{O}_A$ on $|\text{EPR}_q\rangle^{\otimes n}$, and the last step is by Equation (41). The remaining cases of $j \in \{1, \ldots, 8\}$ are similar, and we omit them.

Now, we show that the measurements in $\mathscr{S}$ are consistent. Let $E = \{E_0, E_1\}$ be any of the measurements in the cells of Figure 6. We have shown that $\mathcal{O}_E = E_0 - E_1$ is consistent on $|\psi\rangle$. But for each $b \in \{0, 1\}$, $E_b = (E_0 + E_1 + (-1)^b (E_0 - E_1))/2 = (I + (-1)^b \mathcal{O}_E)/2$, and so each E_b is consistent

on $|\psi\rangle$ by the following calculation

$$E_b \otimes I_B |\psi\rangle = \tfrac{1}{2}(I + (-1)^b \mathcal{O}_E) \otimes I_B |\psi\rangle = \tfrac{1}{2}|\psi\rangle + \tfrac{1}{2}(-1)^b \mathcal{O}_E \otimes I_B |\psi\rangle$$
$$= \tfrac{1}{2}|\psi\rangle + \tfrac{1}{2}(-1)^b I_A \otimes \mathcal{O}_E |\psi\rangle = \tfrac{1}{2} I_A \otimes (I + (-1)^b \mathcal{O}_E)|\psi\rangle = I_A \otimes E_b |\psi\rangle,$$

where the third equality is by the consistency of $\mathcal{O}_E$. As a result, the VARIABLE$_j$ measurements are consistent. As for the CONSTRAINT$_i$ measurements, each such measurement $\{F_{a,b,c}\}_{a,b,c \in \{0,1\}}$ is of the form $F_{a,b,c} = E_a^1 \cdot E_b^2 \cdot E_c^3$, where E_1, E_2, and E_3 are VARIABLE measurements. But then consistency of F follows from the VARIABLE consistencies:

$$F_{a,b,c} \otimes I_B |\psi\rangle = (E_a^1 \cdot E_b^2 \cdot E_c^3)_A \otimes I_B |\psi\rangle = (E_a^1 \cdot E_b^2)_A \otimes (E_c^3)_B |\psi\rangle = (E_a^1)_A \otimes (E_c^3 \cdot E_b^2)_B |\psi\rangle$$
$$= I_A \otimes (E_c^3 \cdot E_b^2 \cdot E_a^1)_B |\psi\rangle = I_A \otimes (E_a^1 \cdot E_b^2 \cdot E_c^3)_B |\psi\rangle = I_A \otimes F_{a,b,c} |\psi\rangle,$$

where the second-to-last step is because the VARIABLE$_j$ measurements in the same row or column commute. Hence, all measurements are consistent.

Since all measurements are consistent, this implies that the answer bit of the player receiving a VARIABLE question is always consistent with the corresponding answer bit of the player receiving the CONSTRAINT question. Similarly, the answers of the player receiving the CONSTRAINT question always satisfy the given constraint; observe that in all rows and the first two columns, the observables multiply to I, whereas the observables in the last column multiply to $-I$. This implies that the strategy is value-1, concluding the proof. $\qquad\square$

7.3 The Pauli basis test

We introduce the quantum low-degree test of [NV18a] in the form of a slight modification to it by [NW19] known as the *Pauli basis test*. Informally, the quantum low-degree test asks the players to measure a large number of qubits and return a highly compressed version of the measurement outcome. The Pauli basis test simply asks that the players return their uncompressed measurement outcomes, and it is designed by direct reduction to the quantum low-degree test. In Section 7.3.1 we describe the Pauli basis test as a nonlocal game $\mathfrak{G}^{\text{PAULI}}$, as we did with the classical low-degree in Section 7.1. In Section 7.3.2, we describe how to generate questions for the Pauli basis test using CL functions. In Section 7.3.3 we exhibit canonical parameters for the Pauli basis test and give bounds on the time complexity of executing the test.

7.3.1 The game

We start by discussing parameter settings. The game $\mathfrak{G}^{\text{PAULI}}$ is parametrized by a tuple

$$\mathsf{qldparams} = (q, m, d, h, H, \Gamma, \pi),$$

where $q, m, d, h, \Gamma \in \mathbb{N}$ are integers, H is a subset of $\mathbb{F}_q$ of size h, and π is a map from $\{1, 2, \ldots, \Gamma\}$ to H^m. We sometimes write $\mathfrak{G}^{\text{PAULI}}_{\mathsf{qldparams}}$ to emphasize the dependence of the Pauli basis test on the parameters.

Informally, the test is meant to certify that the players share a state of the form $|\text{EPR}_q\rangle^{\otimes \Gamma}$. Its question set includes questions that are planes and points in $\mathbb{F}_q^m$, which are meant to correspond to questions in a low-degree test, and questions of the form (PAULI, W), for $W \in \{X, Z\}$. Upon receipt of a question of the latter form, the players are expected to perform the POVM $\{\tau_u^W\}_{u \in \mathbb{F}_q^\Gamma}$ and report the outcome u as their answer.

Definition 7.10 (Admissible parameters). We say that the tuple $\text{qldparams} = (q, m, d, h, H, \Gamma, \pi)$ is *admissible* if q is an admissible field size, $h \leq q$, and $\Gamma \leq h^m$.

Similarly to the game $\mathfrak{G}^{\text{LD}}$, questions in the Pauli basis game come with a "question type" part and a "question content" part. The question types are taken from the set

$$\mathcal{T}^{\text{PAULI}} = \left(\{\text{POINT}, \text{PLANE}, \text{PAULI}, \text{PAIR}\} \times \{X, Z\}\right) \cup \mathcal{T}^{\text{MS}} \cup \{\text{PAIR}\}, \tag{42}$$

where $\mathcal{T}^{\text{MS}}$ is the question type set of the Magic Square game defined in Section 7.2. The question content has a format that depends on the type.
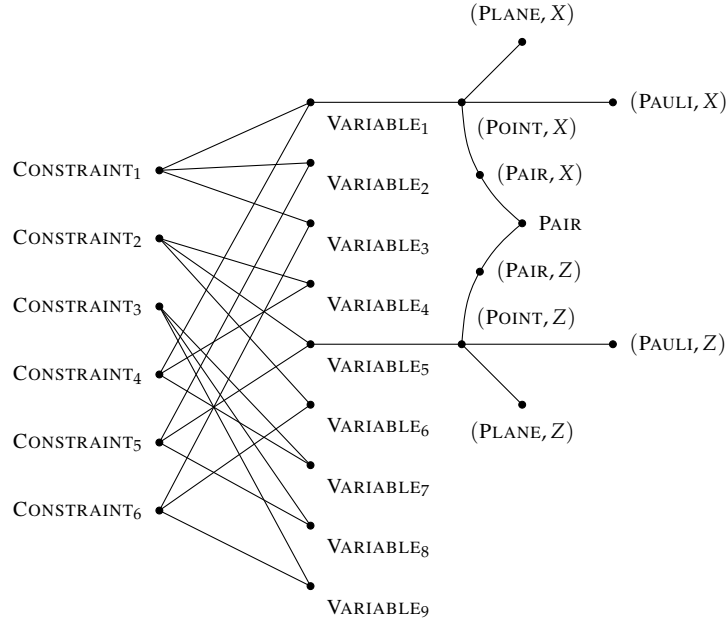
Figure 7: Graph G^{PAULI} for the Pauli basis test. Each vertex also has a self-loop which is not drawn on the figure for clarity.

The distribution μ_{PAULI} over questions $((t_A, x_A), (t_B, x_B))$ in $\mathfrak{G}^{\text{PAULI}}$ is defined through the following sampling procedure:

1. Sample a pair of types by sampling an edge (t_A, t_B) of the graph G^{PAULI} given in Figure 7 uniformly at random (including the self-loops).

2. Sample the following uniformly at random:

   (a) *(Points)* $u_X, u_Z \in \mathbb{F}_q^m$,

   (b) *(Directions)* $v_1, v_2 \in \mathbb{F}_q^m$,

   (c) *(Qubit basis for (anti-)commutation)* $r_X, r_Z \in \mathbb{F}_q$.

3. For $w \in \{A, B\}$ and $W \in \{X, Z\}$,

   (a) If $t_w = (\text{POINT}, W)$, then set $x_w = u_W$,

(b) If $t_w = (\text{PLANE}, W)$, then set $x_w = (u, v_1, v_2)$, where $u = L_{v_1, v_2}^{\text{PL}}(u_W)$,

(c) If $t_w = \text{CONSTRAINT}_i$ for some $i \in \{1, \ldots, 6\}$, then set $x_w = (u_X, u_Z, r_X, r_Z)$,

(d) If $t_w = \text{VARIABLE}_j$ for some $j \in \{1, \ldots, 9\}$, then set $x_w = (u_X, u_Z, r_X, r_Z)$,

(e) If $t_w = \text{PAIR}$, then set $x_w = (u_X, u_Z, r_X, r_Z)$,

(f) If $t_w = (\text{PAIR}, W)$, then set $x_w = (u_X, u_Z, r_X, r_Z)$,

(g) If $t_w = (\text{PAULI}, W)$, then set $x_w = 0$.

As with the game $\mathfrak{G}^{\text{LD}}$, the question content is a bit string that is interpreted as a vector over $\mathbb{F}_q$ (as described in Section 3.3.2).

Decision procedure. The decision procedure for $\mathfrak{G}^{\text{PAULI}}$ is presented in Figure 8. Similarly to Figure 2, we provide a table that summarizes a parsing scheme for the questions and answers, depending on the type of question. The answers are bit strings that are interpreted as more structured objects such as elements over $\mathbb{F}_q$, vectors, or polynomials, depending on the question. In the "low-degree check", the decision procedure $\mathcal{D}^{\text{PAULI}}$ calls the classical low-degree decision procedure $\mathcal{D}^{\text{LD}}$ parametrized by the tuple ldparams $= (q, m, d, 1)$ (defined in Section 7.1) as a subroutine.

We describe an honest, value-1 PCC strategy for the Pauli basis test.

Definition 7.11. Let qldparams $= (q, m, d, h, H, \Gamma, \pi)$ be a parameter tuple. Define the *honest Pauli strategy corresponding to* qldparams as a partial strategy $\mathscr{S}^{\text{PAULI}}$ that uses the state $|\text{EPR}_q\rangle^{\otimes \Gamma}$ and uses the POVM $\{\tau_u^W\}_{u \in \mathbb{F}_q^\Gamma}$ for the question (PAULI, W) for $W \in \{X, Z\}$.

Lemma 7.12. *The Pauli basis test* $\mathfrak{G}_{\text{qldparams}}^{\text{PAULI}}$ *has a value-1 SPCC strategy.*

Proof. We begin by specifying the value-1 strategy $\mathscr{S} = (|\psi\rangle, M)$. The state is

$$|\psi\rangle = |\text{EPR}_q\rangle^{\otimes \Gamma} \otimes |\text{EPR}_2\rangle .$$

Now we specify the measurements. We start with measurements associated with questions of type POINT, PLANE, and PAULI. Using notation introduced in Section 3.4, for $W \in \{X, Z\}$,

$$M_a^{(\text{POINT}, W), y} = \tau_{[g_{\cdot}, \pi(y) = a]}^W \otimes I ,$$
$$M_f^{(\text{PLANE}, W), \mathbf{p}} = \tau_{[(g_{\cdot}, \pi)|_{\mathbf{p}} = f]}^W \otimes I ,$$
$$M_a^{(\text{PAULI}, W)} = \tau_a^W \otimes I .$$

We recall Definition 7.1 for the notation $\mathbf{p}$ and Definition 3.28 for the bracket notation used to post-process measurement outcomes. For example, the measurement on question (POINT, W) corresponds to first performing the measurement $\{\tau_w^W\}_{w \in \mathbb{F}_q^\Gamma}$, receiving an outcome $w \in \mathbb{F}_q^\Gamma$, and then outputting the value $a = g_{w, \pi}(y)$. Next we specify the POVMs associated with questions of type CONSTRAINT, VARIABLE, and PAIR. Questions with these question types have a question content that is a tuple $\omega = (u_X, u_Z, r_X, r_Z) \in (\mathbb{F}_q^m)^2 \times \mathbb{F}_q^2$. Given such a tuple, consider the two $\mathbb{F}_2$-valued POVMs $A = \{A_b\}_{b \in \mathbb{F}_2}$ and $B = \{B_b\}_{b \in \mathbb{F}_2}$ defined as

$$A_b = \tau_{[\text{tr}(g_{\cdot}, \pi(u_X) r_X) = b]}^X \otimes I , \tag{44}$$
$$B_b = \tau_{[\text{tr}(g_{\cdot}, \pi(u_Z) r_Z) = b]}^Z \otimes I . \tag{45}$$

68

Type	Question Content	Answer Format
(POINT, W)	$y \in \mathbb{F}_q^m$	Element of $\mathbb{F}_q$
(PLANE, W)	$v = (u_W, v_1, v_2) \in (\mathbb{F}_q^m)^3$	Polynomial $f : \mathbf{p}(v) \to \mathbb{F}_q$
PAIR	$(u_X, u_Z, r_X, r_Z) \in (\mathbb{F}_q^m)^2 \times \mathbb{F}_q^2$	$(\beta_X, \beta_Z) \in \mathbb{F}_2^2$
(PAIR, W)	$(u_X, u_Z, r_X, r_Z) \in (\mathbb{F}_q^m)^2 \times \mathbb{F}_q^2$	Element of $\mathbb{F}_2$
CONSTRAINT_i	$(u_X, u_Z, r_X, r_Z) \in (\mathbb{F}_q^m)^2 \times \mathbb{F}_q^2$	$(\alpha_{v_1}, \alpha_{v_2}, \alpha_{v_3}) \in \mathbb{F}_2^3$
VARIABLE_j	$(u_X, u_Z, r_X, r_Z) \in (\mathbb{F}_q^m)^2 \times \mathbb{F}_q^2$	Element of $\mathbb{F}_2$
(PAULI, W)	0	Element of $\mathbb{F}_q^\Gamma$

Table: Question and answer format of the Pauli basis game.

On input $(t_A, x_A, t_B, x_B, a_A, a_B)$, the decision procedure $\mathcal{D}^{\text{PAULI}}$ performs the following checks for $w \in \{A, B\}$:

1. (**Consistency check**). If $t_A = t_B$, accept iff $a_A = a_B$.

2. (**Low-degree check**). If $t_w = (\text{POINT}, W), t_{\overline{w}} = (\text{PLANE}, W)$, accept if $\mathcal{D}^{\text{LD}}_{\text{ldparams}}$ accepts $(t_w, x_w, t_{\overline{w}}, x_{\overline{w}}, a_w, a_{\overline{w}})$ with $\text{ldparams} = (q, m, d, 1)$.

3. (**Consistency check**). If $t_w = (\text{POINT}, W), t_{\overline{w}} = (\text{PAULI}, W)$, accept if $g_{a_{\overline{w}}, \pi}(y) = a_w$, where $g_{a_{\overline{w}}, \pi}$ is the low-degree encoding of $a_{\overline{w}} \in \mathbb{F}_q^\Gamma$ defined in Section 3.4.

In the remaining three cases, the decision procedure first computes the number

$$\gamma = \text{tr}\left((\text{ind}_{H,m,\pi}(u_X) r_X) \cdot (\text{ind}_{H,m,\pi}(u_Z) r_Z)\right), \tag{43}$$

where we recall the $\text{ind}_{H,m,\pi}(\cdot)$ notation from Section 3.4.

4. (**Commutation check**). If $t_w = (\text{PAIR}, W), t_{\overline{w}} = \text{PAIR}$, accept if $a_w = \beta_W$ or $\gamma \neq 0$.

5. (**Consistency check**). If $t_w = (\text{POINT}, W), t_{\overline{w}} = (\text{PAIR}, W)$, accept if $\text{tr}(a_w r_W) = a_{\overline{w}}$ or $\gamma \neq 0$.

6. (**Magic square check**). If $t_w = \text{CONSTRAINT}_i, t_{\overline{w}} = \text{VARIABLE}_j$, accept if $\gamma = 0$, or a_w satisfies constraint CONSTRAINT_i and $\alpha_j = a_{\overline{w}}$.

7. (**Consistency check**). If $t_w = (\text{POINT}, W), t_{\overline{w}} = \text{VARIABLE}_j$, accept if $\gamma = 0$ or if $j = 1, W = X$, and $\text{tr}(a_w r_X) = a_{\overline{w}}$ or if $j = 5, W = Z$, and $\text{tr}(a_w r_Z) = a_{\overline{w}}$.

Figure 8: Specification of the decision procedure $\mathcal{D}^{\text{PAULI}}_{\text{qldparams}}$.

We would like to determine when the two observables $\mathcal{O}_A = A_0 - A_1$ and $\mathcal{O}_B = B_0 - B_1$ commute or anti-commute. Towards this we derive alternative expressions for these observables from which their commutativity becomes plain from inspection. We begin by inspecting the first matrices on the right-hand sides of Equations (44) and (45):

$$\tau^W_{[\mathrm{tr}(g_{\cdot,\pi}(u_W)\,r_W)=b]} = \sum_{w:\mathrm{tr}(g_w,\pi(u_W)\,r_W)=b} \tau^W_w$$
$$= \sum_{w:\mathrm{tr}((w\cdot\mathrm{ind}_{H,m,\pi}(u_W))\cdot r_W)=b} \tau^W_w \;, \tag{46}$$

where (46) follows by Definition 12. As a result,

$$\tau^W_{[\mathrm{tr}(g_{\cdot,\pi}(u_W)\,r_W)=0]} - \tau^W_{[\mathrm{tr}(g_{\cdot,\pi}(u_W)\,r_W)=1]} = \sum_{w:\mathrm{tr}((w\cdot\mathrm{ind}_{H,m,\pi}(u_W))\cdot r_W)=0} \tau^W_w - \sum_{w:\mathrm{tr}((w\cdot\mathrm{ind}_{H,m,\pi}(u_W))\cdot r_W)=1} \tau^W_w$$
$$= \sum_w (-1)^{\mathrm{tr}((w\cdot\mathrm{ind}_{H,m,\pi}(u_W))\cdot r_W)}\tau^W_w$$
$$= \tau^W(\mathrm{ind}_{H,m,\pi}(u_W)r_W) \;, \tag{47}$$

where the last step uses (17). As a result, Equation (47) and Equations (44) and (45) imply that

$$\mathcal{O}_A = \tau^X(\mathrm{ind}_{H,m,\pi}(u_X)r_X) \otimes I, \quad \mathcal{O}_B = \tau^Z(\mathrm{ind}_{H,m,\pi}(u_Z)r_Z) \otimes I \;.$$

Now, let $\gamma = \mathrm{tr}((\mathrm{ind}_{H,m,\pi}(u_X)r_X) \cdot (\mathrm{ind}_{H,m,\pi}(u_Z)r_Z)) \in \mathbb{F}_2$, as in Equation (43). Then by Equation (15),

$$\mathcal{O}_A\mathcal{O}_B = (-1)^\gamma \mathcal{O}_B\mathcal{O}_A \;.$$

As a result, γ quantifies whether the observables $\mathcal{O}_A$ and $\mathcal{O}_B$ commute, and therefore whether the measurements A and B commute. If $\gamma = 0$, they commute. If $\gamma = 1$, they anti-commute. We now specify the POVMs associated with questions of type CONSTRAINT, VARIABLE, and PAIR, considering separately the cases when $\gamma = 0$ or $\gamma = 1$.

1. If $\gamma = 0$, for each $\beta_X, \beta_Z \in \mathbb{F}_2$ define

$$M^{\mathrm{PAIR},\omega}_{\beta_X,\beta_Z} = A_{\beta_X} \cdot B_{\beta_Z} \;, \tag{48}$$

$$M^{(\mathrm{PAIR},X),\omega}_a = A_a \;, \qquad M^{(\mathrm{PAIR},Z),\omega}_a = B_a \;.$$

The POVM associated with questions of type CONSTRAINT and VARIABLE are defined to be trivial. In particular, we define

$$M^{\mathrm{CONSTRAINT}_i,\omega}_{0,0,0} = M^{\mathrm{VARIABLE}_j,\omega}_0 = I \;,$$

and the remaining POVM elements in these measurements are set to be zero.

2. If $\gamma = 1$, then $\mathcal{O}_A$ and $\mathcal{O}_B$ anti-commute. In this case we define measurements $M^{\mathrm{CONSTRAINT}_i,\omega}$ and $M^{\mathrm{VARIABLE}_j,\omega}$, for all $i \in \{1,\ldots,6\}$ and $j \in \{1,\ldots,9\}$ to be those guaranteed by Theorem 7.9. Measurements associated with inputs of type PAIR are defined to be trivial. In particular, we define

$$M^{\mathrm{PAIR},\omega}_{0,0} = M^{(\mathrm{PAIR},X),\omega}_0 = M^{(\mathrm{PAIR},Z),\omega}_0 = I \;,$$

and the remaining matrices in these measurements are set to be zero.

This completes the specification of the strategy.

Now, we show that $\mathscr{S}$ is a value-1 SPCC strategy. It is clearly symmetric and projective. To show that it is consistent, we note that all measurements are Pauli basis measurements, which are consistent, or measurements produced by Theorem 7.9, which are also consistent. The only exception is the PAIR measurement in the $\gamma = 0$ case, which by Equation (48) is a product of two commuting, consistent measurements, and so it is therefore also consistent. To show that it is commuting, we note that for each $W \in \{X, Z\}$, all (POINT, W), (PLANE, W), and (PAULI, W) measurements commute as they are all measurements in the Pauli W basis. Next, if $\gamma = 0$ then the CONSTRAINT and VARIABLE measurements commute trivially, and for $W \in \{X, Z\}$, the (POINT, W) measurement commutes with the (PAIR, W) measurement, as they are both W basis measurements, and the measurement $M_{\beta_X, \beta_Z}^{\text{PAIR}, \omega} = A_{\beta_X} \cdot B_{\beta_Z}$ commutes with $M_a^{(\text{PAIR}, W), \omega}$ because A and B commute. On the other hand, if $\gamma = 0$, then the PAIR measurements commute trivially, and the CONSTRAINT and VARIABLE measurements commute by Theorem 7.9. Finally, the (POINT, X) measurement commutes with VARIABLE$_1$ because both are X-basis measurements, and likewise both (POINT, Z) and VARIABLE$_5$ are Z-basis measurements.

It remains to show that $\mathscr{S}$ is value-1. Consider first the first three tests executed by the decision procedure in Figure 8. The strategy passes the consistency checks with probability 1 because it is projective and consistent. It passes the low-degree checks because it answers those consistently with an honest strategy in the classical low-degree test.

Next, consider the remaining four tests. Fix an $\omega = (u_X, u_Z, r_X, r_Z)$ and γ as in (43). If $\gamma = 0$, then the strategy passes the commutation check with probability 1 by construction. As for the consistency check in Item 5, we can write the (POINT, W) measurement as follows:

$$M_{[\text{tr}(\,\cdot\, r_W) = a_{\overline{w}}]}^{(\text{POINT}, W), y} = \tau_{[\text{tr}(g_{\cdot, \pi}(y) r_W) = a_{\overline{w}}]}^{W} \otimes I = M_{a_{\overline{w}}}^{(\text{PAIR}, W), \omega} \otimes I.$$

As a result, due to the consistency of the (PAIR, W) measurement, the consistency check is passed with probability 1. On the other hand, if $\gamma \neq 0$, then the strategy passes the Magic Square check by Theorem 7.9. As for the consistency check in Item 7, we can write the (POINT, X) measurement as follows:

$$M_{[\text{tr}(\,\cdot\, r_X) = a_{\overline{w}}]}^{(\text{POINT}, X), y} = \tau_{[\text{tr}(g_{\cdot, \pi}(y) r_X) = a_{\overline{w}}]}^{X} \otimes I = M_{a_{\overline{w}}}^{\text{VARIABLE}_1, \omega} \otimes I,$$

where the last step is by Theorem 7.9. As these measurements are consistent, this test is passed with probability 1. □

As described in Definition 7.11, the strategy $\mathscr{S}^{\text{PAULI}}$ is a strategy that uses qudit Pauli measurements and maximally entangled states defined over qudits with dimension larger than 2. However, it is more convenient for our application of the Pauli basis test to have a self-test for *qubits*. In particular, we use the Pauli basis test in the "introspection game" of Section 8, where the players are commanded to sample questions according to a sampler $\mathcal{S}$ of a normal form verifier. By definition of normal form verifier, $\mathcal{S}$ is a sampler over $\mathbb{F}_2$, and therefore it is natural to use a test for qubit Pauli observables. This motivates the definition of a *binary Pauli strategy*:

Definition 7.13. Let qldparams $= (q, m, d, h, H, \Gamma, \pi)$ be a parameter tuple. Define the *honest binary Pauli strategy corresponding to* qldparams as the partial strategy $\mathscr{S}^{\text{BP}}$ that uses the state $|\text{EPR}_2\rangle^{\otimes \Gamma \log q}$ and uses the POVM $\{\sigma_u^W\}_{u \in \mathbb{F}_2^{\Gamma \log q}}$ for the question (PAULI, W) for $W \in \{X, Z\}$. Here, σ_u^W denotes the tensor product of qubit Pauli projectors $\bigotimes_i \sigma_{u_i}^W$.

Since we use field sizes that are powers of 2, Lemma 3.29 shows that the strategy $\mathscr{S}^{\text{PAULI}}$ based on qudits is isomorphic to the strategy $\mathscr{S}^{\text{BP}}$ based on qubits.

Soundness of the Pauli basis test. We now state the soundness properties of the Pauli basis test. The following is an adaptation of the self-testing statement in [NW19, Theorem 6.4].

Theorem 7.14. *There exists a function $\delta(\varepsilon, m, d, q) = a(\varepsilon + md/q^c)^b$ for universal constants $a, b, c > 0$ such that the following holds. Let $\mathsf{qldparams} = (q, m, d, h, H, \Gamma, \pi)$ be an admissible parameter tuple and let $\mathscr{S}^{\mathrm{PAULI}}$ be the honest Pauli strategy corresponding to $\mathsf{qldparams}$. The game $\mathfrak{G}^{\mathrm{PAULI}}_{\mathsf{qldparams}}$ is a self-test for the honest binary Pauli strategy $\mathscr{S}^{\mathrm{BP}}$ corresponding to $\mathsf{qldparams}$ with robustness $\delta(\varepsilon, m, d, q)$.*

Proof. This follows from [NW19, Theorem 6.4], which states that the game $\mathfrak{G}^{\mathrm{PAULI}}_{\mathsf{qldparams}}$ is a self-test for the honest Pauli strategy $\mathscr{S}^{\mathrm{PAULI}}$ corresponding to $\mathsf{qldparams}$ with robustness $\delta(\varepsilon, m, d, q)$, and Lemma 3.29, which states that $\mathscr{S}^{\mathrm{PAULI}}$ is isomorphic to the honest binary Pauli strategy $\mathscr{S}^{\mathrm{BP}}$ corresponding to $\mathsf{qldparams}$. $\qquad\square$

7.3.2 Conditional linear functions for the Pauli basis test

We define CL functions that allow us to specify the distribution μ_{PAULI} of the Pauli basis test as a (typed) CL distribution. These CL functions will be used in the sampler for the introspective verifier in Section 8.

Fix a parameter tuple $\mathsf{qldparams} = (q, m, d, h, H, \Gamma, \pi)$. Let V^{PAULI} denote the linear space $\mathbb{F}_q^s$ for $s = 4m + 2$. The space V^{PAULI} is decomposed into a direct sum of the following register subspaces: V_X, V_Z, V_{V1}, V_{V2} (which are m-dimensional), and V_{R_X}, V_{R_Z} (which are 1-dimensional). We define CL functions $L^{\mathsf{t}} : V^{\mathrm{PAULI}} \to V^{\mathrm{PAULI}}$ for every type $\mathsf{t} \in \mathcal{T}^{\mathrm{PAULI}}$:

1. If $\mathsf{t} = (\mathrm{POINT}, W)$ for some $W \in \{X, Z\}$, then $L^{\mathrm{POINT},W}$ is the 1-level CL function L^{PT} (from Section 7.1.2) parameterized by field size q, dimension m, and point register V_W.

2. If $\mathsf{t} = (\mathrm{PLANE}, W)$ for some $W \in \{X, Z\}$, then $L^{\mathrm{POINT},W}$ is the 2-level CL function L^{PL} (from Section 7.1.2) parameterized by field size q, dimension m, point register V_W, and direction registers V_{V1}, V_{V2}.

3. If $\mathsf{t} = \mathrm{CONSTRAINT}_i$ for $i \in \{1, 2, \ldots, 6\}$, then the 1-level CL function $L^{\mathrm{CONSTRAINT}_i}$ is the projector onto $V_X \oplus V_Z \oplus V_{R_X} \oplus V_{R_Z}$.

4. If $\mathsf{t} = \mathrm{VARIABLE}_j$ for $j \in \{1, 2, \ldots, 9\}$, then the 1-level CL function L^{VARIABLE_j} is the projection onto $V_X \oplus V_Z \oplus V_{R_X} \oplus V_{R_Z}$.

5. If $\mathsf{t} = \mathrm{PAIR}$, then the 1-level CL function L^{PAIR} is the projection onto $V_X \oplus V_Z \oplus V_{R_X} \oplus V_{R_Z}$.

6. If $\mathsf{t} = (\mathrm{PAIR}, W)$, then the 1-level CL function $L^{\mathrm{PAIR},W}$ is the projection onto $V_X \oplus V_Z \oplus V_{R_X} \oplus V_{R_Z}$.

7. If $\mathsf{t} = (\mathrm{PAULI}, W)$ for some $W \in \{X, Z\}$, then the 0-level CL function $L^{\mathrm{PAULI},W}$ is identically 0.

Let ν denote the typed CL distribution on $\mathcal{T}^{\mathrm{PAULI}} \times V^{\mathrm{PAULI}} \times \mathcal{T}^{\mathrm{PAULI}} \times V^{\mathrm{PAULI}}$ where $(\mathsf{t}_A, x_A, \mathsf{t}_B, x_B)$ is sampled by first uniformly sampling an edge $(\mathsf{t}_A, \mathsf{t}_B)$ from the graph G^{PAULI} defined in Figure 7, sampling a uniformly random $z \in V^{\mathrm{PAULI}}$, and then setting $x_w = L^{\mathsf{t}_w}(z)$ for $w \in \{A, B\}$.

Lemma 7.15. *The distribution ν is a 2-level typed CL distribution. Furthermore, let $(\mathsf{t}_A, x_A, \mathsf{t}_B, x_B)$ be sampled from μ_{PAULI} described in Section 7.3.1. For $w \in \{A, B\}$ let $\tilde{x}_w$ be the vector x_w, seen as an element of V^{PAULI} (i.e. the vector is padded with zeroes whenever necessary). Then the distribution of $(\mathsf{t}_A, \tilde{x}_A, \mathsf{t}_B, \tilde{x}_B)$ is identical to the distribution ν.*

Proof. That ν is 2-level is immediate from the description. To see that the distributions are identical, the random vectors $u_X, u_Z, v_1, v_2, r_X, r_Z$ sampled in the description of μ_{PAULI} correspond, in the description of ν, to the projection of the random vector $z \in V^{\text{PAULI}}$ to the register subspaces V_X, V_Z, V_{V1}, V_{V2}, V_{R_X}, and V_{R_Z} respectively. $\qquad\square$

7.3.3 Canonical parameters and complexity of the Pauli basis test

We specify a canonical setting of the parameter tuple qldparams as a function of an integer r. We then give bounds on the complexity of computing the decision procedure and CL functions of the Pauli basis test corresponding to the parameter tuple qldparams, as a function of r.

Definition 7.16 (Canonical parameters of the Pauli basis test). Let c_0 denote the smallest of the universal constants a, b, c specified in Theorem 7.14, and let $c_1 = \max\{c_0, 2\}$. For all integers $r \in \mathbb{N}$, define the tuple $\text{introparams}(r) = (q, m, d, h, H, \Gamma, \pi)$ where

$$k = 2\lceil c_1 \log r \rceil + 1, \qquad q = 2^k, \qquad \Gamma = \lceil r / \log q \rceil + 1, \qquad m = \left\lceil \frac{2}{c_0} \cdot \frac{\log \Gamma}{\log r} \right\rceil,$$

and, using notation from Section 3.4.1, define $H = H_{\text{canon},m,k,\Gamma} \subseteq \mathbb{F}_q$, $h = h_{\text{canon},m,k,\Gamma} = |H|$, $\pi : \{1, 2, \ldots, \Gamma\} \to H^m$ as

$$\pi(i) = \pi_{\text{canon},m,k,\Gamma}(i-1)$$

for $i \in \{1, 2, \ldots, \Gamma\}$, where $\pi_{\text{canon},m,k,\Gamma} : \{0, 1, \ldots, \Gamma - 1\} \to H^m$, and $d = m(h-1)$.

Intuitively, this choice of parameter settings is such that the Pauli basis test certifies the presence of an r-qubit EPR state.

Lemma 7.17. *For all integers* $r \in \mathbb{N}$, *the parameter tuple* $\text{introparams}(r)$ *is admissible (see Definition 7.10), and furthermore there exist universal constants* $a', b' > 0$ *such that the function* $\delta(\varepsilon, m, d, q)$ *from Theorem 7.14 is at most* $a'(\varepsilon + \frac{1}{r})^{b'}$.

Proof. We verify the admissibility of $\text{introparams}(r)$ first. The field size $q = 2^k$ is admissible because k is odd. Fix $r \in \mathbb{N}$. From the definition of $h_{\text{canon},m,k,\Gamma}$ in Section 3.4.1, we get that

$$h = h_{\text{canon},m,k,\Gamma} = 2^{\lceil b(\Gamma)/m \rceil}$$

where $b(\Gamma) = \lfloor \log(\Gamma - 1) + 1 \rfloor$. Therefore $h^m \geq 2^{b(\Gamma)} \geq \Gamma$ and

$$\log h \leq \frac{b(\Gamma)}{m} \leq \frac{c_0}{2} \cdot \frac{1 + \log(\Gamma - 1)}{\log \Gamma} \cdot \log r \leq c_0 \log r \leq \log q \tag{49}$$

where in the last equality we used that $c_0 \leq c_1$. This proves admissibility of $\text{introparams}(r)$. Next, we show that md/q^{c_0} is at most an inverse polynomial function in r. We have

$$\frac{md}{q^{c_0}} \leq \frac{m^2 h}{q^{c_0}} \leq \frac{m^2 r^{c_0}}{q^{c_0}} \leq \frac{m^2 r^{c_0}}{r^{c_0 c_1}} = m^2 r^{-(c_1-1)c_0} \leq \lceil 2/c_0 \rceil^2 \cdot \lceil \log(r+2) \rceil^2 \cdot r^{-(c_1-1)c_0} \tag{50}$$

where the first inequality follows from the definition of d, the second inequality follows from (49), the third inequality follows from the definition of q, and the fourth inequality follows from the definition of m and

$\log q = k \geq 1$. Since $c_1 \geq 2$ by definition, the right-hand side of (50) is at most $a_0(1/r)^{b_0}$ for universal constants a_0, b_0.

Thus from the definition of $\delta(\varepsilon, m, d, q)$ from Theorem 7.14 we get,

$$\delta(\varepsilon, m, d, q) \leq a\left(\varepsilon + md/q^{c_0}\right)^b \leq a\left(\varepsilon + a_0(1/r)^{b_0}\right)^b \leq a'\left(\varepsilon + \frac{1}{r}\right)^{b'}$$

for some universal constants $a', b' > 0$. This completes the proof of the lemma. $\qquad\square$

Lemma 7.18. *Given an integer r, let* introparams$(r) = (q, m, d, h, H, \Gamma, \pi)$. *The following can be computed in time* polylog r *given r, written in binary, as input:*

1. *The integers q, m, d, h, Γ.*

2. *Given an additional input $x \in \mathbb{F}_q$, deciding if $x \in H$.*

3. *Given an additional input $x \in \{0, 1, \ldots, \Gamma - 1\}$, the value $\pi(x)$.*

Proof. The first item follows from the definitions of the parameters in Definition 7.16; the second item follows from the fact that H is defined to be the span of the first $\ell(\Gamma, m)$ elements of the canonical self-dual basis of $\mathbb{F}_q$ over $\mathbb{F}_2$ (as specified by Lemma 3.16). The last item follows from Lemma 3.23. $\qquad\square$

Lemma 7.19. *Let r be an integer, and let* introparams(r) *denote the parameter tuple specified by Definition 7.16.*

1. *The time complexity of the decision procedure $\mathcal{D}^{\mathrm{PAULI}}$ parameterized by* introparams(r) *is* poly(r).

2. *The time complexity of computing marginals of the CL functions L^{t} as well as the associated factor spaces, for $\mathsf{t} \in \mathcal{T}^{\mathrm{PAULI}}$, is* polylog r.

3. *The Turing machine description of the decision procedure $\mathcal{D}^{\mathrm{PAULI}}$ parameterized by* introparams *can be computed from the binary presentation of r in* polylog(r) *time.*

Proof. Finite field arithmetic over $\mathbb{F}_q$ can be performed in time polylog q, by Lemma 3.18. The parameters of introparams(r), which the decision procedure $\mathcal{D}^{\mathrm{PAULI}}$ implicitly computes given r, can be computed in time polylog(r), by Lemma 7.18. The most expensive step in $\mathcal{D}^{\mathrm{PAULI}}$ is to evaluate the low-degree encoding $g_{a,\pi}(y)$ where $a \in \mathbb{F}_q^\Gamma$ and $y \in \mathbb{F}_q^m$, which takes time poly$(m, d, \Gamma, \log q) = $ poly(r).

The complexity of computing the CL functions L^{t} for types $\mathsf{t} \in \mathcal{T}^{\mathrm{PAULI}}$ is dominated by the complexity of computing the CL functions L^{PL} and L^{PT} from the classical low-degree test, which takes time poly$(m, \log q) = $ polylog(r).

The factor spaces of L^{t} depend only on the question type t (of which there are only constantly many), and outputting the length-m indicator vectors of the factor spaces takes $O(m) = $ polylog(r) time.

Finally, the time complexity of computing the description of $\mathcal{D}^{\mathrm{PAULI}}$ from the binary representation of r requires polylog r time, because the checks performed in the decision procedure $\mathcal{D}^{\mathrm{PAULI}}$ depend on introparams which ultimately depends on r; we assume that the decision procedure computes the parameter tuple introparams based on r. Thus the time complexity is dominated by the time to write r in binary. $\qquad\square$

8 Introspection Games

8.1 Overview

Consider a normal form verifier $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ (see Definition 5.28). In this section we design a normal form verifier $\mathcal{V}^{\text{INTRO}} = (\mathcal{S}^{\text{INTRO}}, \mathcal{D}^{\text{INTRO}})$ (called the *introspective verifier*) such that in the n-th game $\mathcal{V}_n^{\text{INTRO}}$ (called the *introspection game*; see Definition 5.28 for the definition of the n-th game associated with a normal form verifier) the verifier expects the players to sample for themselves questions x and y distributed as their questions in the game $\mathcal{V}_N$ for index $N = 2^n$—this is the "introspection" step. Note the exponential separation of the indices of the introspection game versus the original game! Our construction of the introspection game generalizes the introspection technique of [NW19].

Recall the execution of the N-th game corresponding to the "original" verifier $\mathcal{V}$ (see Definition 5.28). Let $n \geq 1$, $N = 2^n$, and suppose that the CL functions of $\mathcal{S}$ on index N are $L^{\text{A}}, L^{\text{B}}$ acting on an ambient space V. In the game $\mathcal{V}_N$, the verifier first samples $z \in V$ uniformly at random and gives each player $w \in \{\text{A}, \text{B}\}$ the question $x_w = L^w(z)$. In this context, the string z is referred to as the "seed". The players respond with answers a_{A} and a_{B}, respectively, and the verifier accepts or rejects according to the output of $\mathcal{D}(N, x_{\text{A}}, x_{\text{B}}, a_{\text{A}}, a_{\text{B}})$.

In the introspection game, with some constant probability independent of n, the verifier $\mathcal{V}_n^{\text{INTRO}}$ sends the question pair (INTRO, A) to player w and (INTRO, B) to the other player $\overline{w}$, where $w \in \{\text{A}, \text{B}\}$ and recall that $\overline{w} = \text{B}$ if $w = \text{A}$ and $\overline{w} = \text{A}$ otherwise. The verifier expects player w to measure their share of the state $|\text{EPR}\rangle_V$ using a coarse-grained Z-basis measurement whose outcomes range over $L^{\text{A}}(V)$, and similarly player $\overline{w}$ measures the state $|\text{EPR}\rangle_V$ using a coarse-grained Z-basis measurement with outcomes that range over $L^{\text{B}}(V)$. If the players perform these measurements honestly, then the outcomes $(x_{\text{A}}, x_{\text{B}})$ are distributed exactly according to $\mu_{\mathcal{S}, N}$, the question distribution of the game $\mathcal{V}_N$. Players w and $\overline{w}$ are then expected to respond with the question x_w and $x_{\overline{w}}$ that they each sampled, together with strings a_w and $a_{\overline{w}}$, respectively, which are intended to be the answers for the question pair $(x_{\text{A}}, x_{\text{B}})$ in the game $\mathcal{V}_N$. In other words, the players introspectively sample the question pair $(x_{\text{A}}, x_{\text{B}})$ and then respond with the question itself and an answer for it.

To facilitate comprehension, we call the players that interact with the introspective verifier $\mathcal{V}^{\text{INTRO}}$ the "introspecting players", and the players that interact with the "original" verifier $\mathcal{V}$ the "original players".

To ensure that the introspecting players follow the above procedure honestly, the introspective verifier $\mathcal{V}_n^{\text{INTRO}}$ first uses the (binary) Pauli basis test described in Section 7.3 to force the introspecting players to share the state $|\text{EPR}\rangle_V$. The Pauli basis test also ensures that the players measure σ^W and report the outcome honestly when they receive questions (PAULI, W) for $W \in \{X, Z\}$. For $v \in \{\text{A}, \text{B}\}$ the verifier cross-checks the question pairs (INTRO, v) and (PAULI, Z) to enforce that the honest measurement is performed for question (INTRO, v).

The main difficulty in the soundness analysis is to ensure that the answer of player w who received question (INTRO, v) depends only on $L^v(z)$ and not on any other information about the string z. First assume for simplicity that L^v is a linear function. As shown below (based on Lemma 8.4), $L^v(z)$ can be obtained by measuring a specific collection of σ^Z observables; namely, the set

$$\{\sigma^Z(u) \mid u \in \ker(L^v)^{\perp}\}. \tag{51}$$

To prevent the player from obtaining any additional information the verifier needs to enforce that the player does *not* additionally measure any $\sigma^Z(u)$ for $u \notin \ker(L^v)^{\perp}$. (We refer to any such σ^Z as a "prohibited" σ^Z.)

The introspective verifier achieves this by sometimes sending "hiding questions" (READ, v) and (HIDE_k, v) to the players. When receiving the READ questions, the players are required to also measure observables from the set

$$\{\sigma^X(r) \mid r \in \ker(L^v)\}, \tag{52}$$

which (as shown in Lemma 8.5 below) commute with every Z-basis measurement in (51). On the other hand, any prohibited $\sigma^Z(u)$ must anticommute with at least one of the $\sigma^X(r)$, as otherwise u would be in $\ker(L^v)^\perp$. As a result, honestly measuring the σ^X observables of (52) has the effect of preventing the player from measuring any of the prohibited σ^Z observables, so that the answer a can effectively only depend on the question $L^v(z)$. (In the protocol, the verifier asks the player to measure the function $(L^v)^\perp(z)$ in the X basis, rather than all of the σ^X observables. By Lemma 8.4 the two are equivalent.) Similarly to how questions (PAULI, Z) and (INTRO, v) are cross-checked, the questions (PAULI, X), (HIDE_k, v) and (READ, v) are cross-checked in order to ensure that the honest X-basis measurements are performed for the hiding questions.

When the CL functions L^v are ℓ-level for $\ell > 1$, the introspective verifier sends one of $O(\ell)$ different hiding questions to the players, chosen at random; together these hiding checks ensure that each of the constituent linear maps of L^v are honestly measured. Intuitively, these hiding questions "interpolate" between questions (PAULI, Z), (INTRO, v) and (PAULI, X) in a way that, for all pairs of questions asked by the verifier, the honest measurements commute. (See Figure 11 for an overview of the honest measurements.)

A key property of the introspection game is that the distribution of questions (which include the Pauli basis test questions as well as the introspection questions and the hiding questions) is also conditionally linear. This means that the introspection game can be ultimately specified by a normal form verifier $\mathcal{V}^{\text{INTRO}} = (\mathcal{S}^{\text{INTRO}}, \mathcal{D}^{\text{INTRO}})$, which is crucial for recursive compression of games. Moreover, while the time complexity of the introspective verifier's decider $\mathcal{D}^{\text{INTRO}}$ remains polynomially related to that of $\mathcal{D}$ on index N, the time complexity of the sampler $\mathcal{S}^{\text{INTRO}}$ is $\text{polylog}(N)$ (exponentially smaller), due to the efficiency of the Pauli basis test. Finally, $\mathcal{S}^{\text{INTRO}}$ only depends on $\mathcal{V}$ through the number of levels ℓ of $\mathcal{S}$ and an upper bound on its randomness complexity, as well as upper bounds on the time complexities of $\mathcal{S}$ and $\mathcal{D}$.

8.2 The introspective verifier

Let $\lambda, \ell \in \mathbb{N}$. Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier, where $\mathcal{S}$ is an ℓ-level sampler. We call $\mathcal{V}, \mathcal{S}$ and $\mathcal{D}$ the "original" verifier, sampler, and decider, respectively. We assume that for all $N \in \mathbb{N}$, the original sampler and decider satisfy

$$\max\left\{\text{RAND}_{\mathcal{S}}(N), \text{TIME}_{\mathcal{S}}(N), \text{TIME}_{\mathcal{D}}(N)\right\} \leq (\lambda N)^\lambda. \tag{53}$$

The *introspective verifier corresponding to $\mathcal{V}$ and parameters* (λ, ℓ) is a *typed* normal form verifier $\mathcal{V}^{\text{INTRO}} = (\mathcal{S}^{\text{INTRO}}, \mathcal{D}^{\text{INTRO}})$, sketched in Section 8.1 and specified in detail in the present section (see Section 6 for the definition of typed normal form verifiers). In the following descriptions of the sampler $\mathcal{S}^{\text{INTRO}}$ and decider $\mathcal{D}^{\text{INTRO}}$, all parameters are functions of the index n, the number of levels ℓ of the sampler $\mathcal{S}$, and the parameter λ; we often leave this dependence implicit. We use $N = 2^n$ to denote the index of the verifier $\mathcal{V}$ that is simulated by the introspective verifier $\mathcal{V}^{\text{INTRO}}$ on index n.

Let $r = (\lambda N)^\lambda$, and let $\text{introparams}(r) = (q, m, d, h, H, \Gamma, \pi)$ denote the parameter tuple specified in Section 7.3.3. Note that introparams is implicitly a function of n (since r is a function of n). By (53), the integer r is an upper bound on the randomness complexity $\text{RAND}_{\mathcal{S}}(N)$ of the sampler $\mathcal{S}$ on index N. The associated parameter tuple introparams is intended to parametrize a Pauli basis test that certifies an

r-qubit EPR state; intuitively, the r-qubit EPR state is meant to serve as the source of randomness for the CL functions of the original sampler $\mathcal{S}$.

Recall that the players in the introspection game are referred to as "introspecting players" and the players in the original game are referred to as "original players". We use the following notation in order to distinguish between questions and answers meant for the introspecting players versus the original players. The questions and answers of the introspecting players are denoted by hatted variables (e.g., $\hat{x}$ and $\hat{a}$). Similarly, the associated question types are denoted using hatted variables $\hat{t}$. The questions and answers of the original players in the original game $\mathcal{V}_N$ are denoted using non-hatted variables (e.g. x, a, and so on).

Types and type graph. The type set $\mathcal{T}^{\text{INTRO}}$ for the introspective verifier $\mathcal{V}^{\text{INTRO}}$ is

$$\mathcal{T}^{\text{INTRO}} = \mathcal{T}^{\text{PAULI}} \cup \left(\left(\{ \text{INTRO}, \text{SAMPLE}, \text{READ} \} \cup \left(\bigcup_{k=1}^{\ell} \{ \text{HIDE}_k \} \right) \right) \times \{ \text{A}, \text{B} \} \right),$$

where $\mathcal{T}^{\text{PAULI}}$ is the type set of the Pauli basis test, defined in Section 7.3. The type graph G^{INTRO} is specified in Figure 9.
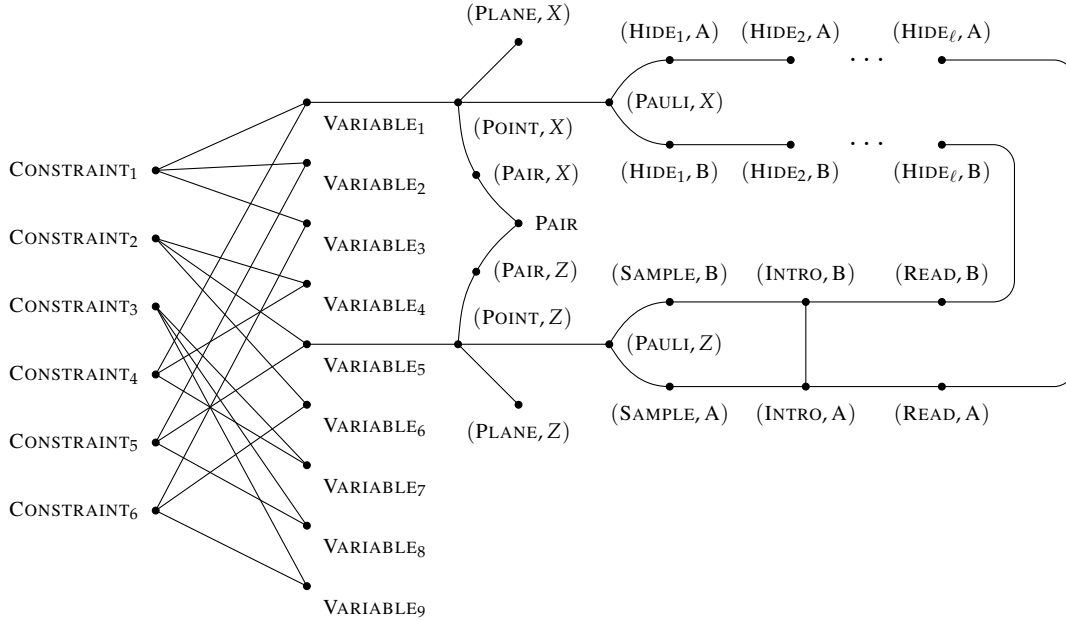
Figure 9: Type graph G^{INTRO} for the introspection game. Each vertex also has a self-loop which is not drawn in the figure for clarity.

Sampler. We first define a 2-level $(\mathcal{T}^{\text{INTRO}}, G^{\text{INTRO}})$-typed sampler $\tilde{\mathcal{S}}^{\text{INTRO}}$, which has field size $q(n)$ and dimension $4m(n) + 2$, where $q(n)$ and $m(n)$ are specified by $\text{introparams}(r)$. Note that the dimension is the same as that of the ambient space of the CL functions of the Pauli basis test for r qubits, specified in Section 7.3.2.

Fix $n \in \mathbb{N}$. We specify the CL functions of $\tilde{\mathcal{S}}^{\text{INTRO}}$ on index n. Since the functions $L_{\hat{t}}^{\text{A}, n}$ and $L_{\hat{t}}^{\text{B}, n}$ are identical for all n and $\hat{t}$, we omit the superscripts A and B. For types $\hat{t} \in \mathcal{T}^{\text{PAULI}}$, the CL functions $L_{\hat{t}}^{n}$ are

given by those specified in Section 7.3.2, parameterized by introparams(r). For types $\hat{\mathfrak{t}} \in \mathcal{T}^{\text{INTRO}} \setminus \mathcal{T}^{\text{PAULI}}$, the associated CL functions are defined to be 0-level CL functions (i.e., they are the 0 map). This means that for question types such as $\hat{\mathfrak{t}} = (\text{INTRO}, v)$ or $\hat{\mathfrak{t}} = (\text{SAMPLE}, v)$ for $v \in \{\text{A}, \text{B}\}$ the associated question is solely comprised of the question type label.

Finally, we define the typed sampler $\mathcal{S}^{\text{INTRO}}$ to be $\kappa(\tilde{\mathcal{S}}^{\text{INTRO}})$, the downsized sampler (Definition 6.6) corresponding to $\tilde{\mathcal{S}}^{\text{INTRO}}$. By Lemma 7.15 the typed CL functions associated with the Pauli basis test are 2-level; using Remark 4.2 and Lemma 4.9 it follows that $\mathcal{S}^{\text{INTRO}}$ is a 2-level typed sampler.

The following lemma establishes the complexity of the sampler $\mathcal{S}^{\text{INTRO}}$ as well as the complexity of computing a description of it from the parameters (λ, ℓ).

Lemma 8.1. *There exists a 2-input Turing machine* ComputeIntroSampler *that on input* (λ, ℓ) *outputs a description of the sampler* $\mathcal{S}^{\text{INTRO}}$ *in time* polylog(λ, ℓ). *Furthermore,*

1. $\text{TIME}_{\mathcal{S}^{\text{INTRO}}}(n) = \text{polylog}((\lambda 2^n)^\lambda, \ell),$

2. $\text{RAND}_{\mathcal{S}^{\text{INTRO}}}(n) = \text{polylog}((\lambda 2^n)^\lambda),$ *and*

3. $\mathcal{S}^{\text{INTRO}}$ *is a* 2-*level typed sampler.*

Proof. Define the following 9-input Turing machine $\hat{\mathcal{S}}^{\text{INTRO}}$, that does not depend on any parameters (so that its description length is constant). On input $(\lambda, \ell, n, x_1, \ldots, x_6)$, $\hat{\mathcal{S}}^{\text{INTRO}}$ computes the output of the typed sampler $\mathcal{S}^{\text{INTRO}}$ (parameterized by (λ, ℓ)) with input tapes set to $(n, x_1, \ldots, x_6)$. In more detail, the Turing machine $\hat{\mathcal{S}}^{\text{INTRO}}$ first computes introparams(r) for $r = (\lambda N)^\lambda$ and $N = 2^n$. Using Lemma 7.18, this computation takes time poly $\log(r)$. Next, depending on the contents of the last 7 input tapes of $\hat{\mathcal{S}}^{\text{INTRO}}$, the Turing machine evaluates the dimension of $\mathcal{S}^{\text{INTRO}}$ (which can easily be computed from introparams(r)), or one of the CL functions, or returns a representation of a factor space of $\mathcal{S}^{\text{INTRO}}$. If the type passed as input is $\hat{\mathfrak{t}} \in \mathcal{T}^{\text{PAULI}}$ then by Lemma 7.19 this takes time polylog(r). If $\hat{\mathfrak{t}} \in \mathcal{T}^{\text{INTRO}} \setminus \mathcal{T}^{\text{PAULI}}$ then this can be done in $O(\log \ell)$ time (to read the input type). Overall, $\hat{\mathcal{S}}^{\text{INTRO}}$ runs in time poly $\log(r, \ell)$.

We now define the Turing machine ComputeIntroSampler: on input (λ, ℓ), it outputs the description of $\hat{\mathcal{S}}^{\text{INTRO}}$ with the first two input tapes hardwired to λ and ℓ, respectively, yielding the sampler $\mathcal{S}^{\text{INTRO}}$ corresponding to parameters (λ, ℓ). Computing this description takes $O(\log \lambda + \log \ell)$ time.

The time complexity of $\mathcal{S}^{\text{INTRO}}$ follows from the time complexity of $\hat{\mathcal{S}}^{\text{INTRO}}$, the randomness complexity follows from the dimension of the ambient space $4m(n) + 2 = \text{polylog}(r)$, and the number of levels is by construction. $\qquad\square$

Decider. The typed decider $\mathcal{D}^{\text{INTRO}}$ is specified in Figure 10. We explain how to interpret the figure, including the notation. (It may also be helpful to review the description of the intended strategy for the players in the game, described in Section 8.3.2.)

Question and answer format. The decider takes as input a tuple $(n, \hat{\mathfrak{t}}_{\text{A}}, \hat{x}_{\text{A}}, \hat{\mathfrak{t}}_{\text{B}}, \hat{x}_{\text{B}}, \hat{a}_{\text{A}}, \hat{a}_{\text{B}})$ where $(\hat{\mathfrak{t}}_w, \hat{x}_w)$ denotes the question for introspecting player $w \in \{\text{A}, \text{B}\}$, and $\hat{a}_w$ denotes their answer. As in the specification of the Pauli basis test, in Figure 10 we include an "answer key" indicating how the players' answers are parsed, depending on the question type. When the question type is from $\mathcal{T}^{\text{PAULI}}$, the question and answer format are as described in Figure 8. When the question type is in $\mathcal{T}^{\text{INTRO}} \setminus \mathcal{T}^{\text{PAULI}}$, the answer format is described in the table at the top of Figure 10.[20] For each such question, there is an associated variable

---

[20]There is no question format specification for the question types in $\mathcal{T}^{\text{INTRO}} \setminus \mathcal{T}^{\text{PAULI}}$, because the question is solely comprised of the question type label.

Type	Answer format
$(\textsc{Intro}, v)$	$(y, a) \in V \times \{0,1\}^*$
$(\textsc{Sample}, v)$	$(z, a) \in V \times \{0,1\}^*$
$(\textsc{Read}, v)$	$(y, y^\perp, a) \in V \times V \times \{0,1\}^*$
$(\textsc{Hide}_k, v)$	$(y, y^\perp, x) \in V \times V \times V$

In the following, whenever $\mathcal{S}$ or $\mathcal{D}$ is called, $\mathcal{D}^{\textsc{Intro}}$ aborts and rejects if the computation takes more than $(\lambda N)^\lambda$ time steps. On input $(n, \hat{t}_A, \hat{x}_A, \hat{t}_B, \hat{x}_B, \hat{a}_A, \hat{a}_B)$, the decider $\mathcal{D}^{\textsc{Intro}}$ first computes the dimension $s(N)$ of V by calling the original sampler $\mathcal{S}$ on input $(N, \textsc{Dimension})$. If $s(N) > (\lambda N)^\lambda$ the decider rejects. The decider then performs an answer length check: if

$$\max\{|\hat{a}_A|, |\hat{a}_B|\} \geq 6s(N) + 2(\lambda N)^\lambda + 3, \tag{54}$$

then the decider rejects. Otherwise, it performs the following tests for all $w, v \in \{A, B\}$. (If no test applies, the decider accepts.)

1. **(Pauli test).** If $\hat{t}_A, \hat{t}_B \in \mathcal{T}^{\textsc{Pauli}}$, accept if $\mathcal{D}^{\textsc{Pauli}}_{\text{introparams}}$ accepts the input $(\hat{t}_A, \hat{x}_A, \hat{t}_B, \hat{x}_B, \hat{a}_A, \hat{a}_B)$.

2. **(Sampling tests).**

   (a) If $\hat{t}_w = (\textsc{Pauli}, Z)$ and $\hat{t}_{\overline{w}} = (\textsc{Sample}, v)$, accept if $\hat{a}^V_w = z_{\overline{w}}$.

   (b) If $\hat{t}_w = (\textsc{Intro}, v)$, $\hat{t}_{\overline{w}} = (\textsc{Sample}, v)$, accept if $y_w = L^v(z_{\overline{w}})$ and $a_w = a_{\overline{w}}$.

3. **(Hiding tests).**

   (a) If $\hat{t}_w = (\textsc{Intro}, v)$, $\hat{t}_{\overline{w}} = (\textsc{Read}, v)$, accept if $y_w = y_{\overline{w}}$ and $a_w = a_{\overline{w}}$.

   (b) If $\hat{t}_w = (\textsc{Hide}_\ell, v)$, $\hat{t}_{\overline{w}} = (\textsc{Read}, v)$, accept if $y_{w, <\ell} = y_{\overline{w}, <\ell}$, and $y^\perp_w = y^\perp_{\overline{w}}$.

   (c) If $\hat{t}_w = (\textsc{Hide}_k, v)$, $\hat{t}_{\overline{w}} = (\textsc{Hide}_{k+1}, v)$ for some $k \in \{1, 2, \ldots, \ell - 1\}$, accept if

   $$y_{w, <k} = y_{\overline{w}, <k}, \quad y^\perp_{w, \leq k} = y^\perp_{\overline{w}, \leq k}, \quad x_{w, >k+1} = x_{\overline{w}, >k+1},$$

   and $y^\perp_{\overline{w}, k+1} = \left(L^v_{k+1, u}\right)^\perp (x_{w, k+1})$ where $u = y_{\overline{w}, \leq k}$.

   (d) If $\hat{t}_w = (\textsc{Pauli}, X)$, $\hat{t}_{\overline{w}} = (\textsc{Hide}_1, v)$, accept if $y^\perp_{\overline{w}, 1} = \left(L^v_1\right)^\perp (\hat{a}^{V_1}_w)$ and $\hat{a}^{V>1}_w = x_{\overline{w}, >1}$.

4. **(Game test).** If $\hat{t}_w = (\textsc{Intro}, A)$ and $\hat{t}_{\overline{w}} = (\textsc{Intro}, B)$, accept if $\mathcal{D}$ accepts $(N, y_w, y_{\overline{w}}, a_w, a_{\overline{w}})$ for $N = 2^n$.

5. **(Consistency test).** If $\hat{t}_A = \hat{t}_B$, accept if and only if $\hat{a}_A = \hat{a}_B$.

Figure 10: The typed decider $\mathcal{D}^{\textsc{Intro}}$ for the introspective verifier, parameterized by integers λ, ℓ, on index n. N denotes 2^n, V is the ambient space for $\mathcal{S}$, and $\{L^v\}_{v \in \{A, B\}}$ the associated CL functions on index N.

$v \in \{A, B\}$ that indicates to the introspecting player which original player it is supposed to impersonate in the introspection game.

In the figure, V denotes the ambient space of the original sampler $\mathcal{S}$ on index $N = 2^n$. Since V is isomorphic to $\mathbb{F}_2^{s(N)}$, where $s(N)$ is the dimension of $\mathcal{S}$, the space V is identified in a canonical way as the register subspace of $\mathbb{F}_2^r$ spanned by $e_1, \ldots, e_{s(N)}$ where e_i is the i-th elementary basis vector. For example, if $\hat{\mathfrak{t}}_w = (\text{READ}, v)$, then syntactically the player's answer is a triple $(y, y^{\perp}, a)$ in $\mathbb{F}_2^r \times \mathbb{F}_2^r \times \{0, 1\}^*$. We assume that the decider $\mathcal{D}$ computes the dimension $s(N)$ of the subspace V by calling $\mathcal{S}$ on input $(N, \text{DIMENSION})$, and if $y, y^{\perp}$ are not presented as vectors in the subspace V, then the decider rejects. Thus in the analysis we directly consider $y, y^{\perp}$ as vectors in V. In Figure 10, the components of the players' answers are subscripted by the player index. For example, if player w receives question (HIDE_k, v), then their answers are denoted by $(y_w, y_{\overline{w}}^{\perp}, x_w)$.

The notation used in the "answer key" is meant to give an indication of the intended meaning of the players' answers. We use y to denote a vector that is supposed to be the result of measuring a CL function L^v; $y^{\perp}$ is supposed to be the result of measuring "dual" linear maps $L^{\perp}$ (as used in Step 3 in Figure 10); x is supposed to be the result of σ^X measurements, and z is supposed to be the result of σ^Z measurements. We use a to denote the introspected answers meant for the original decider $\mathcal{D}$.

CL functions and factor spaces. For $v \in \{A, B\}$, let $L^v = L^{v,N}$ denote the CL function for original player v specified by $\mathcal{S}$ on index $N = 2^n$. For $z \in V$, the decider $\mathcal{D}$ computes $L^v(z)$ by calling $\mathcal{S}$ on input $(N, v, \text{MARGINAL}, \ell, z)$.

For $y \in V$ and $k \in \{1, \ldots, \ell\}$ we define register subspaces $V_k^v(y)$ by induction on k. For $k = 1$, $V_1^v(y)$ is the first factor space[21] of L^v and is independent of y. Suppose $V_j^v(y)$ has been defined for all $j < k$. Then we define the marginal space $V_{<k}^v(y) = \bigoplus_{j=1}^{k-1} V_k^v(y)$, and define $V_k^v(y)$ as the k-th factor space $V_{k,u}^v$ of L^v with prefix $u = y^{V_{<k}^v(y)}$, the projection of y to $V_{<k}^v(y)$. We also define $V_{\leq k}^v(y) = V_{<k+1}^v(y)$, and $V_{>k}^v(y)$ to be the complementary register subspace to $V_{\leq k}^v(y)$ within V.

The decider $\mathcal{D}^{\text{INTRO}}$ computes factor spaces $V_j^v(y)$ from $y \in V$ in the following sequential manner: first, the indicator vector for the factor space V_1^v is computed by calling the original sampler $\mathcal{S}$ on input $(N, v, \text{FACTOR}, 1, 0)$. Let y_1 denote the projection of y to V_1^v. Then, for $j \in \{2, \ldots, \ell\}$, the factor space $V_j^v(y)$ is computed by calling $\mathcal{S}$ on input $(N, v, \text{FACTOR}, j, y_{\leq j-1})$, where $y_{\leq j-1}$ is the projection of y to $V_{\leq j-1}^v(y)$.

We give more details on the implementation of decider $\mathcal{D}^{\text{INTRO}}$ specified in Figure 10.

1. The decider $\mathcal{D}^{\text{INTRO}}$ first checks that the answers are not too long; the maximum length answer should be either a tuple $(y, y^{\perp}, a)$ where $y, y^{\perp} \in V$ and a is an answer intended for the original decider $\mathcal{D}$ on index N (which we assume runs in time at most $(\lambda N)^{\lambda}$), or a tuple $(y, y^{\perp}, x) \in V \times V \times V$. This check is necessary in order to ensure that the decider $\mathcal{D}^{\text{INTRO}}$ halts in time $\text{poly}(N)$. The bound (54) is explained by the encoding for tuples specified in Remark 3.2.

2. If the question types for the players are both in $\mathcal{T}^{\text{PAULI}}$, $\mathcal{D}^{\text{INTRO}}$ executes the decision procedure $\mathcal{D}^{\text{PAULI}}$ for the Pauli basis test parametrized by introparams (see Section 7.3.1 for the definition of $\mathcal{D}^{\text{PAULI}}$). The Pauli basis test is intended to ensure that the players share an r-qubit EPR state, where $r = (\lambda N)^{\lambda}$. Since the EPR state is measured to introspectively sample questions according to the original sampler $\mathcal{S}$, it is necessary that r is at least as large as the randomness complexity of $\mathcal{S}$, which is equal

---

[21] See Lemma 4.4 for the definitions of factor spaces of a CL function.

to the dimension of $\mathcal{S}$ (since for a normal form verifier the field size is taken to be $q(n) = 2$). This is satisfied under the assumption stated in (53).

3. In Step 2a of $\mathcal{D}^{\text{INTRO}}$, player $w \in \{A, B\}$ receives question (PAULI, Z) and player $\overline{w}$ receives question (SAMPLE, v), for some $v \in \{A, B\}$. According to the answer key, player w is expected to return an answer $\hat{a}_w \in \mathbb{F}_2^r$ and player $\overline{w}$ is expected to respond with a pair $(y_{\overline{w}}, a_{\overline{w}}) \in V \times \{0,1\}^*$. Thus, the dimension of answer $\hat{a}_w$ may be larger than that of $y_{\overline{w}}$; this is the reason that Step 2a checks consistency between $y_{\overline{w}}$ and the projection of $\hat{a}_w$ to V.

4. In Step 2b of $\mathcal{D}^{\text{INTRO}}$, player $w \in \{A, B\}$ receives question (INTRO, v) and player $\overline{w}$ receives question (SAMPLE, v). As specified by the "answer key", player w responds with $(y_w, a_w) \in V \times \{0,1\}^*$ and player $\overline{w}$ responds with $(z_{\overline{w}}, a_{\overline{w}}) \in V \times \{0,1\}^*$. The decider $\mathcal{D}^{\text{INTRO}}$ checks that $a_w = a_{\overline{w}}$ and $y_w = L^v(z_{\overline{w}})$ where L^v denotes the CL function for player v specified by $\mathcal{S}$ for index $N = 2^n$. The CL function is computed by calling $\mathcal{S}$ on input $(N, v, \text{MARGINAL}, \ell, z)$.

5. In Step 3b, $\mathcal{D}^{\text{INTRO}}$ checks that the answer of player w who received (HIDE_ℓ, v) is consistent with the answer of player $\overline{w}$ who received (READ, v). One of the checks is that $y_{w, <\ell} = y_{\overline{w}, <\ell}$; these are, respectively, the projections of y_w to $V_{<\ell}^v(y_w)$ and $y_{\overline{w}}$ to $V_{<\ell}^v(y_{\overline{w}})$.

6. In Steps 3c, the vectors $x_{w, >k+1}$ and $x_{\overline{w}, >k+1}$ denote the projections of x_w and $x_{\overline{w}}$ to $V_{>k+1}^v(y_w)$ and $V_{>k+1}^v(y_{\overline{w}})$, respectively. Similarly, $y_{\overline{w}, k+1}^\perp$ denotes the projection of $y_{\overline{w}}^\perp$ to $V_{k+1}^v(y_{\overline{w}})$ and $x_{w, k+1}$ denotes the projection of x_w to $V_{k+1}^v(y_{\overline{w}})$. Note that the factor spaces depend on y_w and $y_{\overline{w}}$.

   The decider also has to compute $(L_{k+1, y_{\overline{w}, \leq k}}^v)^\perp (x_{w, k+1})$. According to Definition 3.11, this requires specifying a basis for $\ker(L_{k+1, y_{\overline{w}, \leq k}}^v)^\perp$. To compute the value, the decider performs the following steps:

   (a) Call $\mathcal{S}$ on input $(N, v, \text{FACTOR}, k+1, y_{w, \leq k})$ to obtain a subset $H = \{h_1, \ldots, h_m\}$ of the canonical basis for $\mathbb{F}_2^{s(N)}$ that is a basis of the register subspace $V_{k+1}^v(y_{\overline{w}})$.

   (b) For $i \in \{1, 2, \ldots, m\}$ compute $c_i = \mathcal{S}(N, v, \text{LINEAR}, k+1, y_{\overline{w}, \leq k}, h_i)$. Compute a matrix representation M for $L_{k+1, y_{\overline{w}, \leq k}}^v$ in the basis H, whose columns are the vectors $c_1, \ldots, c_m$ as elements of $V_{k+1}^v(y_{\overline{w}})$.

   (c) Using a canonical algorithm for Gaussian elimination, compute a basis F for $\ker(M)$.

   (d) Compute the canonical complement S of F, as in Definition 3.7. S is a basis for $\ker(L_{k+1, y_{\overline{w}, \leq k}}^v)^\perp$.

   (e) To compute $(L_{k+1, y_{\overline{w}, \leq k}}^v)^\perp$ on input $x_{w, k+1}$, compute the canonical linear map with kernel basis S (see Definition 3.10) on input $x_{w, k+1}$.

7. In Step 3d, the player w that receives (PAULI, X) is expected to return an answer $\hat{a}_w$ in $\mathbb{F}_2^r$. Part of this step checks that the projection of $\hat{a}_w$ to $V_{>1}$ is equal to $x_{\overline{w}, >1}$ (which is the projection to $V_{>1}$ of the third component of the answer triple of player $\overline{w}$ that receives question (HIDE_1, v)).

The following lemma establishes the complexity of the decider $\mathcal{D}^{\text{INTRO}}$ as well as the complexity of computing a description of it from the parameters (λ, ℓ) and the description of the original verifier $\mathcal{V} = (\mathcal{S}, \mathcal{D})$.

Lemma 8.2. *There exists a 4-input Turing machine* ComputeIntroDecider *that on input* $(\mathcal{V}, \lambda, \ell)$ *outputs a description of the decider* $\mathcal{D}^{\text{INTRO}}$ *in time* $\text{poly}(|\mathcal{V}|, \log \lambda, \log \ell)$. *Furthermore, the decider* $\mathcal{D}^{\text{INTRO}}$ *has time complexity* $\text{TIME}_{\mathcal{D}^{\text{INTRO}}}(n) = \text{poly}((\lambda 2^n)^\lambda, \ell)$.

Proof. Define the following 10-input Turing machine $\hat{\mathcal{D}}^{\textsc{Intro}}$. The Turing machine does not depend on any parameters, so its size is constant. On input

$$(\mathcal{V}, \lambda, \ell, n, x_1, x_2, \ldots, x_6),$$

$\hat{\mathcal{D}}^{\textsc{Intro}}$ computes the output of the decider $\mathcal{D}^{\textsc{Intro}}$ with input tapes set to $(n, x_1, \ldots, x_6)$. In more detail, the Turing machine $\hat{\mathcal{D}}^{\textsc{Intro}}$ first computes $\mathsf{introparams}(r)$ for $r = (\lambda N)^\lambda$ and $N = 2^n$. Using Lemma 7.18, this computation takes time $\mathrm{polylog}(r)$. It then executes the tests described in Figure 10. Write $\mathcal{V} = (\mathcal{S}, \mathcal{D})$. The complexity of performing the entire procedure is subsumed by the complexity of the following steps:

1. Running the decider $\mathcal{D}^{\textsc{Pauli}}$, which takes time $\mathrm{poly}(r)$ by Lemma 7.19;

2. Running the original decider $\mathcal{D}$ (on index $N = 2^n$) for at most $(\lambda N)^\lambda$ steps;

3. Running the original sampler $\mathcal{S}$ (on index N) in order to compute the dimension $s(N)$ and the marginal and factor spaces, and the CL functions as described in Section 8.2. $\mathcal{S}$ is called at most $\mathrm{poly}(s(N), \ell)$ times; due to the timeout, each call takes time at most $(\lambda N)^\lambda$.

4. Computing $(L_{k,u}^v)^\perp(x_{\overline{w},k})$ in Step 3c. This only requires to perform Gaussian elimination and other simple finite field manipulations that can be implemented in $\mathrm{poly}(s(N), \log|\mathbb{F}|)$ time.

All other tests are elementary. Thus the time complexity of $\hat{\mathcal{D}}^{\textsc{Intro}}$ is $\mathrm{poly}(r, \ell)$. Note that the bound is independent of $\mathcal{V}$: due to the abort condition in the definition of $\mathcal{D}^{\textsc{Intro}}$, the Turing machine $\hat{\mathcal{D}}^{\textsc{Intro}}$ aborts if the runtime of $\mathcal{S}$ or $\mathcal{D}$ is larger than $(\lambda N)^\lambda$.

We now define the Turing machine ComputeIntroDecider: on input $(\mathcal{V}, \lambda, \ell)$, it outputs the description of $\hat{\mathcal{D}}^{\textsc{Intro}}$ with the first three input tapes hardwired to $\mathcal{V}$, λ, and ℓ, respectively, yielding the decider $\mathcal{D}^{\textsc{Intro}}$ corresponding to original verifier $\mathcal{V}$ and parameters (λ, ℓ). Computing this description takes $\mathrm{poly}(|\mathcal{V}|, \log \lambda, \log \ell)$ time. The time complexity of $\mathcal{D}^{\textsc{Intro}}$ follows from the time complexity of $\hat{\mathcal{D}}^{\textsc{Intro}}$. $\qquad\square$

8.3 Completeness and complexity of the introspective verifier

In this section we determine the complexity of the introspective verifier and establish the completeness property of the introspection game: if for $N = 2^n$, $\mathcal{V}_N$ has a PCC strategy (see Definition 5.11) with value 1, then so does $\mathcal{V}_n^{\textsc{Intro}}$. For this we describe the actions that are expected of the players in the introspection game (i.e. the "honest strategy"). We first prove several preliminary lemmas that will be used in both the completeness and soundness analysis.

8.3.1 Preliminary lemmas

The lemmas in this section are stated for general fields $\mathbb{F}$ and generalized Pauli observables and projectors, although in the application to introspection games we use $\mathbb{F} = \mathbb{F}_2$, $\omega = -1$, and qubit Pauli observables and projectors. Furthermore, the Pauli observables $\tau^W(v)$ and projectors τ_u^W act on $\mathbb{C}^{\mathbb{F}^k}$ for some integer k (in our application, we set $k = r$).

Lemma 8.3 (Fact 3.2 of [NW19]). *Let V be a subspace of $\mathbb{F}^k$. For all $v \notin V^\perp$,*

$$\mathop{\mathbb{E}}_{u \sim V} \omega^{\mathrm{tr}(u \cdot v)} = 0,$$

where the expectation is over a uniformly random vector u from V.

The next lemma generalizes Eq. (18).

Lemma 8.4. *Let $L : \mathbb{F}^k \to \mathbb{F}^k$ be a linear map, and let $W \in \{X, Z\}$. For each a in the range of L, let $u_a \in \mathbb{F}^k$ be such that $L(u_a) = a$.*

1. For each $v \in \ker(L)^\perp$,

$$\tau^W(v) = \sum_{a \in \mathbb{F}^k} \omega^{\mathrm{tr}(u_a \cdot v)} \tau^W_{[L(\cdot)=a]} \ .$$

2. For all a in the range of L,

$$\tau^W_{[L(\cdot)=a]} = \mathop{\mathbb{E}}_{v \sim \ker(L)^\perp} \omega^{-\mathrm{tr}(v \cdot u_a)} \tau^W(v) \ .$$

Proof. Let V denote the image of $\mathbb{F}^k$ under L. Let $a \in V$, $v \in \ker(L)^\perp$. For all $u, u' \in L^{-1}(a)$, we have that $u - u' \in \ker(L)$ and thus $\mathrm{tr}(u \cdot v) = \mathrm{tr}(u' \cdot v)$. As a result, using (17), for any $v \in \ker(L)^\perp$ it holds that

$$\tau^W(v) = \sum_{u \in \mathbb{F}^k} \omega^{\mathrm{tr}(u \cdot v)} \tau^W_u = \sum_{a \in V} \sum_{u \in L^{-1}(a)} \omega^{\mathrm{tr}(u \cdot v)} \tau^W_u = \sum_{a \in V} \omega^{\mathrm{tr}(u_a \cdot v)} \tau^W_{[L(\cdot)=a]} = \sum_{a \in \mathbb{F}^k} \omega^{\mathrm{tr}(u_a \cdot v)} \tau^W_{[L(\cdot)=a]} \ ,$$

where in the last equality we used that for $a \notin V$, the projector $\tau^W_{[L(\cdot)=a]}$ vanishes. This shows the first item. For the second item,

$$
\begin{aligned}
\tau^W_{[L(\cdot)=a]} &= \sum_{u \in L^{-1}(a)} \tau^W_u \\
&= \sum_{u \in L^{-1}(a)} \mathop{\mathbb{E}}_{v \sim \mathbb{F}^k} \left(\omega^{-\mathrm{tr}(u \cdot v)} \tau^W(v) \right) \\
&= \sum_{u \in \ker(L)} \mathop{\mathbb{E}}_{v \sim \mathbb{F}^k} \left(\omega^{-\mathrm{tr}((u_a + u) \cdot v)} \tau^W(v) \right) \\
&= \frac{|\ker(L)|}{|\mathbb{F}^k|} \sum_{v \in \mathbb{F}^k} \left(\left(\mathop{\mathbb{E}}_{u \sim \ker(L)} \omega^{-\mathrm{tr}(u \cdot v)} \right) \omega^{-\mathrm{tr}(u_a \cdot v)} \tau^W(v) \right) \\
&= \mathop{\mathbb{E}}_{v \in \ker(L)^\perp} \omega^{-\mathrm{tr}(v \cdot u_a)} \tau^W(v) \ ,
\end{aligned}
$$

where the second equality follows from (18) and the last uses the fact that $|\mathbb{F}^k| = |\ker(L)| \, |\ker(L)^\perp|$, as shown in Lemma 3.5, and Lemma 8.3 applied to the expectation over u. $\qquad\square$

Lemma 8.5 (Commuting X and Z measurements). *For all linear maps $L, R : \mathbb{F}^k \to \mathbb{F}^k$ such that*

$$\ker(R)^\perp \subseteq \ker(L) \ ,$$

the measurements $\{\tau^Z_{[L(\cdot)=b]}\}_{b \in \mathbb{F}^k}$ and $\{\tau^X_{[R(\cdot)=d]}\}_{d \in \mathbb{F}^k}$ commute.

Proof. Let $b, d \in \mathbb{F}_k$. If either b is not in the range of L, or d is not in the range of R, then at least one of $\tau^Z_{[L(\cdot)=b]}$ or $\tau^X_{[R(\cdot)=d]}$ is 0, and thus the operators trivially commute. Otherwise, both b and d are in the range of L and R, respectively. Let $a_0 \in L^{-1}(b)$, $c_0 \in R^{-1}(d)$. By Lemma 8.4,

$$\tau^Z_{[L(\cdot)=b]} = \mathop{\mathbb{E}}_{u \in \ker(L)^\perp} \omega^{\mathrm{tr}(u \cdot a_0)} \tau^Z(u) \ ,$$

83

$$\tau^X_{[R(\cdot)=d]} = \mathop{\mathbb{E}}_{v \in \ker(R)^\perp} \omega^{\mathrm{tr}(v \cdot c_0)} \tau^X(v) \ .$$

For any $v \in \ker(R)^\perp$, by assumption $v \in \ker(L)$, so for $u \in \ker(L)^\perp$ it holds that $u \cdot v = 0$. Thus $\tau^Z(u)$ and $\tau^X(v)$ commute, and it follows that $\tau^Z_{[L(\cdot)=b]}$ and $\tau^X_{[R(\cdot)=d]}$ commute as well. $\qquad\square$

8.3.2 Complexity and completeness of the introspective verifier

The following theorem formulates the complexity and completeness properties of the introspective verifier. Since $\mathcal{V}^{\mathrm{INTRO}}$ is a typed verifier, we use the detyping procedure described in Section 6.3 to obtain an untyped normal form verifier.

Theorem 8.6 (Complexity and completeness of the introspective verifier). *Let $\lambda, \ell \in \mathbb{N}$. Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier such that $\mathcal{S}$ is an ℓ-level sampler. Let $\mathcal{V}^{\mathrm{INTRO}} = (\mathcal{S}^{\mathrm{INTRO}}, \mathcal{D}^{\mathrm{INTRO}})$ be the typed introspective verifier corresponding to $\mathcal{V}$ and parameters (λ, ℓ). Let $\mathrm{detype}(\mathcal{V}^{\mathrm{INTRO}}) = (\mathrm{detype}(\mathcal{S}^{\mathrm{INTRO}}), \mathrm{detype}(\mathcal{D}^{\mathrm{INTRO}}))$ denote the detyped verifier.*

1. *(**Completeness**) Suppose that $\mathcal{V}$ satisfies the assumption stated in (53). Then for all $n \in \mathbb{N}$ and $N = 2^n$, if $\mathcal{V}_N$ has a projective, consistent, and commuting (PCC) strategy with value 1 then $\mathrm{detype}(\mathcal{V}^{\mathrm{INTRO}})_n$ also has a PCC strategy with value 1.*

2. *(**Sampler complexity**) The sampler $\mathcal{S}^{\mathrm{INTRO}}$ is a $(\mathcal{T}^{\mathrm{INTRO}}, G^{\mathrm{INTRO}})$-type, 2-level sampler. Moreover, the time and randomness complexities of $\mathrm{detype}(\mathcal{S}^{\mathrm{INTRO}})$ satisfy that for all $n \in \mathbb{N}$,*

$$\mathrm{TIME}_{\mathrm{detype}(\mathcal{S}^{\mathrm{INTRO}})}(n) = \mathrm{poly}\left(\lambda \log(\lambda N), \ell\right) ,$$

$$\mathrm{RAND}_{\mathrm{detype}(\mathcal{S}^{\mathrm{INTRO}})}(n) = \mathrm{poly}\left(\lambda \log(\lambda N), \ell\right) ,$$

   where $N = 2^n$.

3. *(**Decider complexity**) The time complexity of the decider $\mathcal{D}^{\mathrm{INTRO}}$ satisfies that for all $n \in \mathbb{N}$,*

$$\mathrm{TIME}_{\mathrm{detype}(\mathcal{D}^{\mathrm{INTRO}})}(n) = \mathrm{poly}\left((\lambda N)^\lambda, \ell\right) ,$$

   where $N = 2^n$.

4. *(**Efficient computability**) There is a Turing machine* ComputeIntroVerifier *which takes as input a tuple $(\mathcal{V}, \lambda, \ell)$ with $\lambda, \ell \in \mathbb{N}$ and returns the description of the detyped introspective verifier $\mathrm{detype}(\mathcal{V}^{\mathrm{INTRO}}) = (\mathrm{detype}(\mathcal{S}^{\mathrm{INTRO}}), \mathrm{detype}(\mathcal{D}^{\mathrm{INTRO}}))$ corresponding to $\mathcal{V}$ and parameters (λ, ℓ) in time $\mathrm{poly}(|\mathcal{V}|, \log \lambda, \log \ell)$.*

Proof. We analyze the completeness and complexity properties of the *typed* verifier $\mathcal{V}^{\mathrm{INTRO}}$; the corresponding properties of the *detyped* verifier $\mathrm{detype}(\mathcal{V}^{\mathrm{INTRO}})$ follow from Lemma 6.18, and the fact that the type set $\mathcal{T}^{\mathrm{INTRO}}$ has size $O(\ell)$.

Completeness. We first show completeness. Let $n \geq 1$ be an index for $\mathcal{V}^{\mathrm{INTRO}}$ and $N = 2^n$ be the corresponding index for $\mathcal{V}$. The assumption on the time complexity of $\mathcal{V}$ ensures that $\mathcal{D}^{\mathrm{INTRO}}$ never aborts due to a timeout. Let $L^v = L^{v,N}$ denote the CL function of the original sampler $\mathcal{S}$ on index N corresponding to player $v \in \{A, B\}$. Let $r = (\lambda N)^\lambda$, and let $\mathrm{introparams}(r) = (q, m, d, h, H, \Gamma, \pi)$, as in Section 8.2. Set $Q = \Gamma \log q$; this represents the number of qubits that are certified by the Pauli basis test parameterized by $\mathrm{introparams}(r)$. Let $\mathscr{S} = (|\mathrm{AUX}\rangle, A, B)$ be a PCC strategy for $\mathcal{V}_N$ with value 1. We first construct a PCC strategy $\mathscr{S}_n^{\mathrm{INTRO}}$ for the typed verifier $\mathcal{V}_n^{\mathrm{INTRO}}$ with value 1. We then conclude using Lemma 6.18.

	V_1^v	V_2^v	V_3^v	AUX
(INTRO, v)	$\sigma_{L_1}^Z$	$\sigma_{L_2}^Z$	$\sigma_{L_3}^Z$	A^x/B^x
(SAMPLE, v)	σ^Z	σ^Z	σ^Z	A^x/B^x
(READ, v)	$\sigma_{L_1}^Z\ \sigma_{L_1^\perp}^X$	$\sigma_{L_2}^Z\ \sigma_{L_2^\perp}^X$	$\sigma_{L_3}^Z\ \sigma_{L_3^\perp}^X$	A^x/B^x
(HIDE$_3$, v)	$\sigma_{L_1}^Z\ \sigma_{L_1^\perp}^X$	$\sigma_{L_2}^Z\ \sigma_{L_2^\perp}^X$	$\sigma_{L_3^\perp}^X$	I
(HIDE$_2$, v)	$\sigma_{L_1}^Z\ \sigma_{L_1^\perp}^X$	$\sigma_{L_2^\perp}^X$	σ^X	I
(HIDE$_1$, v)	$\sigma_{L_1^\perp}^X$	σ^X	σ^X	I

The left-most column denotes the introspection/hiding questions that a player may receive. The top row denotes the registers corresponding to the factor spaces of a CL function L^v (we note that the partition of the registers depends on the prefixes), as well as the register corresponding to the state $|\text{AUX}\rangle$ coming from the original PCC strategy $\mathscr{S}$. We use $\sigma_{L_j}^Z$ as shorthand for $\sigma_{[L_{j,x_{<j}}^v(\cdot)=x_j]}^Z$ and similarly $\sigma_{L_j^\perp}^X$ for $\sigma_{[(L_{j,x_{<j}}^v)^\perp(\cdot)=x_j^\perp]}^X$. A symbol I means that the register is left unmeasured.

Figure 11: Summary of the honest strategy $\mathscr{S}_n^{\text{INTRO}}$ for $\mathcal{V}_n^{\text{INTRO}}$ for a 3-level sampler.

Remark 8.7. *Note that by definition in $\mathcal{V}_n^{\text{INTRO}}$ the players receive questions (x, y) that are sampled according to the distribution $\mu_{\mathcal{S}^{\text{INTRO}}, n}^{G^{\text{INTRO}}}$ associated with the downsized typed sampler $\kappa(\tilde{\mathcal{S}}^{\text{INTRO}})$. Using the definition of the downsized typed sampler, Definition 6.6, and Lemma 4.10 the distribution is identical to the distribution $\mu_{\tilde{\mathcal{S}}^{\text{INTRO}}, n}^{G^{\text{INTRO}}}$, up to the bijective mapping κ. This mapping can be computed by the players themselves. Therefore, we construct a strategy for players that receive questions from $\mu_{\tilde{\mathcal{S}}^{\text{INTRO}}, n}^{G^{\text{INTRO}}}$, and a strategy for questions from $\mu_{\mathcal{S}^{\text{INTRO}}, n}^{G^{\text{INTRO}}}$ follows immediately.*

The strategy $\mathscr{S}_n^{\text{INTRO}}$ uses the state $|\text{EPR}_2\rangle^{\otimes(Q+1)} \otimes |\text{AUX}\rangle$, where recall that $|\text{EPR}_2\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. For all register subspaces $R \subseteq \mathbb{F}_2^Q$ (see Definition 4.1 for the definition of register subspace), whenever we refer to "register R", we mean the qubits of $|\text{EPR}_2\rangle^{\otimes Q}$ corresponding to R (see Section 3.5). The most frequent register subspace we consider is V, spanned by $e_1, \ldots, e_{s(N)}$. We write $\overline{V}$ for the complement of V, i.e. the register subspace spanned by $e_{s(N)+1}, \ldots, e_Q$. (Note that $s(N) \leq r \leq Q$ by assumption (53) and the definition of Q.) Then $|\text{EPR}_2\rangle^{\otimes Q} = |\text{EPR}\rangle_V \otimes |\text{EPR}\rangle_{\overline{V}}$.

Let $\mathscr{S}_n^{\text{BP}}$ be the honest binary Pauli strategy with respect to introparams defined in the proof of Lemma 7.12. For a question of type in $\mathcal{T}^{\text{PAULI}}$ the player measures the shared state $|\text{EPR}_2\rangle^{\otimes(Q+1)}$ using the measurements specified by $\mathscr{S}_n^{\text{BP}}$, and reports the measurement outcomes. When a player receives questions of type $\hat{\mathfrak{t}} \in \mathcal{T}_n^{\text{INTRO}} \setminus \mathcal{T}^{\text{PAULI}}$, they perform measurements described as follows. (Below, whenever we write a Pauli operator σ_a^W the register on which the operator acts should always be clear from context, and is implicit from the space in which the outcome a ranges.) The reader may find it helpful to consult Figure 11 to see a summary of the honest strategy $\mathscr{S}_n^{\text{INTRO}}$ for the special case when $\ell = 3$.

($\textsc{Intro}, v$): The player performs the measurement

$$\left\{ \sigma^Z_{[L^v(\cdot)=y]} \right\} \tag{55}$$

to obtain an $y \in V$. Intuitively, the player has now introspectively sampled the question y for original player v in game $\mathcal{V}_N$. The player then measures $|\textsc{aux}\rangle$ using player A's measurement $\{A_a^y\}$ from $\mathscr{S}$ if $v = \text{A}$ and using player B's measurement $\{B_a^y\}$ if $v = \text{B}$ to obtain an answer a. The player replies with (y, a).

($\textsc{Sample}, v$): The player measures their share of $|\text{EPR}\rangle_V$ in the Z basis to obtain $z \in V$. Using this, they compute the question $y = L^v(z)$. The player then uses player v's strategy and question y to measure $|\textsc{aux}\rangle$ and obtain outcome a. The player replies with (z, a).

($\textsc{Read}, v$): The player first performs all measurements as in the ($\textsc{Intro}, v$) question for player v and records the outcomes as $y \in L(V)$ and $a \in \{0, 1\}^*$. For $j \in \{1, 2, \ldots, \ell\}$, the player measures the register $V_j^v(y)$ with the measurement

$$\left\{ \sigma^X_{[L_j^\perp(\cdot) = y_j^\perp]} \right\}_{y_j^\perp} \tag{56}$$

to obtain $y^\perp = y_1^\perp + \cdots + y_\ell^\perp$. Here $L_j^\perp$ is shorthand for the function $(L_{j, y_{<j}}^v)^\perp$ defined in Item 6 of the decider description in Section 8.2. (That this is simultaneously measurable with the measurement in (55) follows from Lemma 8.5 and the fact that $\ker(L_j^\perp)^\perp = \ker(L_j)$ by Lemma 3.5 and the definition of $L_j^\perp$ in Section 8.2.) The player measures $|\textsc{aux}\rangle$ with player v's measurement strategy in $\mathscr{S}$ for question y to obtain a and replies with $(y, y^\perp, a)$.

($\textsc{Hide}_k, v$): The player performs the following sequence of measurements: first measure $\{\sigma^Z_{[L_1^v(\cdot) = y_1]}\}$ on register V_1^v to obtain y_1. Then, use y_1 to specify the second linear function $L_{2, y_1}^v(\cdot)$ and measure register V_{2, y_1}^v using $\{\sigma^Z_{[L_{2, y_1}^v(\cdot) = y_2]}\}$ to obtain y_2. This process continues until the $(k - 1)$-th linear map $L_{k-1, y_{<k-1}}^v(\cdot)$ has been measured to obtain y_{k-1} in factor space $V_{k-1, y_{<k-1}}^v$. Let $y = y_1 + y_2 + \cdots + y_{k-1}$. Next, for $j \in \{1, 2, \ldots, k\}$ the player measures

$$\left\{ \sigma^X_{[L_j^\perp(\cdot) = y_j^\perp]} \right\}_{y_j^\perp},$$

where $L_j^\perp$ denotes the linear map $(L_{j, y_{<j}}^v)^\perp$ as in the case ($\textsc{Read}, v$). Let $y^\perp = y_1^\perp + y_2^\perp + \cdots + y_k^\perp$, where each $y_j^\perp$ is a vector in the factor space $V_{j, y_{<j}}^v$. Finally, the player measures register $V_{>k}^v(y)$ using $\{\sigma^X_{x_{>k}}\}$ to obtain outcome $x_{>k}$. Let $x = x_{>k}$. The player replies with $(y, y^\perp, x)$.

By definition, when player w performs the honest measurement for question ($\textsc{Intro}, \text{A}$) and player $\overline{w}$ performs the honest measurement for question ($\textsc{Intro}, \text{B}$), the joint outcome (y, y') has distribution $\mu_{\mathcal{S}, N}$. In this case, the players play according to strategy $\mathscr{S}$ and succeed with probability 1. In all other cases, it is straightforward to verify that the players succeed in all tests performed by $\mathcal{D}^{\textsc{Intro}}$ (Figure 10) with probability 1. As a result, the value of this strategy is 1.

The strategy $\mathscr{S}^{\textsc{Intro}}$ is projective by construction. It is also consistent because of the assumed consistency of the strategy $\mathscr{S}$ as well as consistency of the honest Pauli strategy $\mathscr{S}^{\text{BP}}$. Furthermore, note that $\mathscr{S}^{\textsc{Intro}}$ only calls $\mathscr{S}$ for both players on question pairs such that both types are in $\{\textsc{Intro}, \textsc{Sample}, \textsc{Read}\}$. In all these cases, $\mathscr{S}$ is called on a pair of questions (y, y') distributed as $(L^v(z), L^{v'}(z))$ for $v, v' \in \{\text{A}, \text{B}\}$

and z uniform in V. When $v \neq v'$, any such pair by definition has positive probability under $\mu_{\mathcal{S},n}$, and so by assumption the associated measurements from $\mathscr{S}$ commute. On the other hand, when $v = v'$, then the players apply the same measurements from $\mathscr{S}$, and because $\mathscr{S}$ only uses projective measurements, their measurements commute as well. Examining all other cases, it follows by direct inspection that the strategy commutes on all question pairs whose corresponding types appear as an edge in the graph G^{INTRO}. Thus the strategy commutes with respect to the support of the distribution $\mu_{\mathcal{S}^{\text{INTRO}},n}$.

Complexity.   The complexity parameters of the typed sampler $\mathcal{S}^{\text{INTRO}}$ and typed decider $\mathcal{D}^{\text{INTRO}}$ follow from Lemmas 8.1 and 8.2. The complexity parameters of the detyped sampler and decider $\text{detype}(\mathcal{S}^{\text{INTRO}})$ and $\text{detype}(\mathcal{D}^{\text{INTRO}})$ then follow from Lemma 6.18.

Efficient computability.   The Turing machine ComputeIntroVerifier does the following on input $(\mathcal{V}, \ell)$: it first computes

$$\mathcal{S}^{\text{INTRO}} = \text{ComputeIntroSampler}(\lambda, \ell)\,,$$
$$\mathcal{D}^{\text{INTRO}} = \text{ComputeIntroDecider}(\mathcal{V}, \lambda, \ell)\,,$$

using Lemmas 8.1 and 8.2, runs the detyping procedure from Definition 6.17, and then outputs the resulting detyped verifier. This takes time $\text{poly}(|\mathcal{V}|, \log \lambda, \log \ell)$. $\square$

Remark 8.8. *For future reference, we note that on any input $(\mathcal{V}, \lambda, \ell)$, the Turing machine* ComputeIntroVerifier *always returns a normal form verifier $\mathcal{V}^{\text{INTRO}} = (\mathcal{S}^{\text{INTRO}}, \mathcal{D}^{\text{INTRO}})$. This is because for any two integer λ, ℓ, by construction* ComputeIntroSampler(λ, ℓ) *returns a sampler with field size 2, and for any $\mathcal{V}, \lambda$ and ℓ the decider $\mathcal{D}^{\text{INTRO}}$ specified in Figure 10 takes 7 inputs and always halts with a single-bit output, even if $\mathcal{S}$ or $\mathcal{D}$ themselves do not halt.*

8.4   Soundness of the introspective verifier

The main result of this section is the following theorem which establishes the soundness property of the introspective verifier.

Theorem 8.9 (Soundness of the introspective verifier)**.** *Let $\lambda, \ell \in \mathbb{N}$. Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier such that $\mathcal{S}$ is an ℓ-level sampler. Let $\mathcal{V}^{\text{INTRO}} = (\mathcal{S}^{\text{INTRO}}, \mathcal{D}^{\text{INTRO}})$ be the introspective verifier corresponding to $\mathcal{V}$ and parameters (λ, ℓ), as defined in Section 8.2. Let $\text{detype}(\mathcal{V}^{\text{INTRO}})$ denote the associated detyped verifier. There exists a function $\delta(\varepsilon, n) = \text{poly}(\varepsilon + 1/(\lambda 2^n)^\lambda)$ (where the implicit polynomial may depend arbitrarily on ℓ) such that for all $n \geq 1$, $N = 2^n$, and $\varepsilon \geq 0$ the following hold.*

1. *If $\text{val}^*(\text{detype}(\mathcal{V}^{\text{INTRO}})_n) > 1 - \varepsilon$, then $\text{val}^*(\mathcal{V}_N) \geq 1 - \delta(\varepsilon, n)$.*

2. *Let $\mathscr{E}(\cdot)$ be as defined in Definition 5.12. Then*

$$\mathscr{E}(\text{detype}(\mathcal{V}^{\text{INTRO}})_n, 1 - \varepsilon) \geq \max\left\{\mathscr{E}(\mathcal{V}_N, 1 - \delta(\varepsilon, n)), (1 - \delta(\varepsilon, n))\, 2^{(\lambda N)^\lambda}\right\}\,.$$

8.4.1   The Pauli twirl

A key tool in the proof of Theorem 8.9 is the *Pauli twirl*. In this section we introduce the Pauli twirl and establish several of its properties. The section closely follows Sections 8 and 10 of [NW19].

To begin, we define the twirl with respect to an arbitrary distribution over unitaries.

Definition 8.10 (Twirl). Let μ be a probability distribution over a finite set of unitary matrices. Then for any matrix A, the *twirl of A with respect to μ*, denoted $\mathcal{T}_\mu(A)$, is defined as

$$\mathcal{T}_\mu(A) = \mathop{\mathbb{E}}_{U \sim \mu}\left(UAU^\dagger\right).$$

In the next two lemmas we consider the Pauli twirl, in which the distribution μ is over subsets of Pauli observables. First, we show how the Pauli twirl acts on Pauli matrices. Then, using this, we derive an expression for the Pauli twirl applied to general matrices.

Lemma 8.11 (Pauli twirl of Pauli matrices). *Let V be a subspace of $\mathbb{F}^k$, let $W \in \{X, Z\}$, and let μ be the uniform distribution over $\{\tau^W(w) : w \in V\}$. Let $W' \neq W$ and $u \in \mathbb{F}^k$. Then*

$$\mathcal{T}_\mu(\tau^{W'}(u)) = \begin{cases} \tau^{W'}(u) & \text{if } u \in V^\perp, \\ 0 & \text{if } u \notin V^\perp. \end{cases}$$

Proof. Let $u \in \mathbb{F}^k$. Let $c = 1$ if $W = Z$ and let $c = -1$ if $W = X$. Then

$$\begin{aligned}
\mathcal{T}_\mu(\tau^{W'}(u)) &= \mathop{\mathbb{E}}_{z \sim V}(\tau^W(z)\tau^{W'}(u)\tau^W(z)^\dagger) \\
&= \mathop{\mathbb{E}}_{z \sim V}(\omega^{c \cdot \text{tr}(u \cdot z)}\tau^{W'}(u)\tau^W(z)\tau^W(z)^\dagger) \\
&= \left(\mathop{\mathbb{E}}_{z \sim V}\omega^{c \cdot \text{tr}(u \cdot z)}\right)\tau^{W'}(u).
\end{aligned}$$

The lemma now follows from Lemma 8.3 and the fact that $c \cdot u \in V^\perp$ if and only if $u \in V^\perp$. $\qquad\square$

Lemma 8.12 (Pauli twirl of general matrices). *Let $V = \mathbb{F}^k$, and let $L : V \to V$ be a linear map. Let ζ be the uniform distribution over $\{\tau^Z(z) \mid z \in V\}$ and χ the uniform distribution over $\{\tau^X(x) \mid x \in \ker(L)\}$. Let M be a matrix acting on $\mathbb{C}^V \otimes \mathcal{H}_A$, where $\mathcal{H}_A$ is a finite dimensional Hilbert space. Then there exist matrices $\{M^y\}_{y \in L(V)}$ acting on $\mathcal{H}_A$ such that the twirl of M with respect to ζ and χ is given by*

$$(\mathcal{T}_\chi \circ \mathcal{T}_\zeta \otimes I_A)(M) = \sum_{y \in V} \tau^Z_{[L(\cdot)=y]} \otimes M^y. \tag{57}$$

Moreover, if we apply (57) to each element of a POVM measurement $\{M_a\}$, then for each $y \in V$, the set $\{M_a^y\}$ also forms a POVM measurement.

Proof. The collection $\{\tau^X(x)\tau^Z(z)\}_{x,z \in V}$ forms a basis for the complex linear space of matrices acting on $\mathbb{C}^V$. As a result, we can write

$$M = \sum_{x,z \in V} \tau^X(x)\tau^Z(z) \otimes M_{x,z},$$

for matrices $M_{x,z}$ on $\mathcal{H}_A$. We now use Lemma 8.11 to compute the twirl first with respect to ζ and then with respect to ζ and χ:

$$\begin{aligned}
(\mathcal{T}_\zeta \otimes I_A)(M) &= \sum_{x,z \in V} \mathcal{T}_\zeta(\tau^X(x))\tau^Z(z) \otimes M_{x,z} = \sum_{z \in V} \tau^Z(z) \otimes M_{0,z}, \\
(\mathcal{T}_\chi \circ \mathcal{T}_\zeta \otimes I_A)(M) &= \sum_{z \in V} \mathcal{T}_\chi(\tau^Z(z)) \otimes M_{0,z} = \sum_{z \in \ker(L)^\perp} \tau^Z(z) \otimes M_{0,z}.
\end{aligned}$$

88

For all $y \in V$, let u_y denote an arbitrary element of $L^{-1}(y)$ if y is in the image of L; otherwise, set $u_y = 0$. Expanding $\tau^Z(z)$ using the first part of Lemma 8.4,

$$\sum_{z \in \ker(L)^\perp} \tau^Z(z) \otimes M_{0,z} = \sum_{z \in \ker(L)^\perp} \sum_y \omega^{\mathrm{tr}(u_y \cdot z)} \tau^Z_{[L(\cdot)=y]} \otimes M_{0,z}$$

$$= \sum_y \tau^Z_{[L(\cdot)=y]} \otimes \left(\sum_{z \in \ker(L)^\perp} \omega^{\mathrm{tr}(u_y \cdot z)} M_{0,z} \right). \tag{58}$$

Equation (57) follows by setting $M^y = \sum_{z \in \ker(L)^\perp} \omega^{\mathrm{tr}(u_y \cdot z)} M_{0,z}$.

For the "moreover" part, note first that whenever $M \geq 0$ it holds that any twirl satisfies $0 \leq \mathscr{T}_\mu(M)$. As a result, each matrix M^y must be positive semi-definite due to Equation (58) and the fact that the $\{\tau^Z_{[L(\cdot)=y]}\}_y$ matrices are orthogonal projections. Next, suppose $\{M_a\}$ is a POVM measurement, and write $N = \sum_a M_a$ for the identity matrix. Then by linearity, for each $y \in V$, $\sum_a M_a^y = N^y$. In addition, $N_{0,0} = I$, and $N_{x,z} = 0$ otherwise. As a result, $N^y = N_{0,0} = I$, and so $\{M_a^y\}$ also forms a POVM. $\square$

In the next few lemmas we derive a sufficient condition for a measurement to be close to its own Pauli twirl, namely that it satisfies certain commutation relations with the Pauli basis measurements.

Lemma 8.13 (Commuting with Pauli basis implies commuting with Pauli observables). *Let $\mathcal{X}, \mathcal{A}$ be finite sets and D be a distribution over $\mathcal{X}$. For each $x \in \mathcal{X}$, let V_x be a register subspace of $V = \mathbb{F}^k$, and let $L_x : V_x \to V_x$ be a linear map.*

Consider a state $|\psi\rangle = |\mathrm{EPR}\rangle_V \otimes |\mathrm{AUX}\rangle$, where $|\mathrm{EPR}\rangle_V \in \mathcal{H}_A \otimes \mathcal{H}_B$, for $\mathcal{H}_A, \mathcal{H}_B \cong \mathbb{C}^V$, is defined in Definition 3.27 and $|\mathrm{AUX}\rangle \in \mathcal{H}_{A'} \otimes \mathcal{H}_{B'}$ is arbitrary. For each $x \in \mathcal{X}$, let $\{M_a^x\}_{a \in \mathcal{A}}$ be a measurement on $\mathcal{H}_A \otimes \mathcal{H}_{A'}$. Let $W \in \{X, Z\}$. Then the following are equivalent on state $|\psi\rangle$:

- *On average over $x \sim D$,*

$$\left[M_a^x, \left(\tau^W_{[L_x(\cdot)=y]} \otimes I_{\overline{V_x}} \otimes I_{A'} \right) \right] \otimes I_B \approx_\varepsilon 0.$$

- *On average over $x \sim D$ and v drawn uniformly from $\ker(L_x)^\perp$,*

$$\left[M_a^x, \left(\tau^W(v) \otimes I_{\overline{V_x}} \otimes I_{A'} \right) \right] \otimes I_B \approx_\varepsilon 0.$$

Proof. For $x \in \mathcal{X}$, $a \in \mathcal{A}$, y in the range of L_x, and $v \in \mathbb{F}^k$ define

$$\Delta^x_{a,y} = \left[M_a^x, \left(\tau^W_{[L_x(\cdot)=y]} \otimes I_{\overline{V_x}} \otimes I_{A'} \right) \right] \otimes I_B,$$

$$\Delta^x_a(v) = \left[M_a^x, \left(\tau^W(v) \otimes I_{\overline{V_x}} \otimes I_{A'} \right) \right] \otimes I_B.$$

By the second item of Lemma 8.4, for each $v \in \ker(L_x)^\perp$,

$$\Delta^x_a(v) = \sum_{y \in V_x} \omega^{\mathrm{tr}(u_y \cdot v)} \Delta^x_{a,y},$$

where for every y in the range of L_x, u_y is a fixed element in $L_x^{-1}(y)$. The expression for the closeness of Δ^x_a to 0 on average over $x \sim D$ and $v \sim \ker(L_x)^\perp$ (i.e. the second quantity of the Lemma statement), is equal to

$$\mathop{\mathbb{E}}_{x \sim D} \mathop{\mathbb{E}}_{v \sim \ker(L_x)^\perp} \sum_a \left\| \Delta^x_a(v) |\psi\rangle \right\|^2,$$

89

and can be expanded as

$$\mathbb{E}_{x \sim D} \mathbb{E}_{v \sim \ker(L_x)^\perp} \sum_a \langle \psi | \sum_{y,y'} \omega^{\mathrm{tr}((u'_y - u_y) \cdot v)} (\Delta^x_{a,y})^\dagger \Delta^x_{a,y'} | \psi \rangle \ . \tag{59}$$

If $y \neq y'$, then by definition $L_x(u_y) \neq L_x(u_{y'})$, and so $u'_y - u_y$ is not in $\ker(L_x)$. Lemma 8.3 (with $V = \ker(L_x)^\perp$) thus implies that (59) equals

$$\mathbb{E}_{x \sim D} \sum_a \langle \psi | \sum_y ((\Delta^x_{a,y})^\dagger \Delta^x_{a,y}) | \psi \rangle = \mathbb{E}_{x \sim D} \sum_{a,y} \| \Delta^x_{a,y} | \psi \rangle \|^2$$

which is the closeness of $\Delta^x_{a,y}$ to 0 on average over $x \sim D$. Thus, $\Delta^x_a(v) \approx_\varepsilon 0$ on average over x and v if and only if $\Delta^x_{a,y} \approx_\varepsilon 0$ on average over x. $\qquad\square$

Lemma 8.14 (Commuting implies twirl). *Let $\mathcal{X}$ be a finite set and D a distribution on $\mathcal{X}$. For each $x \in \mathcal{X}$, let $\{M^x_a\}$ be a POVM on $\mathcal{H}_A$, and let μ_x be a distribution over unitary matrices acting on $\mathcal{H}_A$. Suppose that on average over $x \sim D$ and $U \sim \mu_x$ it holds that*

$$[M^x_a, U^\dagger] \otimes I_B \approx_\varepsilon 0 \ ,$$

where the commutator is evaluated on a state $|\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$. Then on average over $x \sim D$,

$$M^x_a \otimes I_B \approx_\varepsilon \mathcal{T}_{\mu_x}(M^x_a) \otimes I_B \ .$$

Proof. Observe that

$$\mathbb{E}_{x \sim D} \sum_a \| (\mathcal{T}_{\mu_x}(M^x_a) - M^x_a) \otimes I_B | \psi \rangle \|^2 = \mathbb{E}_{x \sim D} \sum_a \| \mathbb{E}_{U \sim \mu_x} (U[M^x_a, U^\dagger]) \otimes I_B | \psi \rangle \|^2. \tag{60}$$

Applying Jensen's inequality, the right-hand side of (60) is at most

$$\mathbb{E}_{x \sim D} \mathbb{E}_{U \sim \mu_x} \sum_a \| (U[M^x_a, U^\dagger]) \otimes I_B | \psi \rangle \|^2 = \mathbb{E}_{x \sim D} \mathbb{E}_{U \sim \mu_x} \sum_a \| [M^x_a, U^\dagger] \otimes I_B | \psi \rangle \|^2,$$

using the unitary invariance of the Euclidean norm. This last quantity is $O(\varepsilon)$, by assumption. $\qquad\square$

Lemma 8.15 (Commuting with each implies commuting with both). *Let $\mathcal{X}$ be a finite set and D a distribution on $\mathcal{X}$. For each $x \in \mathcal{X}$, let $\{M^x_a\}$ be a POVM on $\mathcal{H}_A$ and let $\mu_{x,1}, \mu_{x,2}$ be two distributions over unitary matrices acting on $\mathcal{H}_A$. Suppose that for each $i \in \{1,2\}$, on average over $x \sim D$ and $U_i \sim \mu_{x,i}$,*

$$[M^x_a, U^\dagger_i] \otimes I_B \approx_\varepsilon 0, \tag{61}$$

where the expression is evaluated on some state $|\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$, where $\mathcal{H}_B \cong \mathcal{H}_A$. Suppose further that on average over $x \sim D$ and $U_2 \sim \mu_{x,2}$,

$$U^\dagger_2 \otimes I_B \approx_\varepsilon I_A \otimes U_2 \ . \tag{62}$$

(The corresponding statement for $i = 1$ is not needed.) Then on average over $x \sim D$, $U_1 \sim \mu_{x,1}$, and $U_2 \sim \mu_{x,2}$,

$$[M^x_a, U^\dagger_1 U^\dagger_2] \otimes I_B \approx_\varepsilon 0 \ .$$

Proof. The claim follows from the following sequence of approximations:

$$M_a^x U_1^\dagger U_2^\dagger \otimes I_B \approx_\varepsilon M_a^x U_1^\dagger \otimes U_2 \qquad\qquad \text{(by (62))}$$
$$\approx_\varepsilon U_1^\dagger M_a^x \otimes U_2 \qquad\qquad \text{(by (61) for } i = 1\text{)}$$
$$\approx_\varepsilon U_1^\dagger M_a^x U_2^\dagger \otimes I_B \qquad\qquad \text{(by (62))}$$
$$\approx_\varepsilon U_1^\dagger U_2^\dagger M_a^x \otimes I_B \;, \qquad\qquad \text{(by (61) for } i = 2\text{)}$$

where each step also uses Fact 5.18. This is equivalent to $\left[M_a^x, U_1^\dagger U_2^\dagger \right] \otimes I_B \approx_\varepsilon 0$, completing the proof. $\qquad\square$

The following is a slight generalization of [NW19, Fact 4.25], and we give a similar proof.

Lemma 8.16 (Close to sub-measurement implies close to measurement)**.** *Let $\mathcal{X}, \mathcal{A}$ be finite sets and D be a distribution on $\mathcal{X}$. Suppose that for each $x \in \mathcal{X}$, $\{A_a^x\}_{a \in \mathcal{A}}$ is a projective measurement and $\{B_a^x\}_{a \in \mathcal{A}}$ is a set of matrices such that each B_a^x is positive semidefinite and $\sum_a B_a^x \leq I$. Suppose $\{C_a^x\}_{a \in \mathcal{A}}$ is a POVM such that $C_a^x \geq B_a^x$ for all x and a. Then if, on average over $x \sim D$, $A_a^x \approx_\varepsilon B_a^x$, then, on average over $x \sim D$, $A_a^x \approx_{\varepsilon^{1/2}} C_a^x$.*

Proof. By the triangle inequality,

$$\mathbb{E}_x \sum_a \|(A_a^x - C_a^x)|\psi\rangle\|^2 \leq 2 \mathbb{E}_x \sum_a \|(A_a^x - B_a^x)|\psi\rangle\|^2 + 2 \mathbb{E}_x \sum_a \|(C_a^x - B_a^x)|\psi\rangle\|^2.$$

The first term on the right-hand side is $O(\varepsilon)$ by assumption. For the second,

$$\mathbb{E}_x \sum_a \|(C_a^x - B_a^x)|\psi\rangle\|^2 = \mathbb{E}_x \sum_a \langle\psi|(C_a^x - B_a^x)^2|\psi\rangle \leq \mathbb{E}_x \sum_a \langle\psi|(C_a^x - B_a^x)|\psi\rangle$$
$$= 1 - \mathbb{E}_x \sum_a \langle\psi|B_a^x|\psi\rangle \leq 1 - \mathbb{E}_x \sum_a \langle\psi|(B_a^x)^2|\psi\rangle \;,$$

where the middle inequality uses $0 \leq C_a^x - B_a^x \leq I$ for all x, a. Write $1 = \mathbb{E}_x \sum_a \langle\psi|(A_a^x)^2|\psi\rangle$, which holds because A is a projective measurement. Then

$$\mathbb{E}_x \sum_a \langle\psi|((A_a^x)^2 - (B_a^x)^2)|\psi\rangle = \Re\left(\mathbb{E}_x \sum_a \langle\psi|(A_a^x + B_a^x)(A_a^x - B_a^x)|\psi\rangle \right)$$
$$\leq \mathbb{E}_x \sqrt{\sum_a \|(A_a^x + B_a^x)|\psi\rangle\|^2} \cdot \sqrt{\sum_a \|(A_a^x - B_a^x)|\psi\rangle\|^2}$$

where the first equality follows from the fact that A_a^x and B_a^x are Hermitian. For each $x \in X$ the first square root is $O(1)$. This allows us to move the expectation into the second square root by Jensen's inequality. The result is $O(\varepsilon^{1/2})$ by assumption. $\qquad\square$

Now we put everything together to show the main result of this section.

Lemma 8.17. *Let $\mathcal{X}, \mathcal{A}$ be a finite sets and D be a distribution over $\mathcal{X}$. For each $x \in \mathcal{X}$, let V_x be a register subspace of $V = \mathbb{F}^k$, let U_x be a register subspace of V_x, and let $L_x : U_x \to U_x$ be a linear map.*

Consider a state $|\psi\rangle = |\text{EPR}\rangle_V \otimes |\text{AUX}\rangle$, where $|\text{EPR}\rangle_V \in \mathcal{H}_A \otimes \mathcal{H}_B$, for $\mathcal{H}_A, \mathcal{H}_B \cong \mathbb{C}^V$, is defined in Definition 3.27 and $|\text{AUX}\rangle \in \mathcal{H}_{A'} \otimes \mathcal{H}_{B'}$ is arbitrary. For each $x \in \mathcal{X}$, let $\{M_{y,a}^x\}_{y \in U_x, a \in \mathcal{A}}$ be a

projective measurement on $\mathcal{H}_{V_x} \otimes \mathcal{H}_{A'}$. Suppose that on average over $x \sim D$ the following conditions hold.

$$\text{(Consistency):} \quad \left(M_y^x \otimes I_{\overline{V_x}} - \tau^Z_{[L_x(\cdot)=y]} \otimes I_{\overline{U_x}} \otimes I_{A'} \right) \otimes I_B \approx_\varepsilon 0 \,,$$

$$\text{(Commutation):} \quad \left[M_{y,a}^x \otimes I_{\overline{V_x}}, \tau_z^Z \otimes I_{\overline{U_x}} \otimes I_{A'} \right] \otimes I_B \approx_\varepsilon 0 \,,$$

$$\left[M_{y,a}^x \otimes I_{\overline{V_x}}, \tau^X_{[L_x^\perp(\cdot)=y^\perp]} \otimes I_{\overline{U_x}} \otimes I_{A'} \right] \otimes I_B \approx_\varepsilon 0 \,.$$

Here, the projector $\tau^Z_{[L_x(\cdot)=y]}$ acts on the register subspace $\mathcal{H}_{U_x}$, and $\overline{U_x}$ and $\overline{V_x}$ denote the complementary register subspace of U_x and V_x, respectively, within V. Then for each $x \in \mathcal{X}$ and $y \in U_x$, there exists a POVM measurement $\{M_a^{x,y}\}_{a \in \mathcal{A}}$ on $\mathcal{H}_{V_x \setminus U_x} \otimes \mathcal{H}_{A'}$ such that on average over $x \sim D$,

$$(M_{y,a}^x \otimes I_{\overline{V_x}}) \otimes I_B \approx_{\varepsilon^{1/2}} \left(\tau^Z_{[L_x(\cdot)=y]} \otimes M_a^{x,y} \otimes I_{\overline{V_x}} \right) \otimes I_B \,.$$

Proof. For each $x \in X$, let ζ_x be the uniform distribution over U_x and χ_x be the uniform distribution over $\ker(L_x)$. We apply Lemma 8.13 to each of the two commutation assumptions and use the fact that $\ker(L_x^\perp)^\perp = \ker(L_x)$ from Lemma 3.12. Lemma 8.13 implies that on average over $x \sim \mathcal{X}$ and $v \sim \zeta_x$ if $W = Z$ or $v \sim \chi_x$ if $W = X$,

$$\left[M_{y,a}^x \otimes I_{\overline{V_x}}, \tau^W(v) \otimes I_{\overline{U_x}} \otimes I_{A'} \right] \otimes I_B \approx_\varepsilon 0 \,.$$

By Lemma 8.15, this implies that on average over $x \sim D$, $u \sim \zeta_x$, and $v \sim \chi_x$,

$$\left[M_{y,a}^x \otimes I_{\overline{V_x}}, \tau^Z(u)\tau^X(v) \otimes I_{\overline{U_x}} \otimes I_{A'} \right] \otimes I_B \approx_\varepsilon 0 \,.$$

By Lemma 8.14 and Lemma 8.12, this implies that on average over $x \sim D$,

$$M_{y,a}^x \otimes I_{\overline{V_x}} \otimes I_B \approx_\varepsilon \left(\mathscr{T}_{\chi_x} \circ \mathscr{T}_{\zeta_x}(M_{y,a}^x) \right) \otimes I_{\overline{V_x}} \otimes I_B$$

$$= \left(\sum_{y' \in V_x} \tau^Z_{[L_x(\cdot)=y']} \otimes M_{y,a}^{x,y'} \right) \otimes I_{\overline{V_x}} \otimes I_B \,, \tag{63}$$

for some POVM measurement $\{M_{y,a}^{x,y'}\}$ on $\mathcal{H}_{V_x \setminus U_x} \otimes \mathcal{H}_{A'}$.

In the following sequence of equations, whenever an operator does not act on a subsystem it should be assumed that it is appropriately tensored with the identity. For clarity, we explicitly indicate using a subscript A or B whether a Pauli operator acts on $\mathcal{H}_A$ or $\mathcal{H}_B$. Then on average over $x \sim D$ we have

$$\begin{aligned}
M_{y,a}^x &= M_{y,a}^x \cdot M_y^x && (M^x \text{ is projective}) \\
&\approx_\varepsilon M_{y,a}^x \cdot \left(\tau^Z_{[L_x(\cdot)=y]} \right)_A && (\text{Consistency assumption}) \\
&\approx_0 M_{y,a}^x \otimes \left(\tau^Z_{[L_x(\cdot)=y]} \right)_B && (\text{Paulis are self-consistent}) \\
&\approx_\varepsilon \left(\sum_{y'} \left(\tau^Z_{[L_x(\cdot)=y']} \right)_A \otimes M_{y,a}^{x,y'} \right) \otimes \left(\tau^Z_{[L_x(\cdot)=y]} \right)_B && (\text{Equation (63)}) \\
&\approx_0 \sum_{y'} \left(\tau^Z_{[L_x(\cdot)=y']} \tau^Z_{[L_x(\cdot)=y]} \right)_A \otimes M_{y,a}^{x,y'} && (\text{Paulis are self-consistent}) \\
&= \left(\tau^Z_{[L_x(\cdot)=y]} \right)_A \otimes M_{y,a}^{x,y} \,,
\end{aligned}$$

where $\approx_0$ indicates equality with respect to the state $|\text{EPR}\rangle_V \otimes |\text{AUX}\rangle$, and we have repeatedly used Fact 5.18. This is essentially the statement promised by the lemma, except that $\{M_{y,a}^{x,y}\}_a$ does not necessarily sum to identity (since we only sum over a). To remedy this, define $M_a^{x,y} = \sum_{y'} M_{y',a}^{x,y}$ and note that $M_a^{x,y} \geq M_{y,a}^{x,y}$, so by Lemma 8.16 and the fact that M^x is projective,

$$(M_{y,a}^x \otimes I_{\overline{V_x}}) \otimes I_B \approx_{\varepsilon^{1/2}} \left(\tau_{[L_x(\cdot)=y]}^Z \otimes M_a^{x,y} \otimes I_{\overline{V_x}} \right) \otimes I_B . \tag{64}$$

$\{M_a^{xy}\}$ is the POVM measurement guaranteed in the lemma statement, which concludes the proof. $\qquad\square$

8.4.2 Preliminary lemmas

We show a few simple lemmas that allow us to argue about measurements that have a decomposition across a tensor product of two Hilbert spaces, within the space of a single player.

Lemma 8.18. *Let $\mathcal{A}, \mathcal{B}$ be finite sets. Let $|\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$ be a state. Consider the following: for all $a \in \mathcal{A}$,*

1. *Let $\mathcal{H}_{A,a}$, $\mathcal{H}_{B,a}$, $\mathcal{H}_{A',a}$, and $\mathcal{H}_{B',a}$ be Hilbert spaces such that*

$$\mathcal{H}_A = \mathcal{H}_{A,a} \otimes \mathcal{H}_{A',a} \quad and \quad \mathcal{H}_B = \mathcal{H}_{B,a} \otimes \mathcal{H}_{B',a} ,$$

   and let $|\psi_{\text{QUE},a}\rangle \in \mathcal{H}_{A,a} \otimes \mathcal{H}_{B,a}$ be a "question state" and $|\psi_{\text{ANS},a}\rangle \in \mathcal{H}_{A',a} \otimes \mathcal{H}_{B',a}$ be an "answer state" such that $|\psi\rangle = |\psi_{\text{QUE},a}\rangle \otimes |\psi_{\text{ANS},a}\rangle$.

2. *Let Q_a be projectors on $\mathcal{H}_{A,a}$ such that $\{Q_a \otimes I_{\mathcal{H}_{A',a}}\}$ forms a projective measurement on $\mathcal{H}_A$ and let $\{A_b^a\}_{b\in\mathcal{B}}$ and $\{B_b^a\}_{b\in\mathcal{B}}$ be matrices acting on $\mathcal{H}_{A',a}$.*

3. *Let D be the distribution on $\mathcal{A}$ obtained by measuring $|\psi\rangle$ using $\{Q_a \otimes I_{\mathcal{H}_{A',a}}\}_{a\in\mathcal{A}}$.*

Then the following are equivalent:

- *On average over $a \sim D$ and with respect to state $|\psi\rangle$,*

$$(I_{\mathcal{H}_{A,a}} \otimes A_b^a) \otimes I_B \approx_{\varepsilon} (I_{\mathcal{H}_{B,a}} \otimes B_b^a) \otimes I_B .$$

- $(Q_a \otimes A_b^a) \otimes I_B \approx_{\varepsilon} (Q_a \otimes B_b^a) \otimes I_B$ *on state $|\psi\rangle$.*

Proof. Expand

$$\sum_{a,b} \left\| (Q_a \otimes A_b^a - Q_a \otimes B_b^a) \otimes I_B \cdot |\psi_{\text{QUE},a}\rangle \otimes |\psi_{\text{ANS},a}\rangle \right\|^2$$

$$= \sum_{a,b} \left\| Q_a \otimes (A_b^a - B_b^a) \otimes I_B \cdot |\psi_{\text{QUE},a}\rangle \otimes |\psi_{\text{ANS},a}\rangle \right\|^2$$

$$= \sum_{a,b} \left\| Q_a \otimes I_{\mathcal{H}_{B,a}} |\psi_{\text{QUE},a}\rangle \right\|^2 \cdot \left\| (A_b^a - B_b^a) \otimes I_{\mathcal{H}_{B',a}} |\psi_{\text{ANS},a}\rangle \right\|^2$$

$$= \sum_{a,b} \left\| Q_a \otimes I_B |\psi\rangle \right\|^2 \cdot \left\| (A_b^a - B_b^a) \otimes I_{\mathcal{H}_{B',a}} |\psi_{\text{ANS},a}\rangle \right\|^2$$

$$= \mathop{\mathbb{E}}_{a\sim D} \sum_b \left\| (A_b^a - B_b^a) \otimes I_{\mathcal{H}_{B',a}} |\psi_{\text{ANS},a}\rangle \right\|^2$$

$$= \mathop{\mathbb{E}}_{a\sim D} \sum_b \left\| I_{\mathcal{H}_{A,a}} \otimes (A_b^a - B_b^a) \otimes I_B \cdot |\psi_{\text{QUE},a}\rangle \otimes |\psi_{\text{ANS},a}\rangle \right\|^2 .$$

93

In going from the third to the fourth line we used the fact that

$$\left\| Q_a \otimes I_{\mathcal{H}_{\mathrm{B},a}} |\psi_{\mathrm{QUE},a}\rangle \right\|^2 = \left\| Q_a \otimes I_{\mathcal{H}_{\mathrm{A}',a}} \otimes I_{\mathrm{B}} |\psi_{\mathrm{QUE},a}\rangle \otimes |\psi_{\mathrm{ANS},a}\rangle \right\|^2 = \left\| Q_a \otimes I_{\mathcal{H}_{\mathrm{A}',a}} \otimes I_{\mathrm{B}} |\psi\rangle \right\|^2 .$$

Hence, the first line is $O(\varepsilon)$ if and only if the last one is. □

Lemma 8.19. *Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be finite sets. Let $|\psi\rangle \in \mathcal{H}_{\mathrm{A}} \otimes \mathcal{H}_{\mathrm{B}}$ be a state, and let $\{A_{a,b}\}_{a \in \mathcal{A}, b \in \mathcal{B}}$ and $\{B_{a,c}\}_{a \in \mathcal{A}, c \in \mathcal{C}}$ be POVMs acting on $\mathcal{H}_{\mathrm{A}}$. Suppose further that:*

1. *The measurements approximately commute, i.e.*

$$\left[A_{a,b}, B_{a,c} \right] \otimes I_{\mathrm{B}} \approx_\delta 0 ,$$

   where the approximation holds with respect to the state $|\psi\rangle$.

2. *For all $a \in \mathcal{A}$, there exist Hilbert spaces $\mathcal{H}_{\mathrm{A},a}$, $\mathcal{H}_{\mathrm{A}',a}$, $\mathcal{H}_{\mathrm{B},a}$, $\mathcal{H}_{\mathrm{B}',a}$, and states $|\psi_{\mathrm{QUE},a}\rangle \in \mathcal{H}_{\mathrm{A},a} \otimes \mathcal{H}_{\mathrm{B},a}$, $|\psi_{\mathrm{ANS},a}\rangle \in \mathcal{H}_{\mathrm{A}',a} \otimes \mathcal{H}_{\mathrm{B}',a}$ such that*

$$\mathcal{H}_{\mathrm{A}} = \mathcal{H}_{\mathrm{A},a} \otimes \mathcal{H}_{\mathrm{A}',a} ,$$
$$\mathcal{H}_{\mathrm{B}} = \mathcal{H}_{\mathrm{B},a} \otimes \mathcal{H}_{\mathrm{B}',a} ,$$
$$|\psi\rangle = |\psi_{\mathrm{ANS},a}\rangle \otimes |\psi_{\mathrm{QUE},a}\rangle .$$

3. *For all $a \in \mathcal{A}$, there exist projectors Q_a on $\mathcal{H}_{\mathrm{A}',a}$ and matrices $\{A_b^a\}_{b \in \mathcal{B}}$, $\{B_c^a\}_{c \in \mathcal{C}}$ acting on $\mathcal{H}_{\mathrm{A}',a}$ such that $\{Q_a \otimes I_{\mathcal{H}_{\mathrm{A}',a}}\}$ is a projective measurement on $\mathcal{H}_{\mathrm{A}}$ and*

$$A_{a,b} = Q_a \otimes A_b^a , \quad B_{a,c} = Q_a \otimes B_c^a .$$

Then

$$\left[I_{\mathcal{H}_{\mathrm{A},a}} \otimes A_b^a , I_{\mathcal{H}_{\mathrm{A},a}} \otimes B_c^a \right] \otimes I_{\mathrm{B}} \approx_\delta 0 ,$$

on average over $a \sim D$ where D is the distribution on $\mathcal{A}$ obtained by measuring $|\psi\rangle$ using $\{Q_a \otimes I_{\mathcal{H}_{\mathrm{A}',a}}\}_{a \in \mathcal{A}}$.

Proof. The assumptions of the lemma imply that

$$
\begin{aligned}
(Q_a \otimes A_b^a B_c^a) \otimes I_{\mathrm{B}} &= (Q_a \otimes A_b^a \cdot Q_a \otimes B_c^a) \otimes I_{\mathrm{B}} && (Q_a \text{ is a projector}) \\
&= (A_{a,b} \cdot B_{a,c}) \otimes I_{\mathrm{B}} && (\text{Item 3}) \\
&\approx_\delta (B_{a,c} \cdot A_{a,b}) \otimes I_{\mathrm{B}} && (\text{Item 1}) \\
&= (Q_a \otimes B_c^a \cdot Q_a \otimes A_b^a) \otimes I_{\mathrm{B}} && (\text{Item 3}) \\
&= (Q_a \otimes B_c^a A_b^a) \otimes I_{\mathrm{B}} . && (Q_a \text{ is a projector})
\end{aligned}
$$

We apply Lemma 8.18 as follows. The set "$\mathcal{A}$" in Lemma 8.18 is the same as $\mathcal{A}$ here, and the set "$\mathcal{B}$" is the product set $\mathcal{B} \times \mathcal{C}$ here. The matrices "$\{A_b^a\}$" are $\{A_b^a B_c^a\}$ here and "$\{B_b^a\}$" are $\{B_c^a A_b^a\}$ here. We then obtain, on average over $a \sim D$,

$$(I_{\mathcal{H}_{\mathrm{A},a}} \otimes A_b^a B_c^a) \otimes I_{\mathrm{B}} \approx_\delta (I_{\mathcal{H}_{\mathrm{A},a}} \otimes B_c^a A_b^a) \otimes I_{\mathrm{B}} .$$

This implies the conclusion of the lemma. □

Lemma 8.20. *Let $\mathcal{Y}$ be a finite set and for all $y \in \mathcal{Y}$ let $\{A_{x,z}^y\}$ be a POVM on $\mathcal{H}_A$. Let $\{B_{x,y,z}\}$ be a projective measurement on $\mathcal{H}_B$. Suppose that*

$$\sum_{x,y,z} \langle \psi | A_{x,z}^y \otimes B_{x,y,z} | \psi \rangle \geq 1 - \delta . \tag{65}$$

Then with respect to state $|\psi\rangle$,

$$I_A \otimes B_{x,y,z} \approx_\delta A_{x,z}^y \otimes B_{x,y} .$$

Proof. Using the fact that $\{B_{x,y,z}\}$ is projective, we have $B_{x,y,z} = B_{x,y}B_z$ for all x,y,z, so that (65) implies

$$\sum_{x,y,z} \langle \psi | A_{x,z}^y \otimes B_{x,y}B_z | \psi \rangle \geq 1 - \delta .$$

Define $\hat{A}_z = \sum_{x,y} A_{x,z}^y \otimes B_{x,y}$ and $\hat{B}_z = I_A \otimes B_z$. The above equation simplifies to

$$\sum_{y,z} \langle \psi | \hat{A}_z \hat{B}_z | \psi \rangle \geq 1 - \delta .$$

This implies that $\hat{A}_z \approx_\delta \hat{B}_z$ as

$$\sum_z \left\| (\hat{A}_z - \hat{B}_z) |\psi\rangle \right\|^2 = \sum_z \langle \psi | \hat{A}_z^2 + \hat{B}_z^2 | \psi \rangle - 2 \langle \psi | \hat{A}_z \hat{B}_z | \psi \rangle \leq 2\delta ,$$

where the equality uses the fact that $\hat{A}_z$ and $\hat{B}_z$ commute. To conclude the proof, we have

$$I_A \otimes B_{x,y,z} = I_A \otimes B_{x,y}B_z \approx_\delta \left(I_A \otimes B_{x,y} \right) \sum_{x',y'} A_{x',z}^{y'} \otimes B_{x',y'} = A_{x,z}^y \otimes B_{x,y},$$

where the approximation follows from $\hat{A}_z \approx_\delta \hat{B}_z$ and Fact 5.18.

$\qquad\square$

8.4.3 Proof of Theorem 8.9

We analyze the soundness of the introspective verifier.

Proof of Theorem 8.9. Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier such that $\mathcal{S}$ is an ℓ-level sampler. Recall the definition of the introspective verifier $\mathcal{V}^{\text{INTRO}} = (\mathcal{S}^{\text{INTRO}}, \mathcal{D}^{\text{INTRO}})$ corresponding to $\mathcal{V}$ from Section 8.2. Fix an index $n \geq 1$ and let $N = 2^n$. Let $r = (\lambda N)^\lambda$ and $\text{introparams}(r) = (q, m, d, h, H, \Gamma, \pi)$, as in Section 8.2.

As in the proof of Theorem 8.6, we make the following simplifications. First, we analyze the soundness of the typed introspective verifier $\mathcal{V}^{\text{INTRO}}$; the soundness of the detyped verifier $\text{detype}(\mathcal{V}^{\text{INTRO}})$ follows from Lemma 6.18 and the fact that the type set $\mathcal{T}^{\text{INTRO}}$ has size $O(\ell)$. Second, analogously to Remark 8.7 without loss of generality for notational simplicity we consider strategies for questions sampled from $\tilde{\mathcal{S}}^{\text{INTRO}}$ rather than the downsized sampler $\mathcal{S}^{\text{INTRO}}$.

Suppose that $\text{val}^*(\mathcal{V}_n^{\text{INTRO}}) > 1 - \varepsilon$ for some $0 < \varepsilon < 1$, and let $\mathscr{S} = (\psi, \hat{A}, \hat{B})$ be a strategy for $\mathcal{V}_n^{\text{INTRO}}$ with value at least $1 - \varepsilon$. Since $\mathscr{S}$ has success probability that is strictly positive, the decider $\mathcal{D}^{\text{INTRO}}$ does not automatically reject, which means that

$$s(N) \leq (\lambda N)^\lambda . \tag{66}$$

We analyze each of the tests performed by $\mathcal{D}^{\text{INTRO}}$ (see Figure 10) in sequence, and state consequences of each test. We start with Item 1, the Pauli test.

Lemma 8.21. *There is a $\delta_1 = \mathrm{poly}(\varepsilon + 1/r)$ such that the following holds. Let $Q = \Gamma \log q$. There is a projective strategy $\mathscr{S}' = (|\psi\rangle, A, B)$ for $\mathcal{V}_n^{\mathrm{INTRO}}$ that succeeds with probability at least $1 - \delta_1$ and furthermore*

$$|\psi\rangle_{\mathrm{AB}} = |\mathrm{EPR}_2\rangle_{\mathrm{A'B'}}^{\otimes Q} \otimes |\mathrm{AUX}\rangle_{\mathrm{A''B''}} \tag{67}$$

for some bipartite state $|\mathrm{AUX}\rangle$, and for all $W \in \{X, Z\}$,

$$A_x^{\mathrm{PAULI},W} = \sigma_x^W , \qquad B_x^{\mathrm{PAULI},W} = \sigma_x^W , \tag{68}$$

where σ_x^W acts on the first $s(N)$ qubits of player A's share (resp. B's share) of $|\mathrm{EPR}_2\rangle^{\otimes Q}$.

Proof. Given the definition of the type graph G^{INTRO}, for $((\hat{t}_{\mathrm{A}}, \hat{x}_{\mathrm{A}}), (\hat{t}_{\mathrm{B}}, \hat{x}_{\mathrm{B}}))$ sampled according to $\mu_{\mathscr{S}^{\mathrm{INTRO}}, n}$ it holds that $(\hat{t}_{\mathrm{A}}, \hat{t}_{\mathrm{B}}) \in \mathcal{T}^{\mathrm{PAULI}} \times \mathcal{T}^{\mathrm{PAULI}}$ with constant probability. Therefore, conditioned on the Pauli test, Item 1, being executed, $\mathscr{S}$ must succeed in the test with probability $1 - O(\varepsilon)$.

Observe that conditioned on the test being executed, the distribution of $((\hat{t}_{\mathrm{A}}, \hat{x}_{\mathrm{A}}), (\hat{t}_{\mathrm{B}}, \hat{x}_{\mathrm{B}}))$ is, by definition, exactly the distribution of questions in the Pauli basis game with parameters qldparams, as described in Section 7.3.1. By Theorem 7.14 it follows that there exists a local isometry $\phi = \phi_{\mathrm{A}} \otimes \phi_{\mathrm{B}}$ and a state $|\mathrm{AUX}\rangle \in \mathcal{H}_{\mathrm{A''}} \otimes \mathcal{H}_{\mathrm{B''}}$ such that

$$\|\phi(|\psi\rangle) - |\mathrm{EPR}_2\rangle^{\otimes Q} \otimes |\mathrm{AUX}\rangle\|^2 \leq \delta'(\varepsilon, \ell, r) , \tag{69}$$

where $\delta'(\varepsilon, \ell, r)$ is an upper bound on $\delta(O(\varepsilon), q, m, d)$ that only depends on ε and r, as stated in Lemma 7.17. In addition, defining $A_{\hat{a}}^{\hat{x}} = \phi_{\mathrm{A}}(\hat{A}_{\hat{a}}^{\hat{x}})$ for all questions $\hat{x}$ and answers $\hat{a}$, for $W \in \{X, Z\}$ it holds that

$$A_x^{\mathrm{PAULI},W} \otimes I_{\mathrm{B}} \approx_{\delta'(\varepsilon, \ell, r)} \sigma_x^W \otimes I_{\mathrm{B}} , \tag{70}$$

and a similar set of equations hold for operators associated with the second player. Using Naimark's theorem as formulated in [NW19, Theorem 4.2], at the cost of extending the state $|\mathrm{AUX}\rangle$ we may assume that the measurements are projective without loss of generality. Define the strategy $\mathscr{S}'$ which uses the state $|\mathrm{EPR}_2\rangle^{\otimes Q} \otimes |\mathrm{AUX}\rangle$ and measurement operators $\{A_{\hat{a}}^{\hat{x}}\}$ and $\{B_{\hat{a}}^{\hat{x}}\}$ for all questions $\hat{x}$, except for (PAULI, W)-type questions where instead the Pauli measurements $\sigma_{\hat{a}}^W$ are used. Using (69) and (70) the strategy $\mathscr{S}'$ succeeds in $\mathcal{V}_n^{\mathrm{INTRO}}$ with probability at least $1 - \delta'(\varepsilon, \ell, r)$.

The claimed bound on δ_1 follows from the bound given in Lemma 7.17. $\qquad\square$

In the remainder of the proof we analyze the strategy $\mathscr{S}'$ from Lemma 8.21. We use the following notation conventions:

1. We use indices A and B to label each player's Hilbert space after application of the isometry ϕ from Lemma 8.21.

2. We write V for the register subspace of $\mathbb{F}_2^Q$ spanned by $e_1, \ldots, e_{s(N)}$ and $\overline{V}$ for its complement. (Note that by definition of introparams in Section 7.3.3 it holds that $s(N) \leq r \leq Q$, where the first inequality follows from (66).)

3. Whenever we write a Pauli operator σ_a^W the register on which the operator acts should always be clear from context, and is implicit from the space in which the outcome a ranges.

4. For measurement operators in the introspection game, the variables for the measurement outcomes follow the specification of the "answer key" in Figure 8 (for $\mathcal{T}^{\text{PAULI}}$-type questions) and Figure 10 (for all other question types). For example, the measurement operators $\{A_x^{\text{PAULI},W}\}$ corresponding to question type (PAULI, W) are indexed by vectors $x \in \mathbb{F}_q^Q$ where $Q = \Gamma \log q$. The measurement operators corresponding to question type (INTRO, v) for $v \in \{A, B\}$ are indexed by pairs $(y, a) \in V \times \{0, 1\}^{\leq 9r}$.[22] We often refer to marginalized measurement operators, e.g., the operator $A_y^{\text{INTRO},v}$ denotes marginalizing $A_{y,a}^{\text{INTRO},v}$ over all a. In these cases, the part of the answer that is marginalized over will be clear from context.

5. We use the notation δ to denote a function which is polynomial in δ_1, although the exact expression may differ from occurrence to occurrence. The polynomial itself may depend on ℓ, but we leave this dependence implicit; due to the use of inductive steps that involve taking the square root of the error ℓ times in sequence (e.g. Lemma 8.26) the exponent generally depends on ℓ.

The next two lemma derive conditions implied by Items 2 and 3 of the checks performed by the decider $\mathcal{D}^{\text{INTRO}}$ described in Figure 10. As these two parts are performed independently for the two possible values of $v \in \{A, B\}$, we only discuss the case where $v = A$. For notational simplicity, whenever possible we omit v when referring to the measurement operators. For example L, $A_{y,a}^{\text{INTRO}}$ and $B_{z,a}^{\text{SAMPLE}}$ are used as shorthand notation for L^A, $A_{y,a}^{\text{INTRO},A}$, and $B_{z,a}^{\text{SAMPLE},A}$ respectively.

Lemma 8.22 (Sampling test, Item 2 of Figure 10). *For each $k \in \{1, 2, \ldots, \ell\}$,*

$$I_A \otimes B_z^{\text{SAMPLE}} \simeq_\delta \sigma_z^Z \otimes I_B , \tag{71}$$

$$A_{y_{\leq k},a}^{\text{INTRO}} \otimes I_B \simeq_\delta I_A \otimes B_{[L_{\leq k}(\cdot)=y_{\leq k}],a}^{\text{SAMPLE}} , \tag{72}$$

where z ranges over V and $y_{\leq k}$ ranges over $L_{\leq k}(V)$. Moreover, analogous equations hold with operators acting on the other side of the tensor product.

Proof. When $((\hat{t}_A, \hat{x}_A), (\hat{t}_B, \hat{x}_B))$ is sampled according to $\mu_{\mathcal{S}^{\text{INTRO}},n}$, each check in Item 2 of Figure 10 is executed with probability $\Omega(1/\ell)$ (this is due to the number of types in $\mathcal{T}^{\text{INTRO}}$ and the structure of the type graph G^{INTRO}). Therefore, in each of the checks specified by Items 2a and 2b, the strategy $\mathscr{S}'$ succeeds with probability at least $1 - O(\ell\delta)$, conditioned on the test being executed. Item 2a for $w = A$ combined with (68) implies (71). Item 2b for $w = A$, combined with Fact 5.21, implies (72). The lemma follows from repeating the same argument with the tensor factors interchanged. $\square$

Lemma 8.23 (Hiding test, Item 3 of Figure 10). *For each $k \in \{1, \ldots, \ell+1\}$,*

$$A_{y_{<k},a}^{\text{INTRO}} \otimes I_B \simeq_\delta I_A \otimes B_{y_{<k},a}^{\text{READ}} , \tag{73}$$

and if $k \leq \ell$,

$$I_A \otimes B_{y_{<k}}^{\text{HIDE}_k} \approx_\delta \sigma_{[L_{<k}(\cdot)=y_{<k}]}^Z \otimes I_B . \tag{74}$$

Furthermore, for all $j, k \in \{1, \ldots, \ell-1\}$ such that $j \leq k$, we have

$$A_{y_{<j},y_{\leq j}^\perp}^{\text{HIDE}_k} \otimes I_B \approx_\delta A_{y_{<j},y_{\leq j}^\perp}^{\text{HIDE}_{k+1}} \otimes I_B . \tag{75}$$

Analogous equations to (73), (74), and (75) hold with operators acting on the other side of the tensor product.

---

[22]Technically the answer a may be a binary string of any length; however, if a is too long the decider rejects due to the answer length check. Thus we assume without loss of generality that the answer a is a binary string of length at most $6s(N) + 2r + 3 \leq 9r$.

Proof. When $((\hat{t}_A, \hat{x}_A), (\hat{t}_B, \hat{x}_B))$ is sampled according to $\mu_{\mathcal{S}^{\text{INTRO}},n}$, each check in Item 3 of Figure 10 is executed with probability $\Omega(1/\ell)$. Therefore, in each of the checks specified by Items 3a, 3b, and 3c (conditioned on the right types) the strategy $\mathcal{S}'$ succeeds with probability at least $1 - O(\ell\delta)$.

Item 3a for $w = $ A combined with Fact 5.21 implies (73). Combining Eqs. (71) and (72) with (73) yields

$$I_A \otimes B_{y_{<k}}^{\text{READ}} \approx_\delta \sigma_{[L_{<k}(\cdot) = y_{<k}]}^Z \otimes I_B . \tag{76}$$

The fact that $\mathcal{S}'$ is projective and succeeds in Items 3b and 3c of Figure 10 with probability $1 - O(\ell\delta)$, along with Item 1 of Fact 5.17, imply (74).

We now establish the "Furthermore" part of the lemma statement. Let $1 \le j \le k \le \ell - 1$. Item 3c, Item 1 of Fact 5.17, and Fact 5.21 imply that

$$A_{y_{<j}, y_{\le j}^\perp}^{\text{HIDE}_k} \otimes I_B \approx_\delta I_A \otimes B_{y_{<j}, y_{\le j}^\perp}^{\text{HIDE}_{k+1}}. \tag{77}$$

Item 5 and Fact 5.21 imply

$$A_{y_{<j}, y_{\le j}^\perp}^{\text{HIDE}_{k+1}} \otimes I_B \simeq_\delta I_A \otimes B_{y_{<j}, y_{\le j}^\perp}^{\text{HIDE}_{k+1}} . \tag{78}$$

This proves (75).

The lemma follows from repeating the same arguments with the tensor factors interchanged. $\qquad\square$

We exploit the tests performed in Item 3 further to show the following lemma.

Lemma 8.24. *For all $k \in \{1, 2, \ldots, \ell\}$,*

$$I_A \otimes B_{y_{<k}, y_k^\perp, x_{>k}}^{\text{HIDE}_k} \approx_\delta \left(\sigma_{[L_{<k}(\cdot) = y_{<k}]}^Z \otimes \sigma_{[L_k^\perp, y_{<k}(\cdot) = y_k^\perp]}^X \otimes \sigma_{x_{>k}}^X \right) \otimes I_B ,$$

and an analogous equation holds with operators acting on the other side of the tensor product.

Proof. The proof is by induction on k. We first show the case $k = 1$. Under the distribution $\mu_{\mathcal{S}^{\text{INTRO}},n}$, the check in Item 3d of Fig. 10 is executed with probablity $\Omega(1/\ell)$. That part of the check for $w = $ A together with Eq. (68) implies that

$$I_A \otimes B_{y_1^\perp, x_{>1}}^{\text{HIDE}_1} \approx_\delta \left(\sigma_{[L_1^\perp, y_{<1}(\cdot) = y_1^\perp]}^X \otimes \sigma_{x_{>1}}^X \right) \otimes I_B . \tag{79}$$

This proves the case for $k = 1$. Next we perform the induction step. Assume that the lemma holds for some $k \in \{1, 2, \ldots, \ell - 1\}$. The check of Item 3c is executed with probablity $\Omega(1/\ell)$; using that $\mathcal{S}'$ succeeds in Item 3c (conditioned on the right types having been sampled) with probablity at least $1 - O(\ell\delta)$, we have

$$\sum_{y_{\le k}, y_{k+1}^\perp, x_{>k+1}} \langle \psi | A_{y_{<k}, [L_{k+1}^\perp, y_{\le k}(\cdot) = y_{k+1}^\perp], x_{>k+1}}^{\text{HIDE}_k} \otimes B_{y_{\le k}, y_{k+1}^\perp, x_{>k+1}}^{\text{HIDE}_{k+1}} | \psi \rangle \ge 1 - O(\ell\delta) .$$

We now apply Lemma 8.20, choosing the measurements A, B and outcomes x, y, z in the lemma as follows:

$$\text{"}A\text{"} : A_{y_{<k}, [L_{k+1}^\perp, y_{\le k}(\cdot) = y_{k+1}^\perp], x_{>k+1}}^{\text{HIDE}_k} , \quad \text{"}B\text{"} : B_{y_{\le k}, y_{k+1}^\perp, x_{>k+1}}^{\text{HIDE}_{k+1}} ,$$

$$\text{"}x\text{"} : y_{<k} , \quad \text{"}y\text{"} : y_k , \quad \text{"}z\text{"} : (y_{k+1}^\perp, x_{>k+1}) .$$

Note that here A does not depend on y, so we use the same A for all values of y. Lemma 8.20 with the above choices of parameters implies that

$$
\begin{aligned}
I_{\mathrm{A}} \otimes B^{\mathrm{HIDE}_{k+1}}_{y_{\leq k}, y^{\perp}_{k+1}, x_{>k+1}} &\approx_{\delta} A^{\mathrm{HIDE}_k}_{y_{<k}, [L^{\perp}_{k+1, y_{\leq k}}(\cdot)=y^{\perp}_{k+1}], x_{>k+1}} \otimes B^{\mathrm{HIDE}_{k+1}}_{y_{\leq k}} \\
&\approx_{\delta} \left(\sigma^Z_{[L_{<k}(\cdot)=y_{<k}]} \otimes \sigma^X_{[L^{\perp}_{k+1, y_{<k+1}}(\cdot)=y^{\perp}_{k+1}]} \otimes \sigma^X_{x_{>k+1}} \right) \otimes B^{\mathrm{HIDE}_{k+1}}_{y_{\leq k}} \\
&\approx_{\delta} \left(\sigma^Z_{[L_{<k}(\cdot)=y_{<k}]} \sigma^Z_{[L_{\leq k}(\cdot)=y_{\leq k}]} \otimes \sigma^X_{[L^{\perp}_{k+1, y_{<k+1}}(\cdot)=y^{\perp}_{k+1}]} \otimes \sigma^X_{x_{>k+1}} \right) \otimes I_{\mathrm{B}}, \\
&\approx_0 \left(\sigma^Z_{[L_{<k+1}(\cdot)=y_{<k+1}]} \otimes \sigma^X_{[L^{\perp}_{k+1, y_{<k+1}}(\cdot)=y^{\perp}_{k+1}]} \otimes \sigma^X_{x_{>k+1}} \right) \otimes I_{\mathrm{B}},
\end{aligned}
$$

where the input to $L^{\perp}_{k+1, y_{<k+1}}(\cdot)$ is x_{k+1}. The second approximation uses the induction hypothesis, Fact 5.21, Fact 5.17, and Fact 5.18. The third approximation follows from Eq. (74) and Fact 5.18. The fourth approximation follows from the definition of CL functions. This completes the induction. $\qquad\square$

Lemma 8.25. *For all $k \in \{1, \ldots, \ell\}$,*

$$
I_{\mathrm{A}} \otimes B^{\mathrm{READ}}_{y_{<k}, y^{\perp}_k} \approx_{\delta} \left(\sigma^Z_{[L_{<k}(\cdot)=y_{<k}]} \otimes \sigma^X_{[L^{\perp}_{k, y_{<k}}(\cdot)=y^{\perp}_k]} \right) \otimes I_{\mathrm{B}} . \tag{80}
$$

Moreover, analogous equations hold with operators acting on the other side of the tensor product.

Proof. Lemma 8.24 and Fact 5.21 imply that

$$
A^{\mathrm{HIDE}_k}_{y_{<k}, y^{\perp}_k} \otimes I_{\mathrm{B}} \approx_{\delta} \left(\sigma^Z_{[L_{<k}(\cdot)=y_{<k}]} \otimes \sigma^X_{[L^{\perp}_{k, y_{<k}}(\cdot)=y^{\perp}_k]} \right) \otimes I_{\mathrm{B}} . \tag{81}
$$

Since the strategy $\mathscr{S}'$ succeeds in Item 3b with probability at least $1 - O(\ell\delta)$ it follows from Fact 5.21 that

$$
A^{\mathrm{HIDE}_{\ell}}_{y_{<\ell}, y^{\perp}_{\leq \ell}} \otimes I_{\mathrm{B}} \approx_{\delta} I_{\mathrm{A}} \otimes B^{\mathrm{READ}}_{y_{<\ell}, y^{\perp}_{\leq \ell}} . \tag{82}
$$

An inductive argument applied to (75) of Lemma 8.23, combined with Fact 5.21, implies that for all $1 \leq k \leq \ell$ we have

$$
A^{\mathrm{HIDE}_k}_{y_{<k}, y^{\perp}_k} \otimes I_{\mathrm{B}} \approx_{\delta} A^{\mathrm{HIDE}_{\ell}}_{y_{<k}, y^{\perp}_k} \otimes I_{\mathrm{B}}. \tag{83}
$$

Equations (81), (82), and (83), combined with Fact 5.21, then establishes the lemma statement. $\qquad\square$

Lemma 8.26. *For each $k \in \{1, 2, \ldots, \ell+1\}$, there exists a product state $|\mathrm{ANC}_k\rangle = |\mathrm{ANC}_{k,\mathrm{A}}\rangle \otimes |\mathrm{ANC}_{k,\mathrm{B}}\rangle \in \mathcal{H}_{\mathrm{A}'_k} \otimes \mathcal{H}_{\mathrm{B}'_k}$ and, for each $y_{<k} \in L_{<k}(V)$, a projective measurement $\{A^{\mathrm{INTRO}, y_{<k}}_{y_{\geq k}, a}\}$ that acts on $\mathcal{H}_{\mathrm{A}} \otimes \mathcal{H}_{\mathrm{A}'_k}$ such that the following holds. First, for all $y \in V$, the operator $A^{\mathrm{INTRO}, y_{<k}}_{y_{\geq k}, a}$ acts as identity on the register subspace spanned by basis vectors for the subspace $V_{<k}(y_{<k})$, and as a consequence the operator*

$$
A^{\mathrm{INTRO}, Z_{<k}}_{y, a} = \sigma^Z_{[L_{<k}(\cdot)=y_{<k}]} \otimes A^{\mathrm{INTRO}, y_{<k}}_{y_{\geq k}, a} \tag{84}
$$

is well-defined. Second, let $\mathscr{S}''_k$ be the strategy defined as follows. The state is $|\mathrm{EPR}_2\rangle^{\otimes Q} \otimes |\mathrm{AUX}\rangle \otimes |\mathrm{ANC}_k\rangle$. The measurements are identical to those in $\mathscr{S}'$ defined in Lemma 8.21, except that $\{A^{\mathrm{INTRO}}_{y, a}\}$ is replaced with $\{A^{\mathrm{INTRO}, Z_{<k}}_{y, a}\}$. Then $\mathscr{S}''_k$ succeeds with probability at least $1 - \delta$ in the game $\mathcal{V}^{\mathrm{INTRO}}_n$.

Proof. The proof is by induction on k from 1 to $\ell+1$. The case $k = 1$ is trivial by setting $A^{\mathrm{INTRO}, y_{<1}}_{y_{\geq 1}, a} = A^{\mathrm{INTRO}}_{y, a}$ for all y, a. Assume that for some $k \in \{1, 2, \ldots, \ell\}$ there exists projective measurements $\{A^{\mathrm{INTRO}, y_{<k}}_{y_{\geq k}, a}\}$ for every $y_{<k}$ and a strategy $\mathscr{S}''_k$ satisfying the conditions of the lemma statement. We show the statement of the lemma for $k+1$.

Commutation with Z-basis measurements. We first prove that on average over $y_{<k}$, the measurement operator $A_{y_{\geq k}, a}^{\text{INTRO}, y_{<k}}$, which comes from the inductive assumption, commutes with the projective measurement $\{\sigma_{z_k}^Z\}$ where the outcomes z_k range over the factor space $V_k(y_{<k})$.

To do so, we first apply Lemma 5.22 where we we choose the measurements "A", "B", and "C" and outcomes "a", "b", and "c" in the lemma as follows:

$$\text{``}A\text{''}: \{A_{y,a}^{\text{INTRO}, Z_{<k}}\}, \quad \text{``}B\text{''}: \{B_{z,a}^{\text{SAMPLE}}\}, \quad \text{``}C\text{''}: \{\sigma_z^Z\},$$
$$\text{``}a\text{''}: y_{<k}, \quad \text{``}b\text{''}: (y_{\geq k}, a), \quad \text{``}c\text{''}: z.$$

To make sense of how the "B" POVM is indexed by "a", "b", and "c" as described above, we use the following relabelling: for all (z, a), identify $B_{z,a}^{\text{SAMPLE}}$ with $B_{y,a,z}^{\text{SAMPLE}}$ where $y = L(z)$. Similarly, for the "C" POVM, we identify σ_z^Z with the operator $\sigma_{y_{<k}, z}^Z$ where $y_{<k} = L_{<k}(z)$. By applying Lemma 8.22 to $\mathscr{S}_k''$ (the strategy given by the inductive hypothesis) with "k" in Lemma 8.22 set to ℓ, we have that

$$A_{y,a}^{\text{INTRO}, Z_{<k}} \otimes I_{\text{B}} \approx_\delta I_{\text{A}} \otimes B_{y,a}^{\text{SAMPLE}} \tag{85}$$

where $B_{y,a}^{\text{SAMPLE}} = \sum_{z: L(z) = y} B_{z,a}^{\text{SAMPLE}}$. Equations (71) and (85) imply that the conditions of Lemma 5.22 are satisfied, and thus we obtain

$$\left[A_{y,a}^{\text{INTRO}, Z_{<k}}, \sigma_z^Z\right] \otimes I_{\text{B}} \approx_\delta 0 \tag{86}$$

where in the answer summation, y is a deterministic function of z. We now apply Lemma 8.19, choosing the measurements "A", "B", "Q" and outcomes "a", "b", "c" in the lemma as follows:

$$\text{``}A\text{''}: \{A_{y,a}^{\text{INTRO}, Z_{<k}}\}, \quad \text{``}B\text{''}: \{\sigma_{[L_{<k}(\cdot) = y_{<k}]}^Z \otimes \sigma_{z_k}^Z\}, \quad \text{``}Q\text{''}: \{\sigma_{[L_{<k}(\cdot) = y_{<k}]}^Z\},$$
$$\text{``}a\text{''}: y_{<k}, \quad \text{``}b\text{''}: (y_{\geq k}, a), \quad \text{``}c\text{''}: z_k.$$

We choose the Hilbert spaces "$\mathcal{H}_{\text{A},a}$" and "$\mathcal{H}_{\text{B},a}$" as the register subspace $V_{<k}(y_{<k})$, and "$\mathcal{H}_{\text{A}',a}$" and "$\mathcal{H}_{\text{B}',a}$" as the register subspace $V_{\geq k}(y_{<k})$ tensored with $\mathcal{H}_{\text{A}''} \otimes \mathcal{H}_{\text{B}''}$, the Hilbert space of the state $|\text{AUX}\rangle$. Thus for every $y_{<k}$, the state $|\text{EPR}\rangle^{\otimes Q} \otimes |\text{AUX}\rangle$ of the strategy $\mathscr{S}_k''$ can be decomposed into a tensor product of a "question state" and an "answer state" as follows:

$$\left(|\text{EPR}_2\rangle_{V_{<k}(y_{<k})}\right)_{\mathcal{H}_{\text{A},a}\mathcal{H}_{\text{B},a}} \otimes \left(|\text{EPR}_2\rangle_{V_{\geq k}(y_{<k})} \otimes |\text{AUX}\rangle\right)_{\mathcal{H}_{\text{A}',a}\mathcal{H}_{\text{B}',a}}.$$

Let $\mu_{L_{<k}}$ denote the distribution over outcomes $y_{<k}$ generated by performing the "Q" measurement on the state $|\text{EPR}\rangle^{\otimes Q}$, which is equivalent to the distribution generated by the following procedure: (1) sample a uniformly random $z \in V$; (2) compute $y = L(z)$; (3) return $y_{<k}$. Then, since Equation (86) and the inductive hypothesis about the structure of $A_{y_{\leq k}, a}^{\text{INTRO}, Z_{<k}}$ satisfy the conditions of Lemma 8.19, we obtain on average over $y_{<k} \sim \mu_{L_{<k}}$

$$\left[A_{y_{\geq k}, a}^{\text{INTRO}, y_{<k}}, \sigma_{z_k}^Z\right] \otimes I_{\text{B}} \approx_\delta 0. \tag{87}$$

Here, the measurement outcomes z_k range over $V_k(y_{<k})$.

Commutation with X-basis measurements. Next, we first prove that on average over $y_{<k}$, the measurement operator $A_{y_{\geq k}, a}^{\text{INTRO}, y_{<k}}$ commutes with the projective measurement $\{\sigma_{[L_{k, y_{<k}}^\perp(\cdot) = y_k^\perp]}^X\}$ where the outcomes $y_k^\perp$ are elements of the factor space $V_k(y_{<k})$.

We again apply Lemma 5.22, choosing the measurements and outcomes in the lemma as follows:

$$\text{``}A\text{''}: \{A_{y,a}^{\text{INTRO},Z_{<k}}\}\,, \quad \text{``}B\text{''}: \{B_{y,y_k^\perp,a}^{\text{READ}}\}\,, \quad \text{``}C\text{''}: \{\sigma_{[L_{<k}(\cdot)=y_{<k}]}^Z \otimes \sigma_{[L_{k,y_{<k}}^\perp(\cdot)=y_k^\perp]}^X\}\,,$$

$$\text{``}a\text{''}: y_{<k}\,, \quad \text{``}b\text{''}: (y_{\geq k},a)\,, \quad \text{``}c\text{''}: y_k^\perp\,.$$

Equations (73) (with "k" in Lemma 8.23 chosen to be $\ell+1$) and (80) imply the conditions of Lemma 5.22, so we obtain

$$\left[A_{y,a}^{\text{INTRO},Z_{<k}}\,,\, \sigma_{[L_{<k}(\cdot)=y_{<k}]}^Z \otimes \sigma_{[L_{k,y_{<k}}^\perp(\cdot)=y_k^\perp]}^X\right] \otimes I_{\text{B}} \approx_\delta 0\,. \tag{88}$$

We then apply Lemma 8.19 with the following choice of measurements and outcomes:

$$\text{``}A\text{''}: \{A_{y,a}^{\text{INTRO},Z_{<k}}\}\,, \quad \text{``}B\text{''}: \{\sigma_{[L_{<k}(\cdot)=y_{<k}]}^Z \otimes \sigma_{[L_{k,y_{<k}}^\perp(\cdot)=y_k^\perp]}^X\}\,, \quad \text{``}Q\text{''}: \{\sigma_{[L_{<k}(\cdot)=y_{<k}]}^Z\}\,,$$

$$\text{``}a\text{''}: y_{<k}\,, \quad \text{``}b\text{''}: (y_{\geq k},a)\,, \quad \text{``}c\text{''}: y_k^\perp\,.$$

The Hilbert space and state decomposition are the same as in the previous invocation of Lemma 8.19. Equation (88) and the inductive hypothesis satisfy the conditions of Lemma 8.19, and we similarly obtain that on average over $y_{<k} \sim \mu_{L_{<k}}$,

$$\left[A_{y_{\geq k},a}^{\text{INTRO},y_{<k}}\,,\, \sigma_{[L_{k,y_{<k}}^\perp(\cdot)=y_k^\perp]}^X\right] \otimes I_{\text{B}} \approx_\delta 0\,, \tag{89}$$

Applying the Pauli twirl. The last step is to apply the Pauli twirl to decompose the family of measurements $\{A_{y_{\geq k},a}^{\text{INTRO},y_{<k}}\}$ into a tensor product measurement, with the first part of the tensor product measuring the k-th linear map of L.

Again applying Lemma 8.22 to $\mathscr{S}_k''$, and using Facts 5.21 and 5.17, we obtain that

$$A_{y_{\leq k}}^{\text{INTRO},Z_{<k}} \otimes I_{\text{B}} \approx_\delta \sigma_{[L_{\leq k}(\cdot)=y_{\leq k}]}^Z \otimes I_{\text{B}}$$

which is equivalent to, by the inductive hypothesis,

$$\sigma_{[L_{<k}(\cdot)=y_{<k}]}^Z \otimes A_{y_k}^{\text{INTRO},y_{<k}} \otimes I_{\text{B}} \approx_\delta \sigma_{[L_{<k}(\cdot)=y_{<k}]}^Z \otimes \sigma_{[L_{k,y_{<k}}(\cdot)=y_k]}^Z \otimes I_{\text{B}}\,. \tag{90}$$

Applying Lemma 8.18 to Equation (90), we conclude that

$$A_{y_k}^{\text{INTRO},y_{<k}} \otimes I_{\text{B}} \approx_\delta \sigma_{[L_{<k}(\cdot)=y_{<k}]}^Z \otimes I_{\text{B}}\,, \tag{91}$$

on average over $y_{<k} \sim \mu_{L_{<k}}$.

Now we apply Lemma 8.17 with the following identification:

$$\text{``}x\text{''}: y_{<k}\,, \quad \text{``}y\text{''}: y_k\,, \quad \text{``}a\text{''}: (y_{>k},a)\,, \quad \text{``}L_x\text{''}: L_{k,y_{<k}}\,,$$

$$\text{``}U_x\text{''}: V_k(y_{<k})\,, \quad \text{``}M_{y,a}^x\text{''}: A_{y_{\geq k},a}^{\text{INTRO},y_{<k}}\,, \quad \text{``}M_a^{x,y}\text{''}: A_{y_{>k},a}^{\text{INTRO},y_{\leq k}}\,.$$

The "Consistency" condition is implied by Equation (91) and the "Commutation" conditions are implied by Equations (87) and (89). We obtain that for all $y_{\leq k}$ there exists POVM measurements $\{A_{y_{>k},a}^{\text{INTRO},y_{\leq k}}\}$ that act as identity on the register subspace spanned by basis vectors for the subspace $V_{<k}(y_{<k})$ and, on average over $y_{<k} \sim \mu_{L_{<k}}$, we have

$$A_{y_{\geq k},a}^{\text{INTRO},y_{<k}} \otimes I_{\text{B}} \approx_\delta \left(\sigma_{[L_{k,y_{<k}}(\cdot)=y_k]}^Z \otimes A_{y_{>k},a}^{\text{INTRO},y_{\leq k}}\right) \otimes I_{\text{B}}\,. \tag{92}$$

Using the inductive assumption on the structure of $A_{y,a}^{\text{INTRO},Z_{<k}}$ from (84), we get

$$A_{y,a}^{\text{INTRO},Z_{<k}} \otimes I_{\text{B}} = \left(\sigma_{[L_{<k}(\cdot)=y_{<k}]}^{Z} \otimes A_{y_{\geq k},a}^{\text{INTRO},y_{<k}} \right) \otimes I_{\text{B}} \tag{93}$$

$$\approx_{\delta} \left(\sigma_{[L_{\leq k}(\cdot)=y_{\leq k}]}^{Z} \otimes A_{y_{>k},a}^{\text{INTRO},y_{\leq k}} \right) \otimes I_{\text{B}} . \tag{94}$$

where the second line follows from (92) and Lemma 8.18. By (94), replacing the projective measurement $\left\{ A_{y,a}^{\text{INTRO},Z_{<k}} \right\}$ with the POVM

$$\left\{ \sigma_{[L_{\leq k}(\cdot)=y_{\leq k}]}^{Z} \otimes A_{y_{>k},a}^{\text{INTRO},y_{\leq k}} \right\}$$

in the strategy $\mathscr{S}_k''$ results in a strategy that succeeds with probability at least $1-\delta$. To show that $\left\{ A_{y_{>k},a}^{\text{INTRO},y_{\leq k}} \right\}$ can furthermore be turned into a projective measurement we use Naimark's theorem as formulated in [NW19, Theorem 4.2]. By inspecting the proof of Naimark's theorem, one can see that the state $|\text{ANC}'\rangle$ it produces is a product state with no entanglement. This yields a strategy $\mathscr{S}_{k+1}''$ with ancilla state $|\text{ANC}_{k+1}\rangle = |\text{ANC}_k\rangle|\text{ANC}'\rangle$, which establishes the induction hypothesis for $k+1$. This completes the proof of Lemma 8.26. $\qquad\square$

Taking $k = \ell + 1$ in Lemma 8.26 we obtain a strategy $\mathscr{S}'' = \mathscr{S}_{\ell+1}''$ with value $1-\delta$ in which, given question (INTRO, A), player A performs the measurement

$$\left\{ \sigma_{[L^{\text{A}}(\cdot)=y]}^{Z} \otimes A_{a}^{\text{INTRO},y} \right\} , \tag{95}$$

for a family of measurements $\left\{ A_{a}^{\text{INTRO},y} \right\}$ acting on the state $|\text{EPR}\rangle_{\overline{V}} \otimes |\text{AUX}\rangle \otimes |\text{ANC}\rangle$ where $|\text{ANC}\rangle = |\text{ANC}_{\ell+1}\rangle$ is unentangled. An analogous argument for Player B's measurements shows that we may additionally assume Player B responds to the question (INTRO, B) using the measurement

$$\left\{ \sigma_{[L^{\text{B}}(\cdot)=y]}^{Z} \otimes B_{a}^{\text{INTRO},y} \right\} , \tag{96}$$

for a family of measurements $\left\{ B_{a}^{\text{INTRO},y} \right\}$ acting on the state $|\text{EPR}\rangle_{\overline{V}} \otimes |\text{AUX}\rangle \otimes |\text{ANC}'\rangle$, where $|\text{ANC}'\rangle$ is unentangled. Summarizing, the strategy $\mathscr{S}''$ uses the state

$$|\theta\rangle = |\text{EPR}\rangle^{\otimes Q} \otimes |\text{AUX}\rangle \otimes |\text{ANC}\rangle \otimes |\text{ANC}'\rangle ,$$

and the measurements given by Equations (95) and (96).

To conclude the proof of the soundness part of the theorem we analyze Item 4 in Figure 10. The test in Item 4 is executed with probability $\Omega(1/\ell)$, so the strategy $\mathscr{S}''$ succeeds with probability at least $1-\delta$ in that test, conditioned on the right types (here we absorb factors of $O(\ell)$ into δ). Using (95) and (96), conditioned on the test being executed the distribution of the part $(y_{\text{A}}, y_{\text{B}})$ of the players' answers in the test is exactly the distribution $\mu_{\mathcal{S},N}$ associated with game $\mathcal{V}_N$. As a result, the strategy which uses the state $|\text{EPR}\rangle_{\overline{V}} \otimes |\text{AUX}\rangle \otimes |\text{ANC}\rangle \otimes |\text{ANC}'\rangle$ and measurements $\{A_{a}^{\text{INTRO},y}\}, \{B_{a}^{\text{INTRO},y}\}$ succeeds with probability at least $1-\delta$ in the game $\mathcal{V}_N$. Thus, $\text{val}^*(\mathcal{V}_N) \geq 1-\delta$, establishing the first item in the theorem.

To show the second item, we observe that local isometries do not change the Schmidt rank of a state. Define $|\psi'\rangle = \phi(|\psi\rangle) \otimes |\text{ANC}\rangle \otimes |\text{ANC}'\rangle$. Since the strategy $(|\psi'\rangle, \{A_{a}^{\text{INTRO},y}\}, \{B_{a}^{\text{INTRO},y}\})$ is δ-close to $(|\theta\rangle, \{A_{a}^{\text{INTRO},y}\}, \{B_{a}^{\text{INTRO},y}\})$ which has value $1-\delta$, the strategy $(|\psi'\rangle, \{A_{a}^{\text{INTRO},x}\}, \{B_{a}^{\text{INTRO},y}\})$ has value at least $1-2\delta$ in the game $\mathcal{V}_N$, and therefore the Schmidt rank of $\phi(|\psi\rangle)$ (and thus of $|\psi\rangle$) must be at least $\mathscr{E}(\mathcal{V}_N, 1-2\delta)$. Here, we use the fact that ancilla states are product states and therefore have Schmidt rank 1.

Moreover, recall that $\phi(|\psi\rangle)$ is δ-close to $|\text{EPR}_2\rangle^{\otimes Q} \otimes |\text{AUX}\rangle$ whose Schmidt coefficients are all at most $2^{-Q/2}$. For any bipartite state $|a\rangle$ with Schmidt rank at most r and $|b\rangle$ whose Schmidt coefficients are all at most β, it follows from the Cauchy-Schwarz inequality that $|\langle a|b\rangle|^2 \le r\beta^2$. Therefore the Schmidt rank of $\phi(|\psi\rangle)$ (and thus of $|\psi\rangle$) is at least

$$(1-\delta)^2 \cdot 2^Q \ge (1-\delta)^2 \cdot 2^{(\lambda N)^\lambda} \, ,$$

where we used that $Q = \Gamma \log q \ge r$ (using the canonical parameter settings of Definition 7.16) and $\delta \ge \|\phi(|\psi\rangle) - |\theta\rangle\|^2 = 2 - 2\Re\langle\theta|\phi(|\psi\rangle)$. Combining the two lower bounds on the Schmidt rank of $|\psi\rangle$ shows the desired lower bound on $\mathscr{E}(\mathcal{V}_n^{\text{INTRO}}, 1 - \varepsilon)$. $\qquad\square$

9 Oracularization

9.1 Overview

In this section we introduce the *oracularization* transformation. At a high level, the oracularization $\mathfrak{G}^{\text{ORAC}}$ of a nonlocal game $\mathfrak{G}$ is intended to implement the following: one player (called the *oracle player*) is supposed to receive questions (x, y) meant for both players in the original game $\mathfrak{G}$, and the other player (called the *isolated player*) only receives either x or y (but not both), along with a label indicating which player in the original game the question is associated with (we refer to such players as the *original* players, e.g. "original A player" and "original B player"). The oracle player is supposed to respond with an answer pair (a, b), and the other player is supposed to respond with an answer c. The oracle and isolated player win the oracularized game if (x, y, a, b) satisfies the predicate of the original game $\mathfrak{G}$ and the isolated player's answer is consistent with the oracle player's answer.

The oracularization step is needed in preparation to the next section, in which we perform answer reduction on the introspection game. To implement answer reduction we need at least one player to be able to compute a proof, in the form of a PCP, that the decider of the original game would have accepted the questions (x, y) and answers (a, b). This requires the player to have access to both questions, and be able to compute both answers.

9.2 Oracularizing normal form verifiers

Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier such that $\mathcal{S}$ is an ℓ-level sampler for some $\ell \geq 0$. We first specify the *typed oracularized verifier* $\mathcal{V}^{\text{ORAC}} = (\mathcal{S}^{\text{ORAC}}, \mathcal{D}^{\text{ORAC}})$ associated with $\mathcal{V}$ as follows.

Sampler. Define the type set $\mathcal{T}^{\text{ORAC}} = \{\text{ORACLE}, \text{A}, \text{B}\}$. (In the remainder of this section we refer to the types in $\mathcal{T}^{\text{ORAC}}$ as *roles*.) Define the type graph G^{ORAC} that is the complete graph on vertex set $\mathcal{T}^{\text{ORAC}}$ (including self-loops on all vertices). Define the $\mathcal{T}^{\text{ORAC}}$-type sampler $\mathcal{S}^{\text{ORAC}}$ as follows. For a fixed index $n \in \mathbb{N}$, let V be the ambient space of $\mathcal{S}$ and L^w for $w \in \{\text{A}, \text{B}\}$ be the pair of CL functions of $\mathcal{S}$ on index n.

Define two $\mathcal{T}^{\text{ORAC}}$-typed families of CL functions $\{L_t^w : V \to V\}$, for $w \in \{\text{A}, \text{B}\}$ and $t \in \mathcal{T}^{\text{ORAC}}$, as follows:

$$L_t^w = \begin{cases} L^t & \text{if } t \in \{\text{A}, \text{B}\}, \\ \text{Id} & \text{if } t = \text{ORACLE}. \end{cases}$$

In other words, if a player gets the type $t \in \{\text{A}, \text{B}\}$, then they get the question that original player t would have received in the game played by $\mathcal{V}_n$. If they get type $t = \text{ORACLE}$, then they get the entire seed z that is used by the sampler $\mathcal{S}$, from which they can compute both $L^{\text{A}}(z)$ and $L^{\text{B}}(z)$, the pair of questions sampled for the players in game $\mathcal{V}_n$.

By definition, the sampler distribution $\mu_{\mathcal{S}^{\text{ORAC}}, n}$ has the following properties.

1. Conditioned on both players receiving the ORACLE role, both players receive z for a uniformly random $z \in V$.

2. Conditioned on both players receiving the isolated player role, the player(s) with role A (respectively, B) receives $L^{\text{A}}(z)$ (respectively, $L^{\text{B}}(z)$) for a uniformly random $z \in V$.

3. Conditioned on player $w \in \{\text{A}, \text{B}\}$ receiving the ORACLE role and player $\overline{w}$ receiving the isolated player role, their question tuple is distributed according to $\big((\text{ORACLE}, z), (v, L^v(z))\big)$ if $w = \text{A}$ and

$\bigl((v, L^v(z)), (\text{ORACLE}, z)\bigr)$ if $w = \text{B}$, where $z \in V$ is uniformly random and v indicates the role of player $\overline{w}$.

Decider. The typed decider $\mathcal{D}^{\text{ORAC}}$ is specified in Figure 12.

Input to decider $\mathcal{D}^{\text{ORAC}}$: $(n, \mathsf{t}_\text{A}, x_\text{A}, \mathsf{t}_\text{B}, x_\text{B}, a_\text{A}, a_\text{B})$. For $w \in \{\text{A}, \text{B}\}$, if $\mathsf{t}_w = \text{ORACLE}$, then parse a_w as a pair $(a_{w,\text{A}}, a_{w,\text{B}})$. Perform the following steps sequentially.

1. **(Game check)**. For all $w \in \{\text{A}, \text{B}\}$, if $\mathsf{t}_w = \text{ORACLE}$, then compute $x_{w,v} = L^v(x_w)$ for $v \in \{\text{A}, \text{B}\}$. If $\mathcal{D}$ rejects $(n, x_{w,\text{A}}, x_{w,\text{B}}, a_{w,\text{A}}, a_{w,\text{B}})$, then reject.

2. **(Consistency checks)**.
   (a) If $\mathsf{t}_\text{A} = \mathsf{t}_\text{B}$ and $a_\text{A} \neq a_\text{B}$, then reject.
   (b) If for some $w \in \{\text{A}, \text{B}\}$, $\mathsf{t}_w = \text{ORACLE}$, $\mathsf{t}_{\overline{w}} \in \{\text{A}, \text{B}\}$, and $a_{w, \mathsf{t}_{\overline{w}}} \neq a_{\overline{w}}$, then reject.

3. Accept if none of the preceding steps rejects.

Figure 12: Specification of the typed decider $\mathcal{D}^{\text{ORAC}}$.

9.3 Completeness and complexity of the oracularized verifier

We determine the complexity of the oracularized verifier and establish the completeness property.

Theorem 9.1 (Completeness and complexity of the oracularized verifier). *Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier. Let $\mathcal{V}^{\text{ORAC}} = (\mathcal{S}^{\text{ORAC}}, \mathcal{D}^{\text{ORAC}})$ be the corresponding typed oracularized verifier. Then the following hold.*

- *(**Completeness**) For all $n \in \mathbb{N}$, if $\mathcal{V}_n$ has a PCC strategy of value 1, then $\mathcal{V}_n^{\text{ORAC}}$ has a symmetric PCC strategy of value 1.*

- *(**Sampler complexity**) The sampler $\mathcal{S}^{\text{ORAC}}$ depends only on $\mathcal{S}$ (and not on $\mathcal{D}$). Moreover, the time and randomness complexities of $\mathcal{S}^{\text{ORAC}}$ satisfy*

$$\text{TIME}_{\mathcal{S}^{\text{ORAC}}}(n) = O\left(\text{TIME}_{\mathcal{S}}(n)\right),$$
$$\text{RAND}_{\mathcal{S}^{\text{ORAC}}}(n) = O\left(\text{RAND}_{\mathcal{S}}(n)\right).$$

Furthermore, if $\mathcal{S}$ is an ℓ-level sampler, then $\mathcal{S}^{\text{ORAC}}$ is a $\max\{\ell, 1\}$-level typed sampler.

- *(**Decider complexity**) The time complexity of $\mathcal{D}^{\text{ORAC}}$ satisfies*

$$\text{TIME}_{\mathcal{D}^{\text{ORAC}}}(n) = \text{poly}\left(\text{TIME}_{\mathcal{D}}(n), \text{RAND}_{\mathcal{S}}(n)\right).$$

- *(**Efficient computability**) There is a Turing machine ComputeOracleVerifier which takes as input $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ and returns $\mathcal{V}^{\text{ORAC}} = (\mathcal{S}^{\text{ORAC}}, \mathcal{D}^{\text{ORAC}})$ in time $\text{poly}(|\mathcal{V}|)$.*

Proof. We analyze the completeness and complexity properties of the typed verifier $\mathcal{V}^{\text{ORAC}}$.

Completeness. For $n \in \mathbb{N}$, let $\mathscr{S} = (|\psi\rangle, A, B)$ be a PCC strategy for $\mathcal{V}_n$ with value 1. Consider the following symmetric strategy $\mathscr{S}^{\mathrm{ORAC}} = (|\psi\rangle, M)$ for $\mathcal{V}_n^{\mathrm{ORAC}}$. Depending on the role received, each player performs the following:

1. Suppose the player receives role $v \in \{A, B\}$ and question x. Then the player performs the measurement that player v would on question x according to strategy $\mathscr{S}$ to obtain outcome a (either $\{A_a^x\}$ or $\{B_a^x\}$, depending on v). The player replies with a.

2. Suppose the player receives role $v = \mathrm{ORACLE}$ and question x. The player first computes $y_w = L^w(x)$ for $w \in \{A, B\}$ where for $w \in \{A, B\}$, L^w is the CL functions of $\mathscr{S}$ corresponding to player w. Then, the player measures using the POVM $\{M_{a_A, a_B}^{\mathrm{ORACLE}, x}\}$ where

$$M_{a_A, a_B}^{\mathrm{ORACLE}, x} = B_{a_B}^{y_B} A_{a_A}^{y_A} . \tag{97}$$

The projectors $A_{a_A}^{y_A}$ and $B_{a_B}^{y_B}$ commute because (y_A, y_B) is distributed according to $\mu_{\mathscr{S}, n}$ (over the choice of x) and $\mathscr{S}$ is a commuting strategy for $\mathcal{V}_n$. Thus $M_{a_A, a_B}^{\mathrm{ORACLE}, x}$ is a projector. The player replies with (a_A, a_B).

The strategy $\mathscr{S}^{\mathrm{ORAC}}$ is symmetric and projective by construction, and consistency follows from the consistency of $\mathscr{S}$. We now argue that the strategy is commuting and has value 1 in the game $\mathcal{V}_n^{\mathrm{ORAC}}$. We consider all possible pairs of roles.

1. (Oracle, isolated) Suppose without loss of generality that player $w = A$ gets the ORACLE role and player $\overline{w} = B$ gets the isolated player B role. Then player w gets question x and player $\overline{w}$ gets question $L^B(x)$, where x is uniformly sampled from V. The oracle player computes $y_v = L^v(x)$ for all $v \in \{A, B\}$. Notice that (y_A, y_B) is distributed according to $\mu_{\mathscr{S}, n}$. The two players return $((a_A, a_B), a_B')$ with probability

$$
\begin{aligned}
\langle\psi| M_{a_A, a_B}^{\mathrm{ORACLE}, x} \otimes B_{a_B'}^{y_B} |\psi\rangle &= \langle\psi| B_{a_B}^{y_B} A_{a_A}^{y_A} \otimes B_{a_B'}^{y_B} |\psi\rangle \\
&= \langle\psi| A_{a_A}^{y_A} \otimes B_{a_B}^{y_B} B_{a_B'}^{y_B} |\psi\rangle \\
&= \delta_{a_B, a_B'} \langle\psi| A_{a_A}^{y_A} \otimes B_{a_B}^{y_B} |\psi\rangle ,
\end{aligned}
$$

where the first equality uses the definition of $M_{a_A, a_B}^{\mathrm{ORACLE}, x}$ from Eq. (97), the second equality uses the consistency of $\mathscr{S}$, and the third equality uses the projectivity of $\mathscr{S}$. Notice that when $a_B = a_B'$, this is exactly the probability of obtaining answers (a_A, a_B) when player A and player B get question pair (y_A, y_B) in the game $\mathcal{V}_n$. Since $\mathscr{S}$ is value-1, the answers satisfy the decision procedure of $\mathcal{V}_n$ with probability 1. Thus the oracle's answers pass the "Game check" of the oracularized decider with certainty, and furthermore the oracle's answers are consistent with the isolated player's answers and thus pass the "Consistency check" with certainty as well.

   Commutativity of $M_{a_A, a_B}^{\mathrm{ORACLE}, x}$ and $B_{a_B}^{y_B}$ follows from the commutativity of $\mathscr{S}$ for the game $\mathcal{V}_n$.

2. (Both oracle) If both players get the ORACLE role, then both players receive the same question $x \in V$. Using a similar analysis as for the previous item, the players return the same answer pair (thus passing the "Consistency check") and pass the "Game check". Both players' measurements commute because they are identical.

3. (Both isolated) Suppose that both players receive the same isolated player role (e.g., they both receive the isolated player role A). They then perform the same measurements, which produce the same outcomes due to the consistency of the strategy $\mathscr{S}$, and thus they pass the "Consistency check". Otherwise, suppose that one player receives the A role and the other player receives the B role. Then the decider $\mathcal{D}^{\mathrm{ORAC}}$ automatically accepts. Furthermore, their measurements commute because their questions are distributed according to $\mu_{\mathcal{S},n}$, and $\mathscr{S}$ is a commuting strategy with respect to $\mu_{\mathcal{S},n}$.

Complexity. It is clear from the definition that $\mathcal{S}^{\mathrm{ORAC}}$ depends only on $\mathcal{S}$. The time and randomness complexities of the sampler $\mathcal{S}^{\mathrm{ORAC}}$ are dominated by those of the sampler $\mathcal{S}$. The complexity of $\mathcal{D}^{\mathrm{ORAC}}$ is dominated by the complexity of $\mathcal{D}$ and performing consistency checks. The sampler $\mathcal{S}^{\mathrm{ORAC}}$ is a $\max\{\ell, 1\}$-level sampler because $\mathcal{S}$ is an ℓ-level sampler and the new CL functions for $\mathsf{t} = \mathrm{ORACLE}$ are 1-level.

Efficient computability. The description of $\mathcal{S}^{\mathrm{ORAC}}$ can be computed, in polynomial time, from the description of $\mathcal{S}$ alone. The description of $\mathcal{D}^{\mathrm{ORAC}}$ can be computed in polynomial time from the descriptions of $\mathcal{S}$ and $\mathcal{D}$. Moreover, in each case the computation amounts to copying the description of $\mathcal{S}$ and $\mathcal{D}$ respectively, and adding constant-sized additional instructions.

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad$ □

9.4 Soundness of the oracularized verifier

Theorem 9.2 (Soundness of the oracularized verifier). *Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier and $\mathcal{V}^{\mathrm{ORAC}} = (\mathcal{S}^{\mathrm{ORAC}}, \mathcal{D}^{\mathrm{ORAC}})$ be the corresponding typed oracularized verifier. Then there exists a function $\delta(\varepsilon) = \mathrm{poly}(\varepsilon)$ such that for all $n \in \mathbb{N}$ the following hold.*

1. *If $\mathrm{val}^*(\mathcal{V}_n^{\mathrm{ORAC}}) > 1 - \varepsilon$, then $\mathrm{val}^*(\mathcal{V}_n) \geq 1 - \delta(\varepsilon)$.*

2. *For all $\varepsilon > 0$, we have that*

$$\mathscr{E}\left(\mathcal{V}_n^{\mathrm{ORAC}}, 1 - \varepsilon\right) \geq \mathscr{E}\left(\mathcal{V}_n, 1 - \delta(\varepsilon)\right)$$

where $\mathscr{E}(\cdot)$ is as in Definition 5.12.

Proof. Fix $n \in \mathbb{N}$. Let $\mathscr{S}^{\mathrm{ORAC}} = \left(|\psi\rangle, A, B\right)$ be a projective strategy for $\mathcal{V}_n^{\mathrm{ORAC}}$ with value $1 - \varepsilon$ for some $0 < \varepsilon \leq 1$. Let $(\mathsf{t}, x) \in \mathcal{T}^{\mathrm{ORAC}} \times V$ be a question to player $w = \mathrm{A}$. In the event that $\mathsf{t} = \mathrm{ORACLE}$ (which occurs with probability $1/3$), let $y_v = L^v(x)$ for each $v \in \{\mathrm{A}, \mathrm{B}\}$. From the consistency check performed by $\mathcal{D}^{\mathrm{ORAC}}$ and item 1 of Fact 5.17, we have that for all $v \in \{\mathrm{A}, \mathrm{B}\}$ and on average over x sampled by $\mathcal{S}^{\mathrm{ORAC}}$,

$$A_{a_v}^{\mathrm{ORACLE}, x} \otimes I_{\mathrm{B}} \approx_\varepsilon I_{\mathrm{A}} \otimes B_{a_v}^{v, y_v} \, . \tag{98}$$

Here, we used that with probability $1/9$ player $w = \mathrm{A}$ gets the ORACLE role and player $\overline{w} = \mathrm{B}$ gets the isolated player v role; conditioned on this, player B gets question y_v.

Using the fact that the POVM elements $\{A_{a_\mathrm{A}, a_\mathrm{B}}^{\mathrm{ORACLE}, x}\}$ are projective and Fact 5.18, we get

$$
\begin{aligned}
A_{a_\mathrm{A}, a_\mathrm{B}}^{\mathrm{ORACLE}, x} \otimes I_{\mathrm{B}} &\approx_\varepsilon A_{a_\mathrm{A}, a_\mathrm{B}}^{\mathrm{ORACLE}, x} \otimes B_{a_\mathrm{B}}^{\mathrm{B}, y_\mathrm{B}} \\
&\approx_\varepsilon A_{a_\mathrm{A}}^{\mathrm{ORACLE}, x} \otimes B_{a_\mathrm{B}}^{\mathrm{B}, y_\mathrm{B}} \\
&\approx_\varepsilon I_{\mathrm{A}} \otimes B_{a_\mathrm{B}}^{\mathrm{B}, y_\mathrm{B}} B_{a_\mathrm{A}}^{\mathrm{A}, y_\mathrm{A}} \\
&\approx_\varepsilon I_{\mathrm{A}} \otimes B_{a_\mathrm{A}}^{\mathrm{A}, y_\mathrm{A}} B_{a_\mathrm{B}}^{\mathrm{B}, y_\mathrm{B}} \, .
\end{aligned}
\tag{99}
$$

Using Item 2a of the consistency check, we have that on average over a random $y = L^A(x) \in V$,

$$A_a^{A,y} \otimes I_B \approx_\varepsilon I_A \otimes B_a^{A,y} . \tag{100}$$

Define, for all $x \in L^A(V)$, measurement operators $\{C_a^x\}_a$ where $C_a^x = A_a^{A,x}$. Similarly, define $D_a^x = B_a^{B,x}$. This defines a strategy $\mathscr{S} = (|\psi\rangle, C, D)$ for the game $\mathcal{V}_n$ that we now argue succeeds with high probability.

Let $x \in V$ be uniformly random. Let $y_A = L^A(x)$ and $y_B = L^B(x)$.

$$
\begin{aligned}
A_{a_A, a_B}^{\text{ORACLE}, x} \otimes I_B &\approx_\varepsilon I_A \otimes B_{a_B}^{B, y_B} B_{a_A}^{A, y_A} \\
&\approx_\varepsilon A_{a_A}^{A, y_A} \otimes B_{a_B}^{B, y_B} \\
&= C_{a_A}^{y_A} \otimes D_{a_B}^{y_B} .
\end{aligned}
$$

The first approximation follows from Equation (99). The second approximation follows from Equation (100) and Fact 5.18 (where we let C_b^y in Fact 5.18 represent $B_{a_B}^{B, y_B}$). The last equality follows from definition of $C_{a_A}^{y_A}$ and $D_{a_B}^{y_B}$. The pair of questions $(y_A, y_B) \in V \times V$ is distributed according to $\mu_{\mathcal{S}, n}$.

The game check part of $\mathcal{D}^{\text{ORAC}}$ succeeds with probability $1 - O(\varepsilon)$, which implies that the answer pair (a_A, a_B) that arises from the measurement $A_{a_A, a_B}^{\text{ORACLE}, x} \otimes I_B$ is accepted by the decider $\mathcal{D}$ on question pair (y_A, y_B) with probability $1 - O(\varepsilon)$. This in turn implies that the strategy $\mathscr{S} = (|\psi\rangle, C, D)$ succeeds with probability $1 - O(\sqrt{\varepsilon})$ in the game $\mathcal{V}_n$. As an additional consequence, the Schmidt rank of $|\psi\rangle$ must be at least $\mathscr{E}(\mathcal{V}_n, 1 - O(\sqrt{\varepsilon}))$. $\qquad\square$

10 Answer Reduction

In this section we show how to transform a normal form verifier $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ into an "answer reduced" normal form verifier $\mathcal{V}^{AR} = (\mathcal{S}^{AR}, \mathcal{D}^{AR})$ such that the values of the associated nonlocal games are directly related, yet the answer-reduced verifier's decision runtime is only polylogarithmic in the answer length of the original verifier (the answer-reduced verifier's sampling runtime remains polynomially related to the sampling runtime of $\mathcal{S}$). The polylogarithmic dependence is achieved by composing a probabilistically checkable proof (PCP) with the oracularized verifier given in Section 9. This step generalizes the answer reduction technique of [NW19, Part V].

Given index $n \in \mathbb{N}$, the answer reduced verifier $\mathcal{V}^{AR}$ simulates the oracularization $\mathcal{V}^{ORAC}$ of $\mathcal{V}$ on index n. To do so, it first samples questions x and y using the oracularized sampler $\mathcal{S}^{ORAC}$ and distributes them to the players, who compute answers a and b. Let us suppose that the first player is assigned the ORACLE role, and parse their question and answer as pairs $x = (x_A, x_B)$ and $a = (a_A, a_B)$, while the second player is an "isolated" player receiving the question x_A and responding with answer b. Instead of executing the decider $\mathcal{D}^{ORAC}$ on the answers (a, b), the verifier $\mathcal{V}^{AR}$ asks the first player to compute a PCP Π of $\mathcal{D}(n, x_A, x_B, a_A, a_B) = 1$, and the second player to compute an encoding g_b of answer b. $\mathcal{V}^{AR}$ then requests randomly chosen locations of the proof Π and the encoding g_b, and executes the PCP verifier on the players' answers. By the soundness of the PCP, $\mathcal{V}^{AR}$ accepts with high probability only if the player's answers satisfy $\mathcal{D}(n, x_A, x_B, a_A, a_B) = 1$ and $b = a_A$.

There are several challenges that arise when implementing answer reduction. One challenge, already encountered in [NW19], is that we need to ensure the PCP Π computed by the first player can be cross-tested against the encoding g_b computed by the second player (who doesn't know the entire structure of the PCP Π). This was handled in [NW19] by using a special type of PCP called a *probabilistically checkable proof of proximity (PCPP)*, which allows one to efficiently check that a *specific* string x is a satisfying assignment to a Boolean formula φ, as opposed to simply checking that φ is satisfiable. In a PCPP, an encoding of the specific string x is provided separately from the proof of satisfiability. The answer reduction scheme of [NW19] was able to use an "off-the-shelf" PCPP in a relatively black-box fashion to handle this.

In our answer reduction scheme, however, there is a further requirement: we need the question distribution of the answer reduced verifier to be conditionally linear. This is necessary to maintain the invariant that the verifier after each step of the compression procedure (introspection, answer reduction, parallel repetition) is a normal form verifier. Unfortunately, simulating the question distributions of off-the-shelf PCPPs with conditionally linear distributions can be quite cumbersome. Instead, we design a bespoke PCP verifier for the protocol whose question distribution is more easily seen to be conditionally linear.

This section is organized as follows. We start with some preliminaries on formulas and encodings in Section 10.1. In Section 10.2 we show how to use the Cook-Levin reduction to reduce the Bounded Halting problem for deciders to a succinct satisfiability problem called Succinct-3SAT. Following this, in Section 10.3, we reduce the Succinct-3SAT instance to an instance of a related problem called Succinct Decoupled 5SAT, which is easier to use in our answer reduction step. Then in Section 10.4 we introduce a PCP for Succinct Decoupled 5SAT. The verifier for the PCP expects a proof consisting of the evaluation tables of low-degree polynomials, including the low-degree encodings of the players' answers a and b. In Section 10.5 we provide the definition of a normal-form verifier $\mathcal{V}^{AR}$ that executes the composition of $\mathcal{V}^{ORAC}$ with the PCP verifier from Section 10.4. In Section 10.6 we show completeness of the construction and analyze its complexity. In Section 10.7 we prove soundness.

10.1 Circuit preliminaries

Recall the definitions pertaining to Turing machines from Section 3.1.

Remark 10.1 (Plugging integers into circuits). *Let $\mathcal{C}$ be a circuit with a single input of length n. Inputs to $\mathcal{C}$ are strings $x \in \{0,1\}^n$. In this section, we will also allow $\mathcal{C}$ to receive inputs $a \in \{0,1,\ldots,2^n-1\}$. In doing so, we use the convention that a number a between 0 and 2^n-1 is interpreted as its n-digit binary encoding $\text{binary}_n(a)$ (recall Definition 3.21) when provided as input to a set of n single-bit wires. In other words, $\mathcal{C}(a) = \mathcal{C}(x)$, where $x = \text{binary}_n(a)$.*

More generally, if the circuit $\mathcal{C}$ has k different inputs of length $n_1,\ldots,n_k$, then we can evaluate it on inputs $a_1 \in \{0,1,\ldots,2^{n_1}-1\},\ldots,a_k \in \{0,1,\ldots,2^{n_k}-1\}$ as follows:

$$\mathcal{C}(a_1,\ldots,a_k) = \mathcal{C}(x_1,\ldots,x_k),$$

where $x_1 = \text{binary}_{n_1}(a_1),\ldots,x_k = \text{binary}_{n_k}(a_k)$.

A 3SAT formula is a Boolean formula in conjunctive normal form in which at most three literals appear in each clause. More precisely, φ is a 3SAT formula on N variables $x_1, x_2, \ldots, x_N$ if it has the form $\bigwedge_{j=1}^{m} C_j$ and each clause C_j is the disjunction of at most three literals, where a literal is either a variable x_i or its negation $\neg x_i$. We use x_i^o to denote the literal x_i if $o = 1$ and $\neg x_i$ if $o = 0$.

Definition 10.2 (Succinct description of 3SAT formulas). Let $N = 2^n$, and let φ be a 3SAT formula on N variables named $x_0,\ldots,x_{N-1}$. Let $\mathcal{C}$ be a Boolean circuit with 3 inputs of length n and three single-bit inputs. Then $\mathcal{C}$ is a *succinct description of φ* if for each $i_1, i_2, i_3 \in \{0,1,\ldots,N-1\}$ and $o_1, o_2, o_3 \in \{0,1\}$,

$$\mathcal{C}(i_1,i_2,i_3,o_1,o_2,o_3) = 1 \tag{101}$$

if and only if $x_{i_1}^{o_1} \vee x_{i_2}^{o_2} \vee x_{i_3}^{o_3}$ is a clause in φ. In Equation (101), we use the notation from Remark 10.1.

Definition 10.3 (Succinct-3SAT problem). The *Succinct-3SAT* problem is the language containing encodings of circuits $\mathcal{C}$ in which $\mathcal{C}$ is a succinct description of a satisfiable 3SAT formula φ.

10.2 A Cook-Levin theorem for bounded deciders

Definition 10.4 (Bounded Halting problem). The *k-input Bounded Halting* problem is the language BH_k containing the set of tuples $(\alpha, T, z_1, \ldots, z_k)$ where α is the description of a k-input Turing machine, $T \in \mathbb{N}$, $z_1, \ldots, z_k \in \{0,1\}^*$, and $\mathcal{M}_\alpha$ accepts input $(z_1,\ldots,z_k)$ in at most T time steps.

We begin by defining natural encodings of a decider's tape alphabet and set of states.

Definition 10.5 (Decoder encodings). Let $\mathcal{D}$ be a decider with tape alphabet $\Gamma = \{0,1,\sqcup\}$ and set of states K. We will write $\text{enc}_\Gamma : \Gamma \cup \{\square\} \to \{0,1\}^2$ for the function which encodes the elements of Γ, and a special "$\square$" symbol described below, as length-two binary strings in the following manner:

$$\text{enc}_\Gamma(0) = 00, \quad \text{enc}_\Gamma(1) = 01,$$
$$\text{enc}_\Gamma(\sqcup) = 10, \quad \text{enc}_\Gamma(\square) = 11.$$

In addition, we write $\text{enc}_K : K \to \{0,1\}^\kappa$ for some arbitrary fixed κ-bit encoding of the elements of K, where $\kappa = \lceil \log(|K|) \rceil$.

Now, we give the main result of this section. It states that any decider $\mathcal{D}$ can be converted into a circuit $\mathcal{C}$ which succinctly represents a 3SAT formula φ_{3SAT} that carries out the time T computation of $\mathcal{D}$. In addition, $\mathcal{C}$ is extremely small—size $\operatorname{poly}\log(T)$ rather than $\operatorname{poly}(T)$.

Proposition 10.6 (Succinct representation of deciders). *There is an algorithm with the following properties. Let $\mathcal{D}$ be a decider, let n, T, Q, and λ be integers with $Q \leq T$ and $|\mathcal{D}| \leq \lambda$, and let x and y be strings of length at most Q. Then on input $(\mathcal{D}, n, T, Q, \lambda, x, y)$, the algorithm outputs a circuit $\mathcal{C}$ on $3m + 3$ inputs which succinctly describes a 3SAT formula φ_{3SAT} on $M = 2^m$ variables. Furthermore, φ_{3SAT} has the following property:*

- *For all $a, b \in \{0,1\}^{2T}$, there exists a $c \in \{0,1\}^{M-4T}$ such that $w = (a, b, c)$ satisfies φ_{3SAT} if and only if there exist $a_{\text{prefix}}, b_{\text{prefix}} \in \{0,1\}^*$ of lengths $\ell_a, \ell_b \leq T$, respectively, such that*

$$a = \operatorname{enc}_\Gamma(a_{\text{prefix}}, \sqcup^{T-\ell_a}) \quad and \quad b = \operatorname{enc}_\Gamma(b_{\text{prefix}}, \sqcup^{T-\ell_b})$$

and $\mathcal{D}$ accepts $(n, x, y, a_{\text{prefix}}, b_{\text{prefix}})$ in time T.

Finally, the following statements hold:

1. *The parameter m controlling the number of inputs to the circuit depends only on T and λ, and $m(T, \lambda) = O(\log(T) + \log(\lambda))$,*

2. *$\mathcal{C}$ has at most $s(n, T, Q, \lambda) = \operatorname{poly}(\log(n), \log(T), Q, \lambda)$ gates,*

3. *The algorithm runs in time $\operatorname{poly}(\log(n), \log(T), Q, \lambda)$,*

4. *Furthermore, explicit values for $m(T, \lambda)$ and $s(n, T, Q, \lambda)$ can be computed in time polynomial in $n, \log(T), Q, \lambda$.*

Proposition 10.6 is essentially the standard fact that Succinct-3SAT is an NEXP-complete language, i.e. that every nondeterministic computation which takes time 2^n can be represented as a Succinct-3SAT instance of size only $\operatorname{poly}(n)$. However, it has several peculiarities which requires us to prove it from scratch rather than simply appealing to the NEXP-completeness of Succinct-3SAT. First, we require that the coordinates of a and b embed into w not randomly but as its lexicographically first coordinates (for reasons that are explained below in Section 10.3). Second, we need explicit bounds on how quantities such as the size of $\mathcal{C}$ relate to quantities such as λ, an upper bound on the description length of $\mathcal{D}$ in bits.

To prove Proposition 10.6, we follow the standard proof that Succinct-3SAT is NEXP-complete as presented in [Pap94]. This proof observes that the Cook-Levin reduction, which is used to show that 3SAT is NP-complete, produces a 3SAT instance whose clauses follow such a simple pattern that they can be described succinctly using an exponentially-smaller circuit. One key difference in our proof is that we will apply the Cook-Levin reduction directly to the 5-input Turing machine $\mathcal{D}$, which by Section 3.1 has 7 tapes; traditional proofs such as the one in [Pap94] would first convert $\mathcal{D}$ to a single-tape Turing machine $\mathcal{D}_{\text{single}}$, and then apply the Cook-Levin reduction for single-tape Turing machines to $\mathcal{D}_{\text{single}}$. Though this adds notational overhead to our proof, it allows us to more easily track of which variables in φ_{3SAT} correspond to the strings a and b (see Proposition 10.6 to see what these refer to).

Proof of Proposition 10.6. The Cook-Levin reduction considers the *execution tableau* of $\mathcal{D}$ when run for time T. The execution tableau contains, for each time $t \in \{1, \ldots, T\}$, variables describing the state of $\mathcal{D}$ and the contents of each of its tape cells at time t. More formally, it consists of the following three sets of variables.

1. For each time $t \in \{1, \ldots, T\}$, tape $i \in \{1, \ldots, 7\}$, and tape position $j \in \{0, 1, \ldots, T+1\}$, the tableau contains two Boolean-valued variables

$$c_{t,i,j} = (c_{t,i,j,1}, c_{t,i,j,2}) \in \{0,1\}^2$$

which are supposed to correspond to the contents of the j-th tape cell on tape i at time t according to $\mathrm{enc}_\Gamma(\cdot)$. The variables with $j \in \{0, T+1\}$ do not correspond to any cell on the tape; rather, the $j = 0$ variables correspond to the left-boundary of the tape, and the $j = T+1$ variables correspond to the right-boundary of the first T cells on the tape. These are expected to always contain the special boundary symbol "$\square$", i.e. $c_{t,i,j}$ should be equal to $\mathrm{enc}_\Gamma(\square)$ whenever $j \in \{0, T+1\}$. As we will see below, it is convenient to define these so that for each $t \in \{1, \ldots, T\}$, $i \in \{1, \ldots, 7\}$, and $j \in \{1, \ldots, T\}$, the variable $c_{t,i,j}$ also has a variable to its left $c_{t,i,j-1}$ and to its right $c_{t,i,j+1}$.

2. For each time $t \in \{1, \ldots, T\}$, tape $i \in \{1, \ldots, 7\}$, and tape position $j \in \{0, 1, \ldots, T+1\}$, the tableau contains Boolean-valued variables $h_{t,i,j} \in \{0,1\}$ which are supposed to indicate whether the i-th tape head is in cell j at time t. For the boundary cells $j \in \{0, T+1\}$, we expect that $h_{t,i,j} = 0$ for all $t \in \{1, \ldots, T\}$ and $i \in \{1, \ldots, 7\}$.

3. For each time $t \in \{1, \ldots, T\}$, the tableau contains κ Boolean-valued variables

$$s_t = (s_{t,1}, \ldots, s_{t,\kappa}) \in \{0,1\}^\kappa$$

which are supposed to correspond to the state of $\mathcal{D}$ at time t according to $\mathrm{enc}_K(\cdot)$.

Finally, we let $\mathcal{V}$ denote the set of all of these variables. In other words,

$$\mathcal{V} = \{c_{t,i,j,k}\}_{t,i,j,k} \cup \{h_{t,i,j}\}_{t,i,j} \cup \{s_{t,k}\}_{t,k}.$$

In total, the number of variables in the execution tableau is given by

$$|\mathcal{V}| = O\big(T^2 + T \cdot \log(|K|)\big) = O\big(T^2 + T \cdot \log(|\mathcal{D}|)\big). \tag{102}$$

The first term in Equation (102) corresponds to the tape cell encodings $c_{t,i,j}$ and $h_{t,i,j}$, and the second term corresponds to the Turing machine state encodings s_t. The second equality uses the fact that $|K| \leq |\mathcal{D}|$.

As stated above, we expect the variables in $\mathcal{V}$ to correspond to some time-T execution of the decider $\mathcal{D}$. However, in general these are just arbitrary $\{0,1\}$-valued variables. We now describe a set of constraints placed on these variables which, if satisfied, ensure they *do* indeed correspond to some time-T execution of $\mathcal{D}$. These constraints will be split into two categories: (i) the constraints corresponding to the boundary, which ensure that the $t = 1$ variables are initialized to a valid starting configuration and the $j \in \{0, T+1\}$ variables are set according to Items 1 and 2, and (ii) the constraints corresponding to the execution of $\mathcal{D}$, which ensure that the variables at each time $(t+1)$ follow from the variables at time t according to the computation of $\mathcal{D}$. We start with the boundary constraints, which are simple enough to be described with a 3SAT formula.

Definition 10.7. The *boundary formula* $\varphi_{\mathrm{Boundary}}$ is the 3SAT formula on the variables $\mathcal{V}$ described as follows. Let $o = \mathrm{enc}_K(\mathsf{start}) \in \{0,1\}^\kappa$, where $\mathsf{start} \in K$ is the start state of $\mathcal{D}$. For the $t = 1$ boundary,

$\varphi_{\text{Boundary}}$ contains the following set of clauses.

Indices	Clauses	
$i \in \{1,\ldots,7\}$	$h_{1,i,1}$	(103)
$i \in \{1,\ldots,7\}$, $j \neq 1$	$\neg h_{1,i,j}$	(104)
$k \in \{1,\ldots,\kappa\}$	$s_{1,k}^{o_k}$	(105)
$i \in \{1,\ldots,7\}$, $j \in \{1,\ldots,T\}$	$\neg c_{1,i,j,1} \vee \neg c_{1,i,j,2}$	(106)
$i \in \{1,\ldots,5\}$, $j < j' \in \{1,\ldots,T\}$	$\neg c_{1,i,j,1} \vee c_{1,i,j',1}$	(107)
$i \in \{6,7\}$, $j \in \{1,\ldots,T\}$	$c_{1,i,j,1}$ and $\neg c_{1,i,j,2}$	(108)

This is meant to be read as follows: for each row, the "Indices" column specifies the range of the indices that the clauses in the "Clauses" column are quantified over. For example, row (103) specifies that for all $i \in \{1,\ldots,7\}$, $\varphi_{\text{Boundary}}$ contains the clause $h_{1,i,1}$. For the $j \in \{0, T+1\}$ boundary, $\varphi_{\text{Boundary}}$ contains the following set of clauses.

Indices	Clauses	
$t \in \{1,\ldots,T\}$, $i \in \{1,\ldots,7\}$, $j \in \{0, T+1\}$, $k \in \{1,2\}$	$c_{t,i,j,k}$	(109)

Rows (103) and (104) ensure that at time $t = 1$, each tape has exactly one tape head, and it is located on cell $j = 1$. Row (105) ensures that at time $t = 1$, the state is given by the start state start. (Recall the notation x_i^o to denote the literal x_i if $o = 1$ and $\neg x_i$ if $o = 0$.) For the remaining rows, we recall that under the encoding of the tape alphabet, $\text{enc}_\Gamma(\sqcup) = 10$ and $\text{enc}_\Gamma(\square) = 11$. As a result, (i) row (109) ensures that for all times and tapes, the cells $j \in \{0, T+1\}$ contain the $\square$ symbol, (ii) row (106) ensures that for time $t = 1$, no cell $j \notin \{0, T+1\}$ contains the $\square$ symbol, and (iii) row (108) ensures that for time $t = 1$ and tapes 6 and 7, all cells $j \in \{1,\ldots,T\}$ contain the $\sqcup$ symbol. Finally, row (107) says that for tapes $i \in \{1,\ldots,5\}$, if cell j contains $\sqcup$, then every cell $j' > j$ must contain $\sqcup$ as well. This means that the five strings encoded by $c_{1,1},\ldots,c_{1,5}$ each consist of a string of 0's and 1s followed by a string of $\sqcup$'s. In short, suppose we write n, x, y, a, and b for the prefixes of these strings with no $\sqcup$'s. If $\varphi_{\text{Boundary}}$ is satisfied, then the execution tableau correctly encodes that the tapes of $\mathcal{D}$ contain inputs n, x, y, a, and b at time $t = 1$.

Next, we describe the execution constraints. These are more complicated than the boundary constraints, and so we will begin by describing them in terms of a general Boolean circuit known as the *local check circuit*. For any time $t \in \{1,\ldots,T-1\}$ and tape positions $j_1,\ldots,j_7 \in \{1,\ldots,T\}$, the local check circuit can check that the execution tableau properly encodes these tape positions at time $t+1$ by looking only at the encodings of these tape positions and their neighbors (i.e. the tape positions $j_i \pm 1$ for each $i \in \{1,\ldots,7\}$) at time t.

Definition 10.8. In this definition, we will define the *local check circuit* $\mathcal{C}_{\text{Check}}$. It has the following inputs.

- For each $i \in \{1,\ldots,7\}$, it has the eight inputs

$c_{\text{Check},0,i,-1}$	$c_{\text{Check},0,i,0}$	$c_{\text{Check},0,i,1}$		$h_{\text{Check},0,i,-1}$	$h_{\text{Check},0,i,0}$	$h_{\text{Check},0,i,1}$
	$c_{\text{Check},1,i,0}$				$h_{\text{Check},1,i,0}$	

(110)

where the c-inputs are in $\{0,1\}^2$ and the h-inputs are in $\{0,1\}$.

- It has two inputs $s_{\text{Check},0}, s_{\text{Check},1} \in \{0,1\}^\kappa$.

In addition, for each time $t \in \{1, \ldots, T-1\}$ and tape positions $j_1, \ldots, j_7 \in \{1, \ldots, T\}$, we will define a circuit $\mathcal{C}_{\text{Check},t,j_1,\ldots,j_7}$ by associating the inputs of $\mathcal{C}_{\text{Check}}$ with certain variables in $\mathcal{V}$. We do this by associating the following eight inputs from $\mathcal{V}$ with the corresponding variables in Equation (110):

$$c_{t,i,j_i-1} \quad c_{t,i,j_i} \quad c_{t,i,j_i+1} \qquad h_{t,i,j_i-1} \quad h_{t,i,j_i} \quad h_{t,i,j_i+1}$$
$$c_{t+1,i,j_i} \qquad\qquad\qquad h_{t+1,i,j_i}$$

as well as by associating s_t and s_{t+1} from $\mathcal{V}$ with $s_{\text{Check},0}$ and $s_{\text{Check},1}$, respectively.

To define the behavior of $\mathcal{C}_{\text{Check}}$, it will be more convenient to define the behavior of the circuits $\mathcal{C}_{\text{Check},t,j_1,\ldots,j_7}$ for all values of $t, j_1, \ldots, j_7$. However, it will be clear that each of these is in fact the same circuit applied to different inputs, and hence this will define $\mathcal{C}_{\text{Check}}$ as well. Now, the circuit $\mathcal{C}_{\text{Check},t,j_1,\ldots,j_7}$ acts as follows.

- Suppose for each $i \in \{1, \ldots, 7\}$, exactly one of the tape positions $j_i - 1$, j_i, and $j_i + 1$ at time t contains a tape head. First, $\mathcal{C}_{\text{Check},t,j_1,\ldots,j_7}$ computes the transition function of the Turing machine applied to the contents of these 7 tape positions and the state of $\mathcal{D}$ at time t, which produces the state of $\mathcal{D}$ and contents of these tape positions at time $t + 1$, as well as directions to move the 7 tape heads in. Then $\mathcal{C}_{\text{Check},t,j_1,\ldots,j_7}$ checks that the variables for the tape positions $j_1, \ldots, j_7$ and the state of $\mathcal{D}$ at time $t + 1$ match what they should be. In addition, if a tape head is marked as moving into a tape cell containing a $\square$ symbol, that tape head remains in place instead.

- Suppose for each $i \in \{1, \ldots, 7\}$, none of the tape positions $j_i - 1$, j_i, or $j_i + 1$ contain a tape head at time t. Then $\mathcal{C}_{\text{Check},t,j_1,\ldots,j_7}$ checks that the variables for the tape positions $j_1, \ldots, j_7$ at time $t + 1$ match the variables at time t.

- Otherwise, $\mathcal{C}_{\text{Check},t,j_1,\ldots,j_7}$ accepts.

This defines $\mathcal{C}_{\text{Check},t,j_1,\ldots,j_7}$, and therefore $\mathcal{C}_{\text{Check}}$.

Definition 10.8 shows the utility of introducing the boundary variables $c_{t,i,0}$ and $c_{t,i,T+1}$. The circuit $\mathcal{C}_{\text{Check}}$ checks the contents of a cell $c_{t,i,j}$ at time $t + 1$ by looking at the cell and its neighbors $c_{t,i,j-1}, c_{t,i,j+1}$ at time t. However, those cells with $j \in \{1, T\}$ only have either a left neighbor or a right neighbor, and so without the boundary variables we'd have to introduce two other local check circuits designed just for these boundary cases. The boundary variables then allow us to use the same local check circuit for all cells.

Proposition 10.9. *The circuit $\mathcal{C}_{\text{Check}}$ has size at most $\text{poly}(|\mathcal{D}|)$ and can be computed in time $\text{poly}(|\mathcal{D}|)$.*

Proof. The circuit has $7 \cdot 12 + 2 \cdot \kappa = 84 + 2\kappa$ total Boolean inputs, giving a total of $2^{84} \cdot 4^\kappa = O(|K|^2)$ possible input strings. For each possible fixed input string, $\mathcal{C}_{\text{Check}}$ will check if the actual input is equal to the fixed input, which takes $O(\kappa)$ gates, and then it will accept if the fixed input should be accepting. This takes $O(|K|^2 \cdot \kappa)$ gates. Computing $\mathcal{C}_{\text{Check}}$ requires looping over all possible input strings and checking which ones are accepting or rejecting. This requires computing the transition function of $\mathcal{D}$, a task which takes time $\text{poly}(|\mathcal{D}|)$. The proposition follows by noting that $|K| \leq |\mathcal{D}|$. $\square$

Proposition 10.10. *Suppose that the execution tableau satisfies $\varphi_{\text{Boundary}}$ and the circuit $\mathcal{C}_{\text{Check},t,j_1,\ldots,j_7}$, for each $t \in \{1, \ldots, T-1\}$ and $j_1, \ldots, j_7 \in \{1, \ldots, T\}$. Then the execution tableau correctly encodes the execution of $\mathcal{D}$ on input (n, x, y, a, b) when run for time T.*

Proof. To show this, we will show for each time $t \in \{1, \ldots T\}$ that the variables in the execution tableau corresponding to time t correctly encode the state of $\mathcal{D}$ and the contents of the seven tapes at time t. The proof is by induction on t. The base case of $t = 1$ follows from the tableau satisfying $\varphi_{\text{Boundary}}$.

Next we perform the induction step. Assuming the statement holds for time $t \in \{1, \ldots, T-1\}$, we will show it holds for time $t + 1$ as well. Let $i^* \in \{1, \ldots, 7\}$ be a tape, and consider a tape position $j_{i^*} \in \{1, \ldots, T\}$. We will show that the variables correctly encode the contents of this tape position at time $t + 1$. Suppose one of the tape positions $j_{i^*} - 1$, j_{i^*}, or $j_{i^*} + 1$ at time t has a tape head. For each of the other tapes $i \neq i^*$, we select a tape position j_i such that either $j_i - 1$, j_i, or $j_i + 1$ has a tape head at time t. (These positions are guaranteed to exist since each tape has exactly one tape head.)

By the induction hypothesis, for each tape $i \in \{1, \ldots, 7\}$ the variables corresponding to tape cells $j_i - 1$, j_i, and $j_i + 1$ correctly encode the contents of these cells at time t. By assumption, $\mathcal{C}_{\text{Check},t,j_1,\ldots,j_7}$ evaluates to 1. In this case, it calculates the transition function of $\mathcal{D}$ to compute the contents of the tape cells $j_1, \ldots, j_7$ at time $t + 1$ and checks that the corresponding variables encode these contents. As a result, the tape position j_{i^*} on tape i^* is correctly encoded. In addition, it computes the state of $\mathcal{D}$ at time $t + 1$ and checks that the corresponding variables encode this state. This completes the induction step. The case when none of the tape positions $j_{i^*} - 1$, j_{i^*}, and $j_{i^*} + 1$ at time t contain a tape head follows similarly. Finally, the variables for all tape positions $j \in \{0, T + 1\}$ are correctly encoded due to $\varphi_{\text{Boundary}}$ being satisfied. $\qquad\square$

Proposition 10.10 gives a set of constraints that ensure the execution tableau properly encodes the execution of $\mathcal{D}$. Our next step will be to convert these constraints into a single 3SAT formula. This entails transforming each circuit $\mathcal{C}_{\text{Check},t,i_1,\ldots,i_7}$ into a 3SAT formula. We do so using the following reduction.

Proposition 10.11 (Circuit-to-3SAT)**.** *There is an algorithm which, on input a size-r circuit $\mathcal{C}$ on variables $x \in \{0,1\}^n$, runs in time $\text{poly}(r)$ and outputs a 3SAT formula φ on variables $x \in \{0,1\}^n$ and $y \in \{0,1\}^r$ with $O(r)$ clauses such that for all x, $\mathcal{C}(x) = 1$ if and only if there exists a y such that x and y satisfy φ.*

Proof. This is the textbook circuit-to-3SAT reduction. For each gate $i \in \{1, \ldots, r\}$, the algorithm introduces a variable $y_i \in \{0, 1\}$, so that the total collection of variables is $\mathcal{V}_{\text{total}} = \{x_i\}_{i \in \{1,\ldots,n\}} \cup \{y_i\}_{i \in \{1,\ldots,r\}}$. Consider gate $i \in \{1, \ldots, r\}$, and let $z_1, z_2 \in \mathcal{V}_{\text{total}}$ be the pair of variables feeding into it. If gate i is an AND gate, then φ includes the constraints

$$(\neg z_1 \vee \neg z_2 \vee y_i), \quad (z_1 \vee z_2 \vee \neg y_i), \quad (z_1 \vee \neg z_2 \vee \neg y_i), \quad (\neg z_1 \vee z_2 \vee \neg y_i). \tag{111}$$

These constraints are satisfied if and only if $y_i = z_1 \wedge z_2$. The case of gate i being an OR gate follows similarly. Finally, φ includes the constraint (y_{i^*}), where $i^* \in \{1, \ldots, r\}$ is the output gate. It follows that x, y satisfy φ if and only if $\mathcal{C}(x) = 1$ and for each $i \in \{1, \ldots, r\}$, y_i is the value computed by gate i in circuit $\mathcal{C}$ on input x. In total, φ has $4r + 1$ clauses and is computable in time $\text{poly}(r)$. $\qquad\square$

We now apply the algorithm from Proposition 10.11 to $\mathcal{C}_{\text{Check}}$, which by Proposition 10.9 has size $r \leq \text{poly}(|\mathcal{D}|)$. This produces a 3SAT formula φ_{Check} on the variables in $\mathcal{C}_{\text{Check}}$ plus auxiliary variables a_k for $k \in \{1, \ldots, r\}$ added by the reduction. Now, for each $t \in \{1, \ldots, T-1\}$ and $j_1, \ldots, j_7 \in \{1, \ldots, T\}$, we define a 3SAT formula $\varphi_{\text{Check},t,j_1,\ldots,j_7}$ analogously to $\mathcal{C}_{\text{Check},t,j_1,\ldots,j_7}$. We begin by associating those variables in φ_{Check} which come from $\mathcal{C}_{\text{Check}}$'s inputs with the variables in $\mathcal{V}$ as in Definition 10.8. Next, for each $k \in \{1, \ldots, r\}$, we introduce a new variable $a_{t,j_1,\ldots,j_7,k} \in \{0, 1\}$ and associate it with the variable a_k. This defines $\varphi_{\text{Check},t,j_1,\ldots,j_7}$. In summary, the final 3SAT instance produced by the Cook-Levin reduction is

$$\varphi := \varphi_{\text{Boundary}} \wedge \Big(\bigwedge_{t,j_1,\ldots,j_7} \varphi_{\text{Check},t,j_1,\ldots,j_7} \Big). \tag{112}$$

By Proposition 10.11, the execution tableau properly encodes the execution of $\mathcal{D}$ if and only if there exists a setting to the auxiliary variables satisfying the 3SAT formula φ. In total, φ contains

$$|V| + O(T^8) \cdot r \leq O(T^2 + T \cdot \log(|\mathcal{D}|) + T^8 \cdot \text{poly}(|\mathcal{D}|)) = \text{poly}(T, |\mathcal{D}|) \qquad (113)$$

variables. The second term on the left-hand side of Equation (113) corresponds to the auxiliary variables in each of the $O(T^8)$ copies of φ_{Check}. The inequality follows by Equation (102) and the fact that $r \leq \text{poly}(|\mathcal{D}|)$.

Our next is to represent the 3SAT formula φ succinctly. To do this, we will provide a circuit $\mathcal{C}$ which succinctly describes a 3SAT formula φ_{3SAT} which, while not literally equal to φ, will be isomorphic to it. This means that, for example, φ_{3SAT} may not even have the same number of variables as φ, but any variable in φ will correspond in a clear and direct manner to a variable in φ_{3SAT}, and any remaining variables in φ_{3SAT} do not appear in any clauses. This circuit is constructed as follows.

Definition 10.12. In this definition we construct the circuit $\mathcal{C}$. It has three inputs z_1, z_2, z_3 of length m, which we specify below in Equation (114), and three inputs $o_1, o_2, o_3 \in \{0,1\}$. For each $v \in \{1,2,3\}$, each z_v is supposed to specify a variable in φ according to a format we will now specify. If z_v is not properly formatted, then it does not correspond to a variable in φ; if any of z_1, z_2, z_3 is not properly formatted, then $\mathcal{C}$ automatically outputs 0. Below, we will often write substrings of the z_v's as though they are integers from some specified range, i.e. $a \in \{b, \ldots, c\}$. This means that a is represented as a binary string of length $\lceil \log(c+1) \rceil$, which is to be interpreted as the binary encoding of an integer between b and c.

The input z_v is formatted as a string $(\omega, \alpha, \beta_1, \beta_2, \beta_3, \beta_4)$. The first substring ω has length $m - (|\alpha| + |\beta_1| + \cdots + |\beta_4|)$ bits and is formatted to be the all-zeroes string. Its purpose is to pad the inputs to have the length m we specify below. Next, α is formatted as an integer $\alpha \in \{1,2,3,4\}$. The variable encoded by z_v is specified by β_α, and the other three β's should be the all-zeros string. We now specify the encoding of β_α, conditioned on the value of α.

1. β_1 is formatted as (t, i, j, k), where

$$t \in \{1, \ldots, T\}, \quad i \in \{1, \ldots, 7\}, \quad j \in \{0, 1, \ldots, T+1\}, \quad k \in \{1, 2\}.$$

This corresponds to the variable $c_{t,i,j,k}$.

2. β_2 is formatted as (t, i, j), where

$$t \in \{1, \ldots, T\}, \quad i \in \{1, \ldots, 7\}, \quad j \in \{0, 1, \ldots, T+1\}.$$

This corresponds to the variable $h_{t,i,j}$.

3. β_3 is formatted as (t, k), where

$$t \in \{1, \ldots, T\}, \quad k \in \{1, \ldots, \kappa\}.$$

This corresponds to the variable $s_{t,k}$.

4. β_4 is formatted as $(t, j_1, \ldots, j_7, k)$, where

$$t \in \{1, \ldots, T-1\}, \quad j_1, \ldots, j_7 \in \{1, 2, \ldots, T\}, \quad k \in \{1, \ldots, r\}.$$

This corresponds to the variable $a_{t, j_1, \ldots, j_7, k}$.

116

In total, the length of these substrings is

$$|\alpha| + |\beta_1| + \cdots + |\beta_4| = O(\log(T) + \log(\kappa) + \log(r)) = O(\log(T) + \log(|\mathcal{D}|)) .$$

As a result, because $|\mathcal{D}| \leq \lambda$, the length of the "padding" ω can be chosen so that each z_i has length

$$m = O(\log(T) + \log(\lambda)) . \tag{114}$$

Checking that each z_ν is properly formatted can be done by checking that certain substrings of z_1, z_2, and z_3 encode integers which fall within specified ranges. This can be done using $O(m)$ gates.

Having specified the inputs, we can now specify the execution of the circuit, and we may assume that z_1, z_2, z_3 are properly formatted. Implementing the clauses from $\varphi_{\text{Boundary}}$ is simple; we specify how to implement the clause from Equation (103).

- Suppose for input z_1, $\alpha = 2$. Then β_2 can be parsed as (t, i, j). The circuit accepts if $j = 1$ and $o_1 = 1$, regardless of z_1 and z_2. This ensures that φ_{3SAT} includes $x_{z_1} \vee x_{z_2}^{o_2} \vee x_{z_3}^{o_3}$ for any z_2, z_3, o_2, o_3, which is equivalent to including the arity-one clause x_{z_1}.

This can be implemented with $O(m)$ gates. Similar arguments can be used to implement the clauses from Equations (104)-(109) using $O(m)$ gates apiece.

Implementing the clauses from the formulas $\varphi_{\text{Check},t,j_1,\ldots,j_7}$ is more challenging. From Equation (111), we can see that any constraint in this formula always involves a variable of the form $a_{t,j_1,\ldots,j_7,k}$ for some $k \in \{1, \ldots, r\}$.

1. First check if one of its inputs z_1, z_2, z_3 corresponds to such a variable. This can be done with $O(1)$ gates simply by checking if for any of the z_i's, $a = 4$. If so, this specifies the values of $t, j_1, \ldots, j_7$.

2. Each variable in $\varphi_{\text{Check},t,j_1,\ldots,j_7}$ is associated with a variable in φ_{Check}. The circuit checks if all the z_i's are contained in $\varphi_{\text{Check},t,j_1,\ldots,j_7}$ and then it computes which variable in φ_{Check} they are associated with. We include below the example of checking whether z_1 is associated with the variable $c_{\text{Check},1,1,0,0}$.

   - The circuit $\mathcal{C}$ first computes $t + 1$, which takes $O(m)$ gates. It then looks at z_1 and checks if $\alpha = 1$. If so, then $\beta_1 = (t', i', j', k')$, and so it tests the equalities $t' = t + 1$, $i' = 1$, $j' = j_1$, and $k' = 0$, each of which takes $O(m)$ gates to test. If so, then z_1 is associated with $c_{\text{Check},1,1,0,0}$.

   There are $\text{poly}(|\mathcal{D}|)$ variables in φ_{Check} and, for each variable z_i, it takes $O(m)$ gates to determine whether z_i is associated with this variable. As a result, because $|\mathcal{D}| \leq \lambda$, this takes $\text{poly}(\lambda, \log(T))$ gates to compute.

3. Whether z_1, z_2, and z_3 share a clause in $\varphi_{\text{Check},t,j_1,\ldots,j_7}$ depends only on which variables in φ_{Check} they are associated with. As a result, after having computed these variables, the algorithm can hard-code whether the circuit $\mathcal{C}$ should accept.

This completes the description of $\mathcal{C}$. In total, it contains $\text{poly}(\lambda, \log(T))$ gates. Computing $\mathcal{C}$ first requires computing φ_{Check}, which takes time $\text{poly}(|\mathcal{D}|) \leq \text{poly}(\lambda)$ by Propositions 10.9 and 10.11. After that, the steps outlined above for construction $\mathcal{C}$ take time $\text{poly}(\lambda, \log(T))$.

The circuit $\mathcal{C}$ succinctly describes φ in the loose sense described above. Recall that φ accepts if and only if it encodes the execution of $\mathcal{D}$ up to time T. We now modify $\mathcal{C}$ to (i) hard code n, x, and y onto $\mathcal{D}$'s input tapes, and (ii) ensure that $\mathcal{D}$ accepts. We demonstrate how to do so by hard coding n as an example.

- The input n is described by some string ν of length $\ell = O(\log(n))$. We would like to hard-code the string $(\nu, \sqcup^{T-\ell})$ into the first tape of $\mathcal{D}$. To do this, we first check if z_1 corresponds to the variable $c_{t,i,j,k}$ for some values of t, i, j, k. Then, we check if $t = 1$, the time where the inputs appear on the tapes, and $i = 1$, corresponding to the first tape. If so, we branch on whether $j \leq \ell$. If it is, then if $k = 1$ the circuit accepts if $o_1 = 0$, and if $k = 2$ the circuit accepts if $o_1 = \nu_j$. Otherwise, if $j > \ell$, then if $k = 1$ the circuit accepts if $o_1 = 1$, and if $k = 2$ the circuit accepts if $o_1 = 0$. This can be done with $\text{poly}(\log(n), \log(T))$ gates.

This modifies φ so that it only accepts if n is written on its first tape at input. We can similarly hard-code x and y onto the second and third input tapes and hard-code the accepting state as the final state of $\mathcal{D}$. In total, this takes $\text{poly}(\log(n), |x|, |y|, \log(T), \lambda)$, which is $\text{poly}(\log(n), Q, \log(T), \lambda)$ because x and y have length at most Q.

It remains to ensure that the variables corresponding to the 4th and 5th at time $t = 1$ are the lexicographically-first named variables in φ. However, this is simple and can be done using $\text{poly}(\log(n), Q, \log(T), \lambda)$ gates. This concludes the construction. $\qquad\square$

10.3 A succinct 5SAT description for deciders

Proposition 10.6 allows us to convert any decider $\mathcal{D}$ and inputs n, x, y into a 3SAT formula $\varphi_{3\text{SAT}}$, succinctly described by a circuit $\mathcal{C}$, which represents it. However, there are two undesirable properties of this construction, which we describe below.

1. First, evaluating any clause $(w_{i_1}^{o_{i_1}} \lor w_{i_2}^{o_{i_2}} \lor w_{i_3}^{o_{i_3}})$ of $\varphi_{3\text{SAT}}$ requires evaluating the *same* assignment w at three separate points. While this is fine when the assignment w is provided in full to the verifier, it can be a problem when the verifier is only able to query the points in w by interacting with a prover. In this case, the verifier might send the prover the values i_1, i_2, i_3, who responds with three bits $b_1, b_2, b_3 \in \{0, 1\}$, purported to be the values $w_{i_1}, w_{i_2}, w_{i_3}$ for some assignment $w \in \{0, 1\}^M$. As it will turn out, in the answer reduced protocol below, the verifier will actually be able to force the prover to reply using *three* different assignments. In other words, the prover will have three assignments $w_1, w_2, w_3 \in \{0, 1\}^M$ such that, for any i_1, i_2, i_3 provided to it by the verifier, it will respond with $b_1 = w_{1,i_1}, b_2 = w_{2,i_2}, b_3 = w_{3,i_3}$. However, even given this there is no guarantee that the three assignments are the *same* assignments (i.e. that $w_1 = w_2 = w_3$). In the work of [NW19], this was accomplished by an additional subroutine called the *intersecting lines test*, which would enforce consistency between w_1, w_2, and w_3. In this work, on the hand, we would like to relax the assumption that w_1, w_2, and w_3 must be the same. This will allow us to not use the intersecting lines test, simplifying the answer reduction protocol. (In fact, the answer reduced verifier will not be querying the assignments directly, but rather low-degree encodings of these assignments; see Section 9 for details.)

2. Second, we are guaranteed that $w = (a, b, c)$ satisfies $\varphi_{3\text{SAT}}$ if and only if $\mathcal{D}$ accepts (n, x, y, a, b). However, it is inconvenient that a and b are contained as substrings of w. To see why, recall from Section 9 that the oracularized verifier sometimes gives one prover a pair of questions (x, y) and another prover just one of the questions—say, x. The answer reduced verifier will sample its questions similarly; as for its answers, it might expect the first prover to respond with a string $w = (a, b, c)$ that satisfies $\varphi_{3\text{SAT}}$ and the second prover to respond with a string a' such that $a = a'$. Verifying that $a = a'$ requires the verifier to sample a uniformly random point from w, restricted to the coordinates in a. As it turns out, generating a uniform point from a substring is extremely cumbersome, though

118

not impossible, to do when the verifier's questions are expected to be sampled from conditional linear functions. To remove this complication, though, we would instead prefer if the provers' answers were formatted in a way that gave the verifier direct access to the strings a and b.

In the remainder of this section, we will show how to modify the Succinct-3SAT circuits produced by Proposition 10.6 in order to ameliorate these two difficulties. Doing so entails modifying the φ_{3SAT} formula from Proposition 10.6 to produce a *5SAT formula* φ_{5SAT}. The clauses of φ_{5SAT} will be of the form

$$a_{i_1}^{o_1} \vee b_{i_2}^{o_2} \vee w_{1,i_3}^{o_3} \vee w_{2,i_4}^{o_4} \vee w_{3,i_5}^{o_5} \,,$$

where $a, b, w_1, w_2,$ and w_3 are five separate assignments which are not assumed to be equal. The guarantee is that (a, b, w_1, w_2, w_3) satisfies φ_{5SAT} if and only if $\mathcal{D}$ accepts (n, x, y, a, b). This addresses the two concerns from above: each clause is totally *decoupled*, meaning it samples 5 variables from 5 different assignments, and so no consistency check must be performed between the assignments. In addition, the first two strings exactly correspond to a and b, addressing the second item.

We now formally define decoupled 5SAT instances and how they succinctly represent bounded deciders. Following that, we show how the succinct 3SAT instance φ_{3SAT} produced by Proposition 10.6 can be modified to produce a succinct 5SAT instance φ_{5SAT} which represents $\mathcal{D}$.

Definition 10.13 (Decoupled 5SAT and its succinct descriptions). A *block* of variables x_i is a tuple $x_i = (x_{i,0}, \ldots, x_{i,N_i-1})$. A formula φ on 5 blocks $x_1, x_2, \ldots, x_5$ of variables is called a *decoupled 5SAT formula* if every clause is of the form

$$x_{1,i_1}^{o_1} \vee x_{2,i_2}^{o_2} \vee x_{3,i_3}^{o_3} \vee x_{4,i_4}^{o_4} \vee x_{5,i_5}^{o_5} \,, \tag{115}$$

for $i_j \in \{0, 1, \ldots, N_j - 1\}$ and $o_1, \ldots, o_5 \in \{0, 1\}$. (Recall from Definition 10.2 that the notation x^o means x if $o = 1$ and $\neg x$ if $o = 0$.)

For each $i \in \{1, 2, \ldots, 5\}$, suppose each N_i is a power of two, and write it as $N_i = 2^{n_i}$. Let $\mathcal{C}$ be a circuit with five inputs of length $n_1, n_2, \ldots, n_5$ and five single-bit inputs. Then $\mathcal{C}$ *succinctly describes decoupled φ* if, for all $i_j \in \{0, 1, \ldots, N_j - 1\}$ and $o_1, o_2, \ldots, o_5 \in \{0, 1\}$,

$$\mathcal{C}(i_1, i_2, \ldots, i_5, o_1, o_2, \ldots, o_5) = 1 \tag{116}$$

if and only if the clause in (115) is included in φ. As in Definition 10.2, we slightly abuse notation and use the convention that a number a between 0 and $2^{n_i} - 1$ is interpreted as its binary encoding $\text{binary}_{n_i}(a)$ when provided as input to a set of n_i single-bit wires.

Definition 10.14 (Succinct descriptions for bounded deciders). Let $\mathcal{D}$ be a decider. Fix an index $n \in \mathbb{N}$ and a time $T \in \mathbb{N}$. Let $L = 2^\ell$ be the smallest power of two at least as large as $2T$. Let x and y be strings, $r \in \mathbb{N}$ and $R = 2^r$.

Consider a circuit $\mathcal{C}$ with two inputs of length ℓ, three inputs of length r, and 5 single-bit inputs. Let $\varphi_{\mathcal{C}}$ be the decoupled 5SAT instance with two blocks of variables of size L and three blocks of size M which $\mathcal{C}$ succinctly describes. Then we say that $\mathcal{C}$ *succinctly describes $\mathcal{D}$ (on inputs n, x, and y and time T)* if, for all $a, b \in \{0, 1\}^L$, there exists $w_1, w_2, w_3 \in \{0, 1\}^R$ such that a, b, w_1, w_2, w_3 satisfy $\varphi_{\mathcal{C}}$ if and only if there exist $a_{\text{prefix}}, b_{\text{prefix}} \in \{0, 1\}^*$ of lengths $\ell_a, \ell_b \leq T$, respectively, such that

$$a = \text{enc}_\Gamma(a_{\text{prefix}}, \sqcup^{L/2-\ell_a}) \quad \text{and} \quad b = \text{enc}_\Gamma(b_{\text{prefix}}, \sqcup^{L/2-\ell_b})$$

and $\mathcal{D}$ accepts $(n, x, y, a_{\text{prefix}}, b_{\text{prefix}})$ in time T.

In this definition of succinct descriptions, the answers a and b are isolated, in that the first input of $\mathcal{C}$ of length ℓ indexes into a and the second input of length ℓ indexes into b. The next proposition shows how to construct such descriptions.

Proposition 10.15 (Explicit succinct descriptions). *There is a Turing machine* SuccinctDecider *with the following properties. Let $\mathcal{D}$ be a decider, let n, T, Q, and λ be integers with $Q \leq T$ and $|\mathcal{D}| \leq \lambda$, and let x and y be strings of length at most Q. Then on input $(\mathcal{D}, n, T, Q, \lambda, x, y)$,* SuccinctDecider *outputs a circuit $\mathcal{C}$ with two inputs of length $\ell_0(T)$, three of length $r_0(T, \lambda)$, and five single-bit inputs which succinctly describes $\mathcal{D}$ on inputs n, x, and y and time T. Moreover, the following hold.*

1. $\ell_0(T) = \lceil \log(2T) \rceil$.

2. $r_0(T, \lambda) = O(\log(T) + \log(\lambda))$,

3. $\mathcal{C}$ *has at most* $s_0(n, T, Q, \lambda) = \mathrm{poly}(\log(T), \log(n), Q, \lambda)$ *gates,*

4. SuccinctDecider *runs in time* $\mathrm{poly}(\log(T), \log(n), Q, \lambda)$, *and the parameters ℓ_0, r_0, s_0 can be computed from n, T, Q, λ in time* $\mathrm{poly}(\log(T), \log(n), \log(Q), \lambda)$.

Proof. The Turing machine SuccinctDecider begins by running the algorithm in Proposition 10.6 on input $(\mathcal{D}, n, T, Q, \lambda, x, y)$ to produce a circuit $\mathcal{C}_{\mathrm{3SAT}}$ on $3r_0 + 3$ inputs, where $r_0 = O(\log(T) + \log(\lambda))$ is the parameter m from the proposition. Set $\ell_0 = \lceil \log(2T) \rceil$, $L = 2^{\ell_0}$ and $R = 2^{r_0}$. Given this, SuccinctDecider returns the circuit $\mathcal{C}$ with inputs $i_1, i_2 \in \{0, 1, \ldots, L-1\}$, $i_3, i_4, i_5 \in \{0, 1, \ldots, R-1\}$, $o_1, o_2, \ldots, o_5 \in \{0, 1\}$, and

$$\mathcal{C}(i_1, i_2, \ldots, i_5, o_1, o_2, \ldots, o_5) = 1,$$

if one of the following conditions hold.

$$\mathcal{C}_{\mathrm{3SAT}}(i_3, i_4, i_5, o_3, o_4, o_5) = 1,$$
$$(i_1 < 2T) \wedge (i_1 = i_3) \wedge (o_1 \neq o_3),$$
$$(i_2 < 2T) \wedge (i_2 = i_3 - 2T) \wedge (o_2 \neq o_3),$$
$$(i_1 \geq 2T) \wedge (i_1 \text{ is odd}) \wedge (o_1 = 1),$$
$$(i_1 \geq 2T) \wedge (i_1 \text{ is even}) \wedge (o_1 = 0),$$
$$(i_2 \geq 2T) \wedge (i_2 \text{ is odd}) \wedge (o_2 = 1),$$
$$(i_2 \geq 2T) \wedge (i_2 \text{ is even}) \wedge (o_2 = 0),$$
$$(i_3 = i_4) \wedge (o_3 \neq o_4),$$
$$(i_4 = i_5) \wedge (o_4 \neq o_5).$$

It is not hard to verify that testing "$(i_1 < 2T)$" can be done with $O(\ell_0)$ AND and OR gates, and testing $(i_2 = i_3 - 2T)$ can be done with $O(m)$ AND and OR gates. Using similar estimates for the remaining sub-circuits, we compute

$$\mathrm{size}(\mathcal{C}) = \mathrm{size}(\mathcal{C}_{\mathrm{3SAT}}) + O(r_0 + \ell_0) = \mathrm{size}(\mathcal{C}_{\mathrm{3SAT}}) + O(r_0)$$
$$\leq \mathrm{poly}(\log(T), \log(n), Q, \lambda) + O(\log(T) + \log(\lambda)).$$

In addition, due to the simplicity of these modifications, we conclude that the runtime of SuccinctDecider is dominated by the runtime of the algorithm from Proposition 10.6, which is $\mathrm{poly}(\log(T), \log(n), Q, \lambda)$.

Now we show that $\mathcal{C}$ succinctly describes $\mathcal{D}$ on inputs n, x, and y and time T. To begin, we describe the decoupled 5SAT formula $\varphi_{\mathcal{C}}$. Let us first consider the constraints in $\varphi_{\mathcal{C}}$ which are implied by the final constraint, i.e. those of the form

$$a_{i_1}^{o_1} \vee b_{i_2}^{o_2} \vee (w_1)_{i_3}^{o_3} \vee (w_2)_{i_4}^{o_4} \vee (w_3)_{i_5}^{o_5}$$

whenever $i_4 = i_5$ and $o_4 \neq o_5$. For any fixed i_1, i_2, i_3, the negations o_1, o_2, o_3 can take any values, and as a result, the first three bits in the constraint vary over all assignments in $\{0,1\}^3$. This means that these constraints are satisfied if and only if $(w_2)_{i_4}^{o_4} \vee (w_3)_{i_5}^{o_5}$ is satisfied whenever $i_4 = i_5$ and $o_4 \neq o_5$. This, in turn, is equivalent to the constraint $w_3 = w_4$. Carrying out similar arguments for the entire circuit, we can express the formula $\varphi_{\mathcal{C}}$ as follows.

$$\begin{aligned} \varphi_{\mathcal{C}}(a, b, w_1, w_2, w_3) = \varphi_{3\mathrm{SAT}}&(w_1, w_2, w_3) \wedge (w_{1,1} = a_1) \wedge (w_{1,2} = b_1) \\ &\wedge (a_2 = (10)^{L/2-T}) \wedge (b_2 = (10)^{L/2-T}) \wedge (w_1 = w_2) \wedge (w_2 = w_3). \end{aligned}$$

Here, we write $\varphi_{3\mathrm{SAT}}(w_1, w_2, w_3)$ for the formula in which, for each constraint in $\varphi_{3\mathrm{SAT}}$, the first variable is taken from w_1, the second from w_2, and the third from w_3. In addition, we write $a = (a_1, a_2)$, where a_1 is the first $2T$ bits in a and a_2 is the remaining $L - 2T$ bits, and similarly for $b = (b_1, b_2)$. We also write $w_1 = (w_{1,1}, w_{1,2}, w_{1,3})$, where $w_{1,1}$ contains the first $2T$ bits in w_1, $w_{1,2}$ contains the second $2T$ bits, and $w_{1,3}$ contains the remaining $R - 4T$ bits. As a result, $\varphi_{\mathcal{C}}$ is satisfied only if $w_1 = w_2 = w_3 = (a_1, b_1, c)$ for some string $c \in \{0,1\}^{R-4T}$. In this case, calling $w = (a_1, b_1, c)$, $\varphi_{\mathcal{C}}$ is satisfied only if $\varphi_{3\mathrm{SAT}}(w)$ is. By Proposition 10.6, this implies that there exists string a_{prefix} of length $\ell_a \leq T$ such that $a_1 = \mathrm{enc}_\Gamma(a_{\mathrm{prefix}}, \sqcup^{T-\ell_a})$. This, in turn, implies that

$$a = (a_1, a_2) = (\mathrm{enc}_\Gamma(a_{\mathrm{prefix}}, \sqcup^{T-\ell_a}), (10)^{L-2T}) = \mathrm{enc}_\Gamma(a_{\mathrm{prefix}}, \sqcup^{L/2-\ell_a}) \,,$$

using the fact that $\mathrm{enc}_\Gamma(\sqcup) = 10$, and similarly for b. Finally, Proposition 10.6 implies that $\mathcal{D}$ accepts $(n, x, y, a_{\mathrm{prefix}}, b_{\mathrm{prefix}})$ in time T. This completes the proof. $\qquad\square$

As stated above, moving from 3SAT to 5SAT allows us to devote the first two inputs to a and b. In addition, we have added extra constraints into $\varphi_{\mathcal{C}}$ which enforce that $w_1 = w_2 = w_3$, which means that we can relax this assumption on these assignments.

We now show a simple transformation that takes in a succinct circuit $\mathcal{C}$ and outputs another succinct circuit $\mathcal{C}'$ whose 5 inputs are "padded" to contain more input bits. This will be helpful in the PCP proof below, where we sometimes expect the input lengths ℓ and r to be divisible by another integer m.

Proposition 10.16 (Padding). *Let $\mathcal{C}$ be a circuit of size s with two inputs of length ℓ, three inputs of length r, and 5 single-bit inputs. Suppose $\mathcal{C}$ succinctly describes $\mathcal{D}$ on inputs n, x, and y and time T. Then there is an algorithm which takes as input $(\mathcal{C}, \ell, r, \ell', r')$, with $\ell' \geq \ell$ and $r' \geq r$, and in time $\mathrm{poly}(s, \ell', r')$ outputs a circuit $\mathcal{C}'$ with the following properties. First, $\mathcal{C}'$ has two inputs of length ℓ', three inputs of length r', and 5 single-bit inputs, and its size is $s + \mathrm{poly}(\ell', r')$. Second, it succinctly describes $\mathcal{D}$ on inputs n, x, and y and time T.*

Proof. Write $L = 2^\ell, R = 2^r$ and $L' = 2^{\ell'}, R' = 2^{r'}$. The algorithm constructs the circuit $\mathcal{C}'$ which on inputs $i_1, i_2 \in \{0, \dots, L'-1\}$, $i_3, i_4, i_5 \in \{0, \dots, R'-1\}$, and $o_1, \dots, o_5 \in \{0, 1\}$, outputs 1 if and only if

one of the following conditions hold:

$$(i_1, i_2 < L) \wedge (i_3, i_4, i_5 < R) \wedge \mathcal{C}(i_1, i_2, i_3, i_4, i_5, o_1, o_2, o_3, o_4, o_5) = 1 \,,$$
$$(i_1 \geq L) \wedge (i_1 \text{ is odd}) \wedge (o_1 = 1) \,,$$
$$(i_1 \geq L) \wedge (i_1 \text{ is even}) \wedge (o_1 = 0) \,,$$
$$(i_2 \geq L) \wedge (i_2 \text{ is odd}) \wedge (o_2 = 1) \,,$$
$$(i_2 \geq L) \wedge (i_2 \text{ is even}) \wedge (o_2 = 0) \,.$$

Now we show that $\mathcal{C}'$ succinctly describes $\mathcal{D}$ on inputs n, x, and y and time T. To begin, we describe the decoupled 5SAT formula $\varphi_{\mathcal{C}'}$. The second and third constraints imply that for each $i_1 \geq L$, $\varphi_{\mathcal{C}'}$ contains the constraint (a_{i_1}) if i_1 is odd and $(\neg a_{i_1})$ if i_1 is even. Likewise, the fourth and fifth constraints imply that for each $i_2 \geq L$, $\varphi_{\mathcal{C}'}$ contains the constraint (b_{i_2}) if i_2 is odd and $(\neg a_{i_2})$ if i_2 is even. Thus, we can express the formula $\varphi_{\mathcal{C}'}$ as follows.

$$\varphi_{\mathcal{C}'}(a, b, w_1, w_2, w_3) = \varphi_{\mathcal{C}}(a_1, b_1, w_{1,1}, w_{2,1}, w_{3,1}) \wedge (a_2 = (10)^{(L'-L)/2}) \wedge (b_2 = (10)^{(L'-L)/2}). \quad (117)$$

Here, we write $a = (a_1, a_2)$ and $b = (b_1, b_2)$, where a_1, b_1 have length L and a_2, b_2 have length $L' - L$, and for each $i \in \{1, 2, 3\}$, we write $w_i = (w_{i,1}, w_{i,2})$, where $w_{i,1}$ has length R and $w_{i,2}$ has length $R' - R$.

Now, suppose there exist w_1, w_2, w_3 such that a, b, w_1, w_2, w_3 satisfy $\varphi_{\mathcal{C}'}$. Then because $\mathcal{C}$ succinctly represents $\mathcal{D}$, there exists $a_{\text{prefix}}, b_{\text{prefix}}$ of lengths $\ell_a, \ell_b \leq T$ such that $\mathcal{D}$ accepts $(n, x, y, a_{\text{prefix}}, b_{\text{prefix}})$. In addition,

$$a_1 = \text{enc}_\Gamma(a_{\text{prefix}}, \sqcup^{L/2 - \ell_a}),$$

and likewise for b_1. Equation (117) then implies that

$$\begin{aligned}
a = (a_1, a_2) &= (\text{enc}_\Gamma(a_{\text{prefix}}, \sqcup^{L/2 - \ell_a}), (10)^{(L'-L)/2}) \\
&= (\text{enc}_\Gamma(a_{\text{prefix}}, \sqcup^{L/2 - \ell_a}), \text{enc}_\Gamma(\sqcup)^{(L'-L)/2}) \\
&= \text{enc}_\Gamma(a_{\text{prefix}}, \sqcup^{L'/2 - \ell_a}),
\end{aligned}$$

where the third step used the fact that $\text{enc}_\Gamma(\sqcup) = 10$. As a similar statement holds for b, this establishes that $\mathcal{C}'$ succinctly describes $\mathcal{D}$ on inputs n, x, and y and time T. $\qquad \square$

10.4 A PCP for normal form deciders

We give a probabilistically checkable proof (PCP) for the Bounded Halting problem specialized to the case of normal form deciders. Our PCP will use standard techniques from the algebraic, low-degree-code-based PCP literature. In particular, we slightly modify the PCP for Succinct-3SAT described in [NW19, Section 11] (which itself is based on the proof of the PCP theorem in [Har04]) to apply it to the decoupled Succinct-5SAT instances described in Section 10.3. We follow their treatment closely. As the PCPs constructed in this section are only an intermediate object towards the normal form verifier introduced in the next section we do not include standard definitions on PCPs, and refer to these references (in particular [Har04]) for background. We begin with some preliminaries.

10.4.1 Preliminaries

A key part of the PCP will be to design a function $f : \mathbb{F}^n \to \mathbb{F}$ which is zero on a subcube $H_{\text{subcube}} = H_1 \times \cdots \times H_n$, where each H_i is some subset of $\mathbb{F}$. Our next proposition shows that given such a function f,

there is a way of writing it so that the fact that it is zero on H_{subcube} is self-evidently true. Doing so involves showing that f can be written in a simple basis of polynomials which are constructed to be zero on the subcube. This fact is standard in the literature (see, for example, [Har04, Proposition 5.3.5]), and we include its proof for completeness.

Proposition 10.17 (Polynomial basis of zero functions). *Let $\mathbb{F}$ be a field. For each $i \in \{1,\ldots,n\}$, let H_i be a subset of $\mathbb{F}$ of size h_i, and let $\text{zero}_i : \mathbb{F} \to \mathbb{F}$ be the function defined as $\text{zero}_i(x) = \prod_{y \in H_i}(x - y)$. Define $H_{\text{subcube}} = H_1 \times \cdots \times H_n$. Suppose $f : \mathbb{F}^n \to \mathbb{F}$ is a degree-d polynomial such that $f(x) = 0$ for all $x \in H_{\text{subcube}}$. Then there exist polynomials $c_1,\ldots,c_n : \mathbb{F}^n \to \mathbb{F}$ such that for all $x \in \mathbb{F}^n$,*

$$f(x) = \sum_{i=1}^{n} c_i(x) \cdot \text{zero}_i(x_i) .$$

In addition, for each $i \in \{1,\ldots,n\}$, c_i is degree-$(d - h_i)$ if $d \geq h_i$, and otherwise it is equal to the zero polynomial.

Proof. To prove this, we first prove the following statement for each $k \in \{0,1,\ldots,n\}$: there exists a degree-d polynomial $r_k : \mathbb{F}^n \to \mathbb{F}$ and polynomials $c_1,\ldots,c_k : \mathbb{F}^n \to \mathbb{F}$ such that

$$f(x) = \sum_{i=1}^{k} c_i(x) \cdot \text{zero}_i(x_i) + r_k(x) . \tag{118}$$

In addition, for each $i \in \{1,\ldots,k\}$, c_i is degree-$(d - h_i)$ if $d \geq h_i$, and otherwise it is equal to the zero polynomial. Furthermore, for each $i \in \{1,\ldots,k\}$, r_k is degree at most $h_i - 1$ in x_i.

The proof is by induction on k, the base case of $k = 0$ being trivial. Now, we perform the induction step. Assuming that Equation (118) holds for k, we will show that it holds for $k + 1$ as well. Let r_k be the polynomial guaranteed by the inductive hypothesis. We now divide r_k by $\text{zero}_{k+1}(x_{k+1})$ using polynomial division. This guarantees a polynomial c_{k+1} and a degree-d polynomial $r_{k+1}(x)$ such that

$$r_k(x) = c_{k+1}(x) \cdot \text{zero}_{k+1}(x_{k+1}) + r_{k+1}(x)$$

In addition, c_{k+1} is degree-$(d - h_{k+1})$ if $d \geq h_{k+1}$, and otherwise it is equal to the zero polynomial. Furthermore, for each $i \in \{1,\ldots,k+1\}$, r_{k+1} is degree at most $h_i - 1$ in x_i. Plugging this into Equation (118), we see that

$$f(x) = \sum_{i=1}^{k+1} c_i(x) \cdot \text{zero}_i(x_i) + r_{k+1}(x) . \tag{119}$$

This completes the induction.

Applying the $k = n$ case of Equation (118), we see that

$$f(x) = \sum_{i=1}^{n} c_i(x) \cdot \text{zero}_i(x_i) + r(x) , \tag{120}$$

where, for each $i \in \{1,\ldots,n\}$, r is degree-$(h_i - 1)$ in x_i. For each $x \in H_{\text{subcube}}$, because $f(x)$ and the summation on the left-hand side of Equation (120) are zero on x, this implies that $r(x) = 0$ as well. We claim that r must therefore be the zero polynomial. We prove this by showing the following statement for every integer $k \in \{1,\ldots,n\}$: let $s(x_1,\ldots,x_k)$ be a polynomial which is zero on $H_1 \times \cdots \times H_k$. Furthermore, suppose that for each $i \in \{1,\ldots,k\}$, s is degree-$(h_i - 1)$ in x_i. Then s is the zero polynomial.

123

The proof is by induction on k. Consider the base case $k = 1$. Then s is a univariate polynomial of degree at most $h_1 - 1$, but is zero on h_1 points. Thus s must be the zero polynomial. Now we perform the induction step. Assuming the proposition holds for some $k \geq 1$, we will show that it holds for $k + 1$ as well. Assume for contradiction that s is *not* the zero polynomial. Let d be the minimum integer such that s is degree d in variable x_{k+1}. By assumption, $d \leq h_{k+1} - 1$. Write

$$s(x_1, \ldots, x_{k+1}) = \sum_{j=0}^{d} x_{k+1}^{j} \cdot g_j(x_1, \ldots, x_k),$$

where for each $j \in \{0, \ldots, d\}$, the polynomial g_j has degree at most $h_i - 1$ in the variable x_i for $i \in \{1, \ldots, k\}$. Because s is nonzero and d was selected to be minimal, g_d cannot be the zero polynomial. In this case, our induction hypothesis states that $g_d(y) \neq 0$ for some $y \in H_1 \times \cdots \times H_k$. Then $s(y, x_{k+1})$ is a degree-d nonzero univariate polynomial in x_{k+1}. Furthermore, for each $x_{k+1} \in H_{k+1}$, $s(y, x_{k+1}) = 0$, by assumption. But this is a contradiction, as any univariate polynomial of degree at most $h_{k+1} - 1$ which is zero on every point in H_{i+1} must be the zero polynomial. As a result, s must be zero on $H_1 \times \cdots \times H_{k+1}$.

Thus, r is the zero polynomial. Applying this fact to Equation (120), we arrive at the statement in the proposition. $\qquad\square$

10.4.2 The PCP

The problem. The input to the PCP verifier is a tuple $(\mathcal{D}, n, T, Q, \lambda, x, y)$. Here, $\mathcal{D}$ is a decider, n, T, Q, and λ are integers with $Q \leq T$ and $|\mathcal{D}| \leq \lambda$, and x and y are a pair of strings of length at most Q each. The goal of the verifier is to check whether there exists two strings a_{prefix} and b_{prefix} of length at most T such that $\mathcal{D}$ halts on input $(n, x, y, a_{\text{prefix}}, b_{\text{prefix}})$ in time at most T. To do that the verifier makes random queries to a specially encoded *PCP proof* Π, and decides whether to accept or reject based on the parts of Π that it reads. We first set the parameters used in the PCP construction.

Definition 10.18 (Parameters for the PCP). For all integers $n, T, Q, \lambda \in \mathbb{N}$ such that $Q \leq T$ and $|\mathcal{D}| \leq \lambda$ define the tuple $\mathsf{pcpparams}(n, T, Q, \lambda) = (\ell, r, s, m, d, m', q)$ as follows. Let $r_0 = r_0(T, \lambda)$, $\ell_0 = \ell_0(T)$ and $s_0 = s_0(n, t, Q, \lambda)$ be as in Proposition 10.15.

1. Let $m = \lceil r_0 / \log(r_0) \rceil$.

2. Let r, ℓ be the smallest integers such that $r \geq r_0$ and $\ell \geq \ell_0$ and such that r, ℓ are multiples of m. Note that $r < r_0 + m$ and $\ell < \ell_0 + m$.

3. Let $s = s_0$.

4. Let $d = 8s \cdot (2^{r/m} - 1) = 8s \cdot \mathrm{poly}(r)$.

5. Let $m' = 5m + 5 + s$.

6. Let q be the smallest field size such that: $q = 2^k$ for some odd integer k; $q^m \geq 2^r$; and $d(m + m')/q^c \leq \frac{1}{n}$, where c is the smallest of the two universal constants in Lemma 7.4 and Theorem 7.14.

Given n, T, Q, λ, $\mathsf{pcpparams}(n, T, Q, \lambda)$ can be computed in time $\mathrm{poly}(\log(n), \log(T), \log(Q), \lambda)$.

Next, we define the format of a valid PCP proof, which for our construction consists of evaluation tables of low-degree polynomials.

Definition 10.19. Given $n, T, Q, \lambda \in \mathbb{N}$ and $(\ell, r, s, m, d, m', q) = \mathsf{pcpparams}(n, T, Q, \lambda)$, a *low-degree PCP proof* is a tuple Π of evaluation tables of polynomials $g_1, \dots, g_5 : \mathbb{F}_q^m \to \mathbb{F}_q$ and $c_0, \dots, c_{m'} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ with all polynomials having degree at most d. We divide the m' input variables of $c_0, \dots, c_{m'}$ into blocks as follows:

$$\mathbb{F}_q^{m'} \ni z = (\underbrace{x_1}_{\mathbb{F}_q^m}, \dots, \underbrace{x_5}_{\mathbb{F}_q^m}, \underbrace{o}_{\mathbb{F}_q^5}, \underbrace{w}_{\mathbb{F}_q^s}) \ .$$

Definition 10.20. Given a low-degree PCP proof Π and a point $z = (x_1, \dots, x_5, o, w) \in \mathbb{F}_q^{m'}$, where $x_1, \dots, x_5 \in \mathbb{F}_q^m$, $o \in \mathbb{F}_q^5$, and $w \in \mathbb{F}_q^s$, the *evaluation* of Π at z is given by

$$\mathsf{eval}_z(\Pi) = (\alpha_1, \dots, \alpha_5, \beta_0, \dots, \beta_{m'}) \in \mathbb{F}_q^{6+m'} \ ,$$

where $\alpha_i = g_i(x_i)$ and $\beta_j = c_j(z)$.

Theorem 10.21. *There exists a Turing machine $\mathcal{M}_{\mathrm{AR}}$ with the following properties.*

1. *(Input format) The input to $\mathcal{M}_{\mathrm{AR}}$ consists of two parts: a "decider specification" and a "PCP view."*

   (a) *(Decider specification) Let $\mathcal{D}$ be a decider, n, T, Q, and λ be integers with $Q \leq T$ and $|\mathcal{D}| \leq \lambda$, and let x and y be strings of length at most Q. Let $(\ell, r, s, m, d, m', q) = \mathsf{pcpparams}(n, T, Q, \lambda)$ be as in Definition 10.18. Then the decider specification is the tuple $(\mathcal{D}, n, T, Q, \lambda, q, x, y)$.*

   (b) *(PCP view) Let $z \in \mathbb{F}_q^{m'}$ and let $\Xi \in \mathbb{F}_q^{6+m'}$. Then the PCP view is the pair (z, Ξ).*

   The Turing machine $\mathcal{M}_{\mathrm{AR}}$ returns either 1 (accept) or 0 (reject).

For the remaining items, assume a decider specification has been fixed, so we think of $\mathcal{M}_{\mathrm{AR}}$ as a function of the PCP view input only.

2. *(Completeness): Suppose $a_{\mathrm{prefix}}, b_{\mathrm{prefix}} \in \{0, 1\}^*$ are two strings of length ℓ_a, ℓ_b, respectively, such that $\mathcal{D}$ halts in time T on input $(n, x, y, a_{\mathrm{prefix}}, b_{\mathrm{prefix}})$. Setting $L = 2^\ell$, write*

$$a = \mathsf{enc}_\Gamma(a_{\mathrm{prefix}}, \sqcup^{L/2 - \ell_a}) \quad \text{and} \quad b = \mathsf{enc}_\Gamma(b_{\mathrm{prefix}}, \sqcup^{L/2 - \ell_b}) \ .$$

Then there exists a low-degree PCP proof (Definition 10.19) $\Pi = (g_1, \dots, g_5, c_0, \dots, c_{m'})$ with $g_1 = g_a$ and $g_2 = g_b$, the canonical low-degree encodings of a and b with parameters m, q respectively (see Definition 3.24), which causes $\mathcal{M}_{\mathrm{AR}}$ to accept with probability 1 over the choice of a point z uniformly at random:

$$\Pr_{z \in \mathbb{F}_q^{m'}} \left(\mathcal{M}_{\mathrm{AR}}(z, \mathsf{eval}_z(\Pi)) = 1 \right) = 1 \ .$$

3. *(Soundness): Let $\Pi = (g_1, \dots, g_5, c_0, \dots, c_{m'})$ be a low-degree PCP proof such that $\mathcal{M}_{\mathrm{AR}}$ at a uniformly random z accepts with probability larger than $p_{\mathrm{sound}} = 0.9$:*

$$\Pr_{z \in \mathbb{F}_q^{m'}} \left(\mathcal{M}_{\mathrm{AR}}(z, \mathsf{eval}_z(\Pi)) = 1 \right) > p_{\mathrm{sound}} \ .$$

Then there exist degree-d polynomials $f_1, f_2 : \mathbb{F}_q^m \to \mathbb{F}_q$ and strings $a, b \in \{0, 1\}^L$ with the following properties.

(a) *There exist strings $a_{\text{prefix}}, b_{\text{prefix}} \in \{0,1\}^*$ of length ℓ_a, ℓ_b, respectively, such that*

$$a = \text{enc}_\Gamma(a_{\text{prefix}}, \sqcup^{L/2-\ell_a}) \quad \text{and} \quad b = \text{enc}_\Gamma(b_{\text{prefix}}, \sqcup^{L/2-\ell_b}).$$

(b) *$\mathcal{D}$ halts in time T on input $(n, x, y, a_{\text{prefix}}, b_{\text{prefix}})$.*

(c) *For every $i \in \{1, \ldots, 2^\ell\}$, $f_1(\pi_L(i)) = a_i$ and $f_2(\pi_L(i)) = b_i$, where π_L is the canonical injection map given in Definition 3.22, where $n = 2^\ell$ and k, m are as here.*

(d) *On a random $x \in \mathbb{F}_q^m$, the probability that $f_1(x) \neq g_1(x)$ is at most 0.2, and likewise for f_2 and g_2.*

4. (Efficiency): $\mathcal{M}_{\text{AR}}$ *runs in time* $\text{poly}(s, m', 2^{r/m}, \log(q), |\mathcal{D}|)$. *Recalling the setting of parameters in Definition 10.18, this means that* $\mathcal{M}_{\text{AR}}$ *runs in time at most* $\text{poly}(\log(T), \log(n), Q, \log(q), |\mathcal{D}|)$.

Proof of Theorem 10.21. We first give the construction of $\mathcal{M}_{\text{AR}}$ and then show that it satisfies the properties claimed in the theorem. The Turing machine $\mathcal{M}_{\text{AR}}$ begins by computing

$$\mathcal{C}_0 = \text{SuccinctDecider}(\mathcal{D}, n, T, Q, \lambda, x, y),$$

(see Proposition 10.15) which succinctly describes the decoupled 5SAT formula $\varphi_{\mathcal{C}_0}$. The circuit $\mathcal{C}_0$ has two ℓ_0-bit inputs, three $r_0 = O(\log(T) + \log(\lambda))$-bit inputs, and five single-bit inputs, and contains at most $s_0 = \text{poly}(\log(T), \log(n), Q, \lambda)$ AND and OR gates. (We note that $\mathcal{C}_0$ also has NOT gates. However, it will not be necessary for us to keep track of the number of these gates.)

Let $\mathcal{C}$ be the circuit obtained from Proposition 10.16 by padding the input wires of $\mathcal{C}_0$ so that the variable-length inputs of $\mathcal{C}$ are two inputs of length ℓ and three of length r. Henceforth we work with this padded circuit. The circuit $\mathcal{C}$ is a succinct description of $\mathcal{D}$; we recall what this means here. Write $L := 2^\ell$ and $R := 2^r$. Then for all $a, b \in \{0,1\}^L$, there exist $u_1, u_2, u_3 \in \{0,1\}^R$ such that a, b, u_1, u_2, u_3 satisfy $\varphi_{\mathcal{C}}$ if and only if there exist $a_{\text{prefix}}, b_{\text{prefiix}} \in \{0,1\}^*$ of lengths $\ell_a, \ell_b \leq T$, respectively, such that

$$a = \text{enc}_\Gamma(a_{\text{prefix}}, \sqcup^{T-\ell_a}) \quad \text{and} \quad b = \text{enc}_\Gamma(b_{\text{prefix}}, \sqcup^{T-\ell_b})$$

and $\mathcal{D}$ accepts $(n, x, y, a_{\text{prefix}}, b_{\text{prefix}})$ in time T. We refer to the five strings a, b, u_1, u_2, u_3 as the *witness strings*. By Propositions 10.15 and 10.16, computing $\mathcal{C}$ from $(\mathcal{D}, n, T, Q, \lambda, x, y)$ can be done in time $\text{poly}(\log(T), \log(n), \log(Q), \lambda)$.

Encoding the proof. Recall from Definition 3.22 the canonical subspaces $H_L = H_{\text{canon}, m, k, L}$ and $H_R = H_{\text{canon}, m, k, R}$ (where we see H_L as a subset of H_R in a natural way), their sizes $h_L := h_{\text{canon}, m, k, L} = 2^{\ell/m}$ and $h_R := h_{\text{canon}, m, k, R} = 2^{r/m}$, and the following two canonical injections:

$$\pi_L = \pi_{\text{canon}, m, k, L} : \{0, 1, \ldots, L-1\} \to H_L^m,$$
$$\pi_R = \pi_{\text{canon}, m, k, R} : \{0, 1, \ldots, R-1\} \to H_R^m.$$

By Definition 3.22 and the setting of parameters in Definition 10.18, these are both bijections, and h_L and h_R are both at most $2^{r/m} = \text{poly}(r)$.

The PCP proof contains five functions $g_1, \ldots, g_5 : \mathbb{F}_q^m \to \mathbb{F}_q$. $\mathcal{M}_{\text{AR}}$ expects these to be the low-degree encodings $g_{a, \pi_L}, g_{b, \pi_L}, g_{u_1, \pi_R}, g_{u_2, \pi_R}, g_{u_3, \pi_R}$ of five witness strings, the first two with respect to π_L and the last three with respect to π_R (see Equation (12) on page 30 for the definition of low-degree encodings). In

this case, the first two have degree $m(h_L - 1) = \text{poly}(r)$, and the last three degree $m(h_R - 1) = \text{poly}(r)$. In addition, for all $i_1, i_2 \in \{0, 1, \ldots, L-1\}$

$$g_1(\pi_L(i_1)) = a_{i_1}, \quad g_2(\pi_L(i_2)) = b_{i_2},$$

and for all $j \in \{3, 4, 5\}$ and $i_j \in \{0, 1, \ldots, R-1\}$,

$$g_j(\pi_R(i_j)) = u_{j-2, i_j}. \tag{121}$$

We also recall the following maps from Definition 3.25:

$$\nu_L = \nu_{\text{canon}, m, k, L} : \mathbb{F}_q^m \to \mathbb{F}_q^\ell,$$
$$\nu_R = \nu_{\text{canon}, m, k, R} : \mathbb{F}_q^m \to \mathbb{F}_q^r.$$

By Proposition 3.26, these are degree $h_L - 1$ and $h_R - 1$, respectively, can be computed in $\text{poly}(m, h_R, k)$ time, and have the property that for each $i \in \{0, 1, \ldots, L-1\}$,

$$\nu_L(\pi_L(i)) = \text{binary}_\ell(i), \tag{122}$$

and likewise for ν_R.

Encoding the formula. Next, $\mathcal{M}_{\text{AR}}$ modifies $\mathcal{C}$ to make it compatible with low-degree encodings. To begin, it applies the Tseitin transformation (see [NW19, Section 3.8]) to $\mathcal{C}$. This produces a Boolean formula $\mathcal{F}$ with $s' \leq 8s$ AND and OR gates (where we recall that s is an upper bound on the number of AND and OR gates in $\mathcal{C}$) such that for all $i_1, i_2 \in \{0, 1, \ldots, L-1\}$, $i_3, i_4, i_5 \in \{0, 1, \ldots, R-1\}$, and $o \in \{0, 1\}^5$,

$$\mathcal{C}(i_1, i_2, \ldots, i_5, o) = 1$$

if and only if there exists a $w \in \{0, 1\}^s$ such that

$$\mathcal{F}(i_1, i_2, \ldots, i_5, o, w) = 1.$$

Next, $\mathcal{M}_{\text{AR}}$ arithmetizes the formula as in [NW19, Definition 3.28] by setting $\mathcal{F}_{\text{arith}} := \text{arith}_q(\mathcal{F})$. This is a function $\mathcal{F}_{\text{arith}} : \mathbb{F}_q^{2\ell + 3r + 5 + s} \to \mathbb{F}_q$ such that

$$\forall x \in \{0, 1\}^{2\ell + 3r + 5 + s}, \quad \mathcal{F}_{\text{arith}}(x) = \mathcal{F}(x). \tag{123}$$

By [NW19, Proposition 3.29], $\mathcal{F}_{\text{arith}}$ is a degree-s' polynomial. Computing $\mathcal{F}_{\text{arith}}$ involves performing $O(s') = O(s)$ field operations (addition, subtraction, and multiplication), a time $\text{poly}(s, \log q)$ task. Let $m' = 5m + 5 + s$, and define the function $g_\varphi : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ by

$$g_\varphi(x_1, x_2, \ldots, x_5, o, w) = \mathcal{F}_{\text{arith}}(\nu_L(x_1), \nu_L(x_2), \nu_R(x_3), \nu_R(x_4), \nu_R(x_5), o, w). \tag{124}$$

By (122) and (123), for all $i_1, i_2 \in \{0, 1, \ldots, L-1\}$, $i_3, i_4, i_5 \in \{0, 1, \ldots, R-1\}$, $o \in \{0, 1\}^5$, and $w \in \{0, 1\}^s$,

$$g_\varphi(\pi_L(i_1), \pi_L(i_2), \pi_R(i_3), \pi_R(i_4), \pi_R(i_5), o, w) = \mathcal{F}(i_1, i_2, \ldots, i_5, o, w). \tag{125}$$

By construction, g_φ has degree at most $s' \cdot h_R$ and can be computed in time $\text{poly}(s, m, h_R, \log(q))$.

Zero on subcube. Define the function $c_0 : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ as

$$c_0(x, o, w) = g_\varphi(x, o, w) \cdot (g_1(x_1) - o_1) \cdots (g_5(x_5) - o_5) .$$

If the g_i are low-degree encodings of witness strings, then c_0 is a degree $d := s' \cdot h_R + 2mh_L + 3mh_R = O(s \cdot \mathrm{poly}(r))$ polynomial. Next, define the subcube

$$H_{\mathrm{subcube}} := H_{\mathrm{subcube},1} \times \cdots \times H_{\mathrm{subcube},5m+5+s} = H_L^{2m} \times H_R^{3m} \times \{0,1\}^{5+s} .$$

Here, the first $2m$ $H_{\mathrm{subcube},i}$ are H_L, the next $3m$ are H_R, and the remaining are $\{0,1\}$. We would like to evaluate c_0 on the subcube H_{subcube} in the case that the g_i are low-degree encodings of the witness strings. Let $(x, o, w) \in H_{\mathrm{subcube}}$. Then because π_L, π_R are bijections, there exist $i_1, i_2 \in \{0, 1, \ldots, L-1\}$ and $i_3, i_4, i_5 \in \{0, 1, \ldots, R-1\}$ such that $x = (\pi_L(i_1), \pi_L(i_2), \pi_R(i_3), \pi_R(i_4), \pi_R(i_5))$. As a result, by Eqs. (121) and (125),

$$c_0(x, o, w) = \mathcal{F}(i_1, i_2, \ldots, i_5, o, w) \cdot (a_{i_1} - o_1)(b_{i_2} - o_2)(u_{1,i_3} - o_3)(u_{2,i_4} - o_4)(u_{3,i_5} - o_5) . \quad (126)$$

Suppose $\mathcal{F}(i_1, i_2, \ldots, i_5, o, w) \neq 0$. Then $\mathcal{C}(i_1, i_2, \ldots, i_5, o) = 1$, and so $a_{i_1}^{o_1} \vee b_{i_2}^{o_2} \vee u_{1,i_3}^{o_3} \vee u_{2,i_4}^{o_4} \vee u_{3,i_5}^{o_5}$ is a clause in $\varphi_{\mathcal{C}}$. But a, b, u_1, u_2, u_3 satisfy $\varphi_{\mathcal{C}}$, and so one of the terms in Equation (126) must evaluate to 0. In conclusion, c_0 is zero on the subcube H_{subcube}.

For each i, let $\mathrm{zero}_i : \mathbb{F}_q \to \mathbb{F}_q$ be the function defined as $\mathrm{zero}_i(x) = \prod_{y \in H_{\mathrm{subcube},i}} (x - y)$. If c_0 evaluates to zero on the subcube H_{subcube}, then by Proposition 10.17 there exist degree-d polynomials $c_1, \ldots, c_{m'} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ such that for all $z = (x, o, w) \in \mathbb{F}_q^{m'}$,

$$c_0(z) = \sum_{i=1}^{m'} c_i(z) \cdot \mathrm{zero}_i(z_i) .$$

The PCP proof will include these c_i's to certify that c_0 is zero on the subcube. From the preceding discussion, this, in turn, certifies that a, b, u_1, u_2, u_3 are witness strings that satisfy $\varphi_{\mathcal{C}}$.

The honest PCP proof. The PCP for $(\mathcal{D}, n, T, Q, \lambda, q, x, y)$ is given by a tuple Π consisting of truth tables of functions $g_1, \ldots, g_5 : \mathbb{F}_q^m \to \mathbb{F}_q$ and $c_0, \ldots, c_{m'} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$, together with associated planes tables. In the "honest" PCP proof, the functions are expected to satisfy the following:

- $g_1, \ldots, g_5$ are the low-degree encodings of $a, b \in \{0,1\}^L$ and $u_1, u_2, u_3 \in \{0,1\}^R$;

- For every $(x, o, w) \in \mathbb{F}_q^{m'}$,

$$c_0(x, o, w) = g_\varphi(x, o, w)(g_1(x_1) - o_1)(g_2(x_2) - o_2) \cdots (g_5(x_5) - o_5) ; \quad (127)$$

- $c_0, \ldots, c_{m'}$ are such that

$$\forall z = (x, o, w) \in \mathbb{F}_q^{m'} , \quad \sum_{i=1}^{m'} c_i(z) \mathrm{zero}_i(z_i) = c_0(z) . \quad (128)$$

The PCP view. We describe how $\mathcal{M}_{AR}$ decides whether to accept or reject given a view (z, Ξ) of a purported PCP proof. Recall that the input $\Xi \in \mathbb{F}_q^{6+m'}$ is intended to be equal to the evaluation $\mathrm{eval}_z(\Pi)$ of a low-degree PCP proof $\Pi = (g_1, \ldots, g_5, c_0, \ldots, c_{m'})$ at the point $z = (x, o, w) \in \mathbb{F}_q^{m'}$. We denote the elements of Ξ by $\Xi = (\alpha_1, \ldots, \alpha_5, \beta_0, \ldots, \beta_{m'})$. Given a decider specification and a PCP view (z, Ξ), $\mathcal{M}_{AR}$ decides to accept or reject by performing the following steps sequentially.

1. (Zero test) Verify that Eq. (128) holds at $z = (x, o, w)$, i.e. reject if $\sum_{i=1}^{m'} \beta_i \, \mathrm{zero}_i(z_i) \neq \beta_0$.

2. (Formula test) Verify Eq. (127) at $z = (x, o, w) \in \mathbb{F}_q^{m'}$: compute the value $g_\varphi(z)$ of the polynomial g_φ obtained from $(\mathcal{D}, n, T, x, y)$. As noted below Eq. (125), this value can be computed in time $\mathrm{poly}(s, m, h_R, \log(q))$. Reject if

$$\beta_0 \neq g_\varphi(z)(\alpha_1' - o_1) \cdots (\alpha_5' - o_5) \, .$$

3. Otherwise, accept.

This completes the description of the Turing machine $\mathcal{M}_{AR}$. It remains to show the properties claimed in the theorem. Completeness follows by inspection of $\mathcal{M}_{AR}$ and the form of the honest PCP proof. Soundness follows from [NW19, Proposition 11.8]. We now evaluate the running time of $\mathcal{M}_{AR}$. The zero test involves computing m' different $\mathrm{zero}_i(z_i)$ values, each of which is a product of at most h_R different terms. This takes time $\mathrm{poly}(m', h_R, \log(q))$. The formula test involves computing $g_\varphi(z)$. As discussed below Eq. (125), given the decider specification $(\mathcal{D}, n, T, Q, q, x, y)$ and the point $z \in \mathbb{F}_q^{m'}$, $g_\varphi(z)$ can be computed in time $\mathrm{poly}(s, m, h_R, \log(q))$. Taken together, these bounds imply the claimed runtime. $\qquad\square$

10.5 A normal form verifier for the PCP

In this section we show how to convert the PCP from Section 10.4 into a normal form verifier. This results in an "answer reduction" scheme: a way to map a verifier $\mathcal{V}$ into a new verifier $\mathcal{V}^{AR}$ with a smaller answer size.

Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier and (λ, μ, σ) a tuple of integers. In the rest of this section we define the answer-reduced verifier $\mathcal{V}^{AR} = (\mathcal{S}^{AR}, \mathcal{D}^{AR})$ associated with $\mathcal{V}$ and (λ, μ, σ). Completeness, complexity and soundness of the construction are shown in the following sections.

10.5.1 Parameters and notation

First, we recall the parameters set by the PCP construction from Section 10.4. Let n be an index for $\mathcal{V}$. Let

$$T = (\mu 2^n)^\mu \quad \text{and} \quad Q = (\sigma n)^\sigma \, . \tag{129}$$

Even though these are fixed as functions of n and (μ, σ), for clarity we generally keep T and Q as free parameters in the analysis. The answer reduction procedure, when applied to a normal form verifier $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ and parameters (μ, σ), assumes the following bounds on the complexities of the verifier $\mathcal{V}$:

$$|\mathcal{D}| \leq \lambda \, , \qquad \forall n \geq 1 \, , \quad \mathrm{TIME}_\mathcal{D}(n) \leq T(n) \quad \text{and} \quad \mathrm{TIME}_\mathcal{S}(n), \, \mathrm{RAND}_\mathcal{S}(n) \leq Q(n) \, . \tag{130}$$

Let $(\ell, r, s, m, d, m', q) = \mathrm{pcpparams}(n, T, Q, \lambda)$ be as in Definition 10.18. We note that with the choice of T and Q in (129), each of these parameters is $\mathrm{poly}(\log T, Q, \lambda)$.

Let $(\mathcal{D}, n, T, Q, \lambda, q, x, y)$ be a decider specification to be input to the PCP verifier $\mathcal{M}_{AR}$ specified in Theorem 10.21. Recall from Definition 10.19 that a low-degree PCP proof consists of $6 + m'$ polynomials

$g_1, \ldots, g_5 : \mathbb{F}_q^m \to \mathbb{F}_q$ and $c_0, \ldots, c_{m'} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$, where the variables of $c_0, \ldots, c_{m'}$ are divided into blocks $z = (x_1, \ldots, x_5, o, w)$. We frequently view the polynomial g_i not as a function with domain $\mathbb{F}_q^m$ but rather as a function with domain $\mathbb{F}_q^{m'}$ that only depends on the i-th block of m variables, x_i (recall that $m' = 5m + 5 + s$). This allows us to consider all the polynomials as functions over the same domain $\mathbb{F}_q^{m'}$. In the "honest" case, the polynomials g_1, g_2 are expected to be low-degree encodings with parameters m, q (see Definition 3.24) of answers $a, b \in \mathbb{F}_2^L$ respectively, where $L = 2^\ell$.

Next, we recall the parameters of the typed sampler $\mathcal{S}^{\mathrm{ORAC}}$ introduced in Section 9.2. The sampler uses ambient space V^{ORAC} over $\mathbb{F}_2$ with dimension $\hat{s}(n)$ that is identical to the dimension of $\mathcal{S}$. Recall from Definition 4.14 the notation $\mu_\mathcal{S}$ denoting the distribution over pairs of questions $(x_{\mathrm{A}}, x_{\mathrm{B}})$ generated by $\mathcal{S}$. We use $\mu_{\mathcal{S},\mathrm{A}}$ to indicate the marginal distribution of $\mu_\mathcal{S}$ on the first question x_{A}, and $\mathrm{SUPP}(\mu_\mathcal{S})$ to indicate the set of question pairs that have nonzero probability under $\mu_\mathcal{S}$.

Remark 10.22. *In this section, for convenience we often identify the label* A *with* 1 *and* B *with* 2.

10.5.2 The answer-reduced verifier

Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier and (λ, μ, σ) integers. All other required parameters and notation are introduced in Section 10.5.1.

Sampler. The sampler is $\mathcal{S}^{\mathrm{AR}} = \mathcal{S}^{\mathrm{ORAC}} \times \mathcal{S}^{\mathrm{PCP}}$, where $\mathcal{S}^{\mathrm{ORAC}}$ is the typed sampler obtained by oracularizing $\mathcal{S}$, as defined in Section 9.2, and the sampler $\mathcal{S}^{\mathrm{PCP}}$ is an additional typed sampler associated with the PCP that is defined below. As a typed sampler, $\mathcal{S}^{\mathrm{AR}}$ uses type set $\mathcal{T}^{\mathrm{ORAC}} \times \mathcal{T}^{\mathrm{PCP}}$ and type graph $G^{\mathrm{AR}} = G^{\mathrm{ORAC}} \times G^{\mathrm{PCP}}$ with edge set

$$E^{\mathrm{AR}} = \left\{ \{(u,v),(u',v')\} : \{u,u'\} \in E^{\mathrm{ORAC}} \wedge \{v,v'\} \in E^{\mathrm{PCP}} \right\} .$$

We first define the sampler $\hat{\mathcal{S}}^{\mathrm{PCP}}$ with field size function $q(n)$. The sampler $\mathcal{S}^{\mathrm{PCP}}$ is taken to be $\kappa(\hat{\mathcal{S}}^{\mathrm{PCP}})$ (see Definition 4.15 for the definition of downsized samplers). The typed sampler $\hat{\mathcal{S}}^{\mathrm{PCP}}$ is a 2-level $\mathcal{T}^{\mathrm{PCP}}$-typed sampler, where

$$\mathcal{T}^{\mathrm{PCP}} = \{\mathrm{POINT}_1, \ldots, \mathrm{POINT}_6\} \cup \{\mathrm{PLANE}_1, \ldots, \mathrm{PLANE}_6\} ,$$

and the type graph $G^{\mathrm{PCP}} = (\mathcal{T}^{\mathrm{PCP}}, E^{\mathrm{PCP}})$ uses $E^{\mathrm{PCP}} = \mathcal{T}^{\mathrm{PCP}} \times \mathcal{T}^{\mathrm{PCP}}$. The ambient vector space for the sampler is

$$V^{\mathrm{PCP}} = \left(\bigoplus_{i=1}^{5} V_{i,\mathrm{X}} \oplus V_{i,\mathrm{V1}} \oplus V_{i,\mathrm{V2}} \right) \oplus V_{\mathrm{AUX,X}} \oplus V_{\mathrm{AUX,V1}} \oplus V_{\mathrm{AUX,V2}} , \tag{131}$$

where each space $V_{i,\mathrm{X}}, V_{i,\mathrm{V1}}, V_{i,\mathrm{V2}}$ is isomorphic to $\mathbb{F}_q^m$, and $V_{\mathrm{AUX,X}}, V_{\mathrm{AUX,V1}}, V_{\mathrm{AUX,V2}}$ are isomorphic to $\mathbb{F}_q^{5+s}$. In addition, define the following direct sums:

$$V_{6,\mathrm{X}} = \left(\bigoplus_{i=1}^{5} V_{i,\mathrm{X}} \right) \oplus V_{\mathrm{AUX,X}} ,$$

$$V_{6,\mathrm{V1}} = \left(\bigoplus_{i=1}^{5} V_{i,\mathrm{V1}} \right) \oplus V_{\mathrm{AUX,V1}} ,$$

$$V_{6,\mathrm{V2}} = \left(\bigoplus_{i=1}^{5} V_{i,\mathrm{V2}} \right) \oplus V_{\mathrm{AUX,V2}} .$$

The conditionally linear function associated to type $t \in \mathcal{T}^{\mathrm{PCP}}$ is a CL function on V^{PCP} defined as follows.

- For the types $t = \mathrm{POINT}_i$ for $i \in \{1, \ldots, 6\}$, the 1-level CL function $L^{\mathrm{PT}i}$ is identical to the CL function L^{PT} defined in Figure 3, with the 1st factor subspace replaced with $V_{i,X}$ here.

- For the types $t = \mathrm{PLANE}_i$ for $i \in \{1, \ldots, 6\}$, the 2-level CL function $L^{\mathrm{PL}i}$ is identical to the CL function L^{PL} defined in Figure 3, with the 1st factor subspace replaced by $V_{i,V1} \oplus V_{i,V2}$ and the 2nd factor subspace by $V_{i,X}$.

Decider. The decider $\mathcal{D}^{\mathrm{AR}}$ is described in Fig. 13.

10.6 Completeness and complexity of the answer-reduced verifier

The following theorem formulates the complexity and completeness properties of the answer-reduced verifier. Since $\mathcal{V}^{\mathrm{AR}}$ is defined as a typed verifier, we use the detyping procedure described in Section 6.3 to obtain an untyped normal form verifier.

Theorem 10.23. *Let $\lambda, \mu, \sigma \in \mathbb{N}$. Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier such that $\mathcal{S}$ is an ℓ-level sampler. Let $\mathcal{V}^{\mathrm{AR}} = (\mathcal{S}^{\mathrm{AR}}, \mathcal{D}^{\mathrm{AR}})$ be the answer-reduced verifier corresponding to $\mathcal{V}$ and parameters (λ, μ, σ). Let $\mathrm{detype}(\mathcal{V}^{\mathrm{AR}}) = (\mathrm{detype}(\mathcal{S}^{\mathrm{AR}}), \mathrm{detype}(\mathcal{D}^{\mathrm{AR}}))$ denote the detyped verifier. Then $\mathrm{detype}(\mathcal{V}^{\mathrm{AR}})$ is a normal form verifier such that the following hold. Let $T(n)$ and $Q(n)$ satisfy (129).*

1. *(**Completeness**) Assume that $\mathcal{V}$ satisfies assumption (130). Then for all $n \in \mathbb{N}$, if $\mathcal{V}_n$ has a projective, consistent, and commuting (PCC) strategy of value 1, then $\mathrm{detype}(\mathcal{V}^{\mathrm{AR}})_n$ has a symmetric PCC strategy with value 1.*

2. *(**Sampler complexity**) $\mathrm{detype}(\mathcal{S}^{\mathrm{AR}})$ is a $\max\{\ell + 4, 5\}$-level sampler that depends on (μ, σ) and $\mathcal{S}$ only (not on $\mathcal{D}$). Moreover, the time and randomness complexities of $\mathrm{detype}(\mathcal{S}^{\mathrm{AR}})$ satisfy*

$$\mathrm{TIME}_{\mathrm{detype}(\mathcal{S}^{\mathrm{AR}})}(n) = O(\mathrm{TIME}_{\mathcal{S}^{\mathrm{ORAC}}}(n)) + \mathrm{poly}\log(T(n)) = \mathrm{poly}(\mu n, (\sigma n)^\sigma),$$
$$\mathrm{RAND}_{\mathrm{detype}(\mathcal{S}^{\mathrm{AR}})} = \mathrm{RAND}_{\mathcal{S}^{\mathrm{ORAC}}}(n) + \mathrm{poly}\log(T(n)) = \mathrm{poly}(\mu n, (\sigma n)^\sigma).$$

3. *(**Decider complexity**) The time complexity of the decider $\mathcal{D}^{\mathrm{AR}}$ satisfies*

$$\mathrm{TIME}_{\mathrm{detype}(\mathcal{D}^{\mathrm{AR}})}(n) = \mathrm{poly}(\log(T(n)), Q(n), |\mathcal{D}|) = \mathrm{poly}(\mu n, (\sigma n)^\sigma, |\mathcal{D}|).$$

4. *(**Efficient computability**) There is a Turing machine $\mathsf{ComputeARVerifier}$ which takes as input a tuple $(\mathcal{V}, \lambda, \mu, \sigma)$, with $\lambda, \mu, \sigma \in \mathbb{N}$, and returns descriptions of $\mathrm{detype}(\mathcal{S}^{\mathrm{AR}}), \mathrm{detype}(\mathcal{D}^{\mathrm{AR}})$ corresponding to $\mathcal{V} = (\mathcal{S}, \mathcal{D})$. Moreover $\mathsf{ComputeARVerifier}$ given such an input runs in time*

$$\mathrm{poly}(|\mathcal{V}|, \log(\lambda), \log(\mu), \log(\sigma)).$$

Proof. We show each of the claimed properties in turn.

Type	Question Format	Answer Format
$\textsc{Point}_i$ for $i = 1, \ldots, 5$	$y_i \in \mathbb{F}_q^m$	$\alpha_i \in \mathbb{F}_q$
$\textsc{Plane}_i$ for $i = 1, \ldots, 5$	$v_i \in (\mathbb{F}_q^m)^3$	$h_i : \mathbf{p}(v_i) \to \mathbb{F}_q$
$\textsc{Point}_6$	$z = (y, o, w) \in \mathbb{F}_q^{m'}$	$(\alpha'_1, \ldots, \alpha'_5, \beta_0, \ldots, \beta_{m'}) \in \mathbb{F}_q^{m'+6}$
$\textsc{Plane}_6$	$v \in (\mathbb{F}_q^{m'})^3$	$(h'_1, \ldots, h'_5, f_0, \ldots, f_{m'}) : \mathbf{p}(v) \to \mathbb{F}_q^{m'+6}$

Table 1: Question and answer formats for types in $\mathcal{T}^{\text{PCP}}$.

On input $(n, t_A, x_A, t_B, x_B, a_A, a_B)$, the decider $\mathcal{D}^{\text{AR}}$ parses t_A, t_B as $(t_{Q,A}, t_{\Pi,A})$, and $(t_{Q,B}, t_{\Pi,B})$ respectively in $\mathcal{T}^{\text{ORAC}} \times \mathcal{T}^{\text{PCP}}$, parses x_A and x_B as $(x_{Q,A}, x_{\Pi,A})$ and $(x_{Q,B}, x_{\Pi,B})$ respectively. The decider performs the following steps sequentially, for all $w \in \{A, B\}$:

1. (**Global consistency check**): If $t_A = t_B$, reject if $a_A \neq a_B$.

2. (**Input consistency check**): If $t_{Q,w} = \textsc{Oracle}$ and $t_{Q,\overline{w}} = v \in \{A, B\}$, and if $(t_{\Pi,w}, t_{\Pi,\overline{w}}) = (\textsc{Point}_6, \textsc{Point}_v)$, reject if $\alpha_v \neq \alpha'_v$ (where $A \leftrightarrow 1$ and $B \leftrightarrow 2$, as per Remark 10.22).

3. (**Input low degree test**) If $t_{Q,w} = t_{Q,\overline{w}} = v \in \{A, B\}$, and if $(t_{\Pi,w}, t_{\Pi,\overline{w}}) = (\textsc{Point}_v, \textsc{Plane}_v)$, execute $\mathcal{D}^{\text{LD}}_{\text{ldparams}}$ on input $(\textsc{Point}, x_{\Pi,w}, \textsc{Plane}, x_{\Pi,\overline{w}}, a_w, a_{\overline{w}})$, where $\text{ldparams} = (q, m, d, 1)$. Reject if $\mathcal{D}^{\text{LD}}_{\text{ldparams}}$ rejects.

4. (**Proof encoding checks**): If $t_{Q,w} = t_{Q,\overline{w}} = \textsc{Oracle}$,

   (a) (Consistency test) If $(t_{\Pi,w}, t_{\Pi,\overline{w}}) = (\textsc{Point}_i, \textsc{Point}_6)$ for some $i \in \{3, \ldots, 5\}$, reject if $\alpha_i \neq \alpha'_i$.

   (b) (Individual low degree test) If $(t_{\Pi,w}, t_{\Pi,\overline{w}}) = (\textsc{Point}_i, \textsc{Plane}_i)$ for some $i \in \{3, \ldots, 5\}$, execute $\mathcal{D}^{\text{LD}}_{\text{ldparams}}$ on input $(\textsc{Point}, x_{\Pi,w}, \textsc{Plane}, x_{\Pi,\overline{w}}, a_w, a_{\overline{w}})$. Reject if $\mathcal{D}^{\text{LD}}_{\text{ldparams}}$ rejects.

   (c) (Simultaneous low degree test) If $(t_{\Pi,w}, t_{\Pi,\overline{w}}) = (\textsc{Point}_6, \textsc{Plane}_6)$, execute $\mathcal{D}^{\text{LD}}_{\text{ldparams}'}$ on input $(\textsc{Point}, x_{\Pi,w}, \textsc{Plane}, x_{\Pi,\overline{w}}, a_w, a_{\overline{w}})$, where $\text{ldparams}' = (q, m', d, m' + 6)$. Reject if $\mathcal{D}^{\text{LD}}_{\text{ldparams}'}$ rejects.

5. (**Game check**): If $t_{Q,w} = \textsc{Oracle}$, then for $v \in \{A, B\}$, compute $x_{w,v} = L^v(x_{Q,w})$. If $t_{\Pi,w} = \textsc{Point}_6$, reject if $\mathcal{M}_{\text{AR}}((\mathcal{D}, n, T, Q, q, x_{w,A}, x_{w,B}), (z, a_w))$ rejects. Otherwise, accept.

Figure 13: The decision procedure $\mathcal{D}^{\text{AR}}$. Parameters T, Q, q, m, m', d are defined in Section 10.5.1.

Completeness. We first show completeness. Let $n \geq 1$ be an index for $\mathcal{V}^{\text{AR}}$. Let $\mathscr{S}$ be a PCC strategy for $\mathcal{V}_n$ with value 1. By Theorem 9.1 it follows that there exists a symmetric PCC strategy $\mathscr{S}' = (|\psi\rangle, M)$ with value 1 for $\mathcal{V}_n^{\text{ORAC}}$. We define a strategy $\mathscr{S}''$ for the typed verifier $\mathcal{V}_n^{\text{AR}}$ as follows. The shared state is $|\psi\rangle$.

Given the index n and (λ, μ, σ), each player can compute $(\ell, r, s, m, d, m', q) = \mathsf{pcpparams}(n, T, Q, \lambda)$ (see Definition 10.18), where $T = T(n)$ and $Q = Q(n)$ are as in (129).

1. On receipt of a question $((\mathsf{t}_Q, \mathsf{t}_\Pi), (x_Q, x_\Pi))$ a player first measures their share of $|\psi\rangle$ using the projective measurement for $\mathscr{S}^{\mathrm{ORAC}}$ for the typed question (t_Q, x_Q) to obtain an outcome a_Q. The player then computes an answer, depending on t_Q, a_Q and t_Π, x_Π, as follows:

   (a) Suppose $\mathsf{t}_Q = v \in \{A, B\}$ and $\mathsf{t}_\Pi \in \{\mathrm{POINT}_v, \mathrm{PLANE}_v\}$. Let $a'_Q = a_Q$ if a_Q has length at most T, and let a'_Q be the truncation of a_Q to its first T symbols otherwise. Let $\ell_a \le T$ be the length of a'_Q, and set
   $$a''_Q = \mathrm{enc}_\Gamma(a'_Q, \sqcup^{L/2 - \ell_a}).$$
   Next, the player computes the canonical low-degree encoding $g_{a''_Q}$ of a''_Q using the canonical low-degree encoding from Definition 3.22, in the same way as described in Section 10.4.2. The player then returns the restriction of $g_{a''_Q}$ to the subspace specified by x_Π.

   (b) If $\mathsf{t}_Q = \mathrm{ORACLE}$, for $v \in \{A, B\}$ the player computes questions $x_v = L^v(x_Q)$, as in Item 1 of $\mathcal{D}^{\mathrm{ORAC}}$. The player parses a_Q as a pair (a_A, a_B). Let $a'_A = a_A$ if a_A has length at most T, and let a'_A be the truncation of a_A to its first T symbols otherwise. Let $\ell_A \le T$ be the length of a'_A, and set
   $$a''_A = \mathrm{enc}_\Gamma(a'_A, \sqcup^{L/2 - \ell_A}).$$
   Define a''_B similarly. The player computes a PCP proof $\Pi = (g_1, \ldots, g_5, c_0, \ldots, c_{m'})$ as described in the completeness case of Theorem 10.21 for the tuple $(\mathcal{D}, n, T, x_A, x_B)$, where the polynomials g_1, g_2 are low-degree encodings of a''_A and a''_B, respectively.

      i. If $\mathsf{t}_\Pi \in \{\mathrm{POINT}_i, \mathrm{PLANE}_i\}$ for $i \in \{1, \ldots, 5\}$, the player returns the restriction of g_i to the subspace of $\mathbb{F}_q^m$ specified by x_Π.
      ii. If $\mathsf{t}_\Pi \in \{\mathrm{POINT}_6, \mathrm{PLANE}_6\}$, the player returns the restriction of all the polynomials $g_1, \ldots, g_5$, $c_0, \ldots, c_{m'}$ to the subspace of $\mathbb{F}_q^{m'}$ specified by x_Π.

   (c) In all other cases the player returns 0.

The strategy $\mathscr{S}''$ is projective and consistent because $\mathscr{S}'$ is. To show that it has value 1, we first observe that by definition it satisfies all consistency checks. Moreover, the strategy passes all low-degree tests with certainty because it always returns restrictions of consistent polynomials. Finally, it also passes the game check with probability 1. This follows from the completeness statement of the PCP made in Theorem 10.21 and the fact that, if $\mathcal{D}$ accepts the input (n, x_A, x_B, a_A, a_B) in time at most T then it also accepts $(n, x_A, x_B, a'_A, a'_B)$ in time at most T, where a'_A and a'_B are obtained from a_A and a_B by truncating them to strings of length T if their lengths exceed T.

To show that $\mathscr{S}''$ is commuting, note that using the product structure of $\mathcal{S}^{\mathrm{AR}}$ every typed question pair with positive probability consists of a pair of questions $((\mathsf{t}_{Q,A}, x_{Q,A}), (\mathsf{t}_{Q,B}, x_{Q,B}))$ with positive probability for $\mathcal{S}^{\mathrm{ORAC}}$, together with an arbitrary pair $((\mathsf{t}_{\Pi,A}, x_{\Pi,A}), (\mathsf{t}_{\Pi,B}, x_{\Pi,B}))$. Using that $\mathscr{S}^{\mathrm{ORAC}}$ is commuting and that the additional operations associated with $((\mathsf{t}_{\Pi,A}, x_{\Pi,A}), (\mathsf{t}_{\Pi,B}, x_{\Pi,B}))$ amount to classical post-processing it follows that $\mathscr{S}''$ is commuting.

This establishes the existence of a symmetric PCC strategy for $\mathcal{V}_n^{\mathrm{AR}}$ with value 1. By Lemma 6.18 it follows that there exists a symmetric PCC strategy for $\mathrm{detype}(\mathcal{V}^{\mathrm{AR}})_n$ with value 1.

Sampler complexity. The sampler $\mathcal{S}^{\mathrm{AR}}$ depends only on $\mathcal{S}^{\mathrm{ORAC}}$, which itself depends only on $\mathcal{S}$ (see Theorem 9.1). Using Theorem 9.1, $\mathcal{S}^{\mathrm{ORAC}}$ is a $\max\{\ell, 1\}$-level typed sampler. Using Lemma 6.18 for the detyping it follows that $\mathcal{S}^{\mathrm{ORAC}}$ is a $\max\{\ell + 2, 3\}$-level typed sampler. Using Lemma 6.18 again, $\mathrm{detype}(\mathcal{S}^{\mathrm{AR}})$ is a $\max\{\ell + 4, 5\}$-level sampler.

In addition to the space V^{ORAC} used by $\mathcal{S}^{\mathrm{ORAC}}$, $\mathcal{S}^{\mathrm{AR}}$ uses space V^{PCP} of dimension $O(m + s)$ over $\mathbb{F}_q$ defined in Eq. (131), and the claim on the randomness complexity follows. Time complexity is clear as well. The complexity of $\mathrm{detype}(\mathcal{S}^{\mathrm{AR}})$ follows by Lemma 6.18.

Decider complexity. The decider $\mathcal{D}^{\mathrm{AR}}$ executes subroutines $\mathcal{D}^{\mathrm{LD}}$ and $\mathcal{M}_{\mathrm{AR}}$. The runtime of $\mathcal{D}^{\mathrm{LD}}$ is $\mathrm{poly}(m, d, m', \log q)$ by Lemma 7.7, which for our choice of parameters is $\mathrm{poly}(\mu n)$. The runtime of $\mathcal{M}_{\mathrm{AR}}$ is given in Theorem 10.21; for T and Q as in (129) it is $\mathrm{poly}(\mu n, (\sigma n)^\sigma, |\mathcal{D}|)$. In addition to the subroutines, $\mathcal{D}^{\mathrm{AR}}$ performs only simple field manipulations in $\mathbb{F}_q$, which by Lemma 3.18 can also be implemented in $\mathrm{poly}(\mu n)$ time since q is polynomial. The complexity of $\mathrm{detype}(\mathcal{D}^{\mathrm{AR}})$ follows by Lemma 6.18.

Efficient computability. The description of $\mathcal{S}^{\mathrm{AR}}$ can be computed, in polynomial time, from the description of $\mathcal{S}$ alone. The description of $\mathcal{D}^{\mathrm{AR}}$ can be computed in polynomial time from the descriptions of $\mathcal{S}, \mathcal{D}$, as well as $\mathcal{D}^{\mathrm{LD}}$; the latter can be computed in polynomial time by Lemma 7.7. The complexity of computing descriptions of the detyped sampler and decider follows by Lemma 6.18. □

10.7 Soundness of the answer-reduced verifier

Theorem 10.24 (Soundness of the answer-reduced verifier)**.** *Let $\mu, \sigma \in \mathbb{N}$. Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier satisfying assumption (130). Let $\mathcal{V}^{\mathrm{AR}} = (\mathcal{S}^{\mathrm{AR}}, \mathcal{D}^{\mathrm{AR}})$ be the answer-reduced verifier corresponding to $\mathcal{V}$ and parameters (λ, μ, σ), as described in Section 10.5. Let $\mathrm{detype}(\mathcal{V}^{\mathrm{AR}})$ denote the detyped verifier, as in Definition 6.17. There exists a $\delta(\varepsilon, n) = \mathrm{poly}(\varepsilon + 1/n)$ such that the following hold. For all $n \geq 1$,*

1. *If $\mathrm{val}^*(\mathrm{detype}(\mathcal{V}^{\mathrm{AR}})_n) > 1 - \varepsilon$ for some $\varepsilon > 0$ then $\mathrm{val}^*(\mathcal{V}_n) \geq 1 - \delta(\varepsilon, n)$.*

2. *Let $\mathscr{E}(\cdot)$ be as defined in Definition 5.12. Then for all $\varepsilon \geq 0$,*

$$\mathscr{E}(\mathrm{detype}(\mathcal{V}^{\mathrm{AR}})_n, 1 - \varepsilon) \geq \mathscr{E}(\mathcal{V}_n, 1 - \delta(\varepsilon, n)) .$$

Proof. We first show the first item, soundness, for the typed verifier $\mathcal{V}^{\mathrm{AR}}$. Soundness for the detyped verifier $\mathrm{detype}(\mathcal{V}^{\mathrm{AR}})$ follows from Lemma 6.18, with a constant-factor loss using that the type set $\mathcal{T}^{\mathrm{AR}}$ for $\mathcal{V}^{\mathrm{AR}}$ has constant size.

We proceed in two steps. Fix an index $n \geq 1$ and suppose that $\mathrm{val}^*(\mathcal{V}_n^{\mathrm{AR}}) > 1 - \varepsilon$ for some $\varepsilon > 0$. Observe that $\mathcal{S}^{\mathrm{ORAC}}$ and $\mathcal{S}^{\mathrm{PCP}}$ both sample distributions that are invariant under permutation of the two players; therefore, the same holds for $\mathcal{S}^{\mathrm{AR}}$. Moreover, the decider $\mathcal{D}^{\mathrm{AR}}$ treats both players symmetrically. Therefore, the game played by $\mathcal{V}_n^{\mathrm{AR}}$ is a symmetric game. Applying Lemma 5.7 it follows that there exists a symmetric projective strategy $\mathscr{S} = (|\psi\rangle, M)$ for $\mathcal{V}_n^{\mathrm{AR}}$ with value $1 - \varepsilon$.

We use the following shorthand notation. A pair of questions to the players is $((\mathsf{t}_\mathrm{A}, x'_\mathrm{A}), (\mathsf{t}_\mathrm{B}, x'_\mathrm{B}))$ where for $w \in \{\mathrm{A}, \mathrm{B}\}$, $\mathsf{t}_w = (\mathsf{t}_{Q,w}, \mathsf{t}_{\Pi,w})$ and $x'_w = (x'_{Q,w}, x'_{\Pi,w})$. When w is clear from context we omit it from the subscript. Fixing a w, whenever $\mathsf{t}_Q = \mathrm{ORACLE}$ we introduce $x_\mathrm{A} = L^\mathrm{A}(x_Q)$ and $x_\mathrm{B} = L^\mathrm{B}(x'_Q)$ and often write directly the player's question as $x_Q = (x_\mathrm{A}, x_\mathrm{B})$. Whenever $\mathsf{t}_Q = v \in \{\mathrm{A}, \mathrm{B}\}$ we slightly abuse notation and write the question as $x_Q = (x_v, v)$, explicitly including the type to clarify which player it points to.

We denote the measurements used by both players in strategy $\mathscr{S}$ by $\{M(x_Q)_a^{x_\Pi}\}$, where for the sake of clarity we have notationally separated the two parts x_Q and x_Π of the question and omitted explicit mention of the associated types t_Q and t_Π (we include the type and write $M(x_Q)_a^{\mathsf{t}_\Pi,x_\Pi}$ when it is needed for clarity). First we show that the strategy $\mathscr{S}$ is close to a strategy $\mathscr{S}'$ that performs "low-degree" measurements: upon receipt of a typed question $(\mathsf{t},x) = ((\mathsf{t}_Q,\mathsf{t}_\Pi),(x_Q,x_\Pi))$ a player first performs a measurement depending on x_Q to obtain a tuple of low-degree polynomials, and then returns evaluations of those polynomials on the subspaces specified by x_Π. This step of the argument uses the quantum soundness of the low-degree test performed in items 3. and 4 of Figure 13. Next, we "decode" this strategy to produce a strategy $\mathscr{S}''$ for $\mathcal{V}^{\text{ORAC}}$ with a high value. This step makes use of the classical soundness of the underlying PCP shown in Section 10.4. The conclusion of the theorem then follows from the soundness of $\mathcal{V}^{\text{ORAC}}$ (Theorem 9.2). We proceed with the details.

We start by showing a sequence of claims that establish approximations implied by the assumption that $\mathscr{S}^{\text{AR}}$ succeeds with probability at least $1 - \varepsilon$ in the decision procedure implemented by the decider in Figure 13.

Claim 10.25 (Global consistency check, Item 1). *On average over questions* $(\mathsf{t}_A, x_A) = ((\mathsf{t}_Q,\mathsf{t}_\Pi),(x_Q,x_\Pi))$ *sampled from the marginal distribution of* $\mu_{\mathscr{S}^{\text{AR}}}$ *on the first player it holds that*

$$M(x_Q)_a^{x_\Pi} \otimes I \simeq_\varepsilon I \otimes M(x_Q)_a^{x_\Pi} \,. \tag{132}$$

Proof. First we observe that the condition $\mathsf{t}_A = \mathsf{t}_B$ for the global consistency check, item 1 in Figure 13, holds with constant probability over the choice of a pair of questions $(\mathsf{t}_A, x_A), (\mathsf{t}_B, x_B)$ sampled according to $\mu_{\mathscr{S}^{\text{AR}}}$. Thus $\mathscr{S}$ must succeed in this test with probability $1 - O(\varepsilon)$, conditioned on the test being executed: this is because each of $\mathcal{S}^{\text{ORAC}}$ and $\mathcal{S}^{\text{PCP}}$ have a constant probability of returning a pair of questions of the same type.

Moreover, observe that conditioned on $\mathsf{t}_A = \mathsf{t}_B$ a pair of questions $((\mathsf{t}_A, x_A),(\mathsf{t}_A, x_B)) \sim \mu_{\mathscr{S}^{\text{AR}}}$ is such that $x_A = x_B = L^{\mathsf{t}_A}(z)$, where z is the sampler seed and L^{t_A} the CL function of type t_A associated with $\mathscr{S}^{\text{AR}}$. The claim then follows directly from the test and the definition of approximate consistency (Definition 5.14). $\square$

Claim 10.26 (Input consistency check, Item 2). *For all* $v \in \{A, B\}$, *on average over question pairs* $(x_A, x_B) \sim \mu_{\mathscr{S}}$ *and* $z = (y_1, \ldots, y_5, o, w) \in \mathbb{F}_q^{m'}$ *sampled uniformly at random,*

$$M(x_A, x_B)_{\alpha_v}^{\text{POINT6},z} \otimes I \simeq_\varepsilon I \otimes M(x_v, v)_{\alpha_v}^{\text{POINT}_v, y_v} \,, \tag{133}$$

where as in Remark 10.22 we made the identification $1 \leftrightarrow A$ *and* $2 \leftrightarrow B$. *Moreover, an analogous relation holds for operators acting on opposite sides of the tensor product.*

Proof. For $w = A$ and fixed $v \in \{A, B\}$ there is a constant probability that $\mathsf{t}_{Q,w} = \text{ORACLE}$, $\mathsf{t}_{Q,\overline{w}} = v$, and $(\mathsf{t}_{\Pi,w},\mathsf{t}_{\Pi,\overline{w}}) = (\text{POINT}_6, \text{POINT}_v)$. Therefore, the input consistency check in Item 2 is executed with constant probability, and $\mathscr{S}$ must pass it with probability $1 - O(\varepsilon)$, conditioned on the test being executed.

Moreover, conditioned on $\mathsf{t}_{Q,w} = \text{ORACLE}$, $\mathsf{t}_{Q,\overline{w}} = v$, and $(\mathsf{t}_{\Pi,w},\mathsf{t}_{\Pi,\overline{w}}) = (\text{POINT}_6, \text{POINT}_v)$, the distribution of $(x_{Q,w}, x_{Q,\overline{w}})$ is $((x_A, x_B), x_v)$ for $(x_A, x_B) \sim \mu_{\mathscr{S}}$ and the distribution of $(x_{\Pi,w}, x_{\Pi,\overline{w}})$ is (z, y_v) for a uniformly random $z \in \mathbb{F}_q^{m'}$. Eq. (133) then follows directly from the specification of the test and the definition of approximate consistency. The "moreover" part follows from the case $w = B$. $\square$

Claim 10.27 (Input low degree test, Item 3). *For each* $v \in \{A, B\}$ *and for each x in the support of the marginal of $\mu_{\mathscr{S}}$ on player v there exists a measurement* $\{G_{\mathbf{g}}^{x,v}\} \in \text{PolyMeas}(m, d, q)$ *such that the following*

135

hold for some $\delta_1 = O(\delta_{\mathrm{LD}}(O(\varepsilon), q, m, d, 1))$, where δ_{LD} is defined in Lemma 7.4. For all $v \in \{A, B\}$, on average over x chosen from the marginal of $\mu_{\mathcal{S}}$ on player v and $y_v \in \mathbb{F}_q^m$ sampled uniformly at random,

$$M(x, v)_\alpha^{\mathrm{POINT}_v, y_v} \otimes I \simeq_{\delta_1} I \otimes G^{x,v}_{[\mathrm{eval}_{y_v}(\cdot) = \alpha]} , \tag{134}$$

$$I \otimes M(x, v)_\alpha^{\mathrm{POINT}_v, y_v} \simeq_{\delta_1} G^{x,v}_{[\mathrm{eval}_{y_v}(\cdot) = \alpha]} \otimes I , \tag{135}$$

$$G^{x,v}_g \otimes I \simeq_{\delta_1} I \otimes G^{x,v}_g , \tag{136}$$

where we used the notation $\mathrm{eval}_{y_v}(g) = g(y_v)$ for the evaluation map.

Proof. Fix $v \in \{A, B\}$. For any $w \in \{A, B\}$ there is a constant probability that $\mathsf{t}_{Q,w} = \mathsf{t}_{Q,\overline{w}} = v$ and $(\mathsf{t}_{\Pi,w}, \mathsf{t}_{\Pi,\overline{w}}) = (\mathrm{POINT}_v, \mathrm{PLANE}_v)$. Therefore, the input low degree test in Item 3 is executed with constant probability, and $\mathscr{S}$ must pass it with probability $1 - O(\varepsilon)$, conditioned on the test being executed.

Observe that by definition the distribution of $(x_{\Pi,A}, x_{\Pi,B})$ conditioned on $\mathsf{t}_{Q,w} = \mathsf{t}_{Q,\overline{w}} = v$, uniformly random $x_Q = (x_v, v)$, and $(\mathsf{t}_{\Pi,w}, \mathsf{t}_{\Pi,\overline{w}}) = (\mathrm{POINT}_v, \mathrm{PLANE}_v)$, where $w \in \{A, B\}$ is uniformly random, is exactly the distribution of questions in the game $\mathfrak{G}^{\mathrm{LD}}$ described in Section 7.1.1, parametrized by $\mathsf{ldparams} = (q, m, d, 1)$.

For every $v \in \{A, B\}$ and question $x = L^v(z)$ in the support of the marginal distribution of $\mu_{\mathcal{S}}$ on player v let $\varepsilon_{x,v}$ be the probability that $\mathscr{S}$ is accepted in Item 3, conditioned on the test being executed and on average over $w \in \{A, B\}$. Then $\mathbb{E}[\varepsilon_{x,v}] = O(\varepsilon)$, where the expectation is taken over a uniformly random $v \in \{A, B\}$ and $x = L^v(z)$ for uniformly random z.

By definition it follows that the strategy $\mathcal{S}^{\mathrm{AR}}$ conditioned on the first part of the players' questions being $\mathsf{t}_{Q,w} = \mathsf{t}_{Q,\overline{w}} = v$ and $x_{Q,A} = x_{Q,B} = x$ is a projective strategy that succeeds with probability $1 - \varepsilon_{x,v}$ in the low-degree test $\mathcal{D}^{\mathrm{LD}}_{\mathsf{ldparams}}$ executed in Item 3.

We may thus apply Lemma 7.4 to obtain $\{G^{x,v}_g\} \in \mathrm{PolyMeas}(m, d, q)$ such that (134), (135) and (136) each hold with approximation error $O(\delta_{\mathrm{LD}}(\varepsilon_{x,v}, q, m, d, 1))$. Using that for fixed q, m, d the function $\varepsilon \mapsto \delta_{\mathrm{LD}}(\varepsilon, q, m, d, 1)$ is concave, the claim follows from Jensen's inequality. $\square$

Claim 10.28 (Proof encoding checks, Item 4)**.** *For each $x_Q = (x_A, x_B)$ in the support of $\mu_{\mathcal{S}}$ there exist measurements $\{G^{(x_A, x_B), i}_g\} \in \mathrm{PolyMeas}(m, d, q)$ for each $i \in \{3, 4, 5\}$ and*

$$\{J^{(x_A, x_B)}_{f_1, \ldots, f_5, c_0, \ldots, c_{m'}}\} \in \mathrm{PolyMeas}(m', d, q, m' + 6)$$

such that the following hold for some

$$\delta_2 = O\big(\delta_{\mathrm{LD}}(O(\varepsilon), q, m, d, 1) + \delta_{\mathrm{LD}}(O(\varepsilon), q, m', d, m' + 6)\big) .$$

First, for all $i \in \{3, 4, 5\}$, on average over $(x_A, x_B) \sim \mu_{\mathcal{S}}$ and $z = (y_1, \ldots, y_5, o, w)$ of type POINT_6 sampled uniformly at random,

$$I \otimes M(x_A, x_B)_{\alpha_i}^{\mathrm{POINT}_i, y_i} \simeq_\varepsilon M(x_A, x_B)_{\alpha_i}^{\mathrm{POINT}_6, z} \otimes I . \tag{137}$$

Second, for all $i \in \{3, 4, 5\}$ and on average over $(x_A, x_B) \sim \mu_{\mathcal{S}}$ and $y_i \in \mathbb{F}_q^m$ sampled uniformly at random,

$$M(x_A, x_B)_\alpha^{\mathrm{POINT}_i, y_i} \otimes I \simeq_{\delta_2} I \otimes G^{(x_A, x_B), i}_{[\mathrm{eval}_{y_i}(\cdot) = \alpha]} , \tag{138}$$

$$G^{(x_A, x_B), i}_g \otimes I \simeq_{\delta_2} I \otimes G^{(x_A, x_B), i}_g . \tag{139}$$

Third, for all $i \in \{1, \ldots, 5\}$ and $j \in \{0, \ldots, m'\}$, on average over $(x_A, x_B) \sim \mu_S$ and $z \in \mathbb{F}_q^{m'}$ sampled uniformly at random,

$$M(x_A, x_B)_{\alpha_1, \ldots, \alpha_5, \beta_0, \ldots, \beta_{m'}}^{\text{POINT}_6, z} \otimes I \simeq_{\delta_2} I \otimes J_{[\text{eval}_z(\cdot) = (\alpha_1, \ldots, \alpha_5, \beta_0, \ldots, \beta_{m'})]}^{(x_A, x_B)} , \tag{140}$$

$$J_{f_1, \ldots, f_5, c_0, \ldots, c_{m'}}^{(x_A, x_B)} \otimes I \simeq_{\delta_2} I \otimes J_{f_1, \ldots, f_5, c_0, \ldots, c_{m'}}^{(x_A, x_B)} . \tag{141}$$

Moreover, analogous equations to (137), (138) and (140) hold with operators acting on opposite sides of the tensor product.

Proof. The proof of the first item is similar to the proof of Claim 10.26, and we omit it.

The proof of the second and third items is similar to the proof of Claim 10.27, and we include more details. Fix an $i \in \{3, 4, 5\}$. For any $w \in \{A, B\}$ there is a constant probability that $t_{Q,w} = t_{Q,\overline{w}} = \text{ORACLE}$ and $(t_{\Pi,w}, t_{\Pi,\overline{w}}) = (\text{POINT}_i, \text{PLANE}_i)$, in which case the individual low-degree test in Item 4b is executed. Therefore, $\mathscr{S}$ must succeed in that part of the test with probability $1 - O(\varepsilon)$ conditioned on the test being executed.

Furthermore, for fixed $i \in \{3, 4, 5\}$ and uniformly random $w \in \{A, B\}$, conditioned on the test being executed for that i and w the distribution of $(x_{\Pi,A}, x_{\Pi,B})$ is exactly the distribution of questions in the game $\mathfrak{G}^{\text{LD}}$ described in Section 7.1.1, parametrized by $\text{ldparams} = (q, m, d, 1)$.

For every $i \in \{3, 4, 5\}$ and $x = (x_A, x_B)$ in the support of μ_S let $\varepsilon_{x,i}$ be the probability that $\mathscr{S}$ is accepted in Item 4b, conditioned on the test being executed for that i and on average over $w \in \{A, B\}$. Then for each i, $\mathbb{E}[\varepsilon_{x,i}] = O(\varepsilon)$, where the expectation is taken over a uniformly random $x \sim \mu_S$.

By definition of the individual low-degree test it follows from Lemma 7.4 that for every $x = (x_A, x_B)$ in the support of μ_S and $i \in \{3, 4, 5\}$ there is a measurement $\{G_g^{(x_A, x_B), i}\} \in \text{PolyMeas}(m, d, q)$ such that on average over $y_i \in \mathbb{F}_q^{m'}$ of type POINT_i sampled uniformly at random, Eq. (138) and (139) both hold with approximation error $O(\delta_{\text{LD}}(\varepsilon_{x,i}, q, m, d, 1))$. Eq (138) and (139) follow using the concavity of δ_{LD} as a function of ε.

Finally we consider the simultaneous low-degree test, Item 4c. Here as well, using that there is a constant probability that $t_{Q,w} = t_{Q,\overline{w}} = \text{ORACLE}$ and $(t_{\Pi,w}, t_{\Pi,\overline{w}}) = (\text{POINT}_6, \text{PLANE}_6)$ it follows that $\mathscr{S}^{\text{AR}}$ must succeed in that part of the test with probability $1 - O(\varepsilon)$. Using a similar argument as before it follows from Lemma 7.4 (this time for parameters $(q, m', d, m' + 6)$) that for every (x_A, x_B) there is a family of measurements $\{J_{f_1, \ldots, f_5, c_0, \ldots, c_{m'}}^{(x_A, x_B)}\} \in \text{PolyMeas}(m', d, q, m' + 6)$ such that on average over $z \in \mathbb{F}_q^{m'}$ sampled uniformly at random, Eq. (140) and (141) both hold with approximation error $O(\delta_{\text{LD}}(O(\varepsilon), q, m', d, m' + 6))$. $\qquad\square$

The families of measurements $\{G_g^{x_Q, i}\}$ and $\{J_{f_1, \ldots, f_5, c_0, \ldots, c_{m'}}^{x_Q}\}$, for x_Q in the support of μ_S and $i \in \{1, \ldots, 5\}$, whose existence follows from Claim 10.27 and Claim 10.28 have outcomes that are low-degree polynomials: for the first family, degree d polynomials $g : \mathbb{F}_q^m \to \mathbb{F}_q$, and for the second, tuples of degree d polynomials $f_i, c_j : \mathbb{F}_q^{m'} \to \mathbb{F}_q$. Recall that $m' = 5m + 5 + s$ and that an element $z \in \mathbb{F}_q^{m'}$ is written as a triple (y, o, w) with $x = (y_1, \ldots, y_5) \in \mathbb{F}_q^{5m}$, $o \in \mathbb{F}_q^5$ and $w \in \mathbb{F}_q^s$. The following claim, whose proof is based on Lemma 5.23, shows that we can reduce to a situation where the polynomials $f_1, \ldots, f_5$ returned by J are such that for each $i \in \{1, \ldots, 5\}$, f_i only depends on the y_i, and not on the entire variable z.

Claim 10.29. *For all (x_A, x_B) in the support of μ_S and degree d polynomials $g_1, \ldots, g_5 : \mathbb{F}_q^m \to \mathbb{F}_q$ and $c_0, \ldots, c_{m'} : \mathbb{F}_q^{m'} \to \mathbb{F}_q$ define*

$$\Lambda_{g_1, \ldots, g_5, c_0, \ldots, c_{m'}}^{x_A, x_B} = G_{g_1}^{x_A, 1} G_{g_2}^{x_B, 2} G_{g_3}^{(x_A, x_B), 3} \cdots G_{g_5}^{(x_A, x_B), 5} J_{c_0, \ldots, c_{m'}}^{(x_A, x_B)} G_{g_5}^{(x_A, x_B), 5} \cdots G_{g_3}^{(x_A, x_B), 3} G_{g_2}^{x_B, 2} G_{g_1}^{x_A, 1} , \tag{142}$$

where the outcomes $(f_1, \ldots, f_5)$ of the J operator in the middle have been marginalized over. Then there is a

$$\delta_3 = O\left(\left(\delta_{\mathrm{LD}}(\varepsilon, q, m', d, m' + 6) + \frac{d}{q}\right)^{1/2}\right)$$

such that

$$\delta_3 \geq \max\left\{\varepsilon, \delta_1, \delta_2\right\}$$

and on average over $(x_A, x_B) \sim \mu_S$ and $z \in \mathbb{F}_q^{m'}$ sampled uniformly at random,

$$\Lambda_{[\mathrm{eval}_z(\cdot) = (\alpha, \beta)]}^{x_A, x_B} \otimes I \simeq_{\delta_3} I \otimes J_{[\mathrm{eval}_z(\cdot) = (\alpha, \beta)]}^{x_A, x_B} . \tag{143}$$

Moreover, a similar equation holds with the operators acting on opposite sides of the tensor product.

Proof. We apply Lemma 5.23 with the following setting of parameters. The number of sets of functions k is set to 6. The question set $\mathcal{X}$ is set to the support of μ_S, and the distribution μ on it is the distribution μ_S. The sets $\mathcal{G}_i$ for $i \in \{1, \ldots, 5\}$ consist of degree d polynomials over $\mathbb{F}_q^{m'}$ that depend only on the i-th block of m variables. The set $\mathcal{G}_6$ consists of $(m' + 1)$-tuples of degree d polynomials over $\mathbb{F}_q^{m'}$.

We first verify the assumption on the sets of functions. Since all polynomials have degree at most d, by Lemma 3.20 the parameter ε in Lemma 5.23 can be set to d/q.

The family of measurements $\{A_{g_1, \ldots, g_6}^x\}$ in Lemma 5.23 is the family of measurements $\{J_{f_1, \ldots, f_5, c_0, \ldots, c_{m'}}^{(x_A, x_B)}\}$ here, where we set $g_i = f_i$ for $i \in \{1, \ldots, 5\}$ and $g_6 = (c_0, \ldots, c_{m'})$. The measurements $\{G_g^{i, x}\}$ in Lemma 5.23 are $\{G_g^{x_A, i}\}$ for $i \in \{1, 2\}$, $\{G_g^{(x_A, x_B), i}\}$ for $i \in \{3, 4, 5\}$, and $\{J_g^{(x_A, x_B)}\}$ for $i = 6$. To ensure that all polynomials are defined over the same range, we treat $g : \mathbb{F}_q^m \to \mathbb{F}_q$ that is an outcome of some $\{G_g^{i, x}\}$ as a polynomial $g' : \mathbb{F}_q^{m'} \to \mathbb{F}_q$, where the role of the m variables of g is taken by the i-th block of m variables of g'.

We verify assumption (38) in the lemma. For $i \in \{1, 2\}$ the assumption follows by combining (133) and (135) with (140) and Fact 5.21. For $i \in \{3, 4, 5\}$ we use (137) instead of (133) and (138) instead of (135). Finally, for $i = 6$ we use (141) and Fact 5.21. In these derivations, we use Fact 5.20, the triangle inequality for "$\simeq$".

The conclusion follows from Lemma 5.23, using also that $\varepsilon, \delta_1, \delta_2 = O(\delta_{\mathrm{LD}}(\varepsilon, q, m', d, m' + 6))$, as can be verified from the definition of δ_{LD} given in Lemma 7.4.

□

At this point, we have constructed measurements G and Λ that return low-degree polynomials in a similar way as is expected from the honest strategy in $\mathcal{V}_n^{\mathrm{AR}}$, as described in the proof of Theorem 10.23. These measurements can be used to specify a new strategy $\mathscr{S}'$ for the game $\mathcal{V}_n^{\mathrm{AR}}$ as follows. The shared state remains the state $|\psi\rangle$ used in $\mathscr{S}$. For $w \in \{A, B\}$, upon reception of a question (t_w, x_w) player w performs the following. If $t_w = (t_{Q,w}, t_{\Pi,w})$ is such that $t_{Q,w} = \mathrm{ORACLE}$, the player measures their share of $|\psi\rangle$ using the measurement $\Lambda^{x_{Q,w}}$ defined in Claim 10.29 to obtain a tuple $(g_1, \ldots, g_5, c_0, \ldots, c_{m'})$ of polynomials. The player then answers exactly as in the strategy described in the "completeness" part of the proof of Theorem 10.23. Similarly, if $t_{Q,w} = v \in \{A, B\}$ the player first measures their share of $|\psi\rangle$ using the measurement $G^{x_{Q,w}, v}$ from Lemma 10.27 to obtain a polynomial g as outcome; the player then answers according to the same honest strategy.

Lemma 10.30. *The strategy $\mathscr{S}'$ succeeds with probability $1 - O(\delta_3)$ in the game $\mathcal{V}_n^{\mathrm{AR}}$.*

Proof. First we establish useful consistency relations. By combining Equation (143) and Equation (140) and applying Fact 5.21 we obtain that for all $i \in \{1, \ldots, 5\}$, on average over $(x_A, x_B) \sim \mu_S$ and $z \in \mathbb{F}_q^{m'}$ sampled uniformly at random,

$$M(x_A, x_B)_{\alpha_i}^{\text{POINT}6,z} \otimes I \simeq_{\delta_3} I \otimes \Lambda_{[\text{eval}_z(\cdot)_i = \alpha_i]}^{(x_A, x_B)} , \tag{144}$$

and a similar equation holds with the operators acting on opposite sides of the tensor product. Next, combining (144) together with Equation (133) and Equation (134) it follows that for each $v \in \{A, B\}$, on average over $(x_A, x_B) \sim \mu_S$ and $z = (y_1, \ldots, y_5, o, w) \in \mathbb{F}_q^{m'}$ sampled uniformly at random,

$$G_{[\text{eval}_{y_v}(\cdot) = \alpha_v]}^{x_v, v} \otimes I \simeq_{\delta_3} I \otimes \Lambda_{[\text{eval}_z(\cdot)_v = \alpha_v]}^{(x_A, x_B)} . \tag{145}$$

From the Schwartz-Zippel lemma (Lemma 3.20) it follows that the probability that any two distinct degree d polynomials g_v (an outcome of $G^{x_v, v}$) and g_v' (an outcome of $\Lambda^{(x_A, x_B)}$) agree at a uniformly random point $y_v \in \mathbb{F}_q^m$ is at most d/q. It thus follows from (145) that for all $v \in \{A, B\}$, on average over $(x_A, x_B) \sim \mu_S$ and $y_v \in \mathbb{F}_q^M$ sampled uniformly at random,

$$G_{g_v}^{x_v, v} \otimes I \simeq_{\delta_3 + d/q} I \otimes \Lambda_{g_v}^{(x_A, x_B)} . \tag{146}$$

We now show that $\mathscr{S}'$ that is accepted by $\mathcal{V}_n^{\text{AR}}$ with high probability. We bound the probability of succeeding in each subtest.

First note that the strategy is accepted in item 1. For the G measurements, consistency follows from (136). For the Λ measurements, note first that by (143) and (141) it follows that on average over $(x_A, x_B) \sim \mu_S$,

$$\Lambda_{[\text{eval}_z(\cdot) = (\alpha, \beta)]}^{x_A, x_B} \otimes I \simeq_{\delta_3} I \otimes \Lambda_{[\text{eval}_z(\cdot) = (\alpha, \beta)]}^{x_A, x_B} . \tag{147}$$

Using that all outcomes of Λ^{x_A, x_B} are degree d polynomials and the Schwartz-Zippel lemma (Lemma 3.20) it follows that whenever a measurement of $\Lambda^{x_A, x_B} \otimes \Lambda^{x_A, x_B}$ returns distinct outcomes, the outcomes take a different value at a uniformly random z with probability at least $1 - d/q$. It then follows from (147) and the fact that $\delta_3 \geq d/q$ by definition that the strategy $\mathscr{S}'$ is accepted in item 1 with probability $O(\delta_3)$.

Next, the strategy is also accepted in the consistency check performed in item 2 due to (146), and the consistency check in item 4(a) for the same reasons as for item 1. Finally, for the low-degree tests performed in item 3 and items 4(b) and 4(c), the strategy succeeds due to consistency and the fact that, as long as both players obtain the same polynomial outcomes, they pass the low-degree tests with probability 1.

It remains to analyze the strategy's success probability in Item 5, the game check. Note that by assumption the original strategy $\mathscr{S}$ succeeds with probability $1 - O(\varepsilon)$ in that test. Using (143) and (140) together with the consistency relations established at the start of the proof, it follows that $\mathscr{S}$ and $\mathscr{S}'$ generate outcomes a_w in Item 5 that are within total variation distance $O(\delta_3)$. The lemma fellows. $\qquad\square$

We now complete the proof by a reduction to the game $\mathcal{V}_n^{\text{ORAC}}$: from the strategy $\mathscr{S}'$ we construct a symmetric strategy $\mathscr{S}'' = (|\psi\rangle, A)$ for $\mathcal{V}^{\text{ORAC}}$ by "decoding" the low-degree measurements G and Λ. The state $|\psi\rangle$ in $\mathscr{S}''$ is identical to the state used in $\mathscr{S}'$ (which is identical to the state used in $\mathscr{S}$). To begin, we define a decoding map $\text{Dec}(\cdot)$, which takes in a polynomial $g : \mathbb{F}_q^m \to \mathbb{F}_q$ and outputs a string in $\mathbb{F}_q^*$. This map is computed as follows:

- First, compute $a = \text{Dec}_\pi(g) \in \mathbb{F}_2^L$, where Dec_π is the low-degree decoding of the low-degree code defined in Section 3.4 (with $S = L = 2^\ell$, ℓ specified by pcpparams(n, T, Q, λ), and $\pi = \pi_L$ is the canonical map from Definition 3.22).

- If there exists an $a_{\text{prefix}} \in \{0,1\}^*$ of length ℓ_a such that $a = \text{enc}_\Gamma(a_{\text{prefix}}, \sqcup^{L-\ell_a})$, then $\text{Dec}(g) = a_{\text{prefix}}$. Otherwise, $\text{Dec}(g)$ is allowed to be arbitrary.

We can now define the "decoded" measurements $\{A^{x_v,v}\}$ and $\{A^{x_A,x_B}\}$ as follows:

$$A_a^{x_v,v} = G_{[\text{Dec}(\cdot)=a]}^{x_v,v} \, , \quad A_{a_A,a_B}^{x_A,x_B} = \Lambda_{[\text{Dec}(\cdot)_{A,B}=(a_A,a_B)]}^{x_A,x_B} \, . \tag{148}$$

Lemma 10.31. *The strategy $\mathscr{S}''$ succeeds with probability $1 - O(\delta_3)$ in the game $\mathcal{V}_n^{\text{ORAC}}$.*

Proof. We consider the different subtests executed by $\mathcal{D}^{\text{ORAC}}$ (see Figure 12). We start with item 2, the consistency checks. Success in the first check, item 2a, follows from the success of $\mathscr{S}'$ in the global consistency check, item 1 of $\mathcal{D}^{\text{AR}}$, the definition (148), and the fact that conditioned on $\mathsf{t}_{Q,A} = \mathsf{t}_{Q,B} = \text{ORACLE}$, the distribution of $(x_{Q,A}, x_{Q,B})$ in $\mathcal{V}_n^{\text{AR}}$ is the same as the distribution of (x_A, x_B) in $\mathcal{V}_n^{\text{ORAC}}$, conditioned on $\mathsf{t}_A = \mathsf{t}_B = \text{ORACLE}$. Similarly, success in the second check, item 2b, follows from success of $\mathscr{S}'$ in the input consistency check, item 2 of $\mathcal{D}^{\text{AR}}$.

Next we consider the game check of $\mathcal{D}^{\text{ORAC}}$, item 1. To analyze the success probability of $\mathscr{S}''$ we use that $\mathscr{S}'$ succeeds in the game check of $\mathcal{D}^{\text{AR}}$, item 5, and the soundness of the PCP, as shown in Theorem 10.21. Let p_{sound} be as in Theorem 10.21.

Let $w \in \{A, B\}$, (x_A, x_B) be in the support of $\mu_{\mathcal{S}}$, and $\Pi = (g_1, \ldots, g_5, c_0, \ldots, c_{m'})$ an outcome of Λ^{x_A,x_B} such that conditioned on that outcome being obtained by player w in the game check of $\mathcal{D}^{\text{AR}}$, $\mathcal{M}_{\text{AR}}$ accepts the pair of inputs $(\mathcal{D}, n, T, Q, q, x_A, x_B)$ and (z, a_w) with probability at least p_{sound} over the choice of a uniformly random $z \in \mathbb{F}_q^{m'}$ and $a_w = \text{eval}_z(\Pi)$.

For any such Π, the soundness statement of Theorem 10.21 states that there exist $a_A, a_B \in \{0,1\}^*$ such that $\mathcal{D}(n, x_A, x_B, a_A, a_B) = 1$ and degree d polynomials $f_A, f_B : \mathbb{F}_q^m \to \mathbb{F}_q$ such that for $v \in \{A, B\}$, $\text{Dec}(f_v) = a_v$ and f_v agrees with g_v on at least 0.8 fraction of points $z \in \mathbb{F}_q^m$. Since moreover g_v is also a degree d polynomial, it follows from the Schwartz-Zippel lemma (Lemma 3.20) that $f_v = g_v$. (Since $\delta_3 \geq d/q$ by definition, we may assume without loss of generality that $d/q < 0.2$; otherwise, the statement of the lemma is trivial.)

It follows that for any proof Π returned by $\{\Lambda_\Pi^{x_A,x_B}\}$ which is accepted with probability greater than p_{sound} in the game check of $\mathcal{D}^{\text{AR}}$ it holds that $\mathcal{D}(n, x_A, x_B, \text{Dec}(g_A), \text{Dec}(g_B)) = 1$. Using this observation we evaluate the probability q_g'' that the strategy $\mathscr{S}''$ succeeds in the game check of $\mathcal{V}^{\text{ORAC}}$. Let q_g' be the probability that $\mathscr{S}'$ succeeds in the game check of $\mathcal{D}^{\text{AR}}$.

$$
\begin{aligned}
q_g'' &= \mathop{\mathbb{E}}_{(x_A,x_B) \sim \mu_{\mathcal{S}}} \sum_{a_A,a_B : \mathcal{D}(n,x_A,x_B,a_A,a_B)=1} \langle \psi | A_{a_A,a_B}^{(x_A,x_B)} \otimes I | \psi \rangle \\
&= \mathop{\mathbb{E}}_{(x_A,x_B) \sim \mu_{\mathcal{S}}} \sum_{\Pi : \mathcal{D}(n,x_A,x_B,\text{Dec}(g_1),\text{Dec}(g_2))=1} \langle \psi | \Lambda_\Pi^{(x_A,x_B)} \otimes I | \psi \rangle \\
&\geq \mathop{\mathbb{E}}_{(x_A,x_B) \sim \mu_{\mathcal{S}}} \sum_{\Pi : \mathcal{D}(n,x_A,x_B,\text{Dec}(g_1),\text{Dec}(g_2))=1} \langle \psi | \Lambda_\Pi^{(x_A,x_B)} \otimes I | \psi \rangle \cdot \mathop{\text{Pr}}_{z \sim \mathbb{F}_q^{m'}} [V(\text{eval}_z(\Pi)) = 1] \\
&= q_g' - \mathop{\mathbb{E}}_{(x_A,x_B) \sim \mu_{\mathcal{S}}} \sum_{\Pi : \mathcal{D}(n,x_A,x_B,\text{Dec}(g_1),\text{Dec}(g_2))=0} \langle \psi | \Lambda_\Pi^{(x_A,x_B)} \otimes I | \psi \rangle \cdot \mathop{\text{Pr}}_{z \sim \mathbb{F}_q^{m'}} [V(\text{eval}_z(\Pi)) = 1] \\
&\geq q_g' - (1 - q_g'') \cdot p_{\text{sound}} \, .
\end{aligned}
$$

Rearranging terms,

$$q_g'' \geq \frac{q_g' - p_{\text{sound}}}{1 - p_{\text{sound}}} = 1 - \frac{1 - q_g'}{1 - p_{\text{sound}}} \, . \tag{149}$$

Altogether, using Lemma 10.30 we have shown that $\mathscr{S}''$ is accepted in each subtest performed by $\mathcal{D}^{\text{ORAC}}$ with probability at least $1 - O(\delta_3)$. Since every subtest occurs with constant probability, the lemma follows.

$\square$

To conclude the proof of the first part of the theorem we appeal to the soundness statement for $\mathcal{V}^{\text{ORAC}}$, given in Theorem 9.2. To obtain the bound stated in the theorem, observe that by the choice of q in Definition 10.18 it holds that $\delta_3 = \text{poly}(\varepsilon + 1/n)$.

To show the second part, the bound on entanglement, we observe that the strategy $\mathscr{S}''$ for $\mathcal{V}^{\text{ORAC}}$ constructed above uses the same entangled state $|\psi\rangle$ as the strategy $\mathscr{S}$ for $\mathcal{V}^{\text{AR}}$ that we started with; the claimed bound follows.

$\square$

11 Parallel Repetition

In each of the transformations on verifiers presented so far (introspection, oracularization, and answer reduction), the *soundness gap* of the resulting verifier is slightly degraded: while the completeness property, i.e. the property of having a PCC strategy with success probability 1, is preserved, if the starting game $\mathcal{V}_n$ has value at most $1 - \varepsilon$, the resulting game $\mathcal{V}_n'$ has value at most $1 - C\varepsilon^c$ for some universal $c, C \geq 1$. In order to apply the compression procedure recursively we need a way to restore the soundness gap after a sequence of transformations. We accomplish this using (a modification of) the technique of *parallel repetition*. Informally, this amounts to transforming a two-player game $\mathfrak{G}$ into another two-player game $\mathfrak{G}^k$ in which the verifier plays k simultaneous copies of $\mathfrak{G}$ with the players, and accepts if and only if the players' answers correspond to valid answers in each copy.

Intuitively, if the value of $\mathfrak{G}$ is $v < 1$, then one would expect the value of $\mathfrak{G}^k$ to decay exponentially with the number of repetitions k. It is not true in general that the value of $\mathfrak{G}^k$ is v^k, but exponential decay bounds on the (tensor product) value of parallel-repeated games are known for specific classes of games [JPY14, DSV15, BVY17]. In particular, it was shown in [BVY17] that the class of *anchored games* satisfies exponential-decay parallel repetition, and furthermore every game can be efficiently transformed into an equivalent anchored game. Put together, this gives a general soundness amplification procedure called "anchored parallel repetition," which we use in our compression procedure to reset the soundness gap to a fixed constant.

The parallel repetition theorems of [DSV15, Yue16] are also applicable for the purpose of soundness amplification, but are not sufficient for us. The crucial point here is that the anchored parallel repetition result of [BVY17] allows us to relate the amount of entanglement needed to play the repeated game $\mathfrak{G}^k$ to the entanglement needed to play the original anchored game $\mathfrak{G}$: roughly speaking, [BVY17] show that for $v \geq \exp(-c\varepsilon^8 \cdot k/A)$, where c is a universal constant and A an upper bound on the length of answers from the players in $\mathfrak{G}$, we have $\mathscr{E}(\mathfrak{G}^k, v) \geq \mathscr{E}(\mathfrak{G}, 1 - \varepsilon)$. On the other hand, the parallel repetition theorems of [DSV15, Yue16] only imply that $\mathscr{E}(\mathfrak{G}^k, v) \geq \log \mathscr{E}(\mathfrak{G}, 1 - \varepsilon)$,[23] which is not sufficient for our purposes.

11.1 The anchoring transformation

Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier. We present a transformation on the verifier $\mathcal{V}$, called *anchoring*, that produces another normal form verifier $\mathcal{V}^{\textsc{Anch}} = (\mathcal{S}^{\textsc{Anch}}, \mathcal{D}^{\textsc{Anch}})$.

We define the anchoring $\mathcal{V}^{\textsc{Anch}}$ of $\mathcal{V}$ by first defining a typed verifier $\tilde{\mathcal{V}}^{\textsc{Anch}} = (\tilde{\mathcal{S}}^{\textsc{Anch}}, \tilde{\mathcal{D}}^{\textsc{Anch}})$, and then detyping $\tilde{\mathcal{V}}^{\textsc{Anch}}$ using Lemma 6.18 to obtain $\mathcal{V}^{\textsc{Anch}}$. Define the type set $\mathcal{T}^{\textsc{Anch}} = \{\textsc{Game}, \textsc{Anchor}\}$ and type graph $\textsc{Game}^{\textsc{Anch}}$ the complete graph over $\mathcal{T}^{\textsc{Anch}}$ along with self-loops at each vertex.

The sampler $\tilde{\mathcal{S}}^{\textsc{Anch}}$ is a $\mathcal{T}^{\textsc{Anch}}$-typed sampler, with field size $q(n) = 2$ and the same dimension $s(n)$ as that of the sampler $\mathcal{S}$. Fix an integer $n \in \mathbb{N}$. Let $V = \mathbb{F}_2^{s(n)}$ denote the ambient space of $\mathcal{S}$ on index n. Let $L^{\textrm{A}}, L^{\textrm{B}} : V \to V$ denote the CL functions of $\mathcal{S}$ on index n. For $w \in \{\textrm{A}, \textrm{B}\}$ the associated CL functions $\{L_{\textrm{t}}^{\textsc{Anch}, w}\}$ of $\tilde{\mathcal{S}}^{\textsc{Anch}}$ are

$$L_{\textrm{t}}^{\textsc{Anch}, w} = \begin{cases} L^w & \text{if } \textrm{t} = \textsc{Game} \\ 0 & \text{if } \textrm{t} = \textsc{Anchor} . \end{cases}$$

Intuitively, when the type $\textrm{t}$ sampled for player w is $\textsc{Game}$, then they receive a question $L^w(z)$ as they would according to $\mathcal{S}$. Otherwise if $\textrm{t} = \textsc{Anchor}$, then their question is the zero string. Thus if L^w is an ℓ-level CL function, then $L_{\textsc{Game}}^{\textsc{Anch}, w}$ is also an ℓ-level CL function, and $L_{\textsc{Anchor}}^{\textsc{Anch}, w}$ is a 0-level CL function.

---

[23]The reason is due to the use of the "quantum correlated sampling lemma" of [DSV15].

The decider $\tilde{\mathcal{D}}^{\text{ANCH}}$ performs the following: on input $(n, \mathsf{t}_A, x_A, \mathsf{t}_B, x_B, a_A, a_B)$, if either t_A or t_B is equal to the type ANCHOR, then the decider accepts. Otherwise, it accepts only if $\mathcal{D}(n, x_A, x_B, a_A, a_B)$ accepts.

We define the anchoring of $\mathcal{V}$ to be the detyped verifier $\mathcal{V}^{\text{ANCH}} = \text{detype}(\tilde{\mathcal{V}}^{\text{ANCH}})$.

Proposition 11.1. *Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier where $\mathcal{S}$ is an ℓ-level sampler. Let $\mathcal{V}^{\text{ANCH}} = (\mathcal{S}^{\text{ANCH}}, \mathcal{D}^{\text{ANCH}})$ be the anchoring of $\mathcal{V}$. Then the verifier $\mathcal{V}^{\text{ANCH}}$ is a normal form verifier that satisfies the following properties: for all $n \in \mathbb{N}$,*

1. *(**Completeness**) If there is a value-1 PCC strategy for $\mathcal{V}_n$, then there is a value-1 PCC strategy for $\mathcal{V}_n^{\text{ANCH}}$.*

2. *(**Soundness**) For all $\varepsilon > 0$, if $\text{val}^*(\mathcal{V}_n^{\text{ANCH}}) \geq 1 - \varepsilon$, then $\text{val}^*(\mathcal{V}_n) \geq 1 - (4 \cdot 16^2)\varepsilon$. Furthermore,*
$$\mathscr{E}(\mathcal{V}_n^{\text{ANCH}}, 1 - \varepsilon) \geq \mathscr{E}\left(\mathcal{V}_n, 1 - (4 \cdot 16^2)\varepsilon\right).$$

3. *(**Sampler complexity**) The time and randomness complexities of the sampler $\mathcal{S}^{\text{ANCH}}$ satisfy*
$$\text{TIME}_{\mathcal{S}^{\text{ANCH}}}(n) = \text{poly}(\text{TIME}_{\mathcal{S}}(n)), \qquad \text{RAND}_{\mathcal{S}^{\text{ANCH}}}(n) = \text{RAND}_{\mathcal{S}}(n) + O(1).$$
   Furthermore the number of levels of $\mathcal{S}^{\text{ANCH}}$ is $\ell + 2$.

4. *(**Decider complexity**) The time complexity of the decider $\mathcal{D}^{\text{ANCH}}$ satisfies*
$$\text{TIME}_{\mathcal{D}^{\text{ANCH}}}(n) = \text{poly}(\text{TIME}_{\mathcal{D}}(n)).$$

5. *(**Efficient computability**) The descriptions of $\mathcal{S}^{\text{ANCH}}$ and $\mathcal{D}^{\text{ANCH}}$ can be computed in polynomial time from the descriptions of $\mathcal{S}$ and $\mathcal{D}$, respectively. In particular, the sampler $\mathcal{S}^{\text{ANCH}}$ only depends on the sampler $\mathcal{S}$.*

Proof. We analyze the completeness, soundness, and complexity properties of the typed verifier $\tilde{\mathcal{V}}^{\text{ANCH}}$; the corresponding properties of the detyped verifier $\mathcal{V}^{\text{ANCH}}$ follow from Lemma 6.18 and the fact that the type set $\mathcal{T}^{\text{ANCH}}$ has size 2.

Fix an index $n \in \mathbb{N}$. For the completeness property, let $\mathscr{S}$ be a value-1 PCC strategy for $\mathcal{V}_n$. We define a value-1 PCC strategy $\mathscr{S}^{\text{ANCH}}$ for $\tilde{\mathcal{V}}_n^{\text{ANCH}}$: whenever a player receives the ANCHOR type as a question type, they perform the trivial measurement (i.e. measure the identity operator). Otherwise, the player performs the same measurement as in $\mathscr{S}$. This is clearly value-1 and PCC. Item 1 follows from this and Lemma 6.18.

For the soundness property, we observe that if a strategy $\mathscr{S}^{\text{ANCH}}$ has value $1 - \varepsilon$ in $\tilde{\mathcal{V}}_n^{\text{ANCH}}$, then
$$1 - \varepsilon = 1 - \gamma + \gamma p$$
where p is the value of $\mathscr{S}^{\text{ANCH}}$ in the game $\mathcal{V}_n$, and $\gamma = 1/4$ is the probability that neither player receives the question type ANCHOR; this follows from the distribution associated with the typed sampler $\tilde{\mathcal{S}}^{\text{ANCH}}$. Thus $\mathscr{S}^{\text{ANCH}}$ has value $1 - \varepsilon/\gamma$ in $\mathcal{V}_n$, and thus $\text{val}^*(\mathcal{V}_n) \geq 1 - 4\varepsilon$. Item 2 follows from this and Lemma 6.18.

Items 3, 4, and 5 are straightforward and also follow from Lemma 6.18. $\square$

11.2 Parallel repetition of anchored games

We present a second transformation on normal form verifiers that amounts to performing parallel repetition. Fix a function $k : \mathbb{N} \to \mathbb{N}$, and let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier where $\mathcal{S}$ is an ℓ-level sampler with dimension $s(n)$. Let $\mathcal{K}$ denote a 1-input Turing machine that computes a function $k : \mathbb{N} \to \mathbb{N}$ (i.e. the inputs and outputs are interpreted as positive integers). The $\mathcal{K}$-*fold parallel repeated verifier* $\mathcal{V}^{\text{REP}} = (\mathcal{S}^{\text{REP}}, \mathcal{D}^{\text{REP}})$ is defined as follows.

Sampler. The sampler $\mathcal{S}^{\text{REP}}$ is an ℓ-level sampler. On index n, it has field size $q(n) = 2$ and dimension $s^{\text{REP}}(n) = k(n)s(n)$. We treat the ambient space V^{REP} of $\mathcal{S}^{\text{REP}}$ on index n as the $k(n)$-fold direct sum of the ambient space V of $\mathcal{S}$ on index n. For all integers $n \in \mathbb{N}$, $w \in \{A, B\}$, we define the CL functions $L^{\text{REP}, w} : V^{\text{REP}} \to V^{\text{REP}}$ as follows:

$$L^{\text{REP}, w} = \bigoplus_{i=1}^{k(n)} L^w ,$$

where $L^A, L^B : V \to V$ are the CL functions of the sampler $\mathcal{S}$ on index n. The CL functions $L^{\text{REP}, w}$ are ℓ-level; the j-th factor spaces $V_{j,u}^{\text{REP}, w}$ of $L^{\text{REP}, w}$ are defined as

$$V_{j,u}^{\text{REP}, w} = \bigoplus_{i=1}^{k(n)} V_{j, u_i}^w$$

for all $u = (u_1, \ldots, u_{k(n)})$ where V_{j, u_i}^w is the j-th factor space of L^w with prefix $u_i \in V$. In other words, the sampler $\mathcal{S}^{\text{REP}}$ is a $k(n)$-fold product of the sampler $\mathcal{S}$.

Decider. The decider $\mathcal{D}^{\text{REP}}$ is defined as follows: on input (n, x, y, a, b) where x, y, a, b are $k(n)$-tuples of questions and answers, respectively, accept if and only if $\mathcal{D}(n, x_i, y_i, a_i, b_i)$ accepts for all $i \in \{1, \ldots, k(n)\}$.

Theorem 11.2 (Anchored parallel repetition [BVY17])**.** *There exists a universal constant $c > 0$ such that the following holds. Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier where $\mathcal{S}$ is an ℓ-level sampler. Let $\hat{\mathcal{V}} = (\hat{\mathcal{S}}, \hat{\mathcal{D}})$ denote the anchoring of $\mathcal{V}$. Let $k : \mathbb{N} \to \mathbb{N}$ be a function computed by a Turing machine $\mathcal{K}$. Then the $\mathcal{K}$-fold repeated verifier $\hat{\mathcal{V}}^{\text{REP}}$ (called the $\mathcal{K}$-fold anchored repetition of $\mathcal{V}$) is a normal form verifier that satisfies the following properties: for all $n \in \mathbb{N}$,*

1. *(**Completeness**) If $\mathcal{V}_n$ has a value-1 PCC strategy, then $\hat{\mathcal{V}}^{\text{REP}}$ has a value-1 PCC strategy.*

2. *(**Soundness**) For all $\varepsilon > 0$, for all*

$$v > \exp\left(-\frac{c\, \varepsilon^8\, k(n)}{\text{TIME}_{\mathcal{D}}(n)} \right) ,$$

   if $\text{val}^*(\hat{\mathcal{V}}_n^{\text{REP}}) > v$ then $\text{val}^*(\mathcal{V}_n) \geq 1 - \varepsilon$ and furthermore

$$\mathscr{E}(\hat{\mathcal{V}}_n^{\text{REP}}, v) \geq \mathscr{E}(\mathcal{V}_n, 1 - \varepsilon) .$$

3. *(**Sampler complexity**) The time and randomness complexities of $\hat{\mathcal{S}}^{\text{REP}}$ satisfy*

$$\text{TIME}_{\hat{\mathcal{S}}^{\text{REP}}}(n) = O(\text{TIME}_{\mathcal{K}}(n) + k(n) \cdot \text{TIME}_{\mathcal{S}}(n)) ,$$
$$\text{RAND}_{\hat{\mathcal{S}}^{\text{REP}}}(n) = O(k(n) \cdot \text{RAND}_{\mathcal{S}}(n)) .$$

   Furthermore, the number of levels of sampler $\hat{\mathcal{S}}^{\text{REP}}$ is $\ell + 2$.

4. *(**Decider complexity**) The time complexity of $\hat{\mathcal{D}}^{\text{REP}}$ satisfies*

$$\text{TIME}_{\hat{\mathcal{D}}^{\text{REP}}}(n) = O(\text{TIME}_{\mathcal{K}}(n) + k(n) \cdot \text{TIME}_{\mathcal{D}}(n)) .$$

5. (***Efficient computability***) *There exists a* 2-*input Turing machine* ComputeRepeatedVerifier *that on input* $(\mathcal{V}, \mathcal{K})$, *where* $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ *is a normal form verifier and* $\mathcal{K}$ *is a Turing machine, outputs a normal form verifier* $\hat{\mathcal{V}}^{\mathrm{REP}} = (\hat{\mathcal{S}}^{\mathrm{REP}}, \hat{\mathcal{D}}^{\mathrm{REP}})$ *that is the* $\mathcal{K}$-*fold anchored repetition of* $\mathcal{V}$. *The time complexity of* ComputeRepeatedVerifier *is* $\mathrm{poly}(|\mathcal{V}|, |\mathcal{K}|)$. *Furthermore, the description of the sampler* $\hat{\mathcal{S}}^{\mathrm{REP}}$ *only depends on the sampler* $\mathcal{S}$ *and* $\mathcal{K}$.

Proof. Item 1 follows from the following straightforward observation: if $\mathscr{S} = (\psi, A, B)$ is a value-1 PCC strategy for $\mathcal{V}_n$, then the strategy where the players share $k(n)$ copies of $|\psi\rangle$, and for the i-th instance of the game $\hat{\mathcal{V}}_n$, the players use strategy $\mathscr{S}$ on the i-th copy of $|\psi\rangle$ (and performing the identity measurement whenever they receive the ANCHOR type). It is straightforward to check that this strategy has value 1 and is PCC.

To show Item 2 we apply [BVY17, Theorem 17]. The exponential decay bound on the value of $\hat{\mathcal{V}}_n^k$ presented in [BVY17] depends on the answer length of the players in the original game $\mathcal{V}_n$. By Definition 5.28, this answer length is restricted to $\{0,1\}^{\mathsf{TIME}_{\mathcal{D}}(n)}$, so the claimed bound follows.

Item 3 follows from the fact that computing the direct sum of $k(n)$ CL functions and factor spaces of the "single-copy" sampler $\hat{\mathcal{S}}$ requires $k(n)$ times the complexity of the "single-copy" sampler, and the complexity of $\hat{\mathcal{S}}$ follows from Proposition 11.1. The dependence on the complexity of $\mathcal{K}$ comes from the sampler $\hat{\mathcal{S}}^{\mathrm{REP}}$ having to compute the function $k(n)$. Since the CL functions of the sampler $\hat{\mathcal{S}}$ are $(\ell + 2)$-level (by Proposition 11.1), and taking the direct sum of CL functions does not increase the number of levels (by Lemma 4.6), the CL functions $L^{\mathrm{REP}, w}$ are $(\ell + 2)$-level.

Item 4 follows from the repeated decider having to run $k(n)$ instances of the decider $\mathcal{D}^{\mathrm{ANCH}}$, and the complexity of $\mathcal{D}^{\mathrm{ANCH}}$ follows from Proposition 11.1. which in turn runs an instance of the original decider $\mathcal{D}$.

Item 5 follows from the fact that (a) the description of the repeated sampler only depends on the the description of the sampler $\mathcal{S}$ and the description of the Turing machine $\mathcal{K}$, and (b) the description of the repeated decider only depends on $\mathcal{D}$ and $\mathcal{K}$. $\qquad\square$

12   Gap-preserving Compression

We combine the transformations from the previous sections (the introspection games, oracularization, answer reduction, and parallel repetition) to obtain our main technical result, a *gap-preserving compression theorem* for normal form verifiers.

We give a high level explanation of the compression theorem. The theorem is parametrized by an integer λ which controls upper bounds on the following quantities: the randomness and time complexities of the verifier $\mathcal{V}$ to be compressed (i.e. the "input verifier") and the description length of the verifier $\mathcal{V}$. The upper bounds specified by λ are encapsulated in the following definition.

Definition 12.1. Let λ be an integer, and let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ be a normal form verifier. We say that the verifier $\mathcal{V}$ is λ-*bounded* if

- The number of levels of $\mathcal{S}$ is at most 8,

- The description length of $\mathcal{V}$, denoted by $|\mathcal{V}|$ and equal to the sum of the description lengths of $\mathcal{S}$ and $\mathcal{D}$, is at most λ, and

- For all $n \in \mathbb{N}$, $\mathsf{RAND}_{\mathcal{S}}(n)$, $\mathsf{TIME}_{\mathcal{S}}(n)$ and $\mathsf{TIME}_{\mathcal{D}}(n)$ are all at most $(\lambda n)^{\lambda}$.

The compression theorem states the existence of an efficient "compression procedure" Compress that achieves the following. Given as input a λ-bounded verifier $\mathcal{V}$, the procedure returns a "compressed verifier" $\mathcal{V}^{\textsc{Compr}}$ such that the n-th game $\mathcal{V}_n^{\textsc{Compr}}$ simulates the N-th game $\mathcal{V}_N$ for $N = 2^n$, and furthermore the time and randomness complexities of $\mathcal{V}_n^{\textsc{Compr}}$ are exponentially smaller than that of $\mathcal{V}_N$.

Theorem 12.2 (Gap-preserving compression of normal form verifiers). *There is a universal constant γ such that the following holds. For all $\lambda \in \mathbb{N}$ there exist*

- *A Turing machine* Compress *that, when given a normal form verifier $\mathcal{V}$ as input, has time complexity* $\mathrm{poly}(|\mathcal{V}|, \log \lambda)$. *Furthermore, the description of* Compress *is computable from the binary representation of λ in time* $\mathrm{polylog}(\lambda)$.

- *A 8-level sampler $\mathcal{S}^{\textsc{Compr}}$ with randomness complexity* $\mathsf{RAND}_{\mathcal{S}^{\textsc{Compr}}}(n) = \mathrm{poly}(n, \lambda)$, *time complexity* $\mathsf{TIME}_{\mathcal{S}^{\textsc{Compr}}}(n) = \mathrm{poly}(n, \lambda)$, *and description length* $|\mathcal{S}^{\textsc{Compr}}| = \mathrm{polylog}(\lambda)$. *Furthermore, the description of $\mathcal{S}^{\textsc{Compr}}$ is computable from the binary representation of λ in time* $\mathrm{polylog}(\lambda)$.

For normal form verifiers $\mathcal{V}$, the Turing machine Compress *on input $\mathcal{V}$ always returns the description of a normal form verifier $\mathcal{V}^{\textsc{Compr}} = (\mathcal{S}^{\textsc{Compr}}, \mathcal{D}^{\textsc{Compr}})$ such that the decider $\mathcal{D}^{\textsc{Compr}}$ has description length $|\mathcal{D}^{\textsc{Compr}}| = \mathrm{poly}(|\mathcal{V}|, \log \lambda)$ and time complexity $\mathsf{TIME}_{\mathcal{D}^{\textsc{Compr}}}(n) = \mathrm{poly}(n, |\mathcal{V}|, \lambda)$. Furthermore, supposing that $\mathcal{V}$ is a λ-bounded normal form verifier, then for all $n \geq \gamma$ and $N = 2^n$, the verifier $\mathcal{V}^{\textsc{Compr}}$ satisfies the following.*

1. (***Completeness***) *If $\mathcal{V}_N$ has a value-1 PCC strategy, then $\mathcal{V}_n^{\textsc{Compr}}$ has a value-1 PCC strategy.*

2. (***Soundness***) *If* $\mathrm{val}^*(\mathcal{V}_N) \leq \frac{1}{2}$, *then* $\mathrm{val}^*(\mathcal{V}_n^{\textsc{Compr}}) \leq \frac{1}{2}$.

3. (***Entanglement lower bound***) $\mathscr{E}(\mathcal{V}_n^{\textsc{Compr}}, \frac{1}{2}) \geq \max\left\{ \mathscr{E}(\mathcal{V}_N, \frac{1}{2}), \frac{1}{2} \cdot 2^{(\lambda N)^{\lambda}} \right\}.$

12.1 Proof of Theorem 12.2

Recall the following Turing machines.

1. ComputeIntroVerifier takes as input a tuple $(\mathcal{V}, \lambda, \ell)$ and returns a description of the (detyped) introspective verifier $\mathcal{V}^{\mathrm{INTRO}}$ corresponding to the verifier $\mathcal{V}$ and parameters (λ, ℓ) (see Item 4 of Theorem 8.6).

2. ComputeARVerifier takes input $(\mathcal{V}^{\mathrm{INTRO}}, \lambda, \mu, \sigma)$ and returns a description of the answer reduced verifier $\mathcal{V}^{\mathrm{AR}}$ corresponding to $\mathcal{V}^{\mathrm{INTRO}}$ and parameters (λ, μ, σ) (see Item 4 of Theorem 10.23).

3. ComputeRepeatedVerifier takes input $(\mathcal{V}, \mathcal{K})$ and returns the $\mathcal{K}$-fold repeated verifier $\hat{\mathcal{V}}^{\mathrm{REP}}$ corresponding to $\mathcal{V}$, where $\mathcal{K}$ is a 1-input Turing machine computing a function $k(n)$ (see Item 5 of Theorem 11.2).

We specify the Turing machine Compress in Figure 14. The Turing machine depends on the parameter λ as well as universal constants c_{INTRO} and c_{REP} which are specified in the proof of Theorem 12.2.

Input: the description of a normal form verifier $\mathcal{V}$.

1. Compute $\mathcal{V}^{(1)} = \mathsf{ComputeIntroVerifier}(\mathcal{V}, \lambda, \ell)$ where $\ell = 8$.

2. Compute $\mathcal{V}^{(2)} = \mathsf{ComputeARVerifier}(\mathcal{V}^{(1)}, \lambda_{\mathrm{INTRO}}, \mu, \sigma)$ where $\mu = \sigma = \lambda_{\mathrm{INTRO}} = c_{\mathrm{INTRO}} \lambda^{c_{\mathrm{INTRO}}}$.

3. Compute $\mathcal{V}^{(3)} = \mathsf{ComputeRepeatedVerifier}(\mathcal{V}^{(2)}, \mathcal{K})$ where $\mathcal{K}$ is a 1-input Turing machine computing the function $k(n) = c_{\mathrm{REP}} (\lambda \cdot n)^{c_{\mathrm{REP}}}$.

4. Return $\mathcal{V}^{\mathrm{COMPR}} = \mathcal{V}^{(3)}$.

Figure 14: The Turing machine Compress, parameterized by integer λ.

Lemma 12.3. *The time complexity of the Turing machine* Compress *parameterized by λ on input $\mathcal{V}$ is* $\mathrm{poly}(|\mathcal{V}|, \log \lambda)$. *The description of* Compress *is computable from the binary description of λ in time* $\mathrm{polylog}(\lambda)$. *In particular, the description length of* Compress *is* $\mathrm{polylog}(\lambda)$.

Proof. By Theorem 8.6, Theorem 10.23 and Theorem 11.2 respectively it follows that the time complexity of each of the three steps of Figure 14 is $\mathrm{poly}(|\mathcal{V}|, \log \lambda)$. For the first step, this uses that $\ell = 8$ is a constant, and for the third step, that the Turing machine $\mathcal{K}$ can be specified using $\mathrm{poly} \log(\lambda)$ bits.

The description of the Turing machine Compress consists of the descriptions of the Turing machines ComputeIntroVerifier, ComputeARVerifier, ComputeRepeatedVerifier, and $\mathcal{K}$, along with the computation of the parameters μ, σ, and the simulation of these Turing machines. The first three Turing machines are fixed, universal objects, while the Turing machine $\mathcal{K}$ and the parameters μ, σ depend on the binary representation of λ. In particular, one can take $\mathcal{K}$ to be a Turing machine that performs repeated squaring and multiplication of its input n to compute $k(n)$; the complexity of this is $\mathrm{polylog}(n, \lambda)$. Aside from the binary description of λ, the Turing machine $\mathcal{K}$ is some fixed Turing machine that doesn't depend on any other parameters. Thus $|\mathcal{K}| \leq \mathrm{polylog}\, \lambda$.

Therefore the description of Compress can be computed efficiently in polynomial time from the binary description of λ. The bound on the description length follows. $\square$

Lemma 12.4. *Let $\lambda \in \mathbb{N}$, and let* Compress *be the Turing machine parameterized by λ specified in Figure 14. For all verifier $\mathcal{V}$, the output of* Compress$(\mathcal{V})$ *is a normal form verifier $\mathcal{V}^{\text{COMPR}} = (\mathcal{S}^{\text{COMPR}}, \mathcal{D}^{\text{COMPR}})$ such that $\mathcal{S}^{\text{COMPR}}$ does not depend on $\mathcal{V}$ but only on the parameter λ.*

Proof. The verifier $\mathcal{V}^{\text{COMPR}} = \mathcal{V}^{(3)}$ is a normal form verifier because ComputeIntroVerifier always returns a normal form verifier (even if the input verifier is not; See Remark 8.8), and by Theorem 10.23 and Theorem 11.2 the remaining two steps preserve normal form. The introspected verifier computed in Step 1 of Figure 14 has a sampler $\mathcal{S}^{\text{INTRO}}$ that only depends on the parameter λ, as implied by Lemma 8.1. The samplers of the subsequent verifiers (the answer reduced verifier from Step 2 and the repeated verifier from Step 3) only depend on $\mathcal{S}^{\text{INTRO}}$ and λ; for $\mathcal{S}^{\text{AR}}$ this is stated in Item 2 of Theorem 10.23, and for $\mathcal{S}^{\text{REP}}$ it is stated in Item 5 of Theorem 11.2. $\qquad\square$

Proof of Theorem 12.2. Fix an integer $\lambda \in \mathbb{N}$ and let Compress be the Turing machine specified in Figure 14 parametrized by λ. We evaluate the parameters of the verifiers generated in each step of the Compress procedure.

Input verifier. Let $\mathcal{V} = (\mathcal{S}, \mathcal{D})$ denote the "input verifier" to be compressed. We specify its properties in terms of the index N (which should be thought of as $N = 2^n$).

Description lengths	$\begin{array}{l} |\mathcal{S}| \\ |\mathcal{D}| \end{array}$
$\text{TIME}_{\mathcal{S}}(N)$	$(\lambda N)^{\lambda}$
$\text{RAND}_{\mathcal{S}}(N)$	$(\lambda N)^{\lambda}$
$\text{TIME}_{\mathcal{D}}(N)$	$(\lambda N)^{\lambda}$
Sampler levels	$\ell \leq 8$
Completeness	$\mathcal{V}_N$ has a value-1 PCC strategy
Soundness	$\text{val}^*(\mathcal{V}_N) \leq \frac{1}{2}$
Entanglement	$\mathscr{E}(\mathcal{V}_N, \frac{1}{2})$

Figure 15: Parameters and properties of the input verifier $\mathcal{V}$.

All bounds in the table in Figure 15 (except for the description length bounds) follow from the assumptions on the input verifier in Theorem 12.2. Specifically, the bounds on $\text{TIME}_{\mathcal{S}}(N)$, $\text{RAND}_{\mathcal{S}}(N)$, $\text{TIME}_{\mathcal{D}}(N)$, and the number of sampler levels express the assumption that $\mathcal{V}$ is λ-bounded. The "Completeness" entry indicates that in the "Completeness" analysis of the compressed verifier $\mathcal{V}^{\text{COMPR}}$, we assume the game $\mathcal{V}_N$ has a value-1 PCC strategy. The "Soundness" entry of the table indicates that in the "Soundness" analysis of the compressed verifier, we assume the value of $\mathcal{V}_N$ is at most $1/2$. The "Entanglement" entry of the table indicates the assumed lower bound on the Schmidt rank of the entangled state used by any strategy that succeeds with probability greater than $\frac{1}{2}$ in the game $\mathcal{V}_N$.

Introspection. The verifier $\mathcal{V}^{(1)} = (\mathcal{S}^{(1)}, \mathcal{D}^{(1)})$ is the introspective verifier corresponding to $\mathcal{V}$. We state its parameters and properties in Figure 16 in terms of an index $n \in \mathbb{N}$.

Description lengths	$\lvert \mathcal{S}^{(1)} \rvert \leq \mathrm{poly}(\log \lambda)$
	$\lvert \mathcal{D}^{(1)} \rvert \leq \mathrm{poly}(\lvert \mathcal{V} \rvert, \log \lambda)$
$\mathrm{TIME}_{\mathcal{S}^{(1)}}(n)$	$\mathrm{polylog}\left((\lambda N)^{\lambda}\right)$
$\mathrm{RAND}_{\mathcal{S}^{(1)}}(n)$	$\mathrm{polylog}\left((\lambda N)^{\lambda}\right)$
$\mathrm{TIME}_{\mathcal{D}^{(1)}}(n)$	$\mathrm{poly}\left((\lambda N)^{\lambda}\right)$
Sampler levels	4
Completeness	$\mathcal{V}_N$ has value-1 PCC strategy $\Rightarrow \mathcal{V}_n^{(1)}$ has value-1 PCC strategy
Soundness	$\mathrm{val}^*(\mathcal{V}_n^{(1)}) > 1 - \varepsilon_1 \Rightarrow \mathrm{val}^*(\mathcal{V}_N) \geq 1 - \delta_1(\varepsilon_1, n)$
Entanglement	$\mathscr{E}(\mathcal{V}_n^{(1)}, 1 - \varepsilon_1) \geq \max\left\{ \mathscr{E}(\mathcal{V}_N, 1 - \delta_1), (1 - \delta_1) 2^{(\lambda N)^{\lambda}} \right\}$
	for $\delta_1 = \delta_1(\varepsilon_1, n)$

Figure 16: Parameters and properties of the introspective verifier $\mathcal{V}^{(1)}$.

The bounds on description length come from the following: by Lemma 8.1 since here $\ell = 8$ the sampler $\mathcal{S}^{(1)}$ only depends on the parameter λ and has description length $\mathrm{poly} \log(\lambda)$. The decider $\mathcal{D}^{(1)}$ depends on the binary representation of λ, as well as the descriptions of the sampler $\mathcal{S}$ and decider $\mathcal{D}$. The description length $\mathcal{D}^{(1)}$ follows from Item 4 in Theorem 8.6. The number of levels, time and randomness complexities of the sampler and decider are specified by Theorem 8.6 as well. As expressed in the statement of Theorem 12.2, these bounds do *not* depend on any assumption about the input verifier $\mathcal{V}$; instead, they only depend on the index n and the parameter λ.

The statements in the remaining rows of the table assume that the input verifier $\mathcal{V}$ is λ-bounded. The statement in the "Completeness" entry follows from Theorem 8.6; the "Soundness" and "Entanglement" entries follow from Theorem 8.9, from which it follows that it suffices to choose a function $\delta_1(\varepsilon_1, n) = a_1 \left(\varepsilon_1 + \frac{1}{(\lambda 2^n)^{\lambda}} \right)^{b_1}$ for some $a_1, b_1 > 0$ that depend on the number of levels of the input sampler $\mathcal{S}$, which by the λ-bounded assumption is at most 8.

We now specify the universal constant $c_{\mathrm{INTRO}} > 0$. Let it be such that the following inequalities hold for all $n, \lambda \in \mathbb{N}$, under the assumption that $\mathcal{V}$ is λ-bounded: letting $\lambda_{\mathrm{INTRO}} = c_{\mathrm{INTRO}} \cdot \lambda^{c_{\mathrm{INTRO}}}$,

$$\max \left\{ \mathrm{TIME}_{\mathcal{S}^{(1)}}(n), \ \mathrm{RAND}_{\mathcal{S}^{(1)}}(n) \right\} \leq \left(\lambda_{\mathrm{INTRO}} \, n \right)^{\lambda_{\mathrm{INTRO}}},$$
$$\mathrm{TIME}_{\mathcal{D}^{(1)}}(n) \leq \left(\lambda_{\mathrm{INTRO}} \, N \right)^{\lambda_{\mathrm{INTRO}}},$$
$$\lvert \mathcal{D}^{(1)} \rvert \leq \lambda_{\mathrm{INTRO}}.$$

Answer reduction. Let $\mathcal{V}^{(2)} = (\mathcal{S}^{(2)}, \mathcal{D}^{(2)})$ denote the answer reduced verifier corresponding to $\mathcal{V}^{(1)}$ and parameters $\lambda = \lambda_{\mathrm{INTRO}}$ and $\sigma = \mu = \lambda_{\mathrm{INTRO}}$. The table in Figure 17 gives the properties of $\mathcal{V}^{(2)}$.

Description lengths	$|\mathcal{S}^{(2)}| \leq \text{polylog}(\lambda)$
	$|\mathcal{D}^{(2)}| \leq \text{poly}(|\mathcal{V}|, \log \lambda)$
$\text{TIME}_{\mathcal{S}^{(2)}}(n)$	$\text{polylog}\left((\lambda N)^{\lambda}\right)$
$\text{RAND}_{\mathcal{S}^{(2)}}(n)$	$\text{polylog}\left((\lambda N)^{\lambda}\right)$
$\text{TIME}_{\mathcal{D}^{(2)}}(n)$	$\text{poly}\left(\log\left((\lambda N)^{\lambda}\right), |\mathcal{D}|\right)$
Sampler levels	6
Completeness	$\mathcal{V}_n^{(1)}$ has value-1 PCC strategy $\Rightarrow \mathcal{V}_n^{(2)}$ has value-1 PCC strategy
Soundness	$\text{val}^*(\mathcal{V}_n^{(2)}) > 1 - \varepsilon_2 \Rightarrow \text{val}^*(\mathcal{V}_n^{(1)}) \geq 1 - \delta_2(\varepsilon_2, n)$
Entanglement	$\mathscr{E}(\mathcal{V}_n^{(2)}, 1 - \varepsilon_2) \geq \mathscr{E}(\mathcal{V}_n^{(1)}, 1 - \delta_2(\varepsilon_2, n))$

Figure 17: Parameters and properties of the answer reduced verifier $\mathcal{V}^{(2)}$.

The statements and bounds in Figure 17 only depend on the bounds stated for verifier $\mathcal{V}^{(1)}$ in Figure 16 but not on the λ-bounded property of the input verifier $\mathcal{V}$. Note that by the choice of c_{INTRO} made above the complexity bounds on $\mathcal{V}^{(1)}$ satisfy assumption (130). The bounds on the description lengths then follow from Item 4 in Theorem 10.23, together with Item 2 to justify that $\mathcal{S}^{(2)}$ only depends on λ and $\mathcal{S}^{(1)}$ but not on $\mathcal{D}^{(1)}$ (and hence not on $\mathcal{D}$).

The bounds on the number of sampler levels and the time and randomness complexities follow from Items 2 and 3 of Theorem 10.23.

The statement in the "Completeness" entry follows from Theorem 10.23 and assumption (130). The "Soundness" and "Entanglement" entries follow from Theorem 10.24 and assumption (130). It follows from Theorem 10.24 that one can choose $\delta_2(\varepsilon_2, n) = a_2 \left(\varepsilon_2 + \frac{1}{n}\right)^{b_2}$ for some universal constants $a_2, b_2 > 0$.

Parallel repetition. Let $\mathcal{V}^{(3)} = \left(\mathcal{S}^{(3)}, \mathcal{D}^{(3)}\right)$ denote the $\mathcal{K}$-fold anchored repetition (see Section 11.2) of $\mathcal{V}^{(2)}$, where $\mathcal{K}$ is a 1-input Turing machine computing the function $k(n) = c_{\text{REP}}(\lambda \cdot n)^{c_{\text{REP}}}$. As already argued in the proof of Lemma 12.3, there is an explicit choice for $\mathcal{K}$ such that $|\mathcal{K}| \leq \text{polylog}\,\lambda$.

We state the parameters and properties of $\mathcal{V}^{(3)}$ in Figure 18.

Description lengths	$|\mathcal{S}^{(3)}| \leq \text{polylog}(\lambda)$
	$|\mathcal{D}^{(3)}| \leq \text{poly}(|\mathcal{V}|, \log \lambda)$
$\text{TIME}_{\mathcal{S}^{(3)}}(n)$	$\text{poly}\left(k(n), \log\left((\lambda N)^{\lambda}\right)\right)$
$\text{RAND}_{\mathcal{S}^{(3)}}(n)$	$\text{poly}\left(k(n), \log\left((\lambda N)^{\lambda}\right)\right)$
$\text{TIME}_{\mathcal{D}^{(3)}}(n)$	$\text{poly}\left(k(n), \log\left((\lambda N)^{\lambda}\right), |\mathcal{D}|\right)$
Sampler levels	8
Completeness	$\mathcal{V}_n^{(2)}$ has value-1 PCC strategy $\Rightarrow \mathcal{V}_n^{(3)}$ has value-1 PCC strategy
Soundness	$\text{val}^*(\mathcal{V}_n^{(3)}) > 1 - \varepsilon_3 \Rightarrow \text{val}^*(\mathcal{V}_n^{(2)}) \geq 1 - \delta_3(\varepsilon_3)$
Entanglement	$\mathscr{E}(\mathcal{V}_n^{(3)}, 1 - \varepsilon_3) \geq \mathscr{E}(\mathcal{V}_n^{(2)}, 1 - \delta_3(\varepsilon_3))$

Figure 18: Parameters and properties of the parallel repeated verifier $\mathcal{V}^{(3)}$.

The statements and bounds in Figure 18 only depend on the bounds stated for verifier $\mathcal{V}^{(2)}$ in Figure 17, and again do not depend on whether the input verifier $\mathcal{V}$ is λ-bounded.

The bounds on description length follow from Item 5 of Theorem 11.2, the description length bounds of $\mathcal{V}^{(2)}$ stated in Figure 17, and the aforementioned bound on the description length of $\mathcal{K}$. The time and randomness complexities of $\mathcal{S}^{(3)}$ and $\mathcal{D}^{(3)}$ follow from Items 3 and 4 of Theorem 11.2 and the complexity upper bounds on $\mathcal{S}^{(2)}$ and $\mathcal{D}^{(2)}$ specified in Figure 17. The statement in the "completeness" entry follows from Item 1 of Theorem 11.2. The "soundness" and "entanglement" entries follow from Item 2 of Theorem 11.2 for some

$$\delta_3(\varepsilon_3) = \left(\frac{a_3 \mathsf{TIME}_{\mathcal{D}^{(2)}}(n) \log \frac{1}{1-\varepsilon_3}}{k(n)} \right)^{1/8} , \tag{150}$$

for some universal constant $a_3 > 0$.

Putting everything together. The verifier $\mathcal{V}^{\mathrm{COMPR}}$ is $\mathcal{V}^{(3)}$. We now put together the bounds and parameters from Figures 15, 16, 17, and 18 to obtain the stated conclusions of Theorem 12.2.

The universal constants c_{REP} and γ are specified in (153) and (154) respectively. The claimed time complexity and computability of the Turing machine Compress follow from Lemma 12.3. The claimed number of levels, time and randomness complexity of the sampler $\mathcal{S}^{\mathrm{COMPR}}$ follows from the complexity bounds in Figures 16, 17, and 18, which in turn only depend on the parameter λ. The description length of $\mathcal{S}^{\mathrm{COMPR}}$ follows from the description length bound on $\mathcal{S}^{(3)}$. In particular the sampler $\mathcal{S}^{\mathrm{COMPR}}$ is independent of the input verifier $\mathcal{V}$, as shown in Lemma 12.4. Finally, the claimed time complexity to compute the description of $\mathcal{S}^{\mathrm{COMPR}}$ follows from the fact that we can simply run Compress on the verifier defined by two fixed Turing machines $\mathcal{S}'$ and $\mathcal{D}'$ (e.g., these could be trivial Turing machines that halt immediately), and again by the independence property described in Lemma 12.4, the first Turing machine it outputs will be $\mathcal{S}^{\mathrm{COMPR}}$. By Lemma 12.3, this takes time polylog(λ).

The claimed time complexity of $\mathcal{D}^{\mathrm{COMPR}}$ in the theorem statement follows from our setting of $k(n) = c_{\mathrm{REP}}(\lambda n)^{c_{\mathrm{REP}}}$ into the bound on $\mathsf{TIME}_{\mathcal{D}^{(3)}}(n)$. The description length of $\mathcal{D}^{\mathrm{COMPR}}$ follows from the description length bound on $\mathcal{D}^{(3)}$.

We now establish the completeness, soundness, and entanglement properties of $\mathcal{V}^{\mathrm{COMPR}}$. Assume that the input verifier $\mathcal{V}$ is λ-bounded and fix an index $n \in \mathbb{N}$. The completeness property (Item 1 in Theorem 12.2) follows from chaining together the completeness properties of $\mathcal{V}_n^{(3)}, \mathcal{V}_n^{(2)}, \mathcal{V}_n^{(1)}$, and the assumption that $\mathcal{V}_N$ has a value-1 PCC strategy.

We now establish the soundness property (Item 2 in Theorem 12.2). Assume for now that we have set the universal constants γ and c_{REP} so that for all integers λ, n such that $n \geq \gamma$ the following inequality holds:

$$\delta_1 \left(2\delta_2 \left(2\delta_3 \left(\frac{1}{2} \right), n \right), n \right) < \frac{1}{2} . \tag{151}$$

Suppose for contradiction that $n \geq \gamma$, the verifier $\mathcal{V}$ is λ-bounded, and $\mathrm{val}^*(\mathcal{V}_N) \leq \frac{1}{2}$, but $\mathrm{val}^*(\mathcal{V}_n^{(3)}) > \frac{1}{2}$. By chaining together the soundness guarantees of the three verifiers $\mathcal{V}_n^{(3)}, \mathcal{V}_n^{(2)}$, and $\mathcal{V}_n^{(1)}$, we obtain

$$\mathrm{val}^*(\mathcal{V}_N) \geq 1 - \delta_1 \left(2\delta_2 \left(2\delta_3 \left(\frac{1}{2} \right), n \right), n \right) > \frac{1}{2} ,$$

a contradiction.[24]

---

[24]Due to the strict inequality vs. non-strict inequality distinction in the soundness statements, in order to do the chaining, we need that $\delta_j(\varepsilon) < 2\delta_j(\varepsilon)$ for non-zero ε and $j \in \{1, 2, 3\}$, which holds in our case.

It remains to show that there is a suitable choice for the constants γ and c_{REP}. We work backwards and first identify a number β_1 such that $\delta_1(\beta_1, n) = a_1(\beta_1 + \frac{1}{(\lambda 2^n)^\lambda})^{b_1} < \frac{1}{2}$. Setting $\beta_1 = \frac{1}{2}(\frac{1}{4a_1})^{1/b_1}$, we get that for all $n \geq \beta_1^{-1}$

$$\delta_1(\beta_1, n) = a_1\left(\beta_1 + \frac{1}{(\lambda 2^n)^\lambda}\right)^{b_1} \leq a_1(\beta_1 + \frac{1}{n})^{b_1} \leq a_1(2\beta_1)^{b_1} < \frac{1}{2} ,$$

where we used that $\lambda \geq 1$ in the first inequality. Similarly, by setting

$$\beta_2 = \frac{1}{2}\left(\frac{\beta_1}{4a_2}\right)^{1/b_2}$$

we get that $2\delta_2(\beta_2, n) < \beta_1$ for all $n \geq \beta_2^{-1}$. Let $c_4 > 0$ be a universal constant such that

$$\text{TIME}_{\mathcal{D}^{(2)}}(n) \leq c_4 \left(\log(\lambda N)^\lambda \cdot |\mathcal{D}|\right)^{c_4}. \tag{152}$$

The existence of c_4 follows from the upper bound on $\text{TIME}_{\mathcal{D}^{(2)}}(n)$ shown earlier in the proof. Using moreover that $\mathcal{V}$ is λ-bounded, and thus that $|\mathcal{D}| \leq \lambda$, the right-hand side of (152) is at most $c_4\left(\lambda^2 \cdot (n + \log \lambda)\right)^{c_4}$. We specify the constant c_{REP} used in the description of the Turing machine Compress in Figure 14. Let c_{REP} be the minimum integer such that for all $n, \lambda \in \mathbb{N}$,

$$(\beta_2/2)^{-8} \cdot a_3 \cdot c_4 \cdot (\lambda^2 \cdot (n + \log \lambda))^{c_4} \leq c_{\text{REP}}(\lambda \cdot n)^{c_{\text{REP}}} . \tag{153}$$

For this choice of c_{REP}, using that $k(n)$ is defined as $c_{\text{REP}}(\lambda \cdot n)^{c_{\text{REP}}}$ we have that the function δ_3 defined in (150) satisfies $2\delta_3(\frac{1}{2}) < \beta_2$. Let

$$\gamma = \max\{\beta_1^{-1}, \beta_2^{-1}\} . \tag{154}$$

For this choice of γ, the inequality in (151) holds for all $n \geq \gamma$. This establishes Item 2.

Finally we show the entanglement lower bound. For all $n \geq \gamma$,

$$\begin{aligned}
\mathscr{E}(\mathcal{V}_n^{(3)}, 1/2) &\geq \mathscr{E}\left(\mathcal{V}_n^{(2)}, 1 - \beta_2\right) \\
&\geq \mathscr{E}\left(\mathcal{V}_n^{(1)}, 1 - \beta_1\right) \\
&\geq \max\left\{\mathscr{E}(\mathcal{V}_N, \frac{1}{2}), \frac{1}{2} \cdot 2^{(\lambda N)^\lambda}\right\} .
\end{aligned}$$

This establishes Item 3 in Theorem 12.2, and concludes the proof of the theorem. $\qquad\square$

12.2 The Kleene-Rogers fixed point theorem

A fundamental result in computability theory is the Kleene-Rogers fixed point theorem, which states that for every Turing machine $\mathcal{F}$ that halts on every input, there exists a Turing machine $\mathcal{M}$ that is a *fixed point* of $\mathcal{F}$ in the sense that given some description of $\mathcal{M}$, the Turing machine $\mathcal{F}$ computes another Turing machine that computes the same function as $\mathcal{M}$. Surprisingly, such a fixed point can be efficiently computed from the description of $\mathcal{F}$ itself! This Theorem is attributed to Rogers [Rog87], who proved a simpler version of a recursion theorem due to Kleene [Kle54].

In computability theory, the Kleene-Rogers fixed point theorem is commonly used to argue that a Turing machine $\mathcal{M}$ that can call other Turing machines $\mathcal{N}$ on an input that is a description of $\mathcal{M}$ is a well-defined

notion. In other words, the Turing machine $\mathcal{M}$ can "print its own source code." We use the Kleene-Rogers fixed point theorem in Section 12.3 to show the existence of a decider $\mathcal{D}$ that calls the compression procedure Compress on itself.

Since the theorem statement and its proof involve Turing machines acting on descriptions of Turing machines and then outputting a description of yet another Turing machine, to aid comprehension we use the following notation (used only in this subsection) to distinguish between Turing machines and their descriptions. We use calligraphic letters such as $\mathcal{M}$ to denote a Turing machine, which is formally a tuple of parameters that specify the Turing machine's alphabet, transition rules, and so on. We use the notation $\overline{\mathcal{M}}$ to denote a binary string that encodes some Turing machine $\mathcal{M}$. We assume an encoding where the states, the transition rules, and the number of input tapes of $\mathcal{M}$ can be efficiently computed given the description $\overline{\mathcal{M}}$. Conversely, for every $k \geq 1$, every binary string $x \in \{0,1\}^*$ can be interpreted as the description of a k-input Turing machine, and we use $[x]_k$ to denote this Turing machine. If $\mathcal{M}$ is a k-input Turing machine, we have $[\overline{\mathcal{M}}]_k = \mathcal{M}$.

Theorem 12.5 (Kleene's recursion theorem/Roger's fixed-point theorem). *For all $k \in \mathbb{N}$, for all 1-input Turing machines $\mathcal{F}$ computing a total function, there exists a k-input Turing machine $\mathcal{M}$ (called a fixed point of $\mathcal{F}$) that computes the same (partial) function as computed by the k-input Turing machine described by $\mathcal{F}(\overline{\mathcal{M}})$.*

Furthermore, there exists a 1-input Turing machine $\mathsf{ComputeFixedPoint}_k$ that given input a description $\overline{\mathcal{F}}$, outputs a description $\overline{\mathcal{M}}$ of a fixed point $\mathcal{M}$ in time $\mathrm{poly}(|\overline{\mathcal{F}}|)$ where $|\overline{\mathcal{F}}|$ denotes the length of $\overline{\mathcal{F}}$. The time complexity of the fixed point $\mathcal{M}$ on input $x = (x_1, \ldots, x_k)$ is at most

$$\mathrm{poly}(|\overline{\mathcal{F}}|, \mathsf{TIME}_{\mathcal{F}, \overline{\mathcal{M}}}, \mathsf{TIME}_{[\mathcal{F}(\overline{\mathcal{M}})]_k, x}),$$

where we recall the TIME notation from Section 3.1.

We include an elementary proof of this theorem.

Proof. Consider the description $\overline{\mathcal{S}}$ of a 1-input Turing machine $\mathcal{S}$ given in Figure 19.

Input: description $\overline{\mathcal{C}}$ of a 1-input Turing machine.

Output: the description of a k-input Turing machine $\mathcal{D}_{\overline{\mathcal{C}}}$ that, on input x, does the following:

1. Run $\mathcal{C}$ with input tape initialized to $\overline{\mathcal{C}}$. Let $\overline{\mathcal{E}}$ denote its output (if it halts).

2. Run $\mathcal{F}$ on input $\overline{\mathcal{E}}$, and let $\overline{\mathcal{E}'}$ denote its output.

3. Run $\mathcal{E}'$ on input x and return the output (if it halts).

Figure 19: The Turing machine $\mathcal{S}$.

We mention some important properties of $\mathcal{S}$. First, $\mathcal{S}$ never actually runs $\mathcal{C}$ or $\mathcal{F}$; it only performs a computation based on the descriptions $\overline{\mathcal{C}}$ and $\overline{\mathcal{F}}$. Next, the description $\overline{\mathcal{S}}$ is computable from the description $\overline{\mathcal{F}}$, and the length of the description $\overline{\mathcal{S}}$ is $\mathrm{poly}(|\overline{\mathcal{F}}|)$. Finally, the running time of $\mathcal{S}$ on input $\overline{\mathcal{C}}$ is $\mathrm{poly}(|\overline{\mathcal{C}}|, |\overline{\mathcal{F}}|)$. This is the time it takes to construct the description of the Turing machine $\mathcal{D}_{\overline{\mathcal{C}}}$.

Thus, for all Turing machines $\mathcal{C}$, $\mathcal{S}(\overline{\mathcal{C}})$ is the description of a Turing machine $\mathcal{D}_{\overline{\mathcal{C}}}$ such that for all inputs x,

$$\mathcal{D}_{\overline{\mathcal{C}}}(x) = \left[\mathcal{F}(\mathcal{C}(\overline{\mathcal{C}})) \right]_k (x) \,,$$

provided that $\mathcal{C}$ halts when given $\overline{\mathcal{C}}$ as input. Here, we use the fact that $\mathcal{F}$ is a total function, so the Turing machine $\left[\mathcal{F}(\mathcal{C}(\overline{\mathcal{C}})) \right]_k$ is well defined.

Input: description $\overline{\mathcal{F}}$ of a 1-input Turing machine.

Output:

    1. Compute the description $\overline{\mathcal{S}}$ from $\overline{\mathcal{F}}$.

    2. Output $\mathcal{S}(\overline{\mathcal{S}})$.

Figure 20: The Turing machine $\mathsf{ComputeFixedPoint}_k$.

Next, consider the Turing machine $\mathsf{ComputeFixedPoint}_k$ defined in Figure 20. Let $\overline{\mathcal{M}}$ denote the output of $\mathsf{ComputeFixedPoint}_k$ on input $\overline{\mathcal{F}}$. Note that $\mathcal{M} = \mathcal{D}_{\overline{\mathcal{S}}}$, and is a k-input Turing machine by construction. By definition, $\mathcal{S}$ halts on every input as well, hence it halts when given $\overline{\mathcal{S}}$ as input. Thus for all inputs x,

$$\mathcal{M}(x) = \mathcal{D}_{\overline{\mathcal{S}}}(x) = \left[\mathcal{F}(\mathcal{S}(\overline{\mathcal{S}})) \right]_k (x) = \left[\mathcal{F}(\overline{\mathcal{M}}) \right]_k (x)$$

where again the Turing machines in brackets are well-defined because $\mathcal{F}$ is a total function.

This demonstrates that a fixed point $\mathcal{M}$ of $\mathcal{F}$ can be computed from $\overline{\mathcal{F}}$. The time complexity of $\mathsf{ComputeFixedPoint}_k$ on input $\overline{\mathcal{F}}$ is $\mathrm{poly}(|\overline{\mathcal{F}}|)$, based on the description and time complexity of $\mathcal{S}$. The time complexity of $\mathcal{M}$ (equivalently, $\mathcal{D}_{\overline{\mathcal{S}}}$) on input x is equal to some polynomial function of

    1. The time complexity of running $\mathcal{S}$ on input $\overline{\mathcal{S}}$.

    2. The time complexity of running $\mathcal{F}$ on input $\overline{\mathcal{M}}$.

    3. The time complexity of running $[\mathcal{F}(\overline{\mathcal{M}})]_k$ on input x. $\qquad\square$

12.3 An MIP^* protocol for the Halting problem

We use the Kleene-Rogers fixed point theorem to construct, for every Turing machine $\mathcal{M}$, a verifier for a game that decides whether $\mathcal{M}$ halts or not.

First, for every Turing machine $\mathcal{M}$ and integers $\Delta, \lambda \in \mathbb{N}$ in Figure 21 we define a Turing machine $\mathcal{F}$.

Input: description $\mathcal{D}$ of a 5-input Turing machine.

Output: the description $\mathcal{D}'$ of a 5-input Turing machine that performs the following on input (n, x, y, a, b): run the following steps for at most $(\Delta n)^\Delta$ time steps (if it has not halted by that time, then reject):

1. Run $\mathcal{M}$ on the empty tape for n steps. If $\mathcal{M}$ halts, then accept. Otherwise, if $\mathcal{M}$ hasn't halted yet, then proceed to the next step.

2. Compute $\mathcal{V}^{\text{COMPR}} = \text{Compress}(\mathcal{V})$, where $\mathcal{V} = (\mathcal{S}^{\text{COMPR}}, \mathcal{D})$ and Compress given in Theorem 12.2 depends on the parameter λ. Let $\mathcal{V}^{\text{COMPR}} = (\mathcal{S}^{\text{COMPR}}, \mathcal{D}^{\text{COMPR}})$.

3. Accept iff $\mathcal{D}^{\text{COMPR}}(n, x, y, a, b)$ accepts.

Figure 21: The Turing machine $\mathcal{F}$, parameterized by integers λ, Δ and the Turing machine $\mathcal{M}$.

Lemma 12.6. *For all Turing machines $\mathcal{M}$, integers $\Delta, \lambda \in \mathbb{N}$, the corresponding Turing machine $\mathcal{F}$ satisfies the following properties:*

- *$\mathcal{F}$ computes a total function.*

- *The description of $\mathcal{F}$ can be computed in polynomial time given the description of $\mathcal{M}$ and the binary representation of Δ, λ.*

Proof. The first item follows by inspection (the Turing machine $\mathcal{F}$ halts on every input). The second item follows because the description of $\mathcal{F}$ depends on descriptions of Turing machines $\mathcal{M}$, Compress, and $\mathcal{S}^{\text{COMPR}}$ (the last two of which depend on the parameter λ), and the integer Δ. The descriptions of Compress and $\mathcal{S}^{\text{COMPR}}$ can be computed in polynomial time from the binary representation of λ, as shown in Lemma 12.3. $\qquad\square$

Since $\mathcal{F}$ computes a total function, by applying the Kleene-Rogers fixed point theorem (Theorem 12.5) with $k = 5$, there exists a 5-input Turing machine $\mathcal{D}^{\text{HALT}}$ computing the same function as $\mathcal{F}(\mathcal{D}^{\text{HALT}})$, and the description of $\mathcal{D}^{\text{HALT}}$ is polynomial-time computable from the description of $\mathcal{F}$.

Let $\mathcal{V}^{\text{HALT}} = (\mathcal{S}^{\text{HALT}}, \mathcal{D}^{\text{HALT}})$, where $\mathcal{S}^{\text{HALT}}$ is defined to be the sampler $\mathcal{S}^{\text{COMPR}}$ from Theorem 12.2 corresponding to parameter λ. Since $\mathcal{F}$ and $\mathcal{S}^{\text{HALT}}$ are polynomial-time computable from $\mathcal{M}, \lambda$, and Δ, the description of the verifier $\mathcal{V}^{\text{HALT}}$ is also computable from these parameters.

From the definition of $\mathcal{D}^{\text{HALT}}$ as a fixed point of $\mathcal{F}$, we get the following properties.

Lemma 12.7. *Let $\mathcal{M}$ be a Turing machine and λ, Δ integer. Then the verifier $\mathcal{V}^{\text{HALT}}$ has the following properties.*

1. *For any integer n, if $\mathcal{M}$ halts in n steps then $\text{val}^*(\mathcal{V}_n^{\text{HALT}}) = 1$. Furthermore, there is a value-1 PCC strategy for the game $\mathcal{V}_n^{\text{HALT}}$.*

2. *For any integer n, if $\mathcal{M}$ does not halt in n steps then $\mathcal{V}_n^{\text{HALT}}$ has a value-1 PCC strategy if and only if $(\mathcal{V}^{\text{HALT}})_n^{\text{COMPR}}$ does. Furthermore, under the same assumption it holds that for any $\alpha \in [0, 1]$,*

$$\mathscr{E}(\mathcal{V}_n^{\text{HALT}}, \alpha) = \mathscr{E}((\mathcal{V}^{\text{HALT}})_n^{\text{COMPR}}, \alpha).$$

155

Proof. By definition, $\mathcal{D}^{\text{HALT}}$ computes the same function as $\mathcal{F}(\mathcal{D}^{\text{HALT}})$. Suppose $\mathcal{M}$ halts in n steps. Then from the definition of $\mathcal{F}$, it follows that $\mathcal{F}(\mathcal{D}^{\text{HALT}})$ accepts on input (n, x, y, a, b) for all x, y, a, b, and hence $\text{val}^*(\mathcal{V}_n^{\text{HALT}}) = 1$. If $\mathcal{M}$ does not halt in n steps, then again from the definition of $\mathcal{F}$, it follows that $\mathcal{F}(\mathcal{D}^{\text{HALT}})$ accepts on input (n, x, y, a, b) if and only if $(\mathcal{D}^{\text{HALT}})^{\text{COMPR}}$ accepts on (n, x, y, a, b). $\qquad\square$

We now argue that the parameters λ, Δ can be chosen so that $\mathcal{V}^{\text{HALT}}$ is λ-bounded, thus satisfying the conditions of Theorem 12.2.

Lemma 12.8. *For all Turing machines $\mathcal{M}$ and integers $\Delta, \lambda \in \mathbb{N}$, the description of the corresponding verifier $\mathcal{V}^{\text{HALT}}$ can be computed in polynomial time from the description of $\mathcal{M}$ and the binary representation of Δ, λ. Furthermore, $\mathcal{V}^{\text{HALT}}$ is a normal form verifier satisfying*

1. $|\mathcal{S}^{\text{HALT}}|, |\mathcal{D}^{\text{HALT}}| \leq \text{poly}(\log \Delta, |\mathcal{M}|, \log \lambda)$

2. $\text{RAND}_{\mathcal{S}^{\text{HALT}}}(n) \leq \text{poly}(n, \lambda)$

3. $\text{TIME}_{\mathcal{S}^{\text{HALT}}}(n) \leq \text{poly}(n, \lambda)$

4. $\text{TIME}_{\mathcal{D}^{\text{HALT}}}(n) \leq \text{poly}((\Delta n)^\Delta, \log \lambda, |\mathcal{M}|)$

Proof. Lemma 12.6 implies that $\mathcal{F}$ can be computed in polynomial time from $(\mathcal{M}, \lambda, \Delta)$; Theorem 12.5 then implies that $\mathcal{D}^{\text{HALT}}$ can be computed in polynomial time from $(\mathcal{M}, \lambda, \Delta)$. Second, combining Lemmas 12.3 and 12.4 shows that $\mathcal{S}^{\text{HALT}} = \mathcal{S}^{\text{COMPR}}$ (corresponding to parameter λ) can be computed in polynomial time from the binary representation of λ.

We now establish the complexity bounds on $\mathcal{V}^{\text{HALT}}$. Since $\mathcal{S}^{\text{HALT}} = \mathcal{S}^{\text{COMPR}}$, the description length, randomness and time complexities of $\mathcal{S}^{\text{HALT}}$ are determined by Theorem 12.2. To analyze the time complexity of the decider $\mathcal{D}^{\text{HALT}}$, we observe that

- The description length $|\mathcal{F}| = \text{poly}(\log \Delta, |\mathcal{M}|, \log \lambda)$, because Δ is written in binary in the description of $\mathcal{F}$, and the dependence on the parameter λ come from the description length of the compression procedure Compress. All other steps have constant-size descriptions.

- By Theorem 12.5, the description length $|\mathcal{D}^{\text{HALT}}|$ is bounded by $\text{poly}(|\mathcal{F}|)$.

- The time complexity of $\mathcal{F}$ on input $\mathcal{D}^{\text{HALT}}$ is at most $\text{poly}(\log \Delta, |\mathcal{M}|, |\mathcal{D}^{\text{HALT}}|, \log \lambda)$, because it simply writes out the description of the Turing machine $\mathcal{D}'$ described in Figure 21.

- The time complexity of the Turing machine described by $\mathcal{F}(\mathcal{D}^{\text{HALT}})$ on any input (n, x, y, a, b) is, by construction, at most $(\Delta n)^\Delta$.

Putting everything together, Theorem 12.5 implies that the time complexity of $\mathcal{D}^{\text{HALT}}$ on inputs (n, x, y, a, b) is at most $\text{poly}((\Delta n)^\Delta, \log \lambda, |\mathcal{M}|)$. $\qquad\square$

Lemma 12.9 (Self-consistent compression parameters)**.** *For all Turing machines $\mathcal{M}$, there exist integers $\lambda, \Delta \in \mathbb{N}$ and corresponding Turing machines $\mathcal{F}, \mathcal{D}^{\text{HALT}}$ such that the corresponding verifier $\mathcal{V}^{\text{HALT}} = (\mathcal{S}^{\text{HALT}}, \mathcal{D}^{\text{HALT}})$ is λ-bounded (see Definition 12.1), and the Turing machine $\mathcal{F}(\mathcal{D}^{\text{HALT}})$ on input (n, x, y, a, b) does not reject due to exceeding the time limit Δn^Δ. Furthermore, integers λ, Δ satisfying these conditions are polynomial-time computable from the description of $\mathcal{M}$.*

Proof. The time complexity of the three steps of the Turing machine described by $\mathcal{F}(\mathcal{D}^{\text{HALT}})$ on input (n, x, y, a, b) is bounded by

$$\underbrace{\text{poly}(n, |\mathcal{M}|)}_{\text{Step 1}} + \underbrace{\text{poly}(|\mathcal{S}^{\text{HALT}}|, |\mathcal{D}^{\text{HALT}}|, \log \lambda)}_{\text{Step 2}} + \underbrace{\text{poly}(n, \lambda, |\mathcal{S}^{\text{HALT}}|, |\mathcal{D}^{\text{HALT}}|)}_{\text{Step 3}} . \tag{155}$$

Step 1 comes from simulating the Turing machine $\mathcal{M}$, Step 2 comes from the complexity of running the Compress procedure, and Step 3 is the complexity of the decider that is output by the Compress procedure, given by Theorem 12.2. Note that we obtain the bound for Step 3 *without* assuming that the verifier $\mathcal{V}^{\text{HALT}}$ is λ-bounded—this is precisely the statement we are trying to prove! Substituting upper bounds on the description lengths of $\mathcal{S}^{\text{HALT}}$ and $\mathcal{D}^{\text{HALT}}$(as given by Theorem 12.2 and Lemma 12.8), we get that (155) is at most $\text{poly}(n, \lambda, \log \Delta, |\mathcal{M}|)$.

Let $C \in \mathbb{N}$ be a universal constant such that the polynomials on the right hand side of Items 1 through 4 of Lemma 12.8 as well as the polynomial upper bound on (155) all satisfy $\text{poly}(a_1, \ldots, a_k) \le C(a_1 \cdots a_k)^C$ for all integers $a_1, \ldots, a_k \in \mathbb{N}$, where $a_1, \ldots, a_k$ represent possible values for the argument of each polynomial (e.g. n, λ, etc.). To prove the lemma it suffices to identify integers Δ, λ such that for all $n \in \mathbb{N}$,

1. $C(\log \Delta \cdot \log \lambda \cdot |\mathcal{M}|)^C \le \lambda$

2. $C(n \cdot \lambda)^C \le (\lambda n)^\lambda$

3. $C((\Delta n)^\Delta \cdot \log \lambda \cdot |\mathcal{M}|)^C \le (\lambda n)^\lambda$

4. $C(n \cdot \lambda \cdot \log \Delta \cdot |\mathcal{M}|)^C \le (\Delta n)^\Delta$

The first three items are to establish the λ-bounded property of $\mathcal{V}^{\text{HALT}}$ (the first is so that the description length of $|\mathcal{D}^{\text{HALT}}|$ is at most λ, and the next two are to bound the time and randomness complexities of $\mathcal{S}^{\text{HALT}}, \mathcal{D}^{\text{HALT}}$). The fourth item is used to argue that $\mathcal{D}^{\text{HALT}}$ does not reject due to exceeding the "time-out" limit of $(\Delta n)^\Delta$.

We choose $\Delta = 128 \cdot C^2 \cdot |\mathcal{M}|$ and $\lambda = \Delta^{2C}$. To see that these satisfy the inequalities, we note that the power n is raised to on the left-hand sides of the inequalities is always less than the power it's raised to on the right-hand sides. As a result, it is sufficient to check that these satisfy satisfying the inequalities for $n = 1$, which they do. These integers are clearly computable in polynomial-time from the description of $\mathcal{M}$. $\qquad\square$

Putting things together we obtain the following result.

Theorem 12.10. *For all Turing machines $\mathcal{M}$, there exists a game $\mathfrak{G}$ such that*

1. If $\mathcal{M}$ halts on the empty tape, then $\text{val}^*(\mathfrak{G}) = 1$.

2. If $\mathcal{M}$ does not halt on the empty tape, then $\text{val}^*(\mathfrak{G}) \le \frac{1}{2}$.

Furthermore, the game $\mathfrak{G}$ is polynomial-time computable from the description of $\mathcal{M}$.

Proof. Fix $\alpha = \frac{1}{2}$. Fix a Turing machine $\mathcal{M}$. Let $\Delta, \lambda, \mathcal{F}$, and $\mathcal{D}^{\text{HALT}}$ be as promised by Lemma 12.9 for this choice of $\mathcal{M}$. In particular, the parameters Δ, λ are polynomial-time computable from the description of $\mathcal{M}$, and therefore by Lemma 12.8 $\mathcal{V}^{\text{HALT}}$ is polynomial-time computable from $\mathcal{M}$. Furthermore, Lemma 12.9 ensures that $\mathcal{V}^{\text{HALT}}$ is λ-bounded.

Let $n_0 = \gamma$, where γ is the universal constant specified in Theorem 12.2. For all $n \geq n_0$, let $\mathfrak{G}_n$ denote the game corresponding to the verifier $\mathcal{V}_n^{\text{HALT}}$, and let $\mathfrak{G}_n^{\text{COMPR}}$ denote the game corresponding to the compressed verifier $\mathcal{V}_n'$ where $\mathcal{V}' = \text{Compress}(\mathcal{V}^{\text{HALT}})$, where the Compress procedure depends on the parameter λ. Finally, let $\mathfrak{G} = \mathfrak{G}_{n_0}$.

Suppose that $\mathcal{M}$ halts on the empty tape; let T be the minimum number of time steps required for $\mathcal{M}$ to halt on the empty tape. Observe that for all $n \geq T$, by Lemma 12.7 it holds that $\mathfrak{G}_n$ has a value-1 PCC strategy. We will use this to show inductively that $\mathfrak{G}_n$ also has a value-1 PCC strategy, for all $n_0 \leq n < T$.

Claim 12.11. *Let $n_0 \leq n < T$. Suppose that $\mathfrak{G}_m$ has a value 1 PCC strategy for all $m > n$. Then $\mathfrak{G}_n$ also has a value 1 PCC strategy.*

Proof. Since by assumption $\mathcal{M}$ does not halt in n steps, by Lemma 12.7 it holds that $\mathfrak{G}_n$ has a value 1 PCC strategy if $\mathfrak{G}_n^{\text{COMPR}}$ does. Since $\mathcal{V}^{\text{HALT}}$ is λ-bounded and $n \geq n_0$, by Theorem 12.2 it follows that $\mathfrak{G}_n^{\text{COMPR}}$ has a value 1 PCC strategy if $\mathfrak{G}_{2^n}$ does. Since $2^n > n$, this is true by the hypothesis of the claim. Thus, $\mathfrak{G}_n$ has a value 1 PCC strategy as claimed. ☐

By Claim 12.11 and downwards induction on n (with the base case $n = T$), we have that $\mathfrak{G}_n$ has a value-1 PCC strategy for all $n \geq n_0$. In particular, we have $\text{val}^*(\mathfrak{G}) = \text{val}^*(\mathfrak{G}_{n_0}) = 1$. This shows the first item in the theorem statement.

Now suppose that $\mathcal{M}$ does not halt on the empty tape. We have that for all $n \in \mathbb{N}$:

$$\mathscr{E}(\mathfrak{G}_n, \alpha) = \mathscr{E}(\mathfrak{G}_n^{\text{COMPR}}, \alpha) \geq \mathscr{E}(\mathfrak{G}_N, \alpha) \,,$$

where the equality follows from Lemma 12.7 and the inequality follows from Theorem 12.2 (again using the λ-bounded property of $\mathcal{V}^{\text{HALT}}$). By induction, we get that for all $k \in \mathbb{N}$,

$$\mathscr{E}(\mathfrak{G}, \alpha) = \mathscr{E}(\mathfrak{G}_{n_0}, \alpha) \geq \mathscr{E}(\mathfrak{G}_{g^{(k)}(n_0)}, \alpha) = \mathscr{E}(\mathfrak{G}_{g^{(k)}(n_0)}^{\text{COMPR}}, \alpha) \geq \alpha 2^{\lambda (g^{(k)}(n_0))^{\lambda}} \,,$$

where $g^{(k)}(\cdot)$ is the k-fold composition of the function $g(n) = 2^n$ and the second inequality follows from Theorem 12.2 again. Since $g(\cdot)$ is a monotonically increasing function and by definition $\alpha > 0$, this implies that there is no finite upper bound on $\mathscr{E}(\mathfrak{G}, \alpha)$, and therefore every finite dimensional strategy for the game $\mathfrak{G}$ must have success probability at most $\alpha = \frac{1}{2}$. ☐

Recall the definition of the complexity class RE, which stands for the set of *recursively enumerable* languages (also called *Turing-recognizable* languages). Precisely, a language $L \subseteq \{0,1\}^*$ is in RE if and only if there exists a Turing machine $\mathcal{M}$ such that if $x \in L$, then $\mathcal{M}(x)$ halts and outputs 1, and if $x \notin L$, then either $\mathcal{M}(x)$ outputs 0 or it does not halt. The Halting Problem is the language that contains descriptions of Turing machines that halt on the empty input tape. The following well-known lemma shows that the Halting Problem is complete for RE. We include the simple proof for convenience.

Lemma 12.12. *The Halting Problem is complete for* RE.

Proof. To see that the Halting Problem is in RE, define $\mathcal{M}$ to take as input an x that represents a Turing machine $\mathcal{N} = [x]$, and runs the universal Turing machine to simulate $\mathcal{N}$ on the empty tape; if $\mathcal{N}$ halts with a 1 then so does $\mathcal{M}$.

To show that the Halting problem is complete for RE, let $L \in$ RE and $\mathcal{M}$ a Turing machine such that if $x \in L$, then $\mathcal{M}(x)$ halts and outputs 1. For an input x, let $\mathcal{N}_x$ be the following Turing machine. $\mathcal{N}_x$ first runs $\mathcal{M}$ on input x. If $\mathcal{M}$ accepts, then $\mathcal{N}_x$ accepts. On all other outcomes, $\mathcal{N}_x$ goes into an infinite loop. Thus $\mathcal{N}_x$ halts if and only if $x \in L$. ☐

Corollary 12.13. $\mathsf{MIP}^* = \mathsf{RE}$.

Proof. Since the Halting Problem is complete for RE, and by Theorem 12.10 is contained in MIP^*, we have the inclusion $\mathsf{RE} \subseteq \mathsf{MIP}^*$. The reverse inclusion $\mathsf{MIP}^* \subseteq \mathsf{RE}$ follows from the following observation. Let $L \in \mathsf{MIP}^*$. From the definition of MIP^* (see e.g. [VW16, Section 6.1], from which we borrow the notation used here) it follows that there exists a polynomial-time Turing machine $\mathcal{R}$ such that for all $x \in \{0,1\}^*$, $\mathcal{R}(x)$ is the description of an m-turn verifier V_x interacting with k provers, where m and k are both polynomial functions of $|x|$ and such that

$$\mathrm{val}^*(V_x) \geq 2/3 \qquad \text{if } x \in L$$
$$\mathrm{val}^*(V_x) \leq 1/3 \qquad \text{if } x \notin L$$

Consider the following Turing machine $\mathcal{A}$: on input x, it computes $V_x = \mathcal{R}(x)$, and then exhaustively searches over tensor-product strategies of increasing dimension and increasing accuracy to evaluate a lower bound on $\mathrm{val}^*(V_x)$. If $\mathrm{val}^*(V_x) \geq 2/3$, then for arbitrarily small δ there exists a finite dimensional tensor-product strategy for the players that achieves value $2/3 - \delta > 1/3$. When the Turing machine $\mathcal{A}$ identifies such a strategy it terminates, outputting 1. If there is no such strategy, then $\mathcal{A}$ never halts. This implies that $L \in \mathsf{RE}$. $\qquad\square$

12.4 An explicit separation

As discussed in Section 1.3, Theorem 12.10 implies that C_{qa}, the set of approximately finite-dimensional correlations, is a strict subset of C_{qc}, the set of commuting-operator correlations. This is because if $C_{qa} = C_{qc}$, then there exists an algorithm to approximate the entangled value of a given nonlocal game $\mathfrak{G}$ to arbitrary accuracy. On the other hand, Theorem 12.10 shows that deciding whether a game has entangled value 1 or at most $1/2$, promised that one is the case, is undecidable. Therefore the correlation sets must be different.

In fact, Theorem 12.10 implies that there is an infinite family $\mathscr{M}$ of Turing machines that do not halt on an empty input tape such that for all $\mathcal{M} \in \mathscr{M}$, the corresponding game $\mathfrak{G}_{\mathcal{M}}$ has $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M}}) < \mathrm{val}^{co}(\mathfrak{G}_{\mathcal{M}})$, where recall that $\mathrm{val}^{co}(\mathfrak{G}_{\mathcal{M}})$ denotes the *commuting-operator value* of $\mathfrak{G}_{\mathcal{M}}$, which is supremum of success probabilities over all commuting-operator strategies for $\mathfrak{G}_{\mathcal{M}}$.[25] However, it is not immediately clear, given a *specific* non-halting Turing machine $\mathcal{M}$, whether the associated game $\mathfrak{G}_{\mathcal{M}}$ separates the commuting-operator model from the tensor product model of strategies. While Theorem 12.10 implies that $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M}}) \leq \frac{1}{2}$, it could also be the case that $\mathrm{val}^{co}(\mathfrak{G}_{\mathcal{M}}) = \mathrm{val}^*(\mathfrak{G}_{\mathcal{M}})$ in that particular instance. We conjecture that $\mathrm{val}^{co}(\mathfrak{G}_{\mathcal{M}}) = 1$ for *all* non-halting Turing machines $\mathcal{M}$, but it appears to be difficult to identify an explicit value-1 commuting operator strategy that demonstrates this.

In this section we identify an explicit game $\mathfrak{G}$ that separates the tensor product model from the commuting-operator model; we show that $\mathrm{val}^*(\mathfrak{G}) \leq \frac{1}{2}$ but $\mathrm{val}^{co}(\mathfrak{G}) = 1$. Interestingly, the proof does not exhibit an explicit value-1 commuting-operator strategy for $\mathfrak{G}$.

We construct the separating game in a similar manner to the games constructed in Section 12.3. Let $\mathcal{A}$ denote the following 1-input Turing machine: it takes as input a description of a nonlocal game $\mathfrak{G}$ and runs the semidefinite programming hierarchy of [NPA08, DLTW08] to compute a non-increasing sequence

---

[25]To see why this holds, observe that if it were the case that $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M}}) = \mathrm{val}^{co}(\mathfrak{G}_{\mathcal{M}})$ for all but finitely many non-halting $\mathcal{M}$ then we could construct an algorithm $\mathcal{A}$ to decide the Halting problem as follows. On input $\mathcal{M}$, $\mathcal{A}$ first checks if $\mathcal{M}$ is one of the finitely many Turing machines for which $\mathrm{val}^*(\mathfrak{G}_{\mathcal{M}}) < \mathrm{val}^{co}(\mathfrak{G}_{\mathcal{M}})$; if so, then it outputs a hard-coded answer for whether $\mathcal{M}$ halts on the empty tape or not. Otherwise, $\mathcal{A}$ computes the nonlocal game $\mathfrak{G}_{\mathcal{M}}$ and executes the aforementioned algorithm for approximating the entangled value of games assuming that $C_{qa} = C_{qc}$.

$\alpha_1, \alpha_2, \ldots$ of upper bounds on $\text{val}^{\text{co}}(\mathfrak{G})$ such that $\lim_{n\to\infty} \alpha_n = \text{val}^{\text{co}}(\mathfrak{G})$. The Turing machine $\mathcal{A}$ halts if it obtains a bound $\alpha_n < 1$. Thus this algorithm eventually halts whenever $\text{val}^{\text{co}}(\mathfrak{G}) < 1$, and otherwise it runs forever.

Consider the Turing machine $\mathcal{R}$ in Figure 22, parameterized by integers λ, Δ. The only difference between the Turing machine described in Figure 22 and the one described in Figure 21 is that the decider $\mathcal{D}'$ returned by $\mathcal{R}(\mathcal{D})$ runs the algorithm $\mathcal{A}$ on some fixed nonlocal game corresponding to the verifier $\mathcal{V} = (\mathcal{S}^{\text{COMPR}}, \mathcal{D})$.

Input: description $\mathcal{D}$ of a 5-input Turing machine.

Output: the description $\mathcal{D}'$ of a 5-input Turing machine: on input (n, x, y, a, b), $\mathcal{D}'$ runs the following steps for at most $(\Delta n)^{\Delta}$ time steps (if it has not halted by that time, then reject).

1. Let $\mathcal{V} = (\mathcal{S}^{\text{COMPR}}, \mathcal{D})$. Compute an explicit description of the nonlocal game $\mathfrak{G} = \mathcal{V}_{n_0}$, where $n_0 = \gamma$ and γ is the universal constant from Theorem 12.2.

2. Run $\mathcal{A}$ on input $\mathfrak{G}$ for n steps. If $\mathcal{A}$ halts, then accept. Otherwise, if $\mathcal{A}$ hasn't halted yet, then proceed to the next step.

3. Compute $\mathcal{V}^{\text{COMPR}} = \text{Compress}(\mathcal{V})$, where $\mathcal{V} = (\mathcal{S}^{\text{COMPR}}, \mathcal{D})$ and Compress is given by Theorem 12.2 which depends on the parameter λ. Let $\mathcal{D}^{\text{COMPR}}$ be the decider of $\mathcal{V}^{\text{COMPR}}$.

4. Accept iff $\mathcal{D}^{\text{COMPR}}(n, x, y, a, b)$ accepts.

Figure 22: The Turing machine $\mathcal{R}$, parameterized by integers λ, Δ.

We follow the same steps as in Section 12.3. By applying the Kleene-Rogers fixed point theorem to $\mathcal{R}$, we obtain a decider $\mathcal{D}^{\text{SEP}}$ that is a fixed point of $\mathcal{R}$. Let $\mathcal{V}^{\text{SEP}} = (\mathcal{S}^{\text{SEP}}, \mathcal{D}^{\text{SEP}})$ where $\mathcal{S}^{\text{SEP}} = \mathcal{S}^{\text{COMPR}}$. A similar argument to that of Lemmas 12.8 and 12.9 shows that there exist choices of λ, Δ (computable from the description of $\mathcal{A}$) such that $\mathcal{V}^{\text{SEP}}$ is λ-bounded. Fix such parameters λ, Δ and let $\mathcal{V}^{\text{SEP}}$ be the verifier corresponding to these parameters. Define the game $\mathfrak{G}^{\text{SEP}} = \mathcal{V}_{n_0}^{\text{SEP}}$ for $n_0 = \gamma$, where γ is the universal constant from Theorem 12.2. Note that since $\mathcal{D}^{\text{SEP}}$ computes the same function as $\mathcal{R}(\mathcal{D}^{\text{SEP}})$ and the description of the game $\mathfrak{G}$ computed by the decider $\mathcal{D}^{\text{SEP}}$ in Step 1 of Figure 22 only depends on the predicate computed by $\mathcal{D}$ (rather than the details of how the predicate is computed), it follows that $\mathfrak{G}$ is identical to $\mathfrak{G}^{\text{SEP}}$.

Theorem 12.14. *For the game $\mathfrak{G}^{\text{SEP}} = \mathcal{V}_{n_0}^{\text{SEP}}$ it holds that* $\text{val}^*(\mathfrak{G}^{\text{SEP}}) \leq \frac{1}{2}$ *and* $\text{val}^{\text{co}}(\mathfrak{G}^{\text{SEP}}) = 1$.

Proof. Suppose that $\text{val}^{\text{co}}(\mathfrak{G}^{\text{SEP}}) = 1$. The invocation of the Turing machine $\mathcal{A}$ on input $\mathfrak{G}^{\text{SEP}}$ never halts, and therefore the decider $\mathcal{D}^{\text{SEP}}$ never accepts in Step 2 of Figure 22. Applying Theorem 12.2, we get that $\mathscr{E}(\mathcal{V}_n^{\text{SEP}}, \frac{1}{2}) \geq \mathscr{E}(\mathcal{V}_{2^n}^{\text{SEP}}, \frac{1}{2})$ and

$$\mathscr{E}\left(\mathcal{V}_n^{\text{SEP}}, \frac{1}{2}\right) \geq \frac{1}{2} 2^{\lambda(2^n)^{\lambda}}$$

for all $n \geq n_0$. An inductive argument implies that there is no finite upper bound on $\mathscr{E}(\mathcal{V}_n^{\text{SEP}}, \frac{1}{2})$, and thus $\text{val}^*(\mathfrak{G}^{\text{SEP}}) = \text{val}^*(\mathcal{V}_{n_0}^{\text{SEP}}) \leq \frac{1}{2}$, which implies the theorem.

On the other hand, suppose that $\text{val}^{\text{co}}(\mathfrak{G}^{\text{SEP}}) < 1$. Then there exists some $m \geq n_0$ such that $\mathcal{A}$ halts on input $\mathfrak{G}^{\text{SEP}}$ after m steps, so $\mathcal{V}_n^{\text{SEP}}$ has a value-1 PCC strategy for all $n \geq m$ (i.e. the players

do not respond with any answers). Thus by the completeness statement of Theorem 12.2 and an induction argument, we have that $\mathcal{V}_n^{\mathrm{SEP}}$ has a value-1 PCC strategy for all $n \geq n_0$, which implies that $\mathrm{val}^*(\mathfrak{G}^{\mathrm{SEP}}) = 1$, a contradiction because of $\mathrm{val}^*(\mathfrak{G}^{\mathrm{SEP}}) \leq \mathrm{val}^{\mathrm{co}}(\mathfrak{G}^{\mathrm{SEP}})$. This completes the theorem. $\qquad\square$

References

[Ara02]   PK Aravind. A simple demonstration of Bell's theorem involving two observers and no prob-
            abilities or inequalities. *arXiv preprint quant-ph/0206070*, 2002. 7.2, 7.2

[AS98]   Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of
            NP. *Journal of the ACM (JACM)*, 45(1):70–122, 1998. 7

[BFL91]   László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has
            two-prover interactive protocols. *Computational complexity*, 1(1):3–40, 1991. 1.1, 1.1, 1.2, 7

[BSGH+06]   Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust
            PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*,
            36(4):889–974, 2006. 2

[BSS05]   Eli Ben-Sasson and Madhu Sudan. Simple PCPs with poly-log rate and query complexity.
            In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages
            266–275. ACM, 2005. 2

[BVY17]   Mohammad Bavarian, Thomas Vidick, and Henry Yuen. Hardness amplification for entangled
            games via anchoring. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory
            of Computing*, pages 303–316. ACM, 2017. 2, 11, 11.2, 11.2

[CHSH69]   John F Clauser, Michael A Horne, Abner Shimony, and Richard A Holt. Proposed experiment
            to test local hidden-variable theories. *Physical review letters*, 23(15):880, 1969. 1.2

[CHTW04]   Richard Cleve, Peter Hoyer, Benjamin Toner, and John Watrous. Consequences and limits of
            nonlocal strategies. In *Proceedings. 19th IEEE Annual Conference on Computational Com-
            plexity, 2004.*, pages 236–249. IEEE, 2004. 1.2

[CLP15]   Valerio Capraro, Martino Lupini, and Vladimir Pestov. *Introduction to sofic and hyperlinear
            groups and Connes' embedding conjecture*, volume 2136. Springer, 2015. 1.4

[CM14]   Richard Cleve and Rajat Mittal. Characterization of binary constraint system games. In *Inter-
            national Colloquium on Automata, Languages, and Programming*, pages 320–331. Springer,
            2014. 7.2

[Col19]   Andrea Coladangelo. A two-player dimension witness based on embezzlement, and an
            elementary proof of the non-closure of the set of quantum correlations. *arXiv preprint
            arXiv:1904.02350*, 2019. 1.3

[Con76]   Alain Connes. Classification of injective factors cases II_1, II_∞, III_λ, $\lambda \neq 1$. *Annals of
            Mathematics*, pages 73–115, 1976. 1, 1.3, 1.3

[CS18]   Andrea Coladangelo and Jalex Stark. Unconditional separation of finite and infinite-
            dimensional quantum correlations. *arXiv preprint arXiv:1804.05116*, 2018. 1.3

[DLTW08]   Andrew C Doherty, Yeong-Cherng Liang, Ben Toner, and Stephanie Wehner. The quantum
            moment problem and bounds on entangled multi-prover games. In *2008 23rd Annual IEEE
            Conference on Computational Complexity*, pages 199–210. IEEE, 2008. 1.3, 1.4, 12.4

[DPP19]   Ken Dykema, Vern I Paulsen, and Jitendra Prakash. Non-closure of the set of quantum corre-
          lations via graphs. *Communications in Mathematical Physics*, 365(3):1125–1142, 2019. 1.3

[DSV15]   Irit Dinur, David Steurer, and Thomas Vidick. A parallel repetition theorem for entangled
          projection games. *Computational Complexity*, 24(2):201–254, 2015. 11, 23

[Eke91]   Artur K Ekert. Quantum cryptography based on bell's theorem. *Physical review letters*,
          67(6):661, 1991. 1.2

[FJVY19] Joseph Fitzsimons, Zhengfeng Ji, Thomas Vidick, and Henry Yuen. Quantum proof systems
          for iterated exponential time, and beyond. In *Proceedings of the 51st Annual ACM SIGACT
          Symposium on Theory of Computing*, pages 473–480. ACM, 2019. 1.2, 1.2, 1.3, 1.3

[FL92]   Uriel Feige and László Lovász. Two-prover one-round proof systems: Their power and their
          problems. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of comput-
          ing*, pages 733–744. ACM, 1992. 1.1

[FNT14]   Tobias Fritz, Tim Netzer, and Andreas Thom. Can you compute the operator norm? *Proceed-
          ings of the American Mathematical Society*, 142(12):4265–4276, 2014. 1.3, 10, 1.4

[Fri12]   Tobias Fritz. Tsirelson's problem and Kirchberg's conjecture. *Reviews in Mathematical
          Physics*, 24(05):1250012, 2012. 1, 1.3, 1.4

[Har04]   Prahladh Harsha. *Robust PCPs of proximity and shorter PCPs*. PhD thesis, Massachusetts
          Institute of Technology, 2004. 10.4, 10.4.1

[HS66]   Fred C. Hennie and Richard E. Stearns. Two-tape simulation of multitape turing machines.
          Journal of the ACM (JACM), 13(4):533–546, 1966. 3.1

[IKM09]   Tsuyoshi Ito, Hirotada Kobayashi, and Keiji Matsumoto. Oracularization and two-prover one-
          round interactive proofs against nonlocal strategies. In *2009 24th Annual IEEE Conference on
          Computational Complexity*, pages 217–228. IEEE, 2009. 1.2

[IV12]   Tsuyoshi Ito and Thomas Vidick. A multi-prover interactive proof for NEXP sound against en-
          tangled provers. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*,
          pages 243–252. IEEE, 2012. 1.2

[Ji16]   Zhengfeng Ji. Classical verification of quantum proofs. In *Proceedings of the forty-eighth
          annual ACM symposium on Theory of Computing*, pages 885–898. ACM, 2016. 1.2

[Ji17]   Zhengfeng Ji. Compression of quantum multi-prover interactive proofs. In *Proceedings of the
          49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 289–302. ACM, 2017.
          1.2, 1.2, 1.3

[JNP+11] Marius Junge, Miguel Navascues, Carlos Palazuelos, David Perez-Garcia, Volkher B Scholz,
          and Reinhard F Werner. Connes' embedding problem and Tsirelson's problem. *Journal of
          Mathematical Physics*, 52(1):012102, 2011. 1, 1.3, 1.4

[JPY14]   Rahul Jain, Attila Pereszlényi, and Penghui Yao. A parallel repetition theorem for entangled
          two-player one-round games under product distributions. In *2014 IEEE 29th Conference on
          Computational Complexity (CCC)*, pages 209–216. IEEE, 2014. 11

[KKM+11] Julia Kempe, Hirotada Kobayashi, Keiji Matsumoto, Ben Toner, and Thomas Vidick. Entangled games are hard to approximate. *SIAM Journal on Computing*, 40(3):848–877, 2011. 1.2

[Kle54] Stephen C. Kleene. Introduction to Metamathematics. *Journal of Symbolic Logic*, 19(3):215–216, 1954. 14, 12.2

[KPS18] Se-Jin Kim, Vern Paulsen, and Christopher Schafhauser. A synchronous game for binary constraint systems. *Journal of Mathematical Physics*, 59(3):032201, 2018. 1.4

[LFKN90] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *Proceedings of 31st Annual Symposium on Foundations of Computer Science*, pages 2–10. IEEE, 1990. 1.1

[LJ91] H.W. Lenstra Jr. Finding isomorphisms between finite fields. *Mathematics of Computation*, 56(193):329–347, 1991. 3.3.2

[Mer90] David Mermin. Simple unified form for the major no-hidden-variables theorems. *Physical Review Letters*, 65(27):3373, 1990. 7.2

[MP13] Gary L Mullen and Daniel Panario. *Handbook of finite fields*. Chapman and Hall/CRC, 2013. 3.3

[MR18] Magdalena Musat and Mikael Rørdam. Non-closure of quantum correlation matrices and factorizable channels that require infinite dimensional ancilla. *arXiv preprint arXiv:1806.10242*, 2018. 1.3

[NPA08] Miguel Navascués, Stefano Pironio, and Antonio Acín. A convergent hierarchy of semidefinite programs characterizing the set of quantum correlations. *New Journal of Physics*, 10(7):073013, 2008. 1, 1, 1.3, 1.4, 12.4

[NV18a] Anand Natarajan and Thomas Vidick. Low-degree testing for quantum states, and a quantum entangled games PCP for QMA. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 731–742. IEEE, 2018. 1.2, 1.2, 2, 7, 7.1.1, 7.3

[NV18b] Anand Natarajan and Thomas Vidick. Two-player entangled games are NP-hard. In *Proceedings of the 33rd Computational Complexity Conference*, page 20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. 1.2, 2

[NW19] Anand Natarajan and John Wright. NEEXP ⊆ MIP*. *arXiv preprint arXiv:1904.05870v3*, 2019. 1.2, 1.2, 2, 2, 2, 2, 3.4.1, 5.2, 5.17, 5.18, 5.19, 5.20, 5.21, 5.2, 5.2, 7, 7.1.1, 7.3, 7.3.1, 7.3.1, 8.1, 8.3, 8.4.1, 8.4.1, 8.4.3, 8.4.3, 10, 1, 10.4, 10.4.2, 10.4.2, 10.4.2

[Oza13] Narutaka Ozawa. About the Connes embedding conjecture. *Japanese Journal of Mathematics*, 8(1):147–183, 2013. 1.3, 1.4

[Pap94] Christos Papadimitriou. *Computational complexity*. Addison Wesley, 1994. 3.1, 10.2

[Per90] Asher Peres. Incompatible results of quantum measurements. *Physics Letters A*, 151(3-4):107–108, 1990. 7.2

[PV16]   Carlos Palazuelos and Thomas Vidick. Survey on nonlocal games and operator space theory. *Journal of Mathematical Physics*, 57(1):015220, 2016. 7

[Rog87]   Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, Cambridge, MA, USA, 1987. 14, 12.2

[Sch80]   Jacob Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980. 3.20

[Sha90]   Adi Shamir. IP = PSPACE. In *Proceedings of 31st Annual Symposium on Foundations of Computer Science*, pages 11–15. IEEE, 1990. 1.1

[Sho90]   Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. *Mathematics of Computation*, 54(189):435–447, 1990. 3.3.2

[Slo19a]   William Slofstra. The set of quantum correlations is not closed. In *Forum of Mathematics, Pi*, volume 7. Cambridge University Press, 2019. 1, 1.3, 1.4

[Slo19b]   William Slofstra. Tsirelson's problem and an embedding theorem for groups arising from non-local games. *Journal of the American Mathematical Society*, 2019. 1.3

[Tsi93]   Boris S Tsirelson. Some results and problems on quantum bell-type inequalities. *Hadronic Journal Supplement*, 8(4):329–345, 1993. 1.3

[Tsi06]   Boris S Tsirelson. Bell inequalities and operator algebras, 2006. Problem statement from website of open problems at TU Braunschweig (2006), available at http://web.archive.org/web/20090414083019/http://www.imaph.tu-bs.de/qi/problems/33.html. 1, 1.3

[Tur37]   Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1):230–265, 1937. 3.1

[Vid16]   Thomas Vidick. Three-player entangled xor games are np-hard to approximate. *SIAM Journal on Computing*, 45(3):1007–1063, 2016. 1.2

[VW16]   Thomas Vidick and John Watrous. Quantum proofs. *Foundations and Trends in Theoretical Computer Science*, 11(1-2):1–215, 2016. 12.3

[Wan89]   Charles C. Wang. An algorithm to design finite field multipliers using a self-dual normal basis. *IEEE Transactions on Computers*, 38(10):1457–1460, 1989. 3.3.2

[WBMS16]   Xingyao Wu, Jean-Daniel Bancal, Matthew McKague, and Valerio Scarani. Device-independent parallel self-testing of two singlets. *Physical Review A*, 93(6):062121, 2016. 7.2

[Yue16]   Henry Yuen. A parallel repetition theorem for all entangled games. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016. 11

[Zip79]   Richard Zippel. Probabilistic algorithms for sparse polynomials. pages 216–226, 1979. 3.20