Introduction to Quantum Interactive Proof Systems

Liam Salt
APMA 990

April 19, 2022

Abstract

In this paper, we introduce quantum interactive proof systems, demonstrate their advantages over classical systems, and showcase an application by using them to prove certain properties of finite groups. We begin by introducing interactive proof systems as defined classically before extending to the quantum case, whereby we show that any language in PSPACE has a 2-round quantum interactive proof system, which is strictly more powerful than the classical case. Then, we show an application to group theory wherein succinct (polynomial length) proofs for group non-membership are possible with a quantum proof system, which is impossible classically. Construction of efficient proofs for group non-membership also allows us to approach other group theoretical problems, such as finding the maximal normal subgroup and finding whether an integer divides the order of a group.

1 Introduction

1.1 Some Computer Science Background

Before we can begin to define a quantum proof system, we first must introduce some classical computing formalism and terminology. In classical computing, an *interactive proof system* is a Turing machine that encapsulates the mathematical idea of a proof. It models a proof as a sequence of communications between two actors: a potentially dishonest *prover*, and a skeptical, honest *verifier*. Through various *rounds* of communication, the prover attempts to convince the verifier to accept their proof. The typical convention (used in this paper) is to assume the verifier is constrained to be a deterministic polynomial-time Turing machine, whereas we make no constraints to the computational power of the prover. Additionally, an interactive proof system is assumed to satisfy the following two properties:

(i) An interactive proof system is said to be complete or to satisfy completeness if for any true statement, the prover can always convince the verifier of its validity.

(ii) An interactive proof system is said to be sound if for any false statement, the prover can only with negligible probability convince the verifier of its validity.

Remark 1.1. *Often, many of these conditions are loosened in various ways, e.g. the prover only needing to convince the verifier up to some probability.*

Now we can more formally define a (deterministic) interactive proof system.

Definition 1.1. *We say that a formal language L has a deterministic k-round interactive proof system with error probability ε with prover verifier pair (P, V) if all of the following:*

(i) (P, V) either accepts or rejects any input after k rounds.

(ii) $\forall x \in L, (P, V)$ accepts x with probability 1.

(iii) $\forall x \notin L, (P, V)$ accepts x with probability ε.

*Any such languages are said to lie in the class **dIP**.*

Given this definition, a familiar, degenerate example is to show that any language in **NP** has a 1−round interactive proof system with error probability 0. We recall that one definition of **NP** is as the set of decision problems whose positive answers can be verified in polynomial time by a deterministic Turing machine. Accordingly, if a problem is in **NP**, then by definition on input our prover can produce a polynomial-sized certificate, which can be verified in polynomial-time exactly when the input is in the language. In fact, **dIP** is exactly equal to **NP**, that is we can capture the full power of deterministic interactive proof systems exactly with the class **NP**.

We can generalize the above definition by allowing the verifier to also have access to a random number generator, with which it may generate and make use of uniformly random bits in each round. The addition of the verifier's access to randomness brings us to the full definition of an interactive proof system. The complexity class of all languages that have interactive proof systems is termed **IP**, and it can be shown that **IP=PSPACE** ([Kit00]). In other words, problems which can be accepted by an interactive proof system are exactly those problems which require only polynomial-sized space (memory) to answer.

1.2 Arthur-Merlin Protocol

Related to **IP**, another class which will become important in the quantum case is **MA**, which stands for Merlin-Arthur (of King Arthur fame), and is comprised of languages which have a particular type of interactive proof system known as a Merlin-Arthur protocol. This class was originally described in [Bab85] and [Gol89].

Definition 1.2. *A language L is said to have a Merlin-Arthur protocol with prover-verifier pair (Merlin, Arthur) or that L is in the class **MA** if:*

(i) Arthur has access to random bits $y \in B_m$, which are known to Merlin.

(ii) If $x \in L$, then there exists $z \in B_n$ such that $P(Arthur\ accepts\ x|y, z) \geq \frac{2}{3}$.

(iii) If $x \notin L$, then for any $z \in B_n$ such that $P(Arthur\ accepts\ x|y, z) \leq \frac{1}{3}$.

(iv) n, m are both polynomial-length.

For clarity, in the above definition z is the polynomial-sized proof certificate provided by Merlin, and whether Arthur accepts Merlin's proof depends on both the string sent by Merlin and the random string produced by Arthur.

The probability conditions given above are quite flexible. Given a language with a Merlin-Arthur protocol, it can be shown that we can induce a new Arthur-Merlin pair for that language which achieves perfect completeness and soundness with error probability at most $\frac{1}{2}$ ([Wat00]).

Moving forward, it will be important to see how **MA** relates to other complexity classes:

Proposition 1.1. ***MA*** *contains both* ***NP*** = ***dIP*** *and* ***BPP****.*

Proof. The second containment is easy to see given the definition of **BPP**, as Arthur has access to random bits and a polynomial-time Turing machine (with acceptance prob $\frac{2}{3}$ as required in **BPP**); therefore, Arthur need not send anything to Merlin, as he is capable of solving any problem in **BPP** himself.

Similarly for **NP**, Merlin can solve the problem then send Arthur the proof certificate which he can verify deterministically, i.e. without needing his random bits. □

2 Quantum Interactive Proof Systems

Now that we have seen the definition of interactive proof systems in the classical case, we can discuss what happens if we allow the Turing machines access to quantum computations. It has been observed that allowing access to quantum computation lends itself to significant speed-up in many classical problems (integer factorization, computing discrete logarithms, efficient unstructured search, etc.), so it makes sense to ask whether any of the above definitions in Section 1 are significantly affected by access to quantum circuits. It turns out that here, as in the case of Deutsch-Jozsa, we see that the principal speedup comes in the form of query complexity, or in our case, the number of rounds required in our proofs.

Remark 2.1. *The Turing machine model is sufficient, but for the remainder of this paper we will use the equivalent circuit model of quantum computation.*

Specifically, a quantum interactive proof system is one where the verifier has access to quantum circuits which can solve problems in **BQP**, while the prover is still unrestricted (within the boundaries of quantum mechanics). The convention is for the messages exchanged between prover and verifier to be encoded as quantum states in the usual n-qubit Z-basis, however this is not strictly required. Some relevant terminology is that we call the class of all languages that have quantum interactive proof systems Quantum **IP** or **QIP**.

Definition 2.1. *A language L is said to have a quantum interactive proof system with error probability ε, or to be in* ***QIP****, if for any input x:*

(i) The verifier accepts or rejects x using a quantum circuit solving problems in ***BQP****.*

(ii) If $x \in L$, then after polynomial-many rounds of communication, the verifier accepts x with probability 1*.*

(iii) If $x \notin L$, then regardless of the actions of the prover, the verifier accepts x with probability at most ε.

We can see that this definition for **QIP** looks quite similar to the classical definition of **IP**, which we discuss more in the next section. In its current state **QIP** is actually only as powerful as **IP**, and in fact, **QIP**=**IP** = **PSPACE**, which is shown in [Wat09].

2.1 Relationship between QIP and PSPACE

Despite the simplicity of the claim at the end of the previous section, we can say a bit more about the relationship between **QIP** and **PSPACE**. Namely, we can put a very nice bound on the number of queries required in a quantum interactive proof system to determine any language in **PSPACE**. In fact, we can prove that **PSPACE** $\subseteq$ **QIP**[**2**], the class of languages with 2-round quantum interactive proof systems, a much tighter bound than what was claimed in the previous section. Importantly, this also shows that in the constant round case, **IP**[**k**] is strictly outperformed by **QIP**[**2**], demonstrating the power of moving to quantum computations. We therefore introduce the first theorem, which is the principal subject of [Wat99]:

Theorem 2.1. *Every language in* ***PSPACE*** *has a* 2*-round quantum interactive proof system with exponentially small error probability.*

To prove the above theorem, we need to show that any language in **PSPACE** has a 2-round quantum proof system, but it will be sufficient to prove the theorem for a language which is **PSPACE**-complete. In particular, we show the theorem for the language of *true quantified boolean formulas (QBF)*, that is the language of true expressions $Q = Q_1 x_1 \cdots Q_n x_n f(x_1, \ldots, x_n)$, where Q_i is a universal or existential quantifier $(\forall, \exists)$, x_i are boolean valued variables, and f is a boolean formula $B_n \to B$ consisting of operations $\wedge, \vee$, and $\neg$ on the variables $x_1, \ldots, x_n$.

There is a classical protocol outlining an interactive proof system for the QBF problem [She92], but this is general takes $N = \binom{n+1}{2} + n$ rounds of communication, i.e. quadratic rounds with respect to number of variables. A full description of this classical protocol is not shown in this paper, but it is outlined in [Wat99]:

A general outline of the algorithm for a formula $Q = Q_1 x_1 \cdots Q_n x_n f(x_1, \ldots, x_n)$ of length d with n variables is:

1. The verifier sends the prover a uniformly random element r_1 of a chosen finite field, and gets a corresponding function f_1 in response from the prover.

2. They continue this exchange N times, with each response polynomial f_i being at most $\deg d$ and depending only on the previously received random elements $r_1, \ldots, r_i$.

3. After receiving the final f_N, the verifier inputs all of the data they have collected into a single predicate (evaluable in polynomial-time): $E(Q, r_1, \ldots, r_N, f_1, \ldots, f_N)$, which is defined in [She92].

4. The verifier accepts if and only if $E(Q, r_1, \ldots, r_N, f_1, \ldots, f_N)$ evaluates to true.

After some analysis, this serves to prove that in the classical case, **IP** = **PSPACE**, and we therefore have already that **PSPACE** $\subseteq$ **QIP**, as we can simulate the classical circuit in the quantum case. Thus, it remains to show that allowing our system access to quantum circuits helps to reduce the number of rounds.

Now, we outline a proof of Theorem 2.1 by showing existence of a correct $2-$round quantum protocol for the language of true quantified boolean formulas.

Proof. We construct a quantum circuit which consists of three 2D arrays of quantum registers: $\mathbf{R} = \{\mathbf{R}\}_{i,j}, \mathbf{S} = \{\mathbf{S}\}_{i,j}$, and $\mathbf{F} = \{\mathbf{F}\}_{i,j}$, for $1 \leq i \leq m$ and $1 \leq j \leq N$, where $N = \binom{n+1}{2} + n$

as above, and $m = q(n)$ for some polynomial q. There are $m \cdot N$ many registers found in the collections $\mathbf{R}$ and $\mathbf{S}$, each consisting of k qubits, where $\mathbb{F}_{2^k}$ is some chosen finite field.

Note that in the classical protocol, the finite field is chosen such that it is as large as computationally practical, as the error rate goes down exponentially in k.

Each register in $\mathbf{F}$ consists of $k(d+1)$ qubits, where d is the total length of the inputted formula. Each $\mathbf{F}_{i,j}$ represents a polynomial of degree at most d with coefficients in $\mathbb{F}_{2^k}$, for example if $d = 3$ and $k = 2$, we represent the polynomial $3x^3 + 2x + 1$ as:

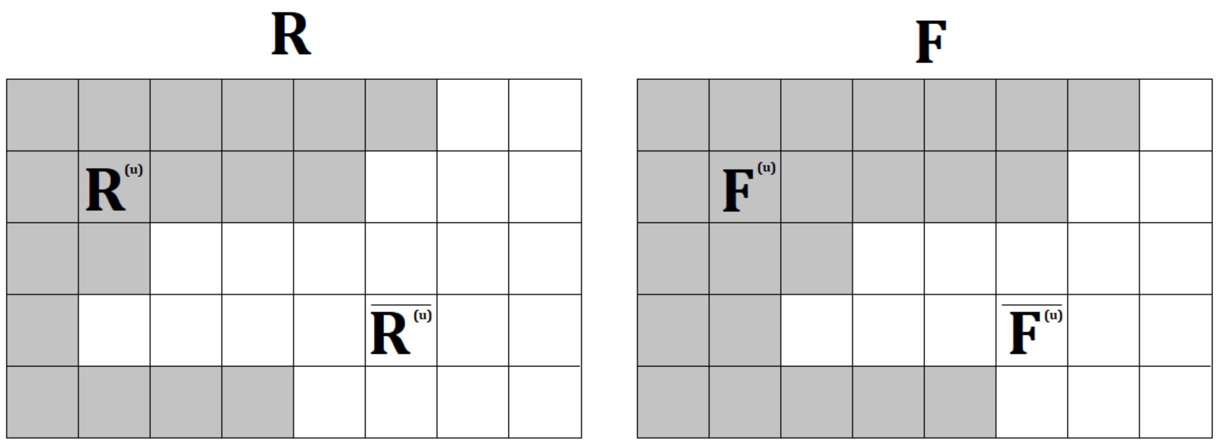
$$3x^3 + x + 1 \rightarrow |11\rangle|00\rangle|10\rangle|01\rangle$$

Additionally, the verifier needs to store classically the value of a uniformly randomly generated vector $u \in \{1, \dots, N\}^m$. Then, according to the vector u, we partition the collections $\mathbf{R}$ and $\mathbf{F}$ into two parts:

$$\begin{array}{rl} R^{(u)} = \{R_{i,j} | j \in \{1, \dots, u_i - 1\}\} & \overline{R^{(u)}} = R \setminus R^{(u)} \\ F^{(u)} = \{F_{i,j} | j \in \{1, \dots, u_i\}\} & \overline{F^{(u)}} = F \setminus F^{(u)} \end{array}$$

In summary the arrays $\mathbf{R}$ and $\mathbf{S}$ each consist of mNk qubits and will later correspond to the random bits from the classical protocol. The array $\mathbf{F}$ consists of $mNk(d + 1)$ many qubits and will later correspond to the polynomials used in the classical protocol. The protocol therefore altogether requires $mNk(d+3)$ qubits and some small number of classical bits to store the vector u.

An example of $\mathbf{R}$ and $\mathbf{F}$ with $N = 8$, $m = 5$, and $u = (7, 6, 3, 2, 5)$ is shown below:



The protocol proceeds as follows:

1. The prover sends $\mathbf{R}$, and $\mathbf{F}$ which they calculate using their unlimited computing power.
2. Using the classical verification protocol described in [She92] for each pair of rows in the above table ($\mathbf{R}_i$,$\mathbf{F}_i$), the verifier rejects the proof if any pair contain an invalid proof that the input formula evaluates to true.
3. The verifier chooses a vector u uniformly randomly from $\{1, \ldots, N\}^m$ and sends it to the prover along with $\overline{\mathbf{F}^{(u)}}$.
4. In the second round, the prover sends S to the verifier, and the verifier calculates $\mathbf{R}_{i,j} - \mathbf{S}_{i,j}$ for each pair (i, j).
5. The verifier applies $H^{\otimes k}$ to each register of $\overline{\mathbf{R}^{(u)}}$ and accepts if and only if $\overline{\mathbf{R}^{(u)}}$ contains only zeros.

The entries in $\mathbf{R}$ and $\mathbf{F}$ correspond to the random values r_i and functions f_i from the classical protocol. A full proof of correctness is found in [Wat99], but we provide an outline here. First suppose that a given formula Q is true. We want to show that the verifier accepts it with certainty. The prover also creates an $m \times N$ matrix R of values drawn uniformly randomly from $\mathbb{F}_{2^k}$, where m, N, and k are defined as above. Using the values of R, the prover also generates a corresponding $m \times N$ matrix of *correct* polynomials F. Each row F_i is formed by defining polynomials $f_{i,j}$ as in the classical protocol, where instead of getting the random values r_j from the verifier, they are taken from the corresponding row R_i of the generated matrix R. Thus, $F_{i,j} = f_{i,j}(R_{i,1}, \ldots, R_{i,N})$, for $i = 1, \ldots, m, j = 1, \ldots, N$. The prover then prepares the following state:

$$\frac{1}{\sqrt{2^{kmN}}} \sum_{i,j} |R_{i,j}\rangle |F_{i,j}\rangle$$

Forming the registers $\mathbf{R}$ and $\mathbf{F}$ as described in the protocol. Additionally, if the formula Q is correct, the prover sets each $\mathbf{S}_{i,j}$ to $\mathbf{R}_{i,j}$. The prover then sends this state to the verifier. As we are assuming Q is true, the honest verifier will not reject at this point, and they will return $\overline{\mathbf{F}^{(u)}}$ and u to the prover in response. Then, for each (i, j) define a unitary:

$$U_{i,j} : |R\rangle|0\rangle \mapsto |R\rangle|F_{i,j}\rangle$$

The prover then applies $U_{i,j}^{-1}$ to registers $\mathbf{S}_{i,j}$ and $\mathbf{F}_{i,j}$ for all $\mathbf{F}_{i,j}$ in $\overline{\mathbf{F}^{(u)}}$, which keeps $\mathbf{S}$ the same, but returns all of $\overline{\mathbf{F}^{(u)}}$ to $|0\rangle$. After the verifier subtracts each $\mathbf{R}_{i,j}$ from each $\mathbf{S}_{i,j}$, the registers in $\overline{\mathbf{R}^{(u)}}$ will no longer be entangled with any other registers; therefore each register in $\overline{\mathbf{R}^{(u)}}$ is sent to $|0\rangle$, and the verifier will accept Q with certainty.

Not discussed in this paper, but available in [Wat99], is the case where Q is not a true formula. In this case, it turns out that the verifier can be made to accept only with some small error probability. An upper bound for this probability is determined by m and k. If we choose $m = (d+1)N$ and $k = 2d + 6 + \lceil \log(dm) \rceil$, then $\varepsilon < 2^{-d}$, i.e. the error probability is exponentially small in the length of Q. □

Therefore, we can show that **PSPACE** $\subseteq$ **QIP[2]**; moreover, there is an $\varepsilon < 2^{-|x|}$ 2-round quantum protocol for any language L in **PSPACE**.

2.2 QMA

Having now introduced quantum interactive proof systems and the classical class **MA**, we can discuss its natural extension **QMA**.

Definition 2.2. *A language L is in said to be in the class* ***QMA*** *if there exists a family of polynomial-time quantum circuits such that:*

(i) If $x \in L$, then there exists a state $|\Psi\rangle$ such that $P(\text{Circuit accepts } |\Psi\rangle) > \frac{2}{3}$

(ii) If $x \notin L$, then for any state $|\Psi\rangle$, $P(\text{Circuit accepts } |\Psi\rangle) < \frac{1}{3}$

There are several important results to do with this class. For example, it has been shown by Kitaev that $\mathbf{QMA} \subseteq \mathbf{P}^{\sharp P}$ ([Kit99]). Another result of Marriott and Watrous is that $\mathbf{QMA} \subseteq \mathbf{PP}$ ([Mar05]), and further, if the reverse containment holds, i.e. if $\mathbf{QMA} = \mathbf{PP}$, it can be shown that $\mathbf{PH} \subseteq \mathbf{PP}$ ([Vya03]).

3 Group Theoretic Applications

Having introduced it in the previous section, we proceed to discuss **QMA** in the context of group theory. The principal result of [Wat00] is a proof of the fact that the group non-membership problem is in **QMA**. This section will review the proof of this fact, along with some corollaries that allow us to recover various other group properties.

3.1 Group Non-Membership

We start by defining the group non-membership problem (GNM).

Definition 3.1. *Given a finite group G and $k+1$ elements $g_1, g_2, \ldots, g_k, h \in G$, the group non-membership problem asks whether h is outside the subgroup generated by the elements g_i,i.e. is it the case that $h \notin (g_1, \ldots, g_k) \subseteq G$?*

Remark 3.1. *It is a result of [Bab91] that there exist oracles with respect to which GNM $\notin$* ***NP*** *and GNM $\notin$* ***BPP****.*

This definition makes no mention of any particular representation of the finite group G, so the convention used in this paper is to represent group elements as finite length binary strings under some relations. To perform the group operation, we employ a *group oracle* which we denote B.

Definition 3.2. *A group oracle B is a family of bijections $\{B_n\}$, each defined $B_n\colon \{0,1\}^{2n+2} \to \{0,1\}^{2n+2}$, satisfying certain properties defined below. Given a subset of strings $S \subseteq \{0,1\}^n$, we can define a group on S, denoted $G_S(B_n)$ and called a black-box group, whose operation is given by queries to B, as follows:*

(i) If $x, y \in G_S(B_n)$, we have that $y \cdot x = z$, where z is given by $B_n(0,0,x,y) = (0,0,x,z)$.

(ii) If $x, y \in G_S(B_n)$, we have that $y \cdot x^{-1} = z$, where z is given by $B_n(1,0,x,y) = (1,0,x,z)$.

(iii) If x or $y \notin G_S(B_n)$, we have that $B_n(c,0,x,y) = (c,1,x,y)$, for $c = 0,1$.

The first and second input bits to the maps B_n are the control and error bits, respectively. The control bit, denoted c, controls whether we take the inverse of the second group element in the operation. The error bit, denoted b, is initialized to $|0\rangle$, and is flipped to $|1\rangle$ exactly when at least one input bit is not a group element. It should be noted that any finite group can be interpreted as a black-box group under some oracle, supposing that we encode its elements as binary strings in a suitable way.

Given that each map in B is defined as a bijection, and is hence invertible, we can take each B_n as a definition for a quantum gate acting on $2n+2$ qubits written in the usual computational basis.

Theorem 3.1. *The Group Non-Membership Problem is in* $\boldsymbol{QMA}^B$ *for an arbitrary group oracle* B.

Proof. Let G be a finite group whose elements are written as binary strings of length n, let $h \in G$ be a distinguished element, and suppose for some $g_1, \ldots, g_k \in G$, we have $H = (g_1, \ldots, g_k) \subseteq G$ a finitely generated subgroup of G. We aim to determine whether $h \in H$. To this end we define a quantum circuit as follows:

For any subset $A \subseteq G$, we define the state $|A\rangle := |A|^{-1/2} \sum_{g \in A} |g\rangle$. Let $\mathbf{R}$ be an input register initialized to the state $|H\rangle$, and $\mathbf{B}$ be an input qubit initialized to $|+\rangle$. Taking the state $|+\rangle|H\rangle$, we multiply by h by applying a controlled B_n gate, whose output will be:

$$\frac{1}{\sqrt{2}}|0\rangle|H\rangle + \frac{|H|^{-1/2}}{\sqrt{2}} \sum_{g \in H} |1\rangle B_n(0,0,g,h) = \frac{1}{\sqrt{2}}|0\rangle|H\rangle + \frac{1}{\sqrt{2}}|1\rangle|Hh\rangle$$

Then apply a Hadamard gate to the first register, taking us to:

$$\frac{1}{2}(|0\rangle(|H\rangle + |Hh\rangle) + |1\rangle(|H\rangle - |Hh\rangle))$$

If $h \in H$, then $H = Hh$, as H is a subgroup. Then, $|H\rangle - |Hh\rangle = 0 \cdot |H\rangle$, so if we measure $\mathbf{B}$ in the Z-basis, we will measure $|0\rangle$ with certainty. If $h \notin H$, then the probability that we measure $|1\rangle$ is $\frac{1}{2}$, as H and Hh are disjoint cosets of G. Thus, if we can create m copies of the state H, a verifier can tell with certainty that $h \in H$ whenever it's true, and will be deceived by the prover with probability $\frac{1}{2^m}$ when $h \notin H$.

Therefore, we have shown that, in the case where $h \not H$, if the prover sends polynomial many copies $|H\rangle$ as a certificate, then the verifier will accept the true statement that $h \notin H$ with near certainty. In particular, the verifier will accept with probability higher than the requisite $\frac{2}{3}$ bound for **QMA**. Additionally, the verifier will reject false claims that $h \notin H$ with certainty, also meeting the **QMA** requirement.

A major caveat for this conclusion is that, in the event that $h \in H$, the dishonest prover need not send $|H\rangle$, and may opt to send any arbitrary $|\Psi\rangle$ instead (to deceive the verifier). The remedy involves making use of the additional power available to the verifier in **QMA** to first perform a verification that the received certificate $|\Psi\rangle$ is equal to $|H\rangle$. The discussion and completion of proof after this consideration is fully detailed in [Wat00], making use of a lemma of [Bab85], but the conclusion is that despite the potential roadblocks presented by the prover's behaviour, we can still recover a QMA protocol that operates with exponentially small error. $\square$

An important result (not proved in this paper) is that there is a strict advantage in moving from **MA** to **QMA** in that, there exists an oracle B relative to which GNM $\notin \mathbf{MA}^B$. That is to say, we achieve a strict advantage in moving to a system involving quantum computations. This separation in the GNM problem additionally leads to the complexity theoretic result that there exists some oracle B such that $\mathbf{BQP}^B \not\subseteq \mathbf{MA}^B$ ([Wat00]).

3.2 Other Applications

We conclude this section by considering some important corollaries of the main result of the previous section. The fact that we can provide succinct (polynomial sized) proofs of group non-membership in the context of **QMA** allows us to also produce results in the same context for problems involving several other group-theoretical properties, including:

1. The **Proper Subgroup** problem: given a group G and elements $g_1, \ldots, g_k, h_1, \ldots, h_l \in G$, is it the case that $(h_1, \ldots, h_l) \subsetneq (g_1, \ldots, g_k)$?
2. The **Divisor of Order** problem: given a group G and an integer N, do we have that N divides $|G|$?
3. The **Subgroup Simplicity** problem: given a group G and a subgroup $H = (g_1, \ldots, g_k) \subseteq G$, is H simple (no non-trivial normal subgroups)?
4. The **Maximal Normal Subgroup** problem: given a group G and elements $g_1, \ldots, g_k$, $h_1, \ldots, h_l \in G$, is $(h_1, \ldots, h_l)$ a maximal normal subgroup of $(g_1, \ldots, g_k)$?

It turns out that as corollaries of GNM$\in\mathbf{QMA}^B$, problems 1 and 2 are also both in $\mathbf{QMA}^B$ for some oracle B, while problems 3 and 4 are actually in $\mathbf{co\text{-}QMA}^B$ ([Bab92]).

4 Conclusion

This paper introduced quantum interactive proof systems primarily by discussing two results. The first result is broad in scope and says that any language in **PSPACE**, that is, a language which can be solved with a polynomial-sized memory, can be equivalently determined using only two rounds of communication between prover and verifier with access to quantum computations. This result was established incrementally over time by Watrous, who originally identified that such languages belong to **QIP[k]** for some constant k. This result was tightened to $k = 3$, and then finally 2. It is possible that this result could be even further tightened to the 1-round case, or to the 0-round case (wherein the verifier can simply solve the problem themselves). It is still open as to what languages are characterized by **QIP[k]** for $k > 2$. We could also establish relationships between **PSPACE** and other classes by asking whether they have quantum proof systems; for example we can ask about the relationship between **NEXP** and **QIP**.

The second portion of this paper was dedicated to the discussion of **QMA** and its relationship with the Group Non-Membership problem. Some additional group-theoretic applications were discussed at the end of Section 3, but there are numerous other applications, e.g. the Graph Non-Isomorphism problem is in **QMA**. The main direction of continuing research from Section 3 mainly comprises complexity theoretic results related to **QMA**. Some of these were mentioned in Section 2.1, but one additional direction suggested in [Wat00] is to ask whether $\mathbf{co\text{-}NP} \subseteq \mathbf{QMA}$, and if so, whether there are any interesting consequences.

References

[Bab85] L. Babai, Trading Group Theory for Randomness, Proceedings of the Seventeenth Annual ACM Symposium on the Theory of Computing, 1985, 421-429.

[Bab91] L. Babai. Local expansion of vertex-transitive graphs and random generation in finite groups. Twenty-Third Annual ACM Symposium on Theory of Computing, 1991, 164-174.

[Bab92] L. Babai. Bounded round interactive proofs in finite groups. SIAM Journal on Discrete Math, 1992, 88-111.

[Gol89] S. Goldwasser, M. Sipser. Private coins versus public coins in interactive proof systems, Randomness and Computation, volume 5 of Advances in Computing Research, JAI Press, 1989, 73-90.

[Kit99] A. Kitaev, Quantum NP, Talk at AQIP'99: Second Workshop on Algorithms in Quantum Information Processing, DePaul University, 1999.

[Kit00] A. Kitaev, J. Watrous, Parallelization, amplification, and exponential time simulation of quantum interactive proof system,32nd Annual ACM Symposium on Theory of Computing, 2000, 608–617.

[Mar05] C. Marriott, J. Watrous, Quantum Arthur-Merlin Games, 2005. https://doi.org/10.48550/arXiv.cs/0506068

[She92] A. Shen. IP = PSPACE: simplified proof. Journal of the ACM, 1992, 878–880.

[Vya03] M. Vyalyi. QMA=PP implies that PP contains PH, 2003.

[Wat99] J. Watrous, PSPACE has 2-round quantum interactive proof systems, 1999. https://doi.org/10.48550/arXiv.cs/9901015

[Wat00] J. Watrous, Succinct quantum proofs for properties of finite groups, 2000. https://doi.org/10.48550/arXiv.cs/0009002

[Wat09] J. Watrous, S. Upadhyay, Z. Ji, R. Jain, QIP=PSPACE, 2009. https://doi.org/10.48550/arXiv.0907.4737