

Quantum Complexities and Interactive Proof Systems

Larisse D. Voufo

December, 14th 2006

Abstract

In complexity studies, Interactive Proof Systems are a very helpful means to establish relationships between major complexity classes. The rise of new complexity classes, which follow the properties of quantum computation, suggests a new set of variants of these systems. This paper focuses on the currently established theories in quantum complexity studies, and studies how quantum complexity classes relate to some variants of Interactive Proof Systems.

Outline. In this paper, we will first overview the history and fundamentals of quantum complexity studies and interactive proof systems. Then, we will go on a journey from classical complexity classes to quantum ones, to introducing interactive proof systems and their relationships with quantum complexity classes. Lastly, we will study the relationships between the complexity classes studied so far.

Keywords. Bounded-error, Circuits, Circuit Model, Classical, Completeness, Complexity Classes, Interactive, Measurement, Merlin-Arthur, (Non) Deterministic, Oracle, Polynomial, Probabilistic, Proof systems, Quantum, Relativity, Reversibility, Universality.

Contents

| | | |
|----------|--|----------|
| 1 | Introduction: Historical Overview | 2 |
| 2 | Fundamentals of quantum complexity studies: An Overview | 3 |
| 2.1 | Quantum algorithm is probabilistic | 3 |
| 2.2 | Probabilistic computation vs. quantum computation. | 3 |
| 2.3 | Issues arising from a quantum circuit model. | 4 |
| 2.3.1 | Universality | 4 |
| 2.3.2 | Better characterizing the resources needed | 4 |
| 2.3.3 | Accuracy | 4 |
| 2.4 | Performance Analysis of Quantum Algorithms over Classical ones | 5 |
| 2.4.1 | Non-Exponential speed-up | 5 |

| | | |
|----------|---|-----------|
| 2.4.2 | “Relativized” exponential speed-up | 5 |
| 2.4.3 | Exponential speed-up for “apparently” hard problems | 5 |
| 3 | Complexity Classes | 6 |
| 3.1 | Exact Classical Complexity Classes | 6 |
| 3.2 | Probabilistic Complexity Classes | 6 |
| 3.3 | Quantum Complexity Classes | 6 |
| 4 | Interactive Proof Systems | 7 |
| 4.1 | From IP to QIP | 7 |
| 4.2 | From NP to QMA, through MA. | 7 |
| 4.3 | Other Variations of IP: | 8 |
| 4.3.1 | MIP: | 8 |
| 4.3.2 | IPP: Unbounded IP | 8 |
| 4.3.3 | compIP: competitive IP | 8 |
| 5 | Relationships between Complexity classes | 9 |
| 5.1 | $P \subseteq NP \subseteq AWPP \subseteq PP \subseteq P\#P \subseteq PSPACE$ | 9 |
| 5.2 | $P \subseteq BPP \subseteq BQP \subseteq AWPP \subseteq PP \subseteq P\#P \subseteq PSPACE$ | 9 |
| 5.3 | NP, Co-NP and BPP are partially disjoint | 9 |
| 5.4 | $BPP \subseteq MA \subseteq IP$ and $EQP \subseteq BQP \subseteq QMA \subseteq QIP$ | 9 |
| 5.5 | $IP = PSPACE$ | 10 |
| 5.6 | Relationships “relative to oracle” | 10 |
| 6 | Conclusion | 10 |
| A | Graphical description of an Interactive Proof System | 11 |
| B | Simulating an Interactive Proof System by a PSPACE machine | 11 |
| C | An Interactive Proof System for Deciding the Truth of QBFs | 13 |

1 Introduction: Historical Overview

The trend towards miniaturization and microcircuitry around the 1980s was a deterministic time for quantum computation. In 1982, Paul Benioff and Richard Feynman (separately) introduced the idea that quantum systems could perform computation. In 1985, for people such as David Deutch, the realization that quantum computers behaved differently from classical Turing machines opened the door to a possibility of new complexity of algorithms. Later on, this idea will be strengthened by the rise of universality of quantum circuits, which yields a machine independent notion of quantum complexity.

Meanwhile, still in 1985, Goldwasser et al. first introduced interactive proof systems. These systems are constituted of a prover (P), with unbounded computational resources,

and a verifier (V), which is a probabilistic polynomial-time machine with access to a random bit string whose length is polynomial on the size n . Here, continuously, P presents a proof that an input x is in some language L and V checks that the presented proof is correct. Once the interactions are completed, after a polynomial number of messages, V must decide whether or not x is in the language with only a $1/3$ chance of error.

Laszlo Babai, later on, introduced a variation of this scheme called the Merlin-Arthur protocol¹, in which Merlin is the prover and Arthur is the verifier. This differs from the previous in that the number of rounds of interaction is bounded by a constant, rather than a polynomial.

2 Fundamentals of quantum complexity studies: An Overview

Before we go further into the study of quantum complexity classes, it is important to understand some fundamental notions of quantum computation. These range from a key property of quantum measurement, to approaches used to compare the efficiencies of quantum algorithms over classical ones, through a study of issues encountered with designing desirable quantum circuit models.

2.1 Quantum algorithm is probabilistic

One key quantum property to remember, in quantum complexity studies, is the fact that the quantum measurement process is random. Due to that fact, an algorithm performed by a quantum computer ought to be probabilistic; meaning that multiple runs of the said algorithm could return different results. However, we should not confuse probabilistic computations with quantum computations

2.2 Probabilistic computation vs. quantum computation.

Let us define a Nondeterministic Computation (NC) as the tree of configurations of Nondeterministic Turing Machines (NTM). While a probabilistic computation is a NC where probabilities are associated with the edges and nodes in the tree, a quantum computation is one where amplitudes are associated with the edges and nodes. Notice that while, in a probabilistic computation, the sum of all the probabilistic outcomes must be 1; in quantum computation, it is the sum of the squares of all the outcomes that must be equal to 1. At this point, it is clear that while a probabilistic computation must obey the rules of classical probability, a quantum computation must obey those of quantum probability. Indeed, the strength of quantum computing lies in the ability to have bad computation paths eliminate each other (through the phenomena of superposition and interference), thus causing some

¹Another variation is called Arthur-Merlin protocol, but defining this is beyond the scope of this paper.

good paths to occur with larger probability. Notice that the ability for quantum machines to take advantage of this interference is tempered by their restriction to unitary operations.

2.3 Issues arising from a quantum circuit model.

Let us define a desired quantum circuit model. In this model, a quantum gate, being a unitary transformation, is reversible. (Notice that a classical reversible computer is a special case of quantum computers.)

$$x^{(n)} \rightarrow y^{(n)} = f(x^{(n)}) \iff U : |xi \rangle \rightarrow |yi \rangle$$

A classical reversible gate performing a permutation of n qubits can be viewed as a unitary transformation $x^{(n)}$ on the computational basis $\{|xi \rangle\}$. This model is unitary because all 2^n strings $|yi \rangle$ are mutually orthogonal. A quantum computation constructed from such classical gates takes $|00 \dots 0 \rangle$ to one of the computational basis states, so that the final measurement is deterministic.

Looking at this model, three issues arise.

2.3.1 Universality

Universality is what we owe our thanks to for the existence of quantum complexity classes. For one thing, the previously defined model would not be of much interest if there exists a transformation in $U(2^n)$ that we cannot reach. Also, it allows the simulation of a quantum computer using another quantum computer, which removes the dependency of complexity classes on the details of the hardware.

2.3.2 Better characterizing the resources needed

This second issue has to do with the fact that, when simulating a quantum computer on a classical computer, there is a need to better characterize the resources needed. We notice that, despite the vastness of Hilbert spaces, a classical computer can still simulate a quantum computer even if limited to an amount of memory space that is polynomial in n .

2.3.3 Accuracy

We define accuracy as the growth of error in measurement as the quantum circuit size increases.

Quantum complexity classes (such as BQP that we will mention again later on), is formally defined under the idealized assumption that quantum gates can be executed with perfect precision. However, that is not always the case and we can get into serious problems when the error in precision becomes very large. As a matter of fact, A polynomial size circuit family that solves a hard problem would not be of much interest if the gates in the circuit were required to have an exponential accuracy. However, An idealized T-gate q-circuit can be simulated with acceptable accuracy by noisy gates, provided that the error probability per gate scales like $1/T$. Moreover, If we have designed a quantum algorithm that solves a decision problem correctly with probability greater than $1/2 + \delta$ (in the ideal case), then we

can perform the gates with an accuracy $T\varepsilon < O(\delta)$. Therefore, a quantum circuit family in the BQP class can really solve hard problems, as long as, we can improve the accuracy of the gates linearly with the computation size T .

2.4 Performance Analysis of Quantum Algorithms over Classical ones

Three approaches are used when comparing the efficiency of quantum algorithms over classical ones.

2.4.1 Non-Exponential speed-up

This corresponds to quantum algorithms that, although computing faster than their classical equivalent, do not change their complexity class. A trivial example of these is Grover's Quantum Speed-up of the Search of an unsorted database, which only produced a quadratic speed-up.

2.4.2 “Relativized” exponential speed-up

Oracles. To understand the concept of oracles, let us imagine a black box containing an unknown function. The point is to perform an algorithm analysis that is oblivious of the function computed within the black box. For instance, David Deutch uses this concept to determine whether the function in the oracle is constant or variant. Hence, the term oracle signifies that the box responds to a query immediately, that is, the time it takes the box to operate is not included in the complexity analysis.

Relativity Very often, we will come across conditions (relations) that are true “relative to a world G ”. This only means that whether the condition is true in general (always) or not, there exists a world G , within which it is unconditionally true. For instance, computer scientist often state that $BPP \neq BQP$ “*relative to oracle*”.

Quantum algorithms for which we observe “relativized” exponential speed-up usually incorporate oracles. Common examples are Simon's exponential quantum speedup for finding the period of 2 to 1 function, and Deutch's square root of not algorithm.

2.4.3 Exponential speed-up for “apparently” hard problems

As the name says, this class of algorithms involves those that produce exponential speed-up for for problem that appear to be hard in classical computations. And example is Shor's factoring algorithm that we are all proud of.

3 Complexity Classes

In this section, we are going to go on a journey from very common non-probabilistic classical complexity classes, to less common ones, to probabilistic ones, and finally to quantum ones.

3.1 Exact Classical Complexity Classes

Here, by “exact”, we mean all those classical complexity classes that are not probabilistic. Classical probabilistic classes will be described later on in section 3.2

P - “easy”. class of languages decided by polynomial-time Turing Machines (TM)

NP: Non-deterministic Polynomials. languages decided by polynomial-time NTM.

The idea is to guess an answer for the given problem, and verify that the answer is correct in polynomial time. Notice that here, the verification returning a “YES” brings us to an accepting state.

NP-hard. Every hard problem can be polynomially reduced to a problem in this class.

NP-complete. Class of NP-hard problems that are also in NP ($\text{NPC} = \text{NP-hard} \cap \text{NP}$).

NPI: Problems in NP of intermediate difficulty. $\text{NPI} = \text{NP} - \text{P} - \text{NPC} = \text{NP} - \text{P} - \text{NP-hard}$

Co-NP. These problems are similar to NP, except for the fact that an answer of “NO” from the verifier brings us to an accepting state. (a counter-example of the problem must be provided).

PSPACE. Decision problems that can be solved in polynomial-space, but may require exponential time.

3.2 Probabilistic Complexity Classes

AWPP. Languages decided by Almost-Wide Probabilistic Polynomial-time NTM.

PP. Languages decided by polynomial-time NTM where the majority of paths gives the correct answer.

P#P. Functions that count the number of accepting paths through an NP machine.

BPP: Bounded-error Probabilistic Polynomial Time. Problems that admit a probabilistic circuit family of polynomial size that always gives the right answer with prob $> 1/2 + \delta$ ($\delta > 0$).

MA. Languages decided by a bounded-error probabilistic Merlin-Arthur protocol.

IP. Problems solvable by an Interactive Proof System.

3.3 Quantum Complexity Classes

BQP: Bounded-error Quantum Polynomial Time. Decision Problems that can be solved, with high probability, by polynomial-size quantum circuits.

EQP (QP). Exact version of BQP

BQNP (= QMA). Quantum analogue of NP

QMA-complete. Quantum analogue of NPC

QIP. Quantum analogue of IP

4 Interactive Proof Systems

We defined Interactive Proof Systems earlier as those systems constituted of a prover (P) (assumed to be infinite in computation and resource), and a verifier (V) (probabilistic polynomial-time machine with access to a random bit string of length polynomial on the size n); Where V must decide whether or not a given input x is in some language L with only a $1/3$ chance of error, after polynomially-bound interactions during which P continuously presents a proof that x is in L while V checks that the presented proof is correct.

Definition. A problem is in the class IP if it is solvable by an Interactive Proof System. By definition,

For any language L , $L \in \text{IP} \Rightarrow \exists V, P \text{ --- } \forall Q, w$:

- $w \in L \Rightarrow \text{Pr}[V \leftrightarrow P \text{ accepts } w] \geq 2/3$
- $w \notin L \Rightarrow \text{Pr}[V \leftrightarrow Q \text{ accepts } w] \leq 1/3$

4.1 From IP to QIP

Take the interactive proof system that describes the complexity class IP as described above. replace the BPP verifier by a BQP one, and the input components of the interaction messages to be quantum data. This results in a description of problems in QIP or Quantum IP. It is not yet known whether QIP strictly contains IP (in other words, whether quantum computations add power to interactive proofs). However, we know that $\text{QIP} = \text{QIP}[3]$, so that more than three rounds are never necessary. In addition, $\text{QIP} \subseteq \text{EXPTIME}$.

4.2 From NP to QMA, through MA.

A Merlin-Arthur protocol is a variation of IP, where the number of messages between P and V is bound by a constant, rather than a polynomial. By setting the verifier Arthur to be a deterministic polynomial-time Turing machine, we get a description of problems in NP. Changing Arthur to a BPP machine “produces” problems in MA. In addition, changing Arthur to BQP yields to the QMA class.

4.3 Other Variations of IP:

There exists multiple variations of interactive proof systems. The following are just a few.

4.3.1 MIP:

the class MIP is described by powerful interactive proof system based on IP, in which there are two independent provers, who cannot communicate with each other once the verifier begins sending messages. This makes it easier to detect a malicious prover trying to trick the verifier, since there is another prover that it can double-check with. So far, we know that $MIP = NEXPTIME$. Also, all languages in NP have zero-knowledge proofs in an MIP system, without any additional assumptions; This is only known for IP assuming the existence of one-way functions.

4.3.2 IPP: Unbounded IP

This is a variant of IP where we replace the BPP verifier by a PP verifier. It modifies the completeness and soundness conditions as follows:

- **Completeness:**

If a string is in the language, the honest verifier will be convinced of this fact by an honest prover with probability at least $1/2$.

- **Soundness:**

If the string is not in the language, no prover can convince the honest verifier that it is in the language, except with probability less than $1/2$.

Currently, we know that $IPP = PSPACE$. However, let us notice that while $IPP=PSPACE$ with respect to all oracles, $IP \neq PSPACE$ with respect to almost all oracles.

4.3.3 compIP: competitive IP

A compIP system weakens the completeness condition in a way that weakens the prover, while IPP and QIP give more power to the verifier:

- **Completeness:**

If a string is in the language L, the honest verifier will be convinced of this fact by an honest prover with probability at least $2/3$. Moreover, the prover will do so in probabilistic polynomial time given access to an oracle for the language L.

Here, the prover is a BPP machine with access to an oracle for the language, but only in the completeness case, not the soundness case. If a language is in compIP, then interactively proving it is in some sense as easy as deciding it. With the oracle, the prover can easily solve the problem, but its limited power makes it

much more difficult to convince the verifier of anything. compIP isn't even known or believed to contain NP.

However, it can solve some problems believed to be hard. For instance, problems such as the graph (non)isomorphism and quadratic non-residuosity are in compIP (oracles).

5 Relationships between Complexity classes

5.1 $P \subseteq NP \subseteq AWPP \subseteq PP \subseteq P\#P \subseteq PSPACE$

From the definitions on these complexity classes, and from the current formalisms of classical complexity theories, it is clear that the chain $P \subseteq NP \subseteq AWPP \subseteq PP \subseteq P\#P \subseteq PSPACE$ is valid. However, although there are some indications that $P \neq NP$, there is still no proof for that relation. Meanwhile, if we can find an NP-hard problem that is also in P, then we have proved that $P = NP$.

5.2 $P \subseteq BPP \subseteq BQP \subseteq AWPP \subseteq PP \subseteq P\#P \subseteq PSPACE$

It is trivial to see that $P \subseteq BPP$, by setting the probability to 1. In addition, since a quantum computer can easily simulate a probabilistic classical one, $BPP \subseteq BQP$ certainly. However, the reverse of this operation (simulating a quantum computer with a probabilistic classical one) cannot be easily done. Thus, we get an indication of BQP being larger than BPP. However, as of today, we do not have a proof for $BPP \neq BQP$ yet.

Moreover, while Bernstein and Vazirani proved that $BQP \subseteq PSPACE$, Adelman, Demarrais and Huang proved that $BQP \subseteq PP$, and Fortnow and Rogers in turn proved that $BQP \subseteq AWPP$.

5.3 NP, Co-NP and BPP are partially disjoint

At this point, we are aware of the chains of complexity classes inclusion described in sections 5.1 and 5.2. However, an information that we miss is the relationship between NP, BPP and BQP. Let us first notice that P is a subset of all NP, co-NP, and BPP. While $NP \neq Co-NP$, $BPP \neq NP$ and $BPP \neq co-NP$, since there are still some problems in BPP that are neither in NP, nor co-NP. Anyways, so far, there is no firm relationship defined between those classes. While $NP \subseteq MA$ by analogy to the P verifier \subseteq BPP verifier, determining whether $NP \subseteq BQP$ is still a subject of research.

5.4 $BPP \subseteq MA \subseteq IP$ and $EQP \subseteq BQP \subseteq QMA \subseteq QIP$

Since we can just have the verifier ignore the prover, we can state that $BPP \subseteq (MA \text{ or } IP)$, and that $MA \subseteq IP$ since a constant bound is definitely lower than a polynomial one. By quantum analogy of the classical relationships between BPP, MA and IP, and by the definitions of EQP and BQP, we can trivially see that $EQP \subseteq BQP \subseteq QMA \subseteq QIP$.

5.5 IP = PSPACE

The proof for $IP = PSPACE$ consists of two main “sub-proofs”: $IP \subseteq PSPACE$ and $PSPACE \subseteq IP$.

Proving that $IP \subseteq PSPACE$ can be trivially done, by presenting a simulation of an interactive proof system by a polynomial space machine.

The proof for $PSPACE \subseteq IP$, however, lays in finding a PSPACE-complete problem, and proving that that problem is in IP. Going in that direction, deciding the truth of QBFs is a problem that has been known to be PSPACE-Complete. Showing that it is in IP as well will then constitute our proof. In other words, let’s represent the problem of deciding the truth of QBFs with TQBF. Then, $(TQBF \subseteq PSPACE\text{-Complete})$ and $(TQBF \subseteq IP) \Rightarrow PSPACE \subseteq IP$.

We can show that $TQBF \subseteq IP$ simply by illustrating the existence of an interactive proof system for solving TQBF.

Further information on this proof are provided in Appendices B and C.

5.6 Relationships “relative to oracle”

Coming back to the terminology of relativity described earlier in section 2.4.2, here are a few cases of its usage. Very often, we will come across computer scientists stating that $BPP \neq BQP$ “relative to oracle”. In addition, if $P = PSPACE$, then $P = AWPP$ “relative to oracle”. Also, $NP = AWPP$ “relative to oracle”.

6 Conclusion

At this point, we have studied the history, fundamentals, and current theories of quantum complexity and interactive proof systems studies. We have established some relations between major complexity classes and variants of interactive proof systems. We have shown that there was a potential for unlimited variants of Interactive proof systems, and at least two problems remain of major interest to us: what is the relationship between NP and BQP? Could we break the two chains from sections 5.1 and 5.2 into a single one? Moreover, what can we do with $IP = PSPACE$? Is IP strictly contained in QIP? All those are just a few of the questions that require further study to answer.

References

- [1] Voufo. 2006. “Interactive Proof Systems: $IP = PSPACE$ ”. <http://www.cs.indiana.edu/~lvoufo/IPsys.pdf>
- [2] Voufo. 2006. “Quantum Complexity Classes”. <http://www.cs.indiana.edu/~lvoufo/qcc.pdf>

A Graphical description of an Interactive Proof System

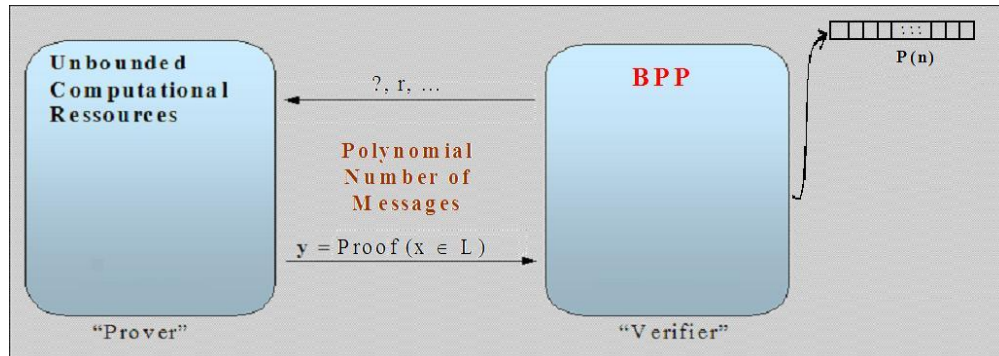


Figure 1: An Interactive Proof System

B Simulating an Interactive Proof System by a PSPACE machine

Let A be a language in IP . On input w with length n , A 's verifier V exchanges exactly $p = \text{poly}(n)$ messages. Let's construct a PSPACE machine M that simulates V . To do this, we define M as follows:

$$\Pr[V \text{ accepts } w] = \max_P \Pr[V \leftrightarrow P \text{ accepts } w]$$

By the definition of IP ,

- $\Pr[V \text{ accepts } w] \geq 1/3$ if $w \in L$.
- $\Pr[V \text{ accepts } w] \leq 1/3$ if $w \notin L$.

Now, we must show that the value can be calculated in PSPACE. Let M_j be the sequence of messages, $m_1 \# m_2 \dots \# m_j$, exchanged by P and V . Let's generalize the interaction of V and P to start with an arbitrary message stream M_j .

- $(V \leftrightarrow P)(w, r, M_j) = \text{accept}$
if M_j can be extended with the messages m_{j+1} through m_p such that:
 1. For $j \leq i < p$, where i is even, $V(w, r, M_j) = m_{i+1}$ ($\Rightarrow V$ sends message)
 2. For $j \leq i < p$, where i is odd, $P(w, r, M_j) = m_{i+1}$ ($\Rightarrow P$ sends message)

3. The final message m_p in the message history is accept (final message is to accept)

((1) and (2) ensure the validity of M_j and (3) ensures that M_j leads to an accept)

Now, let's generalize the earlier definitions further, taking a random string r , of size p , to define:

$$\Pr[V \leftrightarrow P \text{ accepts } w \text{ starting at } M_j] = \Pr[(V \leftrightarrow P)(w, r, M_j) = \text{accept}]$$

Now,

$$\Pr[V \text{ accepts } w \text{ starting at } M_j] = \max_P \Pr[V \leftrightarrow P \text{ accepts } w \text{ starting at } M_j]$$

For every $0 \leq j \leq p$ and every message history M_j , let's define the function N_{M_j} :

$$N_{M_j} = \begin{cases} 0 & \text{if } j = p \text{ and } m_p = \text{reject.} \\ 1 & \text{if } j = p \text{ and } m_p = \text{accept.} \\ \max_{m_{j+1}} N_{M_{j+1}} & \text{if } j < p \text{ and } j \text{ is odd.} \\ \text{wt-avg}_{m_{j+1}} N_{M_{j+1}} & \text{if } j < p \text{ and } j \text{ is even.} \end{cases}$$

Where

$\Pr_r =$ probability taken over r

$$\begin{aligned} \text{wt-avg}_{m_{j+1}} N_{M_{j+1}} &= \sum_{m_{j+1}} (\Pr_r [V(x, r, M_j)]) \\ &= \text{the average of } N_{M_{j+1}}, \text{ weighted by the probability that} \\ &\quad V \text{ sends message } m_{j+1}. \end{aligned}$$

Now, let's take M_0 . We want to show that:

1. N_{M_0} can be computed in polynomial space
2. $N_{M_0} = \Pr[V \text{ accepts } w]$

For (1), To compute N_{M_0} , an algorithm can recursively calculate the values N_{M_j} for every j and M_j . Moreover, since the depth of the recursion is p , only polynomial space is necessary.

For (2), we must show that:

$$\text{For every } 0 \leq j \leq p, \text{ and every } M_j, N_{M_j} = \Pr[V \text{ accepts } w \text{ starting at } M_j]$$

\implies **Proof by induction on j :**

Base case: $j = p$

m_p is either accept or reject.

- If m_p is accept, then $N_{M_j} = 1$, by definition of N_{M_j} , and $\Pr [V \text{ accepts } w \text{ starting at } M_j] = 1$, since the message stream indicates acceptance.
- Similar argument for the case where m_p is reject.

Induction Hypothesis (on $k > 0$): $k = j+1 \leq p$, and $N_{M_{j+1}} = \Pr [V \text{ accepts } w \text{ starting at } M_{j+1}]$.
Prove true at $k - 1$: $N_{M_j} = \Pr [V \text{ accepts } w \text{ starting at } M_j]$

- If j is even, then:
 - m_{j+1} is a message from V to P .
 - By def. of N_{M_j} , $N_{M_j} = \sigma_{m_{j+1}}(\Pr_r [V(x, r, M_j)] N_{M_{j+1}})$.
 - By IH, $N_{M_j} = \sum_{m_{j+1}} (\Pr_r [V(x, r, M_j) = m_{j+1}] \cdot \Pr [V \text{ accepts } w \text{ starting at } M_{j+1}])$.
 - By def., $N_{M_j} = \Pr [V \text{ accepts } w \text{ starting at } M_j]$.
- If j is odd, then:
 - $m_j + 1$ is a message from P to V .
 - By def. of N_{M_j} , $N_{M_j} = \max_{m_{j+1}} N_{M_{j+1}}$.
 - By IH, $N_{M_j} = \max_{m_{j+1}} \cdot \Pr [V \text{ accepts } w \text{ starting at } M_{j+1}]$.
 - $N_{M_j} = \Pr [V \text{ accepts } w \text{ starting at } M_j]$, since:
 1. $\max_{m_{j+1}} \cdot \Pr [V \text{ accepts } w \text{ starting at } M_{j+1}] \leq \Pr [V \text{ accepts } w \text{ starting at } M_j]$, since the prover on the right-hand side could send the message m_{j+1} to maximize the expression on the left-hand side.
 2. $\max_{m_{j+1}} \cdot \Pr [V \text{ accepts } w \text{ starting at } M_{j+1}] \geq \Pr [V \text{ accepts } w \text{ starting at } M_j]$, since the same prover cannot do any better than send that same message.

From the above, we get:

$$\max_{m_{j+1}} \cdot \Pr [V \text{ accepts } w \text{ starting at } M_{j+1}] = \Pr [V \text{ accepts } w \text{ starting at } M_j]$$

Therefore, $IP \subseteq PSPACE$.

C An Interactive Proof System for Deciding the Truth of QBFs

Given the arithmetic form A of a simple QBF B , We want to prove that $A \neq 0 \pmod{p}$ for some polynomially long prime p . For this purpose, there are three important points:

1. P chooses p and sends it to V , along with a written proof of primality, as the first step of the interactive proof, since even an infinitely powerful cheating prover cannot find such a prime if $A = 0$.
2. V randomly chooses p , using the fact that for most p , $A \neq 0 \pmod{p}$ iff B is true.
 - Possibility to prove both membership and non-membership by the same protocol.
 - However, if the chosen p happens to be a divisor of A , even an honest prover may be unable to prove a correct statement. Therefore, this protocol has imperfect completeness.

We've already established, in part II, that the polynomial $q(zi)$ that represents A' can have an exponentially high degree, and that such polynomials cannot be handled by the polynomial time verifier during the interactive proof. However, if B is simple, then the degree of the polynomial $q(zi)$ that describes the functional form of A grows at most linearly with the size of B . Moreover, given a non-simple closed QBF, with a quantifier-free sub-expression that is denoted by B , we can reduce its high degree to 3 (thus representing $q(zi)$ by just four numbers in ZP), by adding sufficiently many dummy variables (as illustrated by the transformation into a simple QBF in part II).

3. The very simple interactive protocol for proving that $A \neq 0 \pmod{p}$.
 - P sends the claimed value a of $A \pmod{p}$ to V , and justifies this claim by considering successively smaller subexpressions of A .
 - At any intermediate stage of the protocol, the current expression A is split into $A1 + A2$ or $A1 \cdot A2$, where $A1$ is a polynomial with fully instantiated variables (whose value $a1$ can be computed by V himself), and $A2$ starts with the leftmost Π or Σ symbol of A .
 - P and V then repeatedly execute the following simplification steps:
 - (a) If $A2$ is *empty*, V stops and accepts the claim iff $a = a1$.
 - (b) If $A1$ is *nonempty*, V replaces A by $A2$, and replaces a by $(a - a1) \pmod{p}$ or $(a/a1) \pmod{p}$ (depending on the operator that connects $A1$ and $A2$).
If V tries to divide a by $a1 = 0 \pmod{p}$, he stops and accepts the claim iff $a = 0 \pmod{p}$.
 - *Otherwise*, P sends the polynomial descriptions $q(zi)$ of A' to V . V checks that $a = (q(0) + q(l)) \pmod{p}$ or $a = (q(0) \cdot q(l)) \pmod{p}$ (depending on the first symbol of $A2$), sends a random $r \in ZP$ to P , replaces A by $(A'(zi = r)) \pmod{p}$, and replaces a by $q(r) \pmod{p}$.

With the final protocol (3), we notice two main things:

1. When B is true and P is honest, V always accepts the proof.

2. When B is false, V accepts the proof with negligible probability.

PROOF :

1. An honest P can always justify his claimed values and polynomials.
2. The proof of this part is similar to the one used by Lund et al. in their permanent proving protocol. A cheating prover who supplied an incorrect value of a must provide an incorrect polynomial $q(z_i)$ to support his claim, since a is checked against $q(0) + q(l)(\text{mod } p)$ or $q(0) \cdot q(l)(\text{mod } p)$. By the interpolation theorem, such an incorrect polynomial of degree t can agree with the correct polynomial on at most t of the p points in ZP . When the value of t is a polynomial and the value of p is exponential in the size of B , there is only a negligible probability that the incorrect q yields a correct value when evaluated at a random point r chosen by V . As a result, a cheating P is forced to provide incorrect values for successively smaller sub-expressions, until he is exposed with overwhelming probability when V evaluates the final sub-expression by himself.

Remark. A single application of this protocol suffices to make the probability of cheating exponentially small, and there is no need to iterate it as in other interactive proofs. However, the protocol seems to be inherently sequential, and it is a major open problem whether it can be executed with a small (e.g., logarithmic) number of rounds. Note that the existence of a constant round protocol for IP would collapse the polynomial hierarchy to its second level, as shown by Boppana et al.