Title: Implications of computer science principles for quantum physics

Date: Dec 02, 2014 03:30 PM

URL: http://pirsa.org/14120016

Abstract: <p>The Church-Turing thesis is one of the pillars of computer science; it postulates that every classical system has equivalent computability power to the so-called Turing machine. While this thesis is crucial for our understanding of computing devices, its implications in other scientific fields have hardly been explored. What if we consider the Church-Turing thesis as a law of nature? In this talk I will present our first results in connection with quantum information theory [1] by showing that computer science laws have profound implications for some of the most fundamental results of quantum theory. First I will show how they question our knowledge on what a mixed quantum state is, as we identified situations in which ensembles of quantum states defining the same mixed state, indistinguishable according to the quantum postulates, do become distinguishable when prepared by a computer (or any classical system). Then I will introduce a new loophole for Bell-like experiments: if some of the parties in a Bell-like experiment use a computer to decide which measurements to make, then the computational resources of an eavesdropper have to be limited in order to have a proper observation of non-locality.</p>

Implications of computer science principles for quantum physics

Ariel Bendersky[1], Gonzalo de la Torre[1], Antonio Acín[1], Gabriel Senno[2], Santiago Figueira[2]

[1]ICFO - Institut de Ciencies Fotoniques, [2]DC - FCEyN - Universidad de Buenos Aires

December 2014 - Perimeter Institute

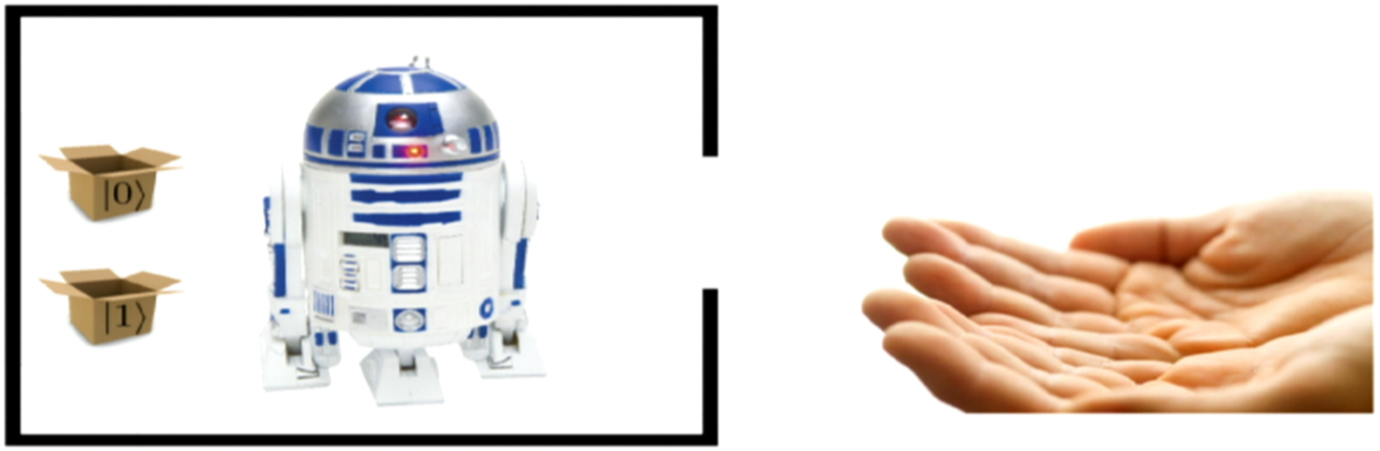
JOHN TEMPLETON FOUNDATION
SUPPORTING SCIENCE - INVESTING IN THE BIG QUESTIONS

ICFO
Institut de Ciències Fotòniques

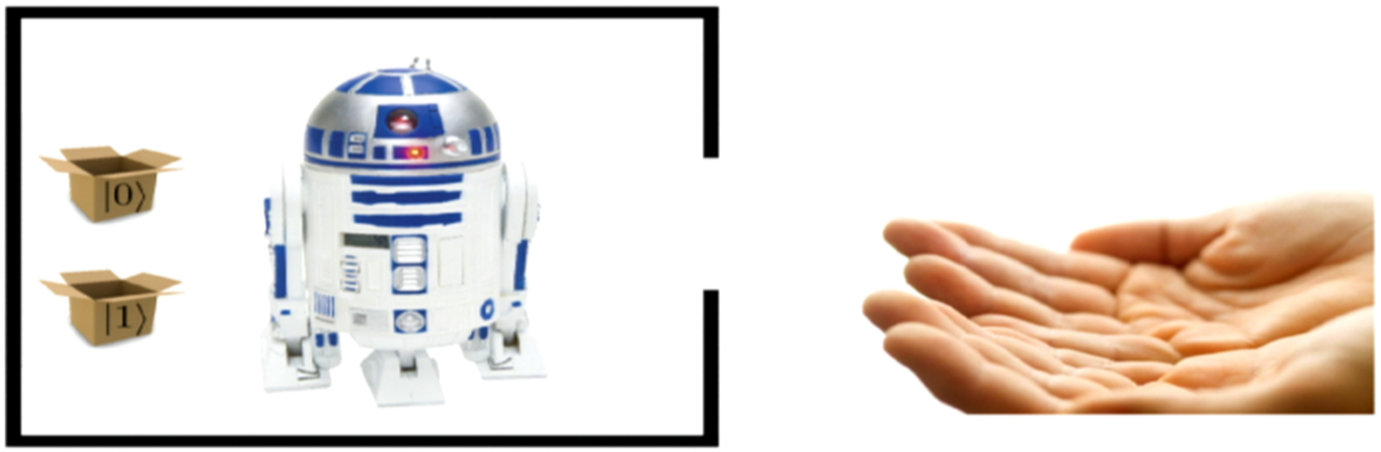
Overview

- Definition of the first problem and its intuitive result.
- Introducing some stuff on computability for physicists.
- Distinguishing the undistinguishable.
- Some generalizations.
- Bell inequality loophole.
- Summary.

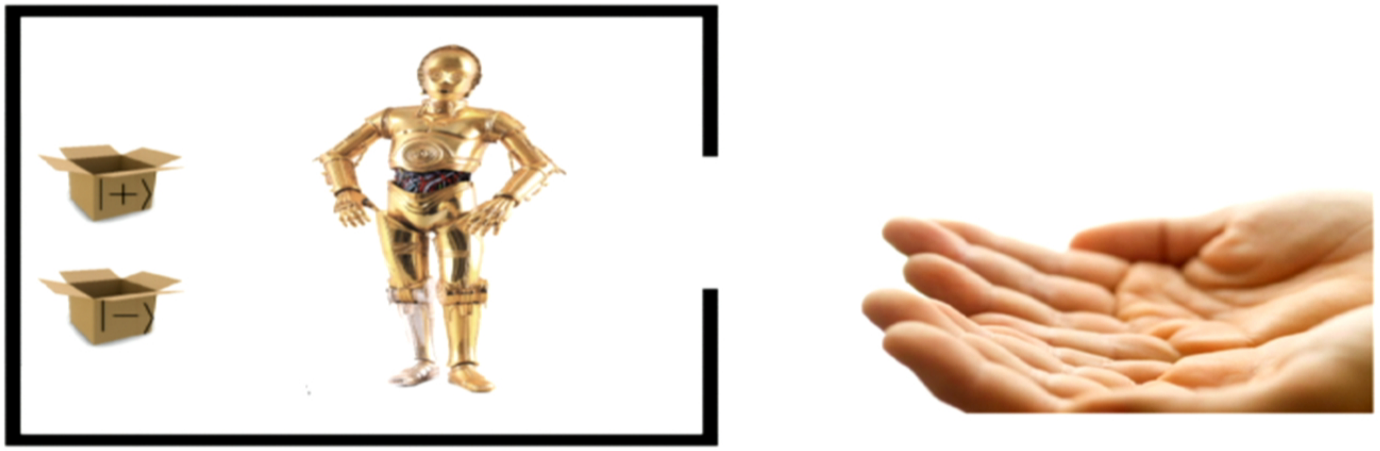
The problem



- R2D2 chooses from each box. The observer only knows that R2D2 will pick half times each state but not how he'll pick each time.
- The state, as described by the observer is $\rho = \frac{\mathbb{I}}{2}$.

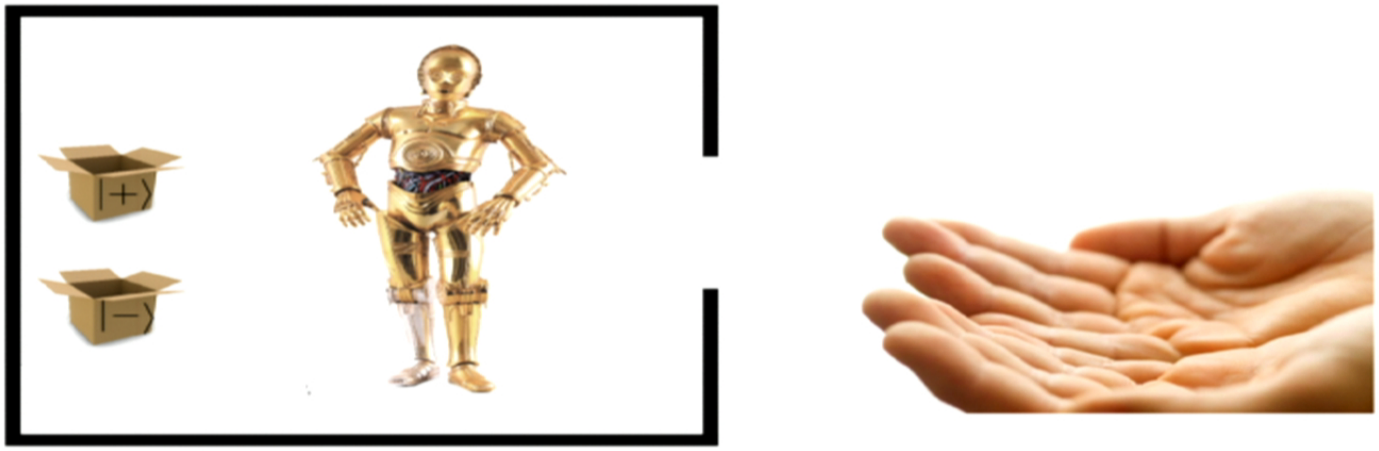


- R2D2 chooses from each box. The observer only knows that R2D2 will pick half times each state but not how he'll pick each time.
- The state, as described by the observer is $\rho = \frac{\mathbb{I}}{2}$.



- C3PO chooses from each box. The observer only knows that C3PO will pick half times each state but not how he'll pick each time.
- The state, as described by the observer is $\rho = \frac{\mathbb{I}}{2}$.

Case 2



- C3PO chooses from each box. The observer only knows that C3PO will pick half times each state but not how he'll pick each time.
- The state, as described by the observer is $\rho = \frac{\mathbb{I}}{2}$.

Undistiguishable

Both situations seem to be undistinguishable.

However...

... they are robots, so they can only choose in a computable manner.
Any classical system used to choose only yields computable choices.

Therefore

We will be able to distinguish both situations.

A little on computability

Turing machines

The Church-Turing thesis

The CT thesis

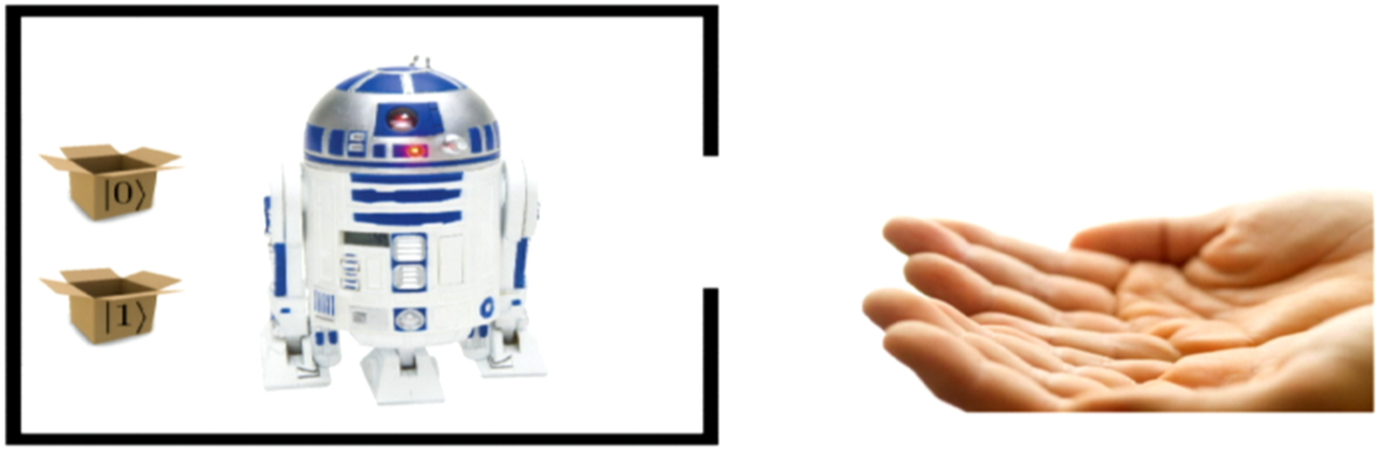
It states that everything that can be computed as a series of steps on a system (effectively calculable) can be computed by a Turing machine. A Turing machine is equivalent to our desktop computers with infinite memory.

The Church-Turing thesis

Properties

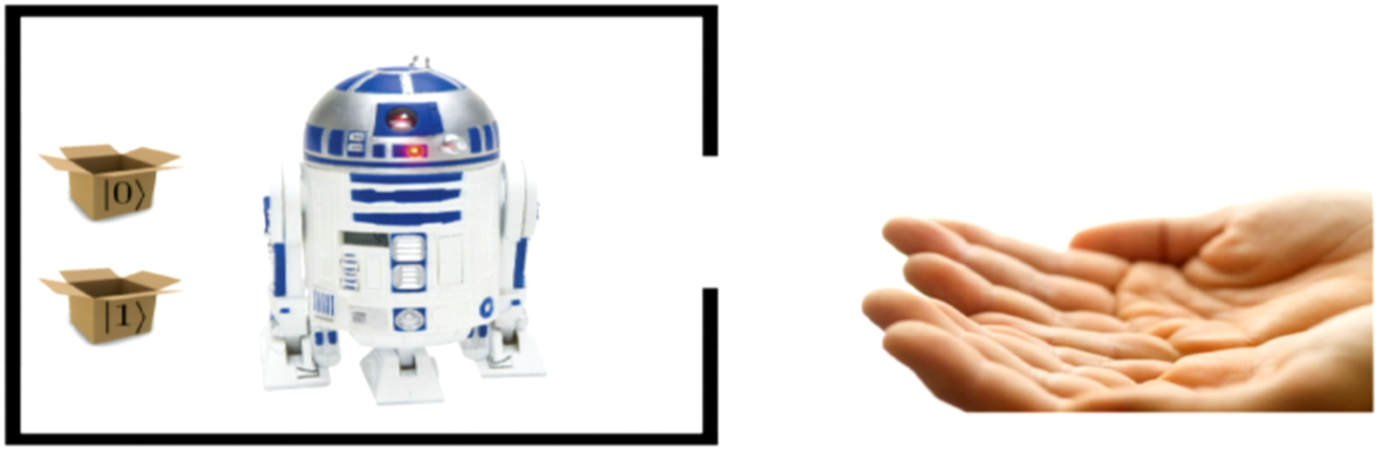
1. Programs (or Turing machines) are enumerable.
2. Some programs don't halt on some inputs, but you can't know (The halting problem is not computable).
3. Most infinite sequences are not computable.
4. There are absolutely normal sequences that are computable.
5. Every finite sequence is computable.
6. Every finite sequence is a prefix for infinitely many computable and infinitely many uncomputable sequences.

Back to our example



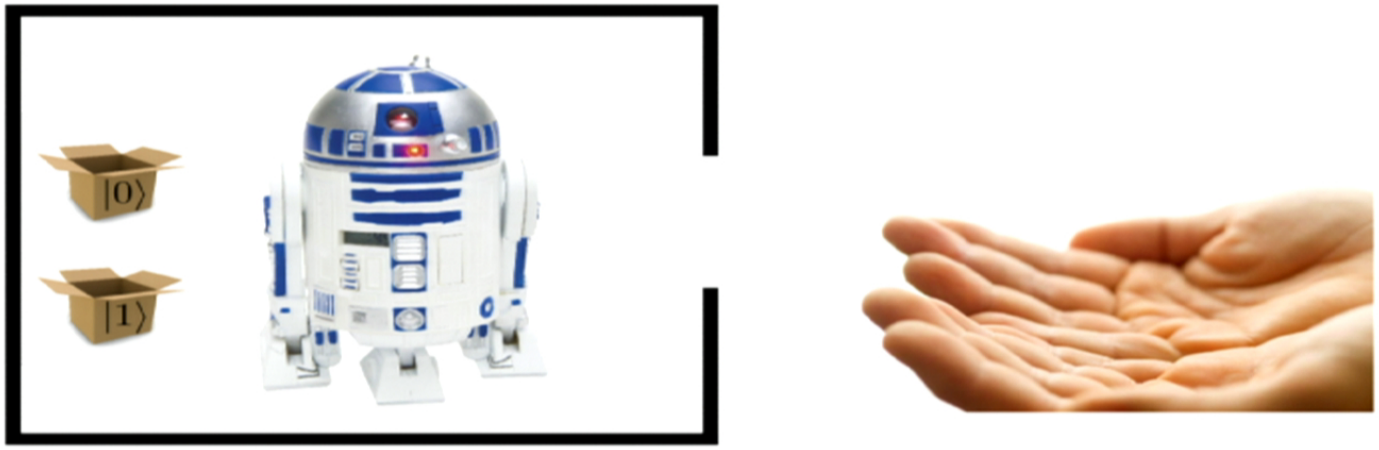
R2D2 and C3PO can only generate computable sequences. Still, the best description for the state (from the observer's perspective) seems to be $\rho = \frac{\mathbb{I}}{2}$. But we can distinguish two apparently undistinguishable situations.

Back to our example



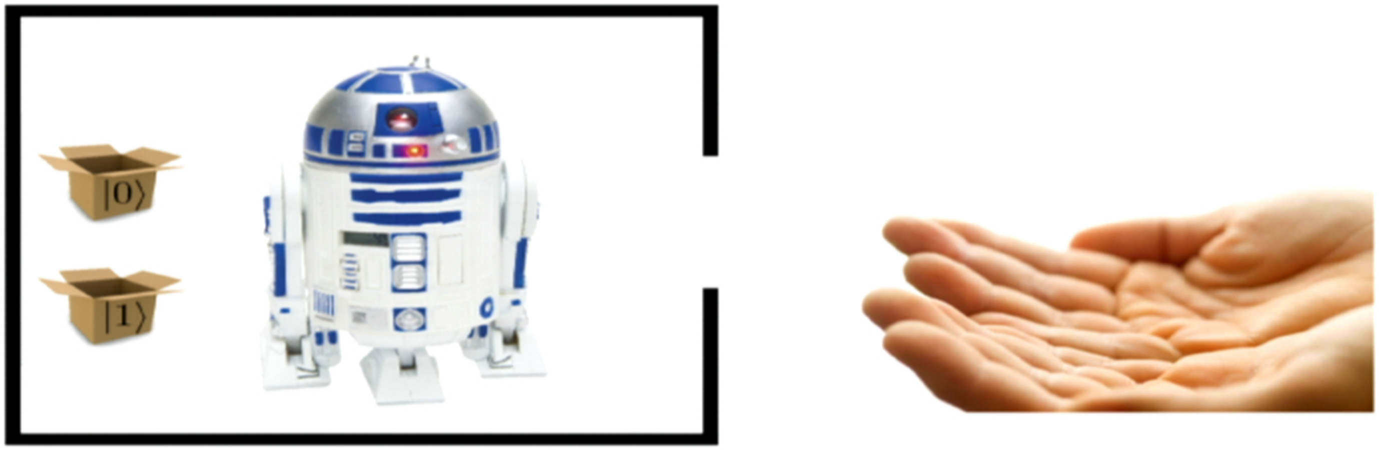
One possible explanation for the distinguishability might be that the fact of having a computable choice changes the state from $\rho = \frac{\mathbb{I}}{2}$ to something else.

Back to our example



However that doesn't seem to be the case. First, no matter what the robot already generated, the next bit can be either a 1 or a 0 and still be computable.

Back to our example



The state description still looks fair.

Distinguishing the undistinguishable

Distinguishing the undistinguishable

Assumption

We have a black box containing one of the two previous situations and we want to know which one of those.

Procedure

We measure every odd qubit on the basis of eigenstates of σ_X and every even qubit on the basis of eigenstates of σ_Z

Distinguishing the undistinguishable

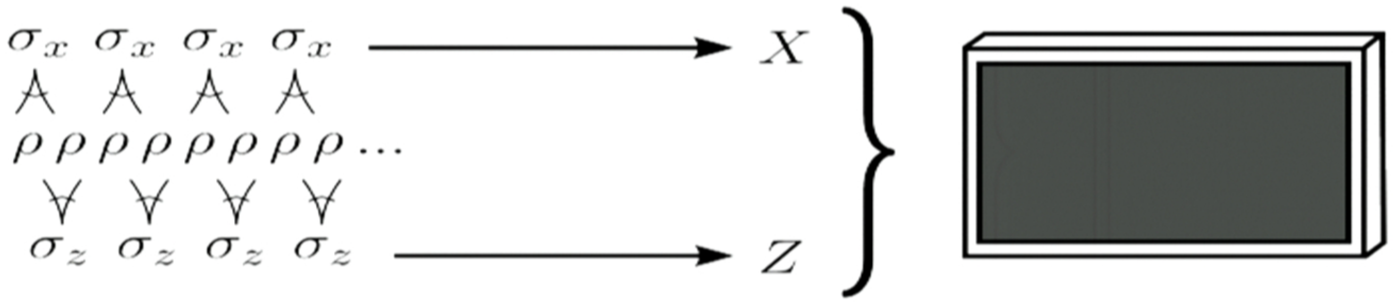
Assumption

We have a black box containing one of the two previous situations and we want to know which one of those.

Procedure

We measure every odd qubit on the basis of eigenstates of σ_X and every even qubit on the basis of eigenstates of σ_Z

Distinguishing the undistinguishable


σ_x σ_x σ_x σ_x
$\rho\,\rho\,\rho\,\rho\,\rho\,\rho\,\rho\,\rho \dots$
σ_z σ_z σ_z σ_z
X
Z

Distinguishing the undistinguishable

Now what?

We obtain two sequences.

- When we measure in the same basis as the preparation, the sequence obtained is computable.
- When we measure in the other basis, the sequence obtained is a fair coin.

Let's go classical

Can we distinguish a random sequence from a computable one?
No!

Distinguishing the undistinguishable

Now what?

We obtain two sequences.

- When we measure in the same basis as the preparation, the sequence obtained is computable.
- When we measure in the other basis, the sequence obtained is a fair coin.

Let's go classical

Can we distinguish a random sequence from a computable one?
No!

However

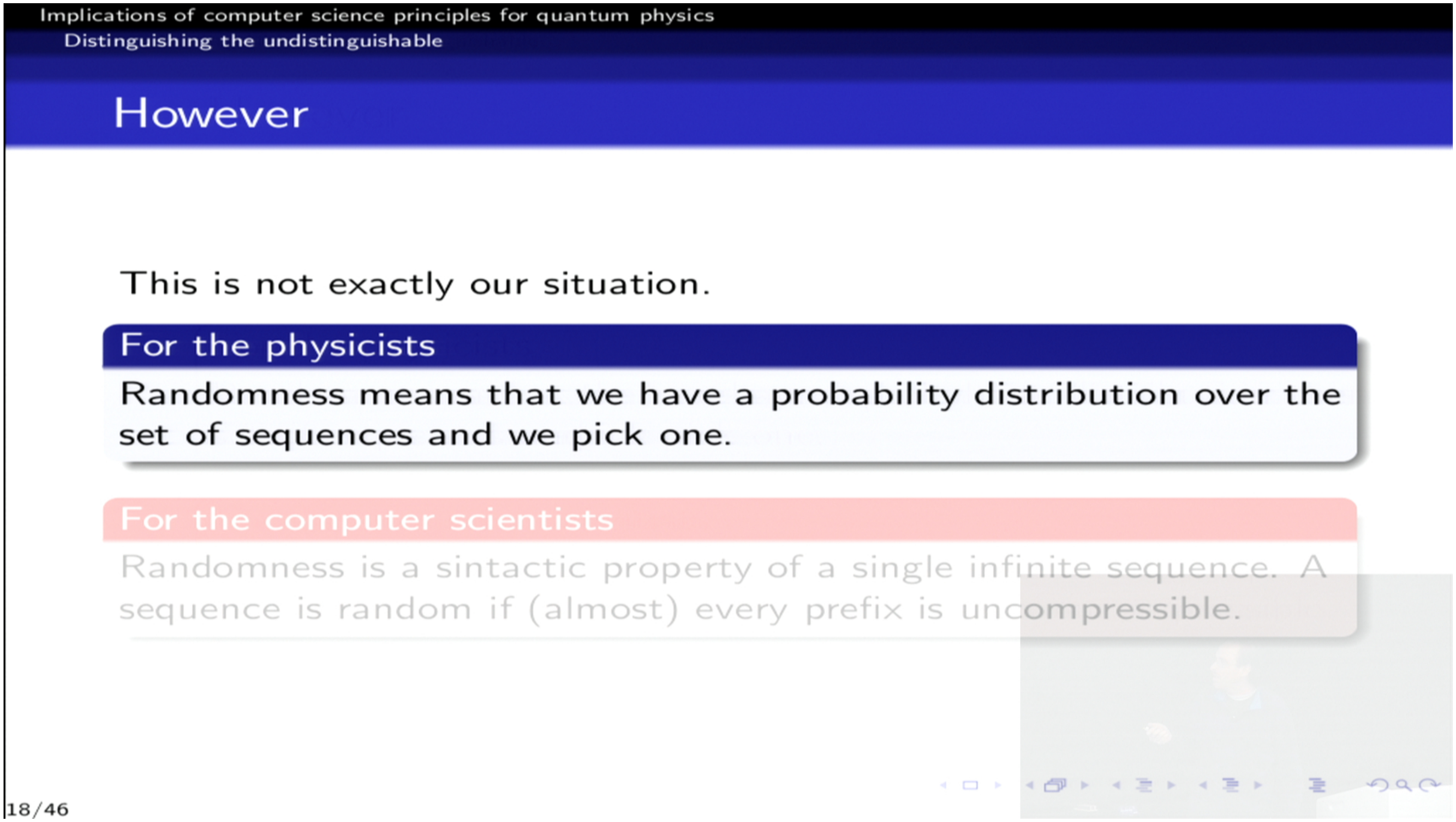
This is not exactly our situation.

For the physicists

Randomness means that we have a probability distribution over the set of sequences and we pick one.

For the computer scientists

Randomness is a sintactic property of a single infinite sequence. A sequence is random if (almost) every prefix is uncompressible.



However

This is not exactly our situation.

For the physicists

Randomness means that we have a probability distribution over the set of sequences and we pick one.

For the computer scientists

Randomness is a sintactic property of a single infinite sequence. A sequence is random if (almost) every prefix is uncompressible.

However

This is not exactly our situation.

For the physicists

Randomness means that we have a probability distribution over the set of sequences and we pick one.

For the computer scientists

Randomness is a sintactic property of a single infinite sequence. A sequence is random if (almost) every prefix is uncompressible.

Distinguishing the undistinguishable

We still have hope

Can we distinguish a computable sequence from one arising from a fair coin with high probability of success?
We proved this to be true.

Recap before getting technical

What we have so far

We reduced the problem of distinguishing two computable preparations of the same mixed state to distinguishing a computable sequence from a sequence obtained by tossing a fair coin. The problem is now classical.

And now...

... we'll go into detail. But not too technical.

Recap before getting technical

What we have so far

We reduced the problem of distinguishing two computable preparations of the same mixed state to distinguishing a computable sequence from a sequence obtained by tossing a fair coin. The problem is now classical.

And now...

... we'll go into detail. But not too technical.

The hand-wavy idea

We have two infinite sequences, X and Y, and it is guaranteed that one is computable and the other came from a fair coin tossing.

- We take a long enough prefix from X. Namely, $X \upharpoonright n$.
- We take a long enough prefix from Y. Namely, $Y \upharpoonright n$.
- We go through all the programs (they are enumerable) until we find one of those prefixes and then we conclude that that one is the computable sequence.

The hand-wavy idea

Not so fast

- What do we mean by *a long enough prefix*?
- We can get a false positive.
- Some programs do not halt, so the previous protocol might never end if we are testing a program that doesn't halt.

Solution

We will see now how to deal with each of these problems.

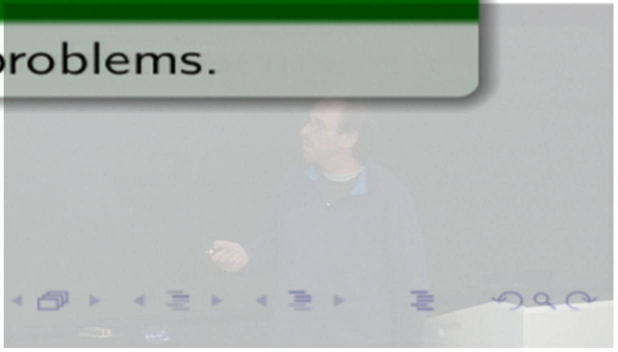
The hand-wavy idea

Not so fast

- What do we mean by *a long enough prefix*?
- We can get a false positive.
- Some programs do not halt, so the previous protocol might never end if we are testing a program that doesn't halt.

Solution

We will see now how to deal with each of these problems.



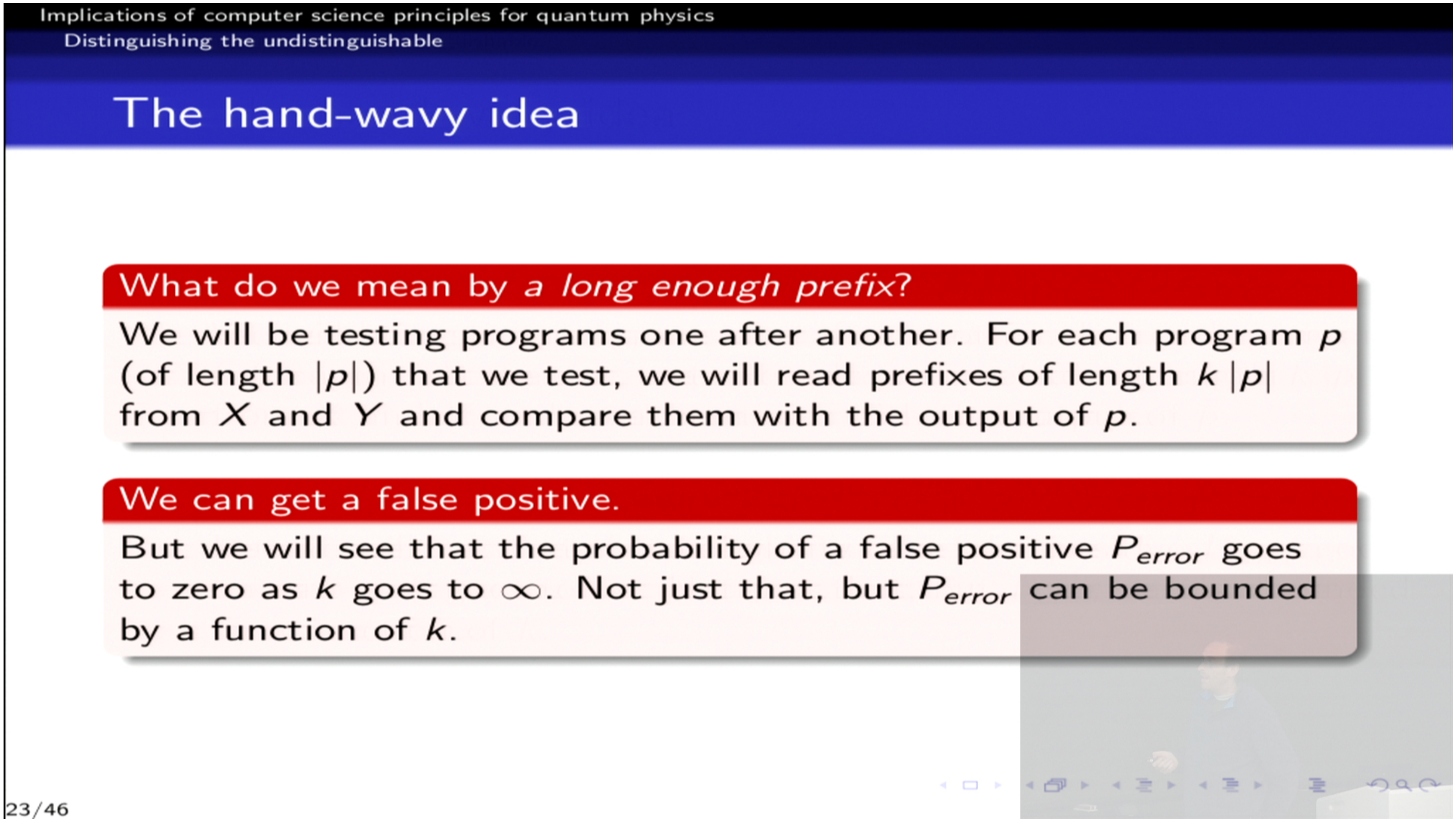
The hand-wavy idea

What do we mean by *a long enough prefix*?

We will be testing programs one after another. For each program p (of length $|p|$) that we test, we will read prefixes of length $k\,|p|$ from X and Y and compare them with the output of p.

We can get a false positive.

But we will see that the probability of a false positive P_{error} goes to zero as k goes to ∞. Not just that, but P_{error} can be bounded by a function of k.



The hand-wavy idea

What do we mean by *a long enough prefix*?

We will be testing programs one after another. For each program p (of length $|p|$) that we test, we will read prefixes of length $k\,|p|$ from X and Y and compare them with the output of p.

We can get a false positive.

But we will see that the probability of a false positive P_{error} goes to zero as k goes to ∞. Not just that, but P_{error} can be bounded by a function of k.

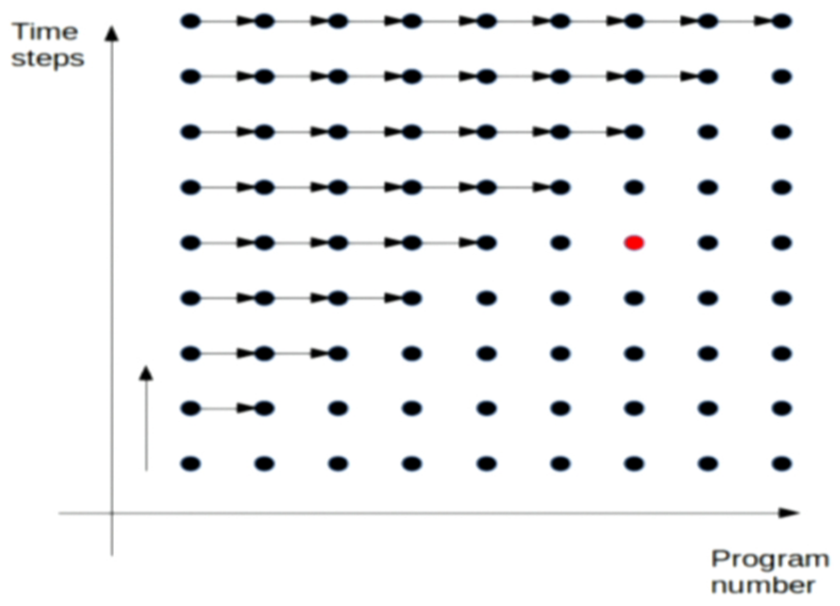
The hand-wavy idea

Some programs do not halt

We will deal with this by dovetailing between programs and execution time.

The hand-wavy idea

Dovetailing



The formal algorithm

Input: $k \in \mathbb{N}$ and $X, Y \in 2^\omega$, one of them being computable
Output: 'X' or 'Y' as the candidate for being computable

```
for $t = 0, 1, \ldots$ do
    for $p = 0, \ldots, t$ do
        if $U_t(p) = X \upharpoonright k|p|$ then
            output '$X$' and halt
        if $U_t(p) = Y \upharpoonright k|p|$ then
            output '$Y$' and halt
```

Observations

- X and Y are treated as oracles.
- The algorithm always halts. Since either $Z = X$ or $Z = Y$ is computable, there exist a program p that generates $Z \upharpoonright k\,|p|$ after some time t_p.
- The only possible error is a misrecognition.

The formal algorithm

```
Input: $k \in \mathbb{N}$ and $X, Y \in 2^\omega$, one of them being computable
Output: '$X$' or '$Y$' as the candidate for being computable
  for $t = 0, 1, \ldots$ do
      for $p = 0, \ldots, t$ do
          if $U_t(p) = X \restriction k|p|$ then
              output '$X$' and halt
          if $U_t(p) = Y \restriction k|p|$ then
              output '$Y$' and halt
```

Observations

- X and Y are treated as oracles.
- The algorithm always halts. Since either $Z = X$ or $Z = Y$ is computable, there exist a program p that generates $Z \restriction k\,|p|$ after some time t_p.
- The only possible error is a misrecognition.

Probability of misrecognition

When can we have a misrecognition?

- It can only happen when the coin produces a sequence of length k that is computable by a program of length 1, or a sequence of length $2k$ generated by a program of length 2, and so on.
- There are at most 2^m sequences generated by programs of length m, and 2^{mk} equiprobable sequences of length mk from a coin.
- How often does a coin give a prefix that is computable by a program k times shorter than the prefix?

Probability of misrecognition

Probability of error

$$P_{\mathrm{error}} \leq \sum_{\ell>0} \frac{2^{\ell}}{2^{k\ell}} = \frac{2^{-(k-1)}}{1-2^{-(k-1)}}$$

which goes to 0 as k goes to ∞.
We pick a k such that the error is lower than what we want, and then we run the recognition algorithm.

Some subtelties

- Our algorithm runs in finite time, but it takes extremely long. It is not something that can be programmed and tested.
- Still, the state has the information on how it was mixed. This is surprising from a fundamental point of view.
- There are computable sequences that look random to the naked eye: The binary digits of pi, some absolutely normal computable numbers, etc. Our algorithm finds those as well.
- The trick comes from the fact that we need i.i.d. variables to do the mix. A program is not i.i.d. But again, nothing classical is according to the CT thesis. The only way that we know to avoid this is by using a quantum coin to choose.

Some subtelties

- Our algorithm runs in finite time, but it takes extremely long. It is not something that can be programmed and tested.
- Still, the state has the information on how it was mixed. This is surprising from a fundamental point of view.
- There are computable sequences that look random to the naked eye: The binary digits of pi, some absolutely normal computable numbers, etc. Our algorithm finds those as well.
- The trick comes from the fact that we need i.i.d. variables to do the mix. A program is not i.i.d. But again, nothing classical is according to the CT thesis. The only way that we know to avoid this is by using a quantum coin to choose.

Why this works?

- A program usually gives simple outputs (in a Kolmogorov complexity sense). Their complexities are, by definition, shorter than the length of the program.
- A coin tipically gives complex outputs (with probability 1 they are Martin-Löf random).
- Our protocol bounds the Kolmogorov complexity of the prefix for the sequence detected as computable. And the bound is roughly that its complexity is $1/k$ of its length. The coin, on the other hand, gives with high probability sequences whose complexity is of the order of its length. Our protocol works by detecting the one with the lower complexity prefix.

Some variations

The MUB case

Mutually unbiased bases

A set of bases $\mathcal{B}_1, \ldots, \mathcal{B}_k$ where $\mathcal{B}_j = \left\{ \left| \psi_1^j \right\rangle, \ldots, \left| \psi_d^j \right\rangle \right\}$ is called a MUB set iif

$$|\langle \psi_p^i | \psi_q^j \rangle| = \delta_{ij}\delta_{pq} + (1 - \delta_{ij})\frac{1}{\sqrt{d}}.$$

MUBs

MUBs are natural generalizations of bases of σ_X and σ_Z. For dimension $d = p^n$ there exist $d + 1$ mutually unbiased bases.

Our algorithm

Still works

If Alice prepares states by classically sampling states from one of a set of MUBs, Bob can still distinguish from which of the bases she is picking by using the same presented algorithm.

The non-orthogonal case

Back to the qubit case.

Giving more freedom to Alice

Now Alice can choose one among k pre-established directions on the Bloch sphere. She then prepares either the $+$ or the $-$ state in that direction.

What does Bob do?

He measures once in every direction yielding k sequences. Now there is one computable and many non-computable sequences that are not ML-Random. It's harder to distinguish.

The trick

Depending on the distribution of directions, Bob starts checking on programs of a certain length skiping short programs.

The non-orthogonal case

Back to the qubit case.

Giving more freedom to Alice

Now Alice can choose one among k pre-established directions on the Bloch sphere. She then prepares either the $+$ or the $-$ state in that direction.

What does Bob do?

He measures once in every direction yielding k sequences. Now there is one computable and many non-computable sequences that are not ML-Random. It's harder to distinguish.

The trick

Depending on the distribution of directions, Bob starts checking on programs of a certain length skiping short programs.

The non-orthogonal case

Back to the qubit case.

Giving more freedom to Alice

Now Alice can choose one among k pre-established directions on the Bloch sphere. She then prepares either the $+$ or the $-$ state in that direction.

What does Bob do?

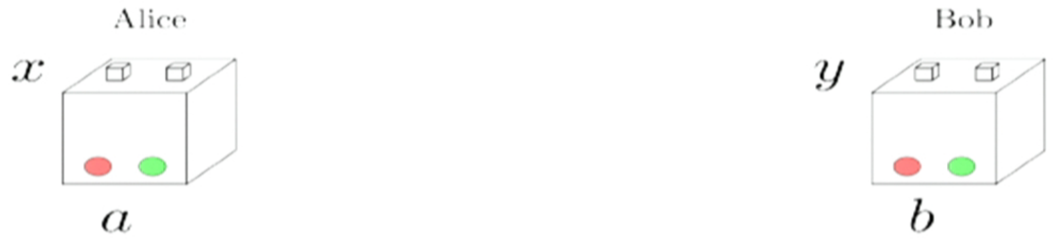
He measures once in every direction yielding k sequences. Now there is one computable and many non-computable sequences that are not ML-Random. It's harder to distinguish.

The trick

Depending on the distribution of directions, Bob starts checking on programs of a certain length skiping short programs.

Bell loophole

Bell scenario and non-locality



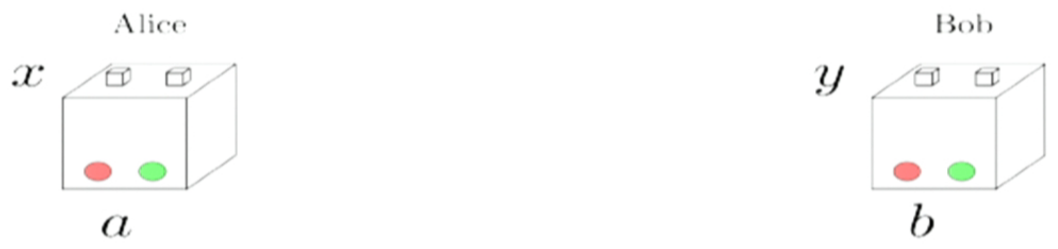
Locality

The system is local iff

$$P(a, b|x, y) = \sum_{\lambda} c_\lambda p_\lambda^A(a|x)\, p_\lambda^B(b|y)\,.$$

In any other case, the distribution is called non-local.

Bell scenario and non-locality

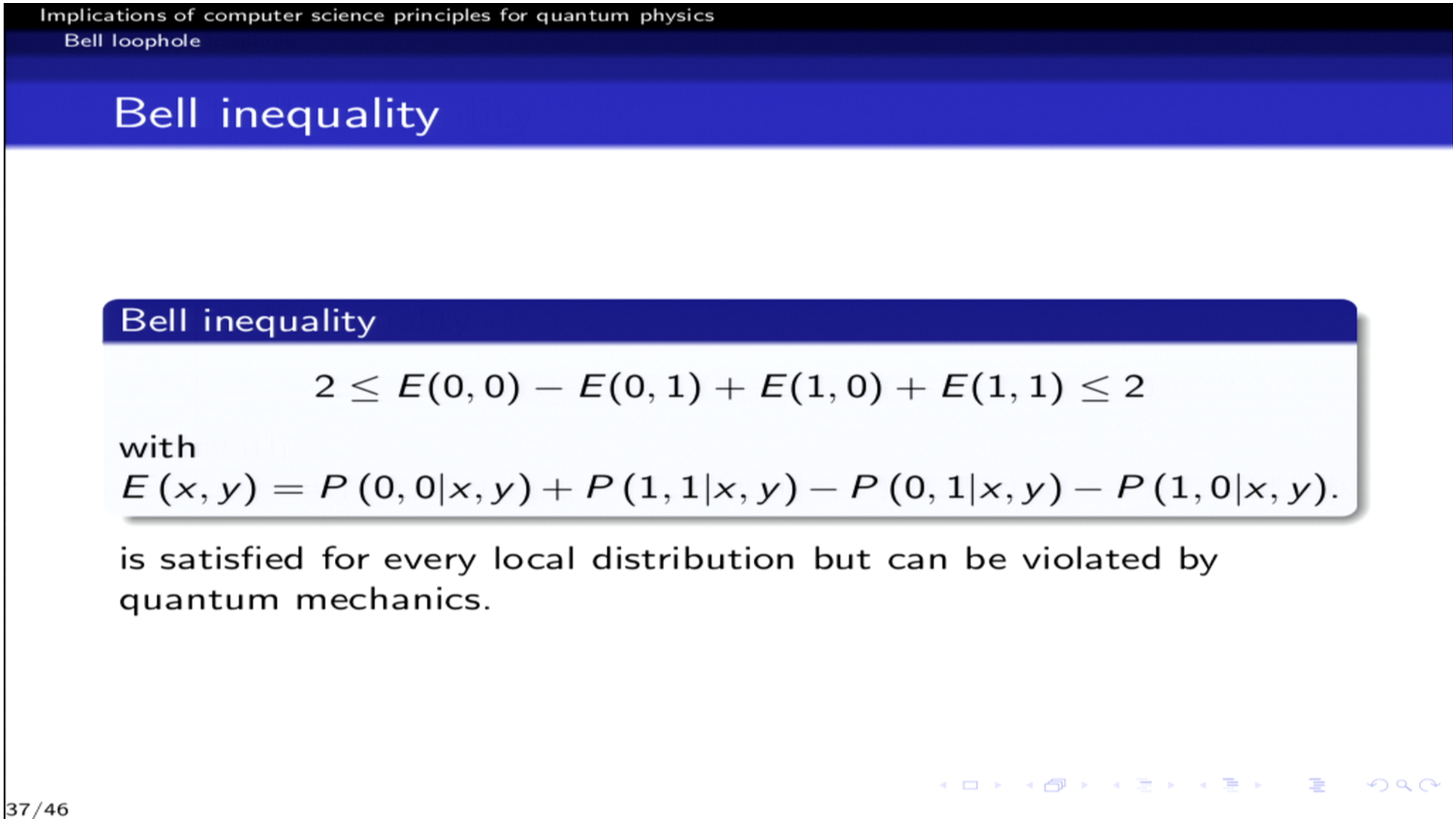


Locality

The system is local iff

$$P(a,b|x,y) = \sum_{\lambda} c_{\lambda} p_{\lambda}^{A}(a|x)\, p_{\lambda}^{B}(b|y)\,.$$

In any other case, the distribution is called non-local.



Bell inequality

Bell inequality

$$2 \leq E(0,0) - E(0,1) + E(1,0) + E(1,1) \leq 2$$

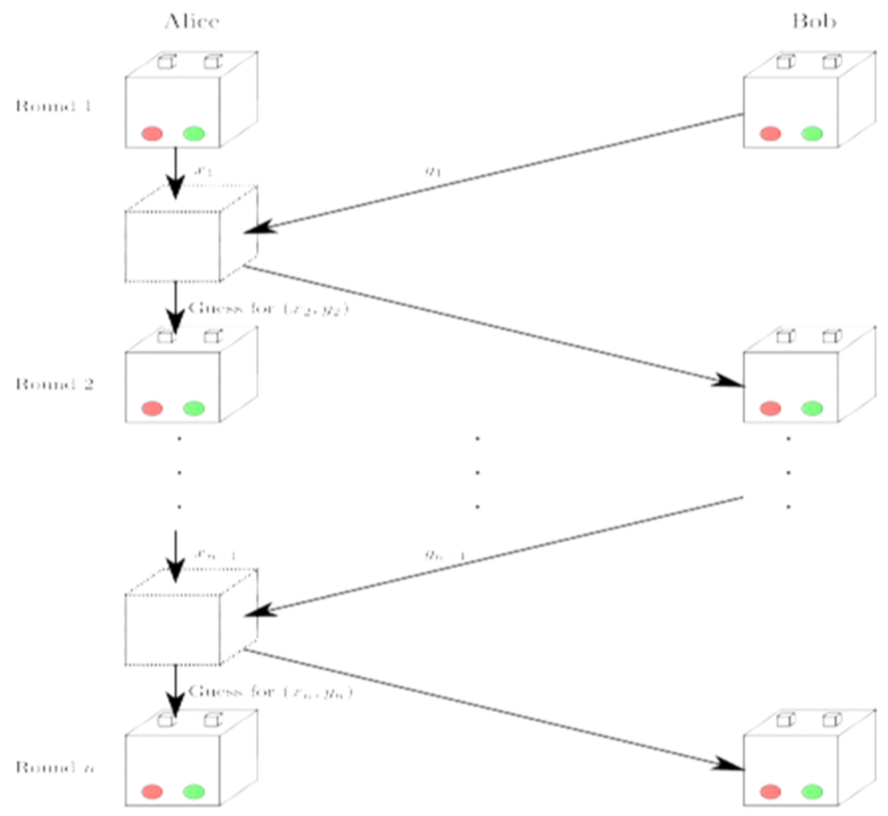
with
$E(x,y) = P(0,0|x,y) + P(1,1|x,y) - P(0,1|x,y) - P(1,0|x,y)$.

is satisfied for every local distribution but can be violated by quantum mechanics.

The memory scenario

We will introduce an eavesdropper that prepares the boxes locally on every round with information on every input from previous rounds. This memory scenario still allows to see non-locality (Barret et al PRA 66:042111, Pironio et al Nature 464(7291):1021-1024, Pironio et al PRA 87:012336).

The memory scenario



The key idea

If the eavesdropper can predict the forthcoming inputs by Alice and Bob, she could prepare the boxes to give whatever probability distribution she wants.

Therefore

The problem is reduced to that of learnability of infinite sequences from prefixes. This problem has already been studied (Solomonoff, Gold, Zeugmann).

The key idea

If the eavesdropper can predict the forthcoming inputs by Alice and Bob, she could prepare the boxes to give whatever probability distribution she wants.

Therefore

The problem is reduced to that of learnability of infinite sequences from prefixes. This problem has already been studied (Solomonoff, Gold, Zeugmann).

Learnability

Main result

Any time (or space) bounded complexity class can be learned from prefixes. It means that after seeing a long enough prefix, every bit is predicted correctly. This classes include P, NP, PR, etc.

The idea

These classes are computably enumerable. The algorithm works as follows:

Seen bits:	1	0	1						
s_1	0	0	0	0	0	0	0	…	
s_2	0	0	1	1	0	1	1	…	
s_3	1	0	0	0	1	0	1	…	
s_4	1	1	1	1	0	0	0	…	
s_5	0	1	0	1	0	1	0	…	
First match: s_6	1	0	1	0	1	1	1	…	Next guess: 0
s_7	1	1	0	1	1	0	1	…	
$\vdots$									

Learnability

Main result

Any time (or space) bounded complexity class can be learned from prefixes. It means that after seeing a long enough prefix, every bit is predicted correctly. This classes include P, NP, PR, etc.

The idea

These classes are computably enumerable. The algorithm works as follows:

Seen bits:		1	0	1						
	s_1	0	0	0	0	0	0	0	...	
	s_2	0	0	1	1	0	1	1	...	
	s_3	1	0	0	0	1	0	1	...	
	s_4	1	1	1	1	0	0	0	...	
	s_5	0	1	0	1	0	1	0	...	
First match:	s_6	1	0	1	0	1	1	1	...	Next guess: 0
	s_7	1	1	0	1	1	0	1	...	
	$\vdots$									

Results

- If Alice and Bob use computable functions (they belong to some time complexity class), they can't rule out an eavesdropper learning a class bigger than the one they are using, therefore they can't conclude nonlocality.
- The eavesdropper, by choosing the class he learns, forces Alice and Bob to have to resort to harder functions. For instance, if the eavesdropper picks NP, then Alice and Bob will have to go beyond NP to have a proper violations of a Bell inequality.

Properties and random thoughts

- Once the sequence is learned, the overhead is small (the eavesdropper keeps simulating the same machine). The eavesdropper does not need more computational power than Alice and Bob.
- The first bits, before the sequences are learned, can give a proper violation. However, is this violation valid if in the long run there is a local model?

Properties and random thoughts

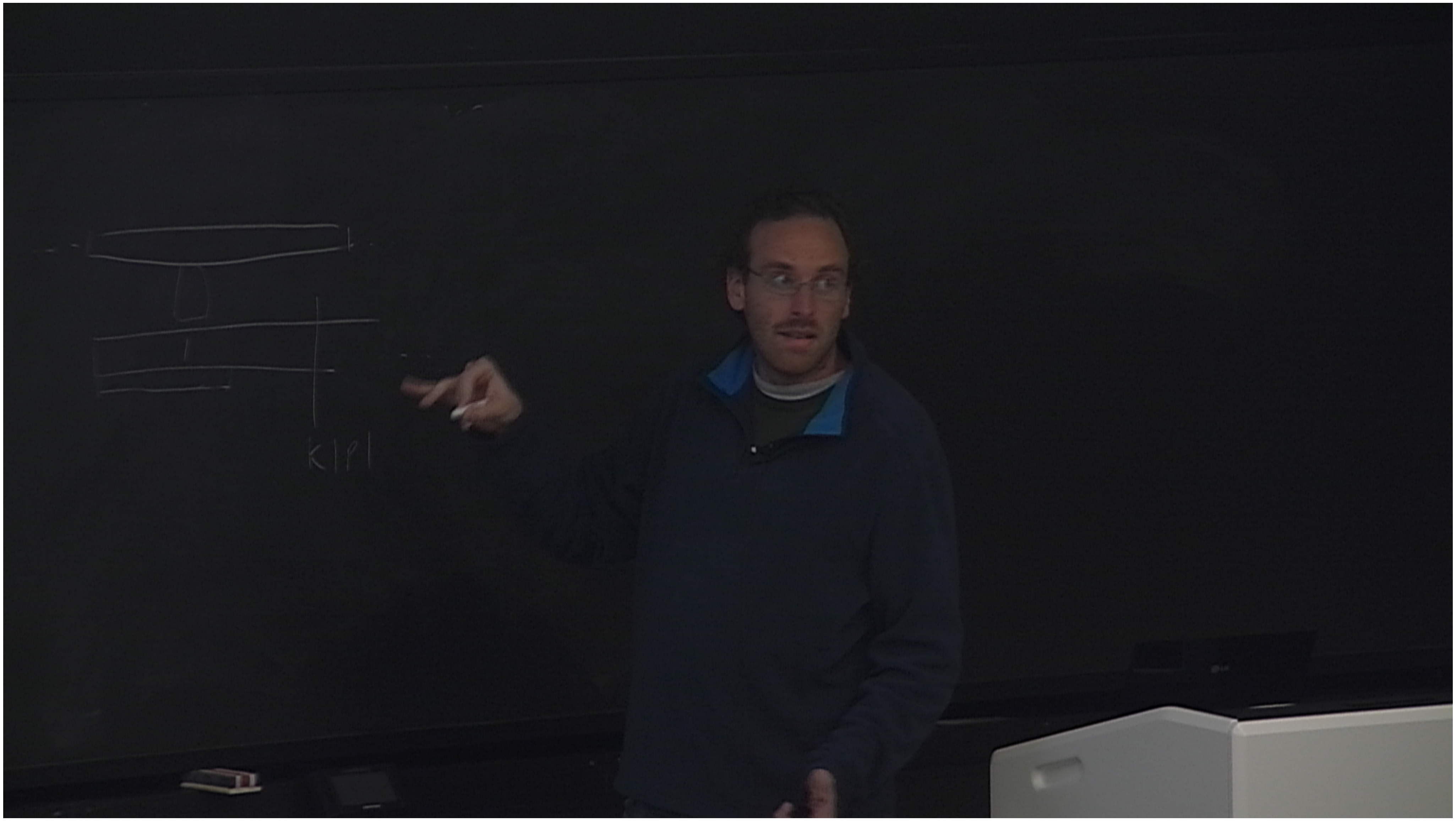
- Once the sequence is learned, the overhead is small (the eavesdropper keeps simulating the same machine). The eavesdropper does not need more computational power than Alice and Bob.
- The first bits, before the sequences are learned, can give a proper violation. However, is this violation valid if in the long run there is a local model?

Summary

- If a mixed state is mixed by mixing pure states in a computable way, they retain information about the basis that was mixed.
- Furthermore, we presented an algorithm that allows to distinguish two mixtures that, before adding the computable requisite, where indistinguishable.
- This implies that, in order for a classical mixture to generate a proper mixed state, some non-computable resource must be available.
- In a Bell experiment, if Alice and Bob use a computer to generate their inputs, they can't rule out an eavesdropper preparing their boxes.
- Such eavesdropper can decide how hard, in terms of time (or space) complexity, it is for Alice and Bob to show a proper violation of a Bell inequality.

Thank you!

quant-ph:1407.0604


K|P|