Is QSEP-CIRCUIT QMA-hard?

E. Hanson[1]

[1]Departments of Math and Physics
McGill University

May 2013

Is QSEP-CIRCUIT QMA-hard?

Maybe!

E. Hanson[1]

[1]Departments of Math and Physics
McGill University

May 2013

What are QMA and QSEP-CIRCUIT?

- QMA is a complexity class that is the quantum analogue of NP

What are QMA and QSEP-CIRCUIT?

- QMA is a complexity class that is the quantum analogue of NP
- QSEP-CIRCUIT is the problem of determining whether not a state is separable (not entangled), when given the state as a quantum circuit

Outline

Outline

What is a complexity class?

Example

PSPACE is the set of problems that can be solved by a deterministic Turing machine in an amount of space (memory) polynomial in the input size.

What is a complexity class?

Example

PSPACE is the set of problems that can be solved by a deterministic Turing machine in an amount of space (memory) polynomial in the input size. E.g. $\text{memory}(n) \leq n^{54}$

What is a complexity class?

- A set of problems that can be solved with an amount of resources that scale with the input in a certain way

Example

PSPACE is the set of problems that can be solved by a deterministic Turing machine in an amount of space (memory) polynomial in the input size. E.g. $\text{memory}(n) \leq n^{54}$

Definitions and notation

Definition

- A *string* is a sequence of characters; say, abc. The length of a string x is denoted $|x|$ and is the number of characters in the string ($|\mathrm{abc}| = 3$).

Definitions and notation

Definition

- A *string* is a sequence of characters; say, abc. The length of a string x is denoted $|x|$ and is the number of characters in the string ($|\text{abc}| = 3$).
- A *decision problem* is a problem with a yes/no answer. That is, it is a formal language L: a set of strings that obey certain rules. A string x is in L if x is an input to to the problem with an "yes" answer.

Definitions and notation

Definition

- A *string* is a sequence of characters; say, abc. The length of a string x is denoted $|x|$ and is the number of characters in the string ($|\mathrm{abc}| = 3$).
- A *decision problem* is a problem with a yes/no answer. That is, it is a formal language L: a set of strings that obey certain rules. A string x is in L if x is an input to to the problem with an "yes" answer.
  - The set of valid characters for a string in a language L is called the *alphabet* associated to the L

Definitions and notation

Definition

- A *string* is a sequence of characters; say, abc. The length of a string x is denoted $|x|$ and is the number of characters in the string ($|\text{abc}| = 3$).
- A *decision problem* is a problem with a yes/no answer. That is, it is a formal language L: a set of strings that obey certain rules. A string x is in L if x is an input to to the problem with an "yes" answer.
  - The set of valid characters for a string in a language L is called the *alphabet* associated to the L
- A *polynomial time verifier* is an algorithm that takes in an input x and a string y, and outputs "yes" or "no" in time polynomial in $|x|$

Decision problem example

Example

- Say I want to consider the decision problem: Given an integer $n \in \mathbb{Z}$, decide either:
  - YES: $n \geq 5$
  - NO: $n < 5$

Decision problem example

Example

- Say I want to consider the decision problem: Given an integer $n \in \mathbb{Z}$, decide either:
  - YES: $n \geq 5$
  - NO: $n < 5$
- Let's call the formal version of this decision problem the formal language L. We'll say the alphabet associated to L is $\Sigma = \{n, =\} \cup \mathbb{Z}$.

Decision problem example

Example

- Say I want to consider the decision problem: Given an integer $n \in \mathbb{Z}$, decide either:
  - YES: $n \geq 5$
  - NO: $n < 5$
- Let's call the formal version of this decision problem the formal language L. We'll say the alphabet associated to L is $\Sigma = \{n, =\} \cup \mathbb{Z}$.
- Furthermore, we'll require that every string where the first character is n, the second character is $=$, and the third character is any integer $z \in \mathbb{Z}$ greater than or equal to 5 is in the language L, and no other strings are in L.

Decision problem example

Example

- Say I want to consider the decision problem: Given an integer $n \in \mathbb{Z}$, decide either:
  - YES: $n \geq 5$
  - NO: $n < 5$
- Let's call the formal version of this decision problem the formal language L. We'll say the alphabet associated to L is $\Sigma = \{n, =\} \cup \mathbb{Z}$.
- Furthermore, we'll require that every string where the first character is n, the second character is $=$, and the third character is any integer $z \in \mathbb{Z}$ greater than or equal to 5 is in the language L, and no other strings are in L.
  - So "$n = 6$" $\in L$, but "abc" $\notin L$, "$n = 2$" $\notin L$, and "$= n6$" $\notin L$

NP

- NP is the set of decision problems that can be solved in polynomial time by a non-deterministic Turing machine

NP

- NP is the set of decision problems that can be solved in polynomial time by a non-deterministic Turing machine
- Or, NP is the set of decision problems whose "yes" instances have solutions that can be checked in polynomial time by a deterministic Turing machine

NP

- NP is the set of decision problems that can be solved in polynomial time by a non-deterministic Turing machine
- Or, NP is the set of decision problems whose "yes" instances have solutions that can be checked in polynomial time by a deterministic Turing machine

Example

Factoring: Given a huge number N with $1 \leq M \leq N$, (say, RSA-210: $N = 245246644900278211976517663573088018467026787678332759743414451715061600830038587216952208399332071549103626827191679864079776723243005600592035631246561218465817904100131859299619933817012149335034875870551067$) does N have a factor d with $1 < d < M$? Very difficult to do in general. But given a prospective factor, just divide to check.

NP

- Alternatively, NP is the set of decision problems ("languages", L) ...

NP

- Alternatively, NP is the set of decision problems ("languages", L) where there exists a polynomial time verifier V and polynomial p such that: (...)

NP

- Alternatively, NP is the set of decision problems ("languages", L) where there exists a polynomial time verifier V and polynomial p such that: for instances of the problem x that have "yes" answers ($x \in L$), then there exists a proof y with $|y| \leq p(|x|)$ where $V(x, y) = \text{yes}$. (...)

NP

- Alternatively, NP is the set of decision problems ("languages", L) where there exists a polynomial time verifier V and polynomial p such that: for instances of the problem x that have "yes" answers ($x \in L$), then there exists a proof y with $|y| \leq p(|x|)$ where $V(x, y) = \text{yes}$. For instances x with a "no" answer ($x \notin L$), for any prospective proof y with $|y| \leq p(|x|)$, $V(x, y) = \text{no}$.

MA

- MA is the probabilistic equivalent of NP; it is the set of decision problems whose solutions can be checked in polynomial time by a probabilistic Turing machine

Definition

- A *probabilistic Turing machine* is simply a Turing machine that can use randomness and is allowed to have some (small) probability of error

MA

- MA is the probabilistic equivalent of NP; it is the set of decision problems whose solutions can be checked in polynomial time by a probabilistic Turing machine
- That is, MA is the set of decision problems L where there exists a probabilistic polynomial time verifier V and polynomial p such that: for instances of the problem x that have "yes" answers ($x \in L$), then there exists a proof y with $|y| \leq p(|x|)$ where $V(x, y) =$ yes with probability at least $2/3$. For instances x with a "no" answer ($x \notin L$), for any proof y with $|y| \leq p(|x|)$, $V(x, y) =$ no with probability at least $2/3$.

Definition

- A *probabilistic polynomial time verifier* is a polynomial time verifier that is allowed to use randomness

NP

- Or, NP is the set of decision problems whose "yes" instances have solutions that can be checked in polynomial time by a deterministic Turing machine
- Alternatively, NP is the set of decision problems ("languages", L) where there exists a polynomial time verifier V and polynomial p such that: for instances of the problem x that have "yes" answers ($x \in L$), then there exists a proof y with $|y| \leq p(|x|)$ where $V(x,y) = \text{yes}$. For instances x with a "no" answer ($x \notin L$), for any prospective proof y with $|y| \leq p(|x|)$, $V(x,y) = \text{no}$.

Outline

Reductions

Definition

- A reduction is an algorithm for solving one problem by assuming you can solve a different one

Reductions

- A problem is called "hard" for a complexity class if it can be reduced to any other problem in the class

Definition

- A reduction is an algorithm for solving one problem by assuming you can solve a different one

Reductions

- A problem is called "hard" for a complexity class if it can be reduced to any other problem in the class
- A problem is called complete if it is in the class and hard for the class

Definition

- A reduction is an algorithm for solving one problem by assuming you can solve a different one

Reductions

- A problem is called "hard" for a complexity class if it can be reduced to any other problem in the class
- A problem is called complete if it is in the class and hard for the class
- If problem A can be reduced to problem B, that is, if we can solve A by assuming we can solve B, then we say $A \leq B$

Definition

- A reduction is an algorithm for solving one problem by assuming you can solve a different one

How to prove a problem is hard

- Any problem reduced to a problem already shown to be hard for a class is also hard

How to prove a problem is hard

- Any problem reduced to a problem already shown to be hard for a class is also hard
- To find the first hard problem, usually the definition of a class is recast as an automatically-hard problem

One more definition

Definition

- A promise problem is a problem with only two outcomes ("yes" or "no") like a decision problem, but only some inputs are valid.

One more definition

Definition

- A promise problem is a problem with only two outcomes ("yes" or "no") like a decision problem, but only some inputs are valid. E.g. is n below 5 or above 10?

One more definition

Definition

- A promise problem is a problem with only two outcomes ("yes" or "no") like a decision problem, but only some inputs are valid. E.g. is n below 5 or above 10? If you give it $n = 7$, it can output "YES" or "NO"; either would be correct.

A non-trivial example of a reduction

A non-trivial example of a reduction

- We will consider two promise problems: one called called "$\text{WMEM}_{\delta'}(K)$", the weak membership problem with two-sided error, and one called "$\text{WMEM}^1_{\delta}(K)$", the weak membership problem with one-sided error.

A non-trivial example of a reduction

- We will consider two promise problems: one called called "$\mathrm{WMEM}_{\delta'}(K)$", the weak membership problem with two-sided error, and one called "$\mathrm{WMEM}^1_\delta(K)$", the weak membership problem with one-sided error.
- The idea is that if we have certain type of set of points called a convex set, and we know if a point we ask about, y, is either deep inside the set, or far away from the set, we can make an algorithm that does even better: our algorithm will tell us if a point is inside the set (even if it's close to the boundary), or far from the set.

A non-trivial example of a reduction

- We will consider two promise problems: one called called "$\text{WMEM}_{\delta'}(K)$", the weak membership problem with two-sided error, and one called "$\text{WMEM}^1_{\delta}(K)$", the weak membership problem with one-sided error.
- The idea is that if we have certain type of set of points called a convex set, and we know if a point we ask about, y, is either deep inside the set, or far away from the set, we can make an algorithm that does even better: our algorithm will tell us if a point is inside the set (even if it's close to the boundary), or far from the set.
- In this way, we've reduced the weak membership problem with one-sided error to the weak membership problem with two-sided error – in effect, we've shown that getting rid of half the error doesn't make the problem any easier!

Defining our terms

Definition

- A convex set $K \subseteq R^n$ is a set of points such that for any two points $x, y \in K$, the line segment connecting x and y is also in K.

Defining our terms

Definition

- A convex set $K \subseteq R^n$ is a set of points such that for any two points $x, y \in K$, the line segment connecting x and y is also in K.
  - That is, for all $\lambda \in [0,1]$, if $x, y \in K$, then $(1-\lambda)x + \lambda y \in K$

Defining our terms

Definition

- A convex set $K \subseteq R^n$ is a set of points such that for any two points $x, y \in K$, the line segment connecting x and y is also in K.
  - That is, for all $\lambda \in [0,1]$, if $x, y \in K$, then $(1-\lambda)x + \lambda y \in K$
- The ball centered at point $p \in \mathbb{R}^n$ of radius r is $B(p,r) = \{x \in \mathbb{R}^n : \|x - p\| \leq r\}$

Defining our terms

Definition

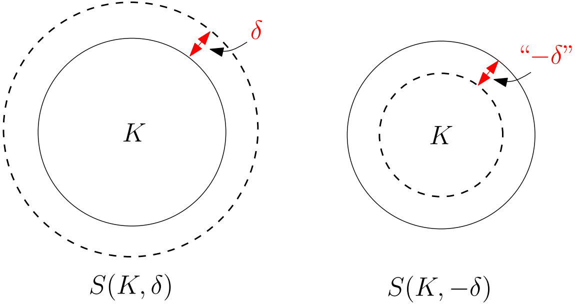
- A convex set $K \subseteq R^n$ is a set of points such that for any two points $x, y \in K$, the line segment connecting x and y is also in K.
  - That is, for all $\lambda \in [0,1]$, if $x, y \in K$, then $(1-\lambda)x + \lambda y \in K$
- The ball centered at point $p \in \mathbb{R}^n$ of radius r is $B(p, r) = \{x \in \mathbb{R}^n : \|x - p\| \leq r\}$
- The ball of radius ε around K is $S(K, \varepsilon) = \{x \in \mathbb{R}^n : \exists\, y \in K \text{ such that } \|y - x\| \leq \varepsilon\}$

Defining our terms

Definition

- A convex set $K \subseteq R^n$ is a set of points such that for any two points $x, y \in K$, the line segment connecting x and y is also in K.
  - That is, for all $\lambda \in [0, 1]$, if $x, y \in K$, then $(1-\lambda)x + \lambda y \in K$
- The ball centered at point $p \in \mathbb{R}^n$ of radius r is $B(p, r) = \{x \in \mathbb{R}^n : \|x - p\| \leq r\}$
- The ball of radius ε around K is $S(K, \varepsilon) = \{x \in \mathbb{R}^n : \exists\, y \in K \text{ such that } \|y - x\| \leq \varepsilon\}$
- The interior of K with radius ε is $S(K, -\varepsilon) = \{x \in \mathbb{R}^n : B(x, \varepsilon) \subseteq K\}$

What do these sets look like?



The Problems

- First, we're going to put a two more constraints on our convex set K:

The Problems

- First, we're going to put a two more constraints on our convex set K:
  - We require there exists a $R > 0$ such that $K \subseteq B(0, R)$

The Problems

- First, we're going to put a two more constraints on our convex set K:
  - We require there exists a $R > 0$ such that $K \subseteq B(0, R)$
  - There exists $p \in \mathbb{R}^n$ and a $r > 0$ such that $B(p, r) \subseteq K$

The Problems

- First, we're going to put a two more constraints on our convex set K:
  - We require there exists a $R > 0$ such that $K \subseteq B(0, R)$
  - There exists $p \in \mathbb{R}^n$ and a $r > 0$ such that $B(p, r) \subseteq K$
- The weak membership problem with 2-sided error, $WMEM_\delta(K)$: For (K, n, p, R, r) as before, given $y \in \mathbb{R}^n$, $\delta > 0$, decide:

The Problems

- First, we're going to put a two more constraints on our convex set K:
  - We require there exists a $R > 0$ such that $K \subseteq B(0, R)$
  - There exists $p \in \mathbb{R}^n$ and a $r > 0$ such that $B(p, r) \subseteq K$
- The weak membership problem with 2-sided error, $WMEM_\delta(K)$: For (K, n, p, R, r) as before, given $y \in \mathbb{R}^n$, $\delta > 0$, decide:
  - YES: $y \in S(K, -\delta)$

The Problems

- First, we're going to put a two more constraints on our convex set K:
  - We require there exists a $R > 0$ such that $K \subseteq B(0, R)$
  - There exists $p \in \mathbb{R}^n$ and a $r > 0$ such that $B(p, r) \subseteq K$

- The weak membership problem with 2-sided error, $WMEM_\delta(K)$: For (K, n, p, R, r) as before, given $y \in \mathbb{R}^n$, $\delta > 0$, decide:
  - YES: $y \in S(K, -\delta)$
  - NO: $y \notin S(K, \delta)$

The Problems

- First, we're going to put a two more constraints on our convex set K:
  - We require there exists a $R > 0$ such that $K \subseteq B(0, R)$
  - There exists $p \in \mathbb{R}^n$ and a $r > 0$ such that $B(p, r) \subseteq K$
- The weak membership problem with 2-sided error, $WMEM_\delta(K)$: For (K, n, p, R, r) as before, given $y \in \mathbb{R}^n$, $\delta > 0$, decide:
  - YES: $y \in S(K, -\delta)$
  - NO: $y \notin S(K, \delta)$
- The weak membership problem with 1-sided error, $WMEM^1_{\delta'}(K)$: For (K, n, p, R, r) as before, given $y \in \mathbb{R}^n$, $\delta' > 0$, decide:

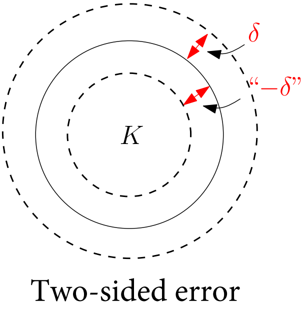
The Problems

- First, we're going to put a two more constraints on our convex set K:
  - We require there exists a $R > 0$ such that $K \subseteq B(0, R)$
  - There exists $p \in \mathbb{R}^n$ and a $r > 0$ such that $B(p, r) \subseteq K$
- The weak membership problem with 2-sided error, $WMEM_\delta(K)$: For (K, n, p, R, r) as before, given $y \in \mathbb{R}^n$, $\delta > 0$, decide:
  - YES: $y \in S(K, -\delta)$
  - NO: $y \notin S(K, \delta)$
- The weak membership problem with 1-sided error, $WMEM^1_{\delta'}(K)$: For (K, n, p, R, r) as before, given $y \in \mathbb{R}^n$, $\delta' > 0$, decide:
  - YES: $y \in K$

The Problems

- First, we're going to put a two more constraints on our convex set K:
  - We require there exists a $R > 0$ such that $K \subseteq B(0, R)$
  - There exists $p \in \mathbb{R}^n$ and a $r > 0$ such that $B(p, r) \subseteq K$
- The weak membership problem with 2-sided error, $WMEM_{\delta}(K)$: For (K, n, p, R, r) as before, given $y \in \mathbb{R}^n$, $\delta > 0$, decide:
  - YES: $y \in S(K, -\delta)$
  - NO: $y \notin S(K, \delta)$
- The weak membership problem with 1-sided error, $WMEM^1_{\delta'}(K)$: For (K, n, p, R, r) as before, given $y \in \mathbb{R}^n$, $\delta' > 0$, decide:
  - YES: $y \in K$
  - NO: $y \notin S(K, \delta)$

What do these problems look like?



The Algorithm

- Goal: Solve WMEM with 1-sided error $\delta > 0$ by assuming we can solve WMEM with 2-sided error $\delta' = \frac{r}{4R}\delta$.

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$

The Algorithm

- Goal: Solve WMEM with 1-sided error $\delta > 0$ by assuming we can solve WMEM with 2-sided error $\delta' = \frac{r}{4R}\delta$.
- If $\|y - p\| \geq 2R$, answer "NO"

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$

The Algorithm

- Goal: Solve WMEM with 1-sided error $\delta > 0$ by assuming we can solve WMEM with 2-sided error $\delta' = \frac{r}{4R}\delta$.
- If $\|y - p\| \geq 2R$, answer "NO"
  - In that case, $2R \leq \|y - p\| \leq \|y\| + \|p\| \leq \|y\| + R$, so $R \leq \|y\|$

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$

The Algorithm

- Goal: Solve WMEM with 1-sided error $\delta > 0$ by assuming we can solve WMEM with 2-sided error $\delta' = \frac{r}{4R}\delta$.
- If $\|y - p\| \geq 2R$, answer "NO"
  - In that case, $2R \leq \|y - p\| \leq \|y\| + \|p\| \leq \|y\| + R$, so $R \leq \|y\|$
- Otherwise, solve $\mathrm{WMEM}(K)_{\delta'}$ for the point $y' = \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right) p$, and output the answer.

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$

Case: $WMEM_{\delta'}$ outputs "YES" for y'

- In this case, we know $y' \in S(K, \delta')$. (Either could have output "YES" because $y' \in S(K, -\delta')$, or because y' is not a valid input and is too close to the border).

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$
- $y' = \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right) p$

Case: $WMEM_{\delta'}$ outputs "YES" for y'

- In this case, we know $y' \in S(K, \delta')$. (Either could have output "YES" because $y' \in S(K, -\delta')$, or because y' is not a valid input and is too close to the border).
- Then $\exists\, x \in K$ with $\|y' - x\| \le \delta'$ by definition of $S(K, \delta')$

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$
- $y' = \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right) p$

Case: $WMEM_{\delta'}$ outputs "YES" for y'

- In this case, we know $y' \in S(K, \delta')$. (Either could have output "YES" because $y' \in S(K, -\delta')$, or because y' is not a valid input and is too close to the border).
- Then $\exists\, x \in K$ with $\|y' - x\| \leq \delta'$ by definition of $S(K, \delta')$
- Also, $\|y - y'\| = \frac{\delta}{4R}\|y - p\| < \delta/2$ ($\|y - p\| < 2R$, otherwise would have output "NO" earlier)

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$
- $y' = \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right) p$

Case: $WMEM_{\delta'}$ outputs "YES" for y'

- In this case, we know $y' \in S(K, \delta')$. (Either could have output "YES" because $y' \in S(K, -\delta')$, or because y' is not a valid input and is too close to the border).
- Then $\exists\, x \in K$ with $\|y' - x\| \leq \delta'$ by definition of $S(K, \delta')$
- Also, $\|y - y'\| = \frac{\delta}{4R}\|y - p\| < \delta/2$ ($\|y - p\| < 2R$, otherwise would have output "NO" earlier)
- $\|y - x\| = \|(y' - x) + (y - y')\| \leq \|y' - x\| + \|y - y'\| < \delta' + \delta/2 < \delta/4 + \delta/2 < \delta$

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$
- $y' = \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right) p$

Case: $WMEM_{\delta'}$ outputs "YES" for y'

- In this case, we know $y' \in S(K, \delta')$. (Either could have output "YES" because $y' \in S(K, -\delta')$, or because y' is not a valid input and is too close to the border).
- Then $\exists x \in K$ with $\|y' - x\| \leq \delta'$ by definition of $S(K, \delta')$
- Also, $\|y - y'\| = \frac{\delta}{4R}\|y - p\| < \delta/2$ ($\|y - p\| < 2R$, otherwise would have output "NO" earlier)
- $\|y - x\| = \|(y' - x) + (y - y')\| \leq \|y' - x\| + \|y - y'\| < \delta' + \delta/2 < \delta/4 + \delta/2 < \delta$
- So $y \in S(K, \delta)$, and we were correct to output "YES"

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$
- $y' = \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right) p$

Case: $WMEM_{\delta'}$ outputs "NO" for y'

- Then we know that $y \notin (K, -\delta)$. We want to show then that $y \notin K$, so assume $y \in K$ to try to reach a contradiction

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$
- $y' = \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right) p$

Case: $WMEM_{\delta'}$ outputs "NO" for y'

- Then we know that $y \notin (K, -\delta)$. We want to show then that $y \notin K$, so assume $y \in K$ to try to reach a contradiction
- We know for any $d \in \mathbb{R}^n$ with $\|d\| < r$ that $p + d \in K$ (because $B(p, r) \subseteq K$)

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$
- $y' = \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right) p$

Case: $WMEM_{\delta'}$ outputs "NO" for y'

- Then we know that $y \notin (K, -\delta)$. We want to show then that $y \notin K$, so assume $y \in K$ to try to reach a contradiction
- We know for any $d \in \mathbb{R}^n$ with $\|d\| < r$ that $p + d \in K$ (because $B(p, r) \subseteq K$)
- So $\alpha := \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right) (p + d) \in K$, because this point lies on the line between y and $p + d$, and K is convex

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$
- $y' = \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right) p$

Case: $WMEM_{\delta'}$ outputs "NO" for y'

- Then we know that $y \notin (K, -\delta)$. We want to show then that $y \notin K$, so assume $y \in K$ to try to reach a contradiction
- We know for any $d \in \mathbb{R}^n$ with $\|d\| < r$ that $p + d \in K$ (because $B(p, r) \subseteq K$)
- So $\alpha := \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right)(p + d) \in K$, because this point lies on the line between y and $p + d$, and K is convex
- $\alpha = y' + \frac{\delta}{4R} d \in K$, so $B(y', \frac{r\delta}{4R}) \subseteq K$, i.e. $y \in S(K, -\delta')$

Reminder

- $\delta' = \frac{r\delta}{4R}$
- $B(p, r) \subseteq K \subseteq B(0, R) \subseteq \mathbb{R}^n$
- $y' = \left(1 - \frac{\delta}{4R}\right) y + \left(\frac{\delta}{4R}\right) p$

What have we done and why?

- We've shown that weak membership with 1-sided error is just as hard as weak membership with 2-sided error by a reduction, by creating an algorithm that assumes we can solve weak membership with 2-sided error in order to solve weak membership with 1-sided error

What have we done and why?

- We've shown that weak membership with 1-sided error is just as hard as weak membership with 2-sided error by a reduction, by creating an algorithm that assumes we can solve weak membership with 2-sided error in order to solve weak membership with 1-sided error
- The set of separable states is convex, so this is actually relevant to QSEP-CIRCUIT

Outline

Interactive Proof Systems

- An interactive proof system models computation as the exchange of messages between a “prover” and “verifier”.

Interactive Proof Systems

- An interactive proof system models computation as the exchange of messages between a “prover” and “verifier”.
- The prover has unlimited computational power but cannot be trusted

Interactive Proof Systems

- An interactive proof system models computation as the exchange of messages between a "prover" and "verifier".
- The prover has unlimited computational power but cannot be trusted
  - so the prover always wants to convince the verifier that the input is a "yes" instance of the problem

Interactive Proof Systems

- An interactive proof system models computation as the exchange of messages between a "prover" and "verifier".
- The prover has unlimited computational power but cannot be trusted
  - so the prover always wants to convince the verifier that the input is a "yes" instance of the problem
- The verifier only has bounded computational power

Interactive Proof Systems

- An interactive proof system models computation as the exchange of messages between a "prover" and "verifier".
- The prover has unlimited computational power but cannot be trusted
  - so the prover always wants to convince the verifier that the input is a "yes" instance of the problem
- The verifier only has bounded computational power
- The verifier and prover exchange messages

Interactive Proof Systems

- An interactive proof system models computation as the exchange of messages between a "prover" and "verifier".
- The prover has unlimited computational power but cannot be trusted
  - so the prover always wants to convince the verifier that the input is a "yes" instance of the problem
- The verifier only has bounded computational power
- The verifier and prover exchange messages
- Require completeness: if the statement is true, the verifier will be convinced by the prover

Interactive Proof Systems

- An interactive proof system models computation as the exchange of messages between a "prover" and "verifier".
- The prover has unlimited computational power but cannot be trusted
  - so the prover always wants to convince the verifier that the input is a "yes" instance of the problem
- The verifier only has bounded computational power
- The verifier and prover exchange messages
- Require completeness: if the statement is true, the verifier will be convinced by the prover
- and completeness: if the statement is false, the prover will only have a small probability of convincing the verifier

NP and MA (again)

NP and MA (again)

- NP is the set of problems that can be solved by a computationally unbounded prover sending a single polynomial-length message to a polynomial-time deterministic verifier such that if the prover is given a "yes" instance, he can always send a message that is accepted by the verifier as a correct proof, and if he is given a "no" instance, any message the prover sends will be rejected.

NP and MA (again)

- NP is the set of problems that can be solved by a computationally unbounded prover sending a single polynomial-length message to a polynomial-time deterministic verifier such that if the prover is given a "yes" instance, he can always send a message that is accepted by the verifier as a correct proof, and if he is given a "no" instance, any message the prover sends will be rejected.
- MA is the set of problems that can be solved by a computationally unbounded prover ("Merlin") sending a single polynomial-length message to a polynomial-time probabilistic verifier ("Arthur") such that if the prover is given a "yes" instance, he can always send a message that is accepted by the verifier as a correct proof with probability $\geq 2/3$, and if he is given a "no" instance, any message the prover sends will be rejected with probability $\geq 2/3$.

Outline

Postulate 1: Quantum States

- To every isolated quantum system (say, a single particle, or the universe) there is an associated complex inner product space (a "Hilbert space") called the state space. A quantum state is a vector in the state space.

Postulate 1: Quantum States

- To every isolated quantum system (say, a single particle, or the universe) there is an associated complex inner product space (a "Hilbert space") called the state space. A quantum state is a vector in the state space.
- We only need to consider finite dimensional spaces, in which case the space is isomorphic to $\mathbb{C}^n$. This is just the set of vectors of length n that have complex numbers as entries with which we can perform vector addition, scalar multiplication, and a generalization of dot product.

Postulate 1: Quantum States

- To every isolated quantum system (say, a single particle, or the universe) there is an associated complex inner product space (a "Hilbert space") called the state space. A quantum state is a vector in the state space.
- We only need to consider finite dimensional spaces, in which case the space is isomorphic to $\mathbb{C}^n$. This is just the set of vectors of length n that have complex numbers as entries with which we can perform vector addition, scalar multiplication, and a generalization of dot product.
- We ignore "global phases" - multiplying a state by $e^{i\theta}$ doesn't make a new state

Postulate 1: Quantum States

- To every isolated quantum system (say, a single particle, or the universe) there is an associated complex inner product space (a "Hilbert space") called the state space. A quantum state is a vector in the state space.
- We only need to consider finite dimensional spaces, in which case the space is isomorphic to $\mathbb{C}^n$. This is just the set of vectors of length n that have complex numbers as entries with which we can perform vector addition, scalar multiplication, and a generalization of dot product.
- We ignore "global phases" - multiplying a state by $e^{i\theta}$ doesn't make a new state
- We don't care about the vector's length, so we make the convention that states are unit vectors (length 1)

More notation!

Definition

- A quantum proof is a quantum state $|y\rangle$

More notation!

Definition

- A quantum proof is a quantum state $|y\rangle$
- The size of a quantum proof $|y\rangle$, denoted $||y\rangle|$ is the number of qubits: log base 2 of the dimension of the vector

More notation!

Definition

- A quantum proof is a quantum state $|y\rangle$
- The size of a quantum proof $|y\rangle$, denoted $||y\rangle|$ is the number of qubits: log base 2 of the dimension of the vector
- A *quantum polynomial time verifier* is an algorithm that in an input state $|x\rangle$ and a quantum proof, a state $|y\rangle$, is allowed to use randomness, and outputs "yes" or "no" in time polynomial in $||x\rangle|$.

Outline

QMA

- QMA is just the quantum analogue of MA: it is the set of promise problems whose solutions can be checked in polynomial time by a quantum, probabilistic Turing machine.

Reminder

- A promise problem is a problem with only two outcomes ("yes" or "no") like a decision problem, but only some inputs are valid. E.g. is n below 5 or above 10?

QMA

- QMA is just the quantum analogue of MA: it is the set of promise problems whose solutions can be checked in polynomial time by a quantum, probabilistic Turing machine.
- That is, the set of promise problems L where there exists a quantum polynomial time verifier V and polynomial p such that: for instances of the problem $|x\rangle$ that have "yes" answers ($x \in L$), then there exists a proof $|y\rangle$ with $||y\rangle| \leq p(||x\rangle|)$ where $V(|x\rangle, |y\rangle) =$ yes with probability at least $2/3$. For instances x with a "no" answer ($x \notin L$), for any proof $|y\rangle$ with $||y\rangle| \leq p(|x|)$, $V(|x\rangle, |y\rangle) =$ no with probability at least $2/3$.

Reminder

- A promise problem is a problem with only two outcomes ("yes" or "no") like a decision problem, but only some inputs are valid. E.g. is n below 5 or above 10?

MA

- MA is the probabilistic equivalent of NP; it is the set of decision problems whose solutions can be checked in polynomial time by a probabilistic Turing machine
- That is, MA is the set of decision problems L where there exists a probabilistic polynomial time verifier V and polynomial p such that: for instances of the problem x that have "yes" answers ($x \in L$), then there exists a proof y with $|y| \leq p(|x|)$ where $V(x, y) = \text{yes}$ with probability at least $2/3$. For instances x with a "no" answer ($x \notin L$), for any proof y with $|y| \leq p(|x|)$, $V(x, y) = \text{no}$ with probability at least $2/3$.

QMA and MA (again)

- MA is the set of decision problems that can be solved by a computationally unbounded prover ("Merlin") sending a single polynomial-length message to a polynomial-time probabilistic verifier ("Arthur") such that if the prover is given a "yes" instance, he can always send a message that is accepted by the verifier as a correct proof with probability $\geq 2/3$, and if he is given a "no" instance, any message the prover sends will be rejected with probability $\geq 2/3$.

QMA and MA (again)

- QMA is the set of promise problems that can be solved by a computationally unbounded quantum prover ("Merlin") sending a single polynomial-length message to a polynomial-time quantum verifier ("Arthur") such that if the prover is given a "yes" instance, he can always send a message that is accepted by the verifier as a correct proof with probability $\geq 2/3$, and if he is given a "no" instance, any message the prover sends will be rejected with probability $\geq 2/3$.

Easily generalized: QIP(k)

- QIP(k) is the set of promise problems that can be solved by a computationally unbounded quantum prover exchanging k polynomial-length messages with a polynomial-time quantum verifier such that if the prover is given a "yes" instance, he can always send a message that is accepted by the verifier as a correct proof with probability $\geq 2/3$, and if he is given a "no" instance, any message the prover sends will be rejected with probability $\geq 2/3$.

Easily generalized: QIP(k)

- QIP(k) is the set of promise problems that can be solved by a computationally unbounded quantum prover exchanging k polynomial-length messages with a polynomial-time quantum verifier such that if the prover is given a "yes" instance, he can always send a message that is accepted by the verifier as a correct proof with probability $\geq 2/3$, and if he is given a "no" instance, any message the prover sends will be rejected with probability $\geq 2/3$.
- QIP is the set of promise problems that can be solved by a computationally unbounded quantum prover exchanging a polynomial number of polynomial-length messages with a polynomial-time quantum verifier such that if the prover is given a "yes" instance, he can always send a message that is accepted by the verifier as a correct proof with probability $\geq 2/3$, and if he is given a "no" instance, any message the prover sends will be rejected with probability $\geq 2/3$.

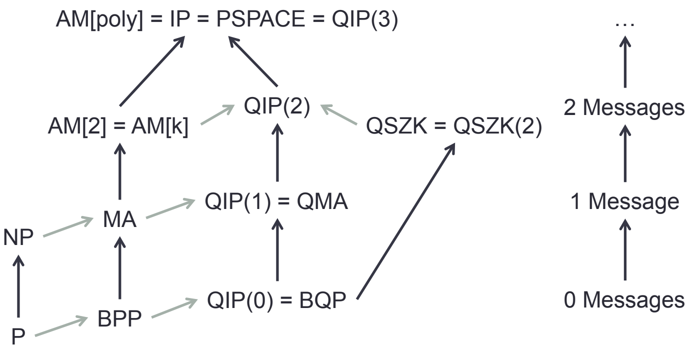
Theorem

PSPACE = QIP = QIP(3)

QSZK

- QSZK stands for quantum statistical zero-knowledge. It is just like QIP, except that for a "yes" instance, the verifier has to be able to generate the proof on his own: that is, the proof must be able to be generated in polynomial time

Hierarchy of Complexity Classes



What do we know about quantum complexity classes?

- We know several QMA-complete problems (unlike the thousands of known NP-complete problems)
- We know very little about QIP(2); the only QIP(2)-complete problem we know of is the trivial rewriting of the definition
- We know a lot about QIP(3) $=$ QIP because it equals PSPACE

Outline

Postulate 2: Time evolution

- Time evolution is unitary. That is, if $|\varphi\rangle$ is the quantum state a system is in, then some time later it will be in a state $U|\varphi\rangle$ for a unitary matrix U.

Postulate 2: Time evolution

- Time evolution is unitary. That is, if $|\varphi\rangle$ is the quantum state a system is in, then some time later it will be in a state $U|\varphi\rangle$ for a unitary matrix U.
- A matrix U is unitary if $UU^\dagger = U^\dagger U = \mathbb{1}$. $\mathbb{1}$ is the identity matrix, and $U^\dagger$ is the complex conjugate transpose of U.

Postulate 2: Time evolution

- Time evolution is unitary. That is, if $|\varphi\rangle$ is the quantum state a system is in, then some time later it will be in a state $U|\varphi\rangle$ for a unitary matrix U.
- A matrix U is unitary if $UU^\dagger = U^\dagger U = \mathbb{1}$. $\mathbb{1}$ is the identity matrix, and $U^\dagger$ is the complex conjugate transpose of U.
- Unitary matrices are those that preserve inner product; in $\mathbb{R}^3$ for example, they are rotations and reflections.

Postulate 3: Measurement

- Measurement is modelled by a collection $\{M_j\}$ of operators on state space with $\sum M_j^\dagger M_j = \mathbb{1}$

Postulate 3: Measurement

- Measurement is modelled by a collection $\{M_j\}$ of operators on state space with $\sum M_j^\dagger M_j = \mathbb{1}$
- The index j labels the different measurement outcomes

Postulate 3: Measurement

- Measurement is modelled by a collection $\{M_j\}$ of operators on state space with $\sum M_j^\dagger M_j = \mathbb{1}$
- The index j labels the different measurement outcomes
- The probability of getting outcome j when the system is in state $|\varphi\rangle$ is $p(j|\varphi) = \langle\varphi|M_j^\dagger M_j|\varphi\rangle$ where $\langle\varphi| := |\varphi\rangle^\dagger$

Postulate 3: Measurement

- Measurement is modelled by a collection $\{M_j\}$ of operators on state space with $\sum M_j^\dagger M_j = \mathbb{1}$
- The index j labels the different measurement outcomes
- The probability of getting outcome j when the system is in state $|\varphi\rangle$ is $p(j|\varphi) = \langle\varphi|M_j^\dagger M_j|\varphi\rangle$ where $\langle\varphi| := |\varphi\rangle^\dagger$
  - This is just the inner product of $M_j|\varphi\rangle$ with itself

Postulate 3: Measurement

- Measurement is modelled by a collection $\{M_j\}$ of operators on state space with $\sum M_j^\dagger M_j = \mathbb{1}$
- The index j labels the different measurement outcomes
- The probability of getting outcome j when the system is in state $|\varphi\rangle$ is $p(j|\varphi) = \langle\varphi|M_j^\dagger M_j|\varphi\rangle$ where $\langle\varphi| := |\varphi\rangle^\dagger$
  - This is just the inner product of $M_j|\varphi\rangle$ with itself
- After measurement, the system is in the state $\dfrac{M_j|\varphi\rangle}{\sqrt{\langle\varphi|M_j^\dagger M_j|\varphi\rangle}}$ (which is just $M_j|\varphi\rangle$ normalized to be a unit vector).

Postulate 4: Composition of Systems

- Classically, it's easy to combine two bits: just concatenate

Postulate 4: Composition of Systems

- Classically, it's easy to combine two bits: just concatenate
- We need something that results in an inner product space: the tensor product

Postulate 4: Composition of Systems

- Classically, it's easy to combine two bits: just concatenate
- We need something that results in an inner product space: the tensor product
- The tensor product of two vectors $|\varphi\rangle$ and $|\psi\rangle$ is denoted $|\varphi\rangle \otimes |\psi\rangle$

Postulate 4: Composition of Systems

- Classically, it's easy to combine two bits: just concatenate
- We need something that results in an inner product space: the tensor product
- The tensor product of two vectors $|\varphi\rangle$ and $|\psi\rangle$ is denoted $|\varphi\rangle \otimes |\psi\rangle$
  - The tensor product is linear in both arguments

Postulate 4: Composition of Systems

- Classically, it's easy to combine two bits: just concatenate
- We need something that results in an inner product space: the tensor product
- The tensor product of two vectors $|\varphi\rangle$ and $|\psi\rangle$ is denoted $|\varphi\rangle \otimes |\psi\rangle$
  - The tensor product is linear in both arguments
- The tensor product of vector spaces V, U is denoted $V \otimes U$ and contains all linear combinations of $|v\rangle \otimes |u\rangle$ for every $|v\rangle \in V$ and $|u\rangle \in U$.

Postulate 4: Composition of Systems

- Classically, it's easy to combine two bits: just concatenate
- We need something that results in an inner product space: the tensor product
- The tensor product of two vectors $|\varphi\rangle$ and $|\psi\rangle$ is denoted $|\varphi\rangle \otimes |\psi\rangle$
  - The tensor product is linear in both arguments
- The tensor product of vector spaces V, U is denoted $V \otimes U$ and contains all linear combinations of $|v\rangle \otimes |u\rangle$ for every $|v\rangle \in V$ and $|u\rangle \in U$.
  - This can contain vectors that cannot be written as $|v\rangle \otimes |u\rangle$ for a $|v\rangle \in V$ and a $|u\rangle \in U$. E.g. $\frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B)$. An entangled state!

Postulate 4: Composition of Systems

- Classically, it's easy to combine two bits: just concatenate
- We need something that results in an inner product space: the tensor product
- The tensor product of two vectors $|\varphi\rangle$ and $|\psi\rangle$ is denoted $|\varphi\rangle \otimes |\psi\rangle$
  - The tensor product is linear in both arguments
- The tensor product of vector spaces V, U is denoted $V \otimes U$ and contains all linear combinations of $|v\rangle \otimes |u\rangle$ for every $|v\rangle \in V$ and $|u\rangle \in U$.
  - This can contain vectors that cannot be written as $|v\rangle \otimes |u\rangle$ for a $|v\rangle \in V$ and a $|u\rangle \in U$. E.g. $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B)$. An entangled state!

Postulate 4: Composition of Systems (continued)

- The state space associated to the composition of two systems is the tensor product of the two individual state spaces

Postulate 4: Composition of Systems (continued)

- The state space associated to the composition of two systems is the tensor product of the two individual state spaces
- Subscripts or superscripts are used to denote which space a state lives in (e.g., $|\varphi\rangle_{AB}, |\varphi\rangle^{AB}$)

Density operators

- We considered quantum states as vectors, but we could equivalently consider them as matrices called density operators. The density operator associated with a state $|\varphi\rangle$ is simply $|\varphi\rangle\langle\varphi|$.

Density operators

- We considered quantum states as vectors, but we could equivalently consider them as matrices called density operators. The density operator associated with a state $|\varphi\rangle$ is simply $|\varphi\rangle\langle\varphi|$.
- A density operator represents a state if it has trace 1 and is positive semi-definite

Density operators

- We considered quantum states as vectors, but we could equivalently consider them as matrices called density operators. The density operator associated with a state $|\varphi\rangle$ is simply $|\varphi\rangle\langle\varphi|$.
- A density operator represents a state if it has trace 1 and is positive semi-definite
- Density operators make it easy when we have uncertainty about a system: $\rho = \sum_i p_i \sigma_i$ is a valid density operator for $p_i \geq 0$, $\sum p_i = 1$, and density operators σ_i. i.e. a probabilistic combination of density operators is a density operator

Density operators

- We considered quantum states as vectors, but we could equivalently consider them as matrices called density operators. The density operator associated with a state $|\varphi\rangle$ is simply $|\varphi\rangle\langle\varphi|$.
- A density operator represents a state if it has trace 1 and is positive semi-definite
- Density operators make it easy when we have uncertainty about a system: $\rho = \sum_i p_i \sigma_i$ is a valid density operator for $p_i \geq 0$, $\sum p_i = 1$, and density operators σ_i. i.e. a probabilistic combination of density operators is a density operator
- A density operator ρ is separable iff it can be written as $\rho = \sum_i p_i \sigma_i \otimes \omega_j$ for $p_i \geq 0$, $\sum p_i = 1$, and density operators σ_i, ω_j. Otherwise it is entangled.

Distance measures

- Trace distance: $\|A\|_1 = \operatorname{tr}(\sqrt{A^\dagger A})$ where $B = \sqrt{C} \iff B^2 = C$

Distance measures

- Trace distance: $\|A\|_1 = \mathrm{tr}\left(\sqrt{A^\dagger A}\right)$ where $B = \sqrt{C} \iff B^2 = C$
- One-way Local operations and classical communication distance: if Alice and Bob wanted to distinguish two states, σ_{AB} and ρ_{AB}, using only operations in their own labs and communicating classically in one direction (e.g. by mail), what success rate would they have above guessing? This is the $1 - LOCC$ distance: $\|\sigma_{AB} - \rho_{AB}\|_{1-LOCC}$

Distance measures

- Trace distance: $\|A\|_1 = \mathrm{tr}\,(\sqrt{A^\dagger A})$ where $B = \sqrt{C} \iff B^2 = C$
- One-way Local operations and classical communication distance: if Alice and Bob wanted to distinguish two states, σ_{AB} and ρ_{AB}, using only operations in their own labs and communicating classically in one direction (e.g. by mail), what success rate would they have above guessing? This is the $1 - LOCC$ distance: $\|\sigma_{AB} - \rho_{AB}\|_{1-LOCC}$
- Fact: $\| * \|_{1-LOCC} \leq \| * \|_1$

Outline

Quantum Gates

- Quantum gates map states to states; because quantum states are unit vectors, gates need to preserve the inner product, and because time evolution is linear, quantum gates should be linear.

Quantum Gates

- Quantum gates map states to states; because quantum states are unit vectors, gates need to preserve the inner product, and because time evolution is linear, quantum gates should be linear.
- Hence, a quantum gate is a $n \times n$ unitary matrix; the composition of unitaries is unitary, so the composition of gates is a gate

Quantum Gates

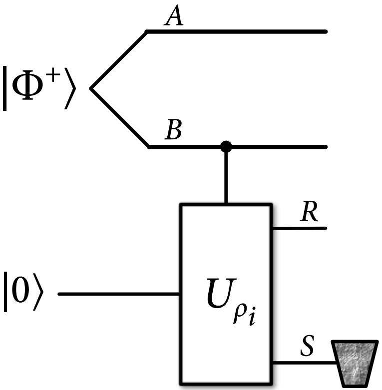
- Quantum gates map states to states; because quantum states are unit vectors, gates need to preserve the inner product, and because time evolution is linear, quantum gates should be linear.
- Hence, a quantum gate is a $n \times n$ unitary matrix; the composition of unitaries is unitary, so the composition of gates is a gate
  - The number of qubits it acts on is $\log_2 n$, so a quantum gate that acts on a single qubit is simply a 2×2 unitary matrix.

Quantum Gates

- Quantum gates map states to states; because quantum states are unit vectors, gates need to preserve the inner product, and because time evolution is linear, quantum gates should be linear.
- Hence, a quantum gate is a $n \times n$ unitary matrix; the composition of unitaries is unitary, so the composition of gates is a gate
  - The number of qubits it acts on is $\log_2 n$, so a quantum gate that acts on a single qubit is simply a 2×2 unitary matrix.
- A quantum circuit is a sequence of quantum gates connected by "wires"

Quantum Circuits

Example



How to throw away one system?

- The partial trace over the B system of a density operator $\rho_{AB} = |a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|$, is $\rho_A = \mathrm{tr}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) = |a_1\rangle\langle a_2| \mathrm{tr}\left(|b_1\rangle\langle b_2|\right)$

How to throw away one system?

- The partial trace over the B system of a density operator $\rho_{AB} = |a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|$, is $\rho_A = \mathrm{tr}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) = |a_1\rangle\langle a_2|\mathrm{tr}\left(|b_1\rangle\langle b_2|\right)$
- Requiring linearity completes the definition

Generating states

- $|0\rangle$ is usually written for the first standard basis vector.
  E.g. for a qubit, $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

Generating states

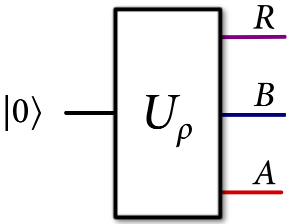
- $|0\rangle$ is usually written for the first standard basis vector. E.g. for a qubit, $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
- Any quantum state can be written as the output of a circuit acting on $|0\rangle$

Generating states

- $|0\rangle$ is usually written for the first standard basis vector. E.g. for a qubit, $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
- Any quantum state can be written as the output of a circuit acting on $|0\rangle$

Example

$|0\rangle \in R \otimes A \otimes B$, want to generate $\rho_{AB} = \mathrm{tr}_R(U_\rho |0\rangle)$



Outline

QSEP-CIRCUIT

- QSEP-CIRCUIT is the problem of determining whether a state ρ_{AB} generated by a circuit is close to a separable state or not.

QSEP-CIRCUIT

- QSEP-CIRCUIT is the problem of determining whether a state ρ_{AB} generated by a circuit is close to a separable state or not.
- Formally, QSEP-CIRCUIT(δ_c,δ_s) is: Given a quantum circuit to generate an n-qubit state ρ_{AB} and a labelling of qubits in the reference system R and the output qubits for A and B, decide whether

$$\min_{\sigma_{AB}\in S} \|\rho_{AB} - \sigma_{AB}\|_1 \leq \delta_c$$

or

$$\min_{\sigma_{AB}\in S} \|\rho_{AB} - \sigma_{AB}\|_{1-LOCC} \geq \delta_s$$

Quantum separability problem

- The problem of deciding if a state is separable or not, given its density matrix, is known to be NP-hard. This is just the weak membership problem!

Quantum separability problem

- The problem of deciding if a state is separable or not, given its density matrix, is known to be NP-hard. This is just the weak membership problem!
- Specifying the state as a circuit has some advantages:

Quantum separability problem

- The problem of deciding if a state is separable or not, given its density matrix, is known to be NP-hard. This is just the weak membership problem!
- Specifying the state as a circuit has some advantages:
  - More physical way to specify a state

Quantum separability problem

- The problem of deciding if a state is separable or not, given its density matrix, is known to be NP-hard. This is just the weak membership problem!
- Specifying the state as a circuit has some advantages:
  - More physical way to specify a state
  - More reasonable scaling: polynomial in the number of qubits the gates act on is exponential in the number of entries of a density matrix

Quantum separability problem

- The problem of deciding if a state is separable or not, given its density matrix, is known to be NP-hard. This is just the weak membership problem!
- Specifying the state as a circuit has some advantages:
  - More physical way to specify a state
  - More reasonable scaling: polynomial in the number of qubits the gates act on is exponential in the number of entries of a density matrix
    - So the quantum separability problem considers states much smaller than those considered by QSEP-CIRCUIT, which is why QSEP-CIRCUIT is a harder problem

NP-hardness of QSEP-CIRCUIT

- Can simply reduce to the quantum separability problem: if we could solve QSEP-CIRCUIT (figure out if big states are separable) then we could solve the quantum separability problem (figure out if small states are separable).

NP-hardness of QSEP-CIRCUIT

- Can simply reduce to the quantum separability problem: if we could solve QSEP-CIRCUIT (figure out if big states are separable) then we could solve the quantum separability problem (figure out if small states are separable).
- The quantum separability problem is already NP-hard, so QSEP-CIRCUIT is NP-hard as well

What do we know about QSEP-CIRCUIT?

- It is in QIP(2)

What do we know about QSEP-CIRCUIT?

- It is in QIP(2)
- It is NP-hard

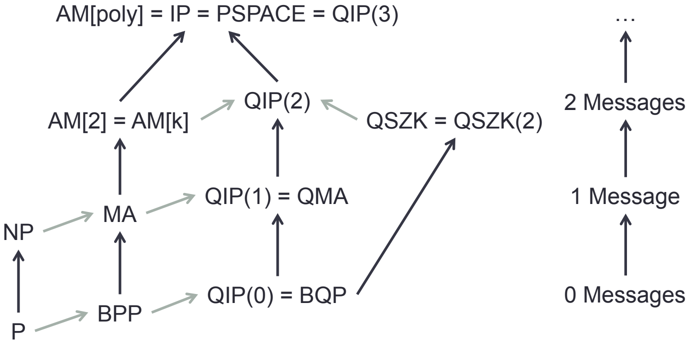
What do we know about QSEP-CIRCUIT?

- It is in QIP(2)
- It is NP-hard
- It is QSZK-hard

What do we know about QSEP-CIRCUIT?

- It is in QIP(2)
- It is NP-hard
- It is QSZK-hard
- A slightly modified problem about channel separability instead of state separability is QIP-complete

Hierarchy of Complexity Classes



So why do we care if QSEP-CIRCUIT is QMA-hard?

- It would strongly suggest QSEP-CIRCUIT is QIP(2)-complete, because we think QSZK and QMA are separate classes and QIP(2) includes both

So why do we care if QSEP-CIRCUIT is QMA-hard?

- It would strongly suggest QSEP-CIRCUIT is QIP(2)-complete, because we think QSZK and QMA are separate classes and QIP(2) includes both
- It seems easier to prove than QIP(2)-hardness: QMA is the analogue of NP, so maybe we can find an analogous proof

Proof that the quantum separability problem is NP-hard:

CLIQUE $\leq_m$ WMQS $\leq_m$ RSDF $\leq_m$ $\mathrm{WOPT}_{\varepsilon}(\mathcal{S}_{M,N}) \leq_T$ $\mathrm{WMEM}^1_{\delta}(\mathcal{S}_{M,N}) \leq_m \mathrm{WMEM}_{\beta}(\mathcal{S}_{M,N})$

Summary

- QMA is a quantum complexity class analogous to NP

Summary

- QMA is a quantum complexity class analogous to NP
- QSEP-CIRCUIT is the problem of determining whether or not a state is separable when given a circuit specifying the state

Summary

- QMA is a quantum complexity class analogous to NP
- QSEP-CIRCUIT is the problem of determining whether or not a state is separable when given a circuit specifying the state
- Quantum information theory is fun!