COMPSCI 755: Unconventional Models of Computation

Lecturer: Professor Cristian S. Calude, Room 252, ext 5751, e-mail : cristian@cs.auckland.ac.nz

Teaching Assistant: Joshua Arulanandham, Room 116, ext 7595, e-mail : hi_josh@hotmail.com

Topics

- Fundamental mathematical constraints on computation
- Fundamental physical constraints on computation
- Potential future computing technologies
- Implications for the mind theories

Format

The course will consist of:

1. Lectures introducing each topic and reviewing the readings
2. Reading assignments from the primary research literature
3. Written assignments to encourage and verify participation
4. Open-discussion and question-and-answer sessions
5. Written exam with questions from the topics discussed in class

Course Work: 60%

You will have *weekly reading assignments*, *fortnightly written assignments*, and *two open-discussion and question-and-answer sessions.*

You will be given a few papers from the primary research literature to read every week. Skim through the readings, and read more thoroughly, at your leisure, the ones that you think you will get most out of. Don't worry if you don't understand every bit of what you read. In this course we will be reading materials that span a wide range of levels of depth and sophistication, and not everyone will understand every phrase and formula in every paper.

In the written assignments (1-2 pages) you may do one of the following:

1. *Summarize* what you learned from the fortnight's lectures and/or readings.
2. *Write* a summary, review, or critique of one or more of the articles/chapters that you read.
3. *Describe and elaborate on* any creative or interesting ideas/thoughts relating to the subject matter that might have been stimulated in your mind as you were listening to/reading/reflecting on the material.
4. *Set up and carry out* any interesting analysis, calculation or simulation relating to any of the quantitative/technical ideas covered during the fortnight.

5. *Correct* any statement that was made in class or in one of the readings which in your opinion is wrong or inaccurate (explain why).

6. *Do a bit of research* on your own. Summarize what you learned and cite your references.

Textbook

C. S. Calude, G. Păun. *Computing with Cells and Atoms*, Taylor & Francis Publishers, London, 2001.

Recommended Books

- J. Gruska. *Quantum Computing,* McGraw-Hill, London, 1999.

  The most comprehensive textbook in Quantum Computing.

- J. G. Hey and R. W. Allen, (eds.). *Feynman Lectures on Computation,* Addison-Wesley, Reading, Massachusetts, 1996.

  Feynman's lecture notes from the course "The Potentialities and Limitations of Computing Machines" taught at Caltech in the early eighties.

- J. G. Hey (ed.). *Feynman and Computation. Exploring the Limits of Computers,* Perseus Books, Reading, Massachusetts, 1999.

  Companion volume to *Feynman Lectures on Computation*, this book collects old and recent articles on the physics of computing by Feynman and his colleagues in physics, electrical engineering, and computer science who were guest lecturers in his course.

- Gh. Păun, G. Rozenberg, A. Salomaa. *DNA Computing. New Computing Paradigms,* Springer-Verlag, Berlin, 1998.

  The best textbook in DNA Computing.

- C. P. Williams, S. H. Clearwater. *Explorations in Quantum Computing,* Springer-Verlag, New York, 1997.

  A very good book in Quantum Computing which comes with software which some may be interested in playing with.

- C. P. Williams, S. H. Clearwater. *Ultimate Zero and One: Computing at the Quantum Frontier*, Springer-Verlag, Heidelberg, 2000.

  A continuation of *Explorations in Quantum Computing*.

Why UMC?

The computer seems to be the only important instrument ever to get exponentially better as it gets cheaper. Its capacity for handling information has been growing about ten million times faster than it did in nervous systems during our entire evolution. The power

- doubled every two years up until 1980s,
- doubled every 18 months in the 1980s (Gordon Moore's 1965 law), and
- is now doubling each year.

By 1993 personal computers provided 10 MIPS (MIPS = million of instructions per second), by 1995 it was 30 MIPS, in 1997 it was over 100 MIPS, now it's about 200 MIPS.

For the sake of a comparison: the human retina uses about 1,000 MIPS to handle edge and motion detectors, while the whole human brain—which is roughly 100,000 times larger than the retina—is worth perhaps 100 million MIPS.

Computers are reading text, recognizing speech, and robots are driving themselves across Mars.

Yet, this exponential race will not guarantee solutions to the many intractable/undecidable problems challenging computer science.

Even worse, it is predictable that this trend of conventional technology will hit the wall in less than 20 years. This is a reason to believe that conventional computation is approaching a critical phase where new technologies will be required to provide significant further progress.

Fundamental Mathematical Constraints on Computation

Church-Turing Thesis

Church-Turing Thesis, a prevailing paradigm in classical computation theory, states that no realizable computing device can be "globally" more powerful, that is, aside from relative speedups, than a universal Turing machine. The modern form of Church-Turing Thesis states that

> *any "reasonable" model of computation can be effectively simulated by a (probabilistic) Turing machine.*

The above statement is a *thesis*, and not a theorem, as it relates an informal notion–a realizable computing device–to the mathematical notion of (probabilistic) Turing machine. Here are some reasons supporting Church-Turing Thesis:

- *Philosophical argument*: Due to Turing's analysis it seems very difficult to imagine some other method which falls outside the scope of his description.
- *Mathematical evidence*: Every mathematical notion of computability which has been proposed was proven equivalent to Turing computability.
- *Sociological evidence*: No example of classical computing device which cannot be simulated by a Turing machine has been given, i.e., the thesis has not been disproved despite having been proposed for more than 60 years.

Church-Turing's Thesis includes a syntactic as well a physical claim. In particular, it specifies which types of computations are physically realisable. According to Deutsch (1982):

> *The reason why we find it possible to construct, say, electronic calculators, and indeed why we can perform mental arithmetic, cannot be found in mathematics or logic.* The reason is that the laws of physics "happen" to permit the existence of physical models for the operations of arithmetic *such as addition, subtraction and multiplication. If they did not, these familiar operations would be non-computable functions. We might still know* of *them and invoke them in mathematical proofs (which would be presumably called "non-constructive") but we could not perform them.*

Church-Turing Thesis was challenged by logicians (Kalmar, Davis, Kreisel), computer scientists (Rosen, Hogarth, Siegelmann) and physicists (Landauer, Svozil). For example, Davis asks himself:

> *"...how can we ever exclude the possibility of our presented, some day (perhaps by some extraterrestrial visitors), with a (perhaps extremely complex) device or "oracle" that "computes" an uncomputable function?"*

Thinking is an essential, if not the most essential, component of human life–it is a mark of "intelligence". Descartes placed the essence of being in thinking. Church-Turing Thesis has been used to approach formally the notion of "intelligent being". In simple terms, Church-Turing Thesis was stated as follows:

> *What is human computable is computable by a universal Turing machine.*

Fundamental Physical Constraints on Computation

An operation is "logically reversible" if it can be *undone*, if it can be run *backwards*, that is, if its inputs can always be deduced from the outputs. Most logical gates are irreversible; a typical example is the NAND gate

$$(a, b) \mapsto \neg(a \wedge b) \tag{1}$$

which has two input bits and only *one* output bit. We cannot recover a unique input from the output bit because the result 1 can be obtained from three distinct inputs: $(0, 0), (0, 1), (1, 0)$.

Assume we operate the gate NAND with two Boolean variables, a, b, and suppose that the four initial states, $(0,0),(0,1),(1,0),(1,1)$, have the same probability distribution, $\frac{1}{4}$. Then, the initial entropy, which is calculated with Shannon's formula:

$$H = -\sum_i p_i \cdot \log p_i,$$

is then

$$H_{initial} = -4 \cdot (\frac{1}{4} \log \frac{1}{4}) = 2 \text{ bits.}$$

The result will be a system with only two possible states, 0 and 1, the outcome 0 appearing with probability $\frac{1}{4}$ and the outcome 1 appearing with probability $\frac{3}{4}$. Consequently, the final entropy is

$$H_{final} = -(\frac{3}{4} \log \frac{3}{4} + \frac{1}{4} \log \frac{1}{4}) = 2 - \frac{3}{4} \log 3 \text{ bits,}$$

which means a loss of

$$H_{initial} - H_{final} = \frac{3}{4} \log 3 \text{ bits.}$$

Assume now that we operate the gate

$$(a, b) \mapsto (a \vee b, a \wedge b),$$

and, again, suppose that the four initial states of the Boolean variables a, b have the same probability distribution, $\frac{1}{4}$. This gate has finally only *three* final states, namely $(0, 0), (1, 0), (1, 1)$, two of them with probability $\frac{1}{4}$ and one with probability $1/2$. Consequently, the final entropy is

$$H_{final} = -(2 \cdot \frac{1}{4} \log \frac{1}{4} + \frac{1}{2} \log \frac{1}{2}) = 1.5 \text{ bits}.$$

In this case, the gate decreases the entropy by 0.5 bits.

The first gate is "more irreversible" than the second one, since it decreases more the entropy.

In thermodynamics the entropy is defined by

$$S = -k \cdot \sum_i p_i \cdot \ln\,(p_i),$$

where $k \approx 1.38 \times 10^{-23}$ joule/°kelvin is Boltzmann's constant.

This notion is coupled to energy through the temperature T of the system: when the entropy of a system is decreased by some amount, then the system dissipates energy equal to the amount of entropy reduction times the temperature. Von Neumann noticed that the two entropies are related by some constant factor, so they are in fact the same notion. When the probability distribution of the system is changed so that the entropy H is decreased by 1 bit, then the entropy S is decreased by $k \cdot \ln 2$ joule/°kelvin, and the system dissipates $kT \cdot \ln 2$ joules of energy in the form of heat. So, a challenging question arises:

> *what is the minimum energy required to carry out a computation?*

Does the above analysis apply to computation? In 1961 Landauer has produced evidence for the affirmative answer. To operate a computer we have to make sure that distinct logical states are represented by distinct physical states. Each bit has two values, 0, 1, so it has one degree of freedom; it corresponds to one or more degrees of freedom of physical states. In general, a set of n bits has n degrees of freedom; they correspond to 2^n physical states. If we erase n bits, say we reset all to 0, then we have compressed 2^n logical states into a single state, a loss of entropy. The irreversible loss information increases temperature of the system, which means, heat dissipation. Consequently, operations which are not one-to-one, which map distinct logical states into a common one, *cost energy.* This cost is expressed by *Landauer's principle*:

erasure of information is a dissipative process.

Here is a simple "home" example. We need two basketballs to design a system of representing information. Put one on the floor by your left foot and hold the other in your (right) hand.

Zero (0) is represented by the ball on the floor; one (1) is represented by the ball in your hand.

Assume that we want to *erase the bit 1*, that is the bit in your hand. To do this you have to *drop the ball.* Simple?

Not really, as the ball does not get *directly* into the floor (to become a 0), but in fact bounces for a while. With a perfectly elastic basketball and a good hard floor the ball may bounce close to your hand, i.e. to the 1 position!

To settle down into 0 the ball has to encounter friction, with the air molecules and the floor. Eventually friction slows down the ball, so 1 has been erased. We could do it because *the energy from bouncing the ball has been transmitted to the floor and the air*. In a vacuum with a perfect frictionless floor erase would be impossible! Energy is consumed in the process of erasure.

A more elaborate example. One can store one bit of information by placing a single molecule in a box, either on the left side or the right side of a partition that divides the box. In this context, erasure means that we choose to move the molecule to the left (or right) side irrespective of whether it started out on the left or right. However, one can suddenly remove the partition, and then slowly compress the one-molecule "gas" with a piston until the molecule reaches the left side. This procedure reduces the entropy of the gas by $\Delta S = k \cdot \ln 2$, and a flow of heat from the box to the environment is produced. Assume now that the process is isothermal at temperature T.

Then work

$$W = kT \cdot \ln 2 \tag{2}$$

is performed on the box, and this work has to be provided. If one decides to erase information, then "a power bill will be generated" and should be paid. For example, according to formula (2), the execution of the gate $(a, b) \mapsto (a \vee b, a \wedge b)$ dissipates at least $\frac{1}{2}kT \cdot \ln 2$ joules of energy. This is a theoretical limit expressing how long the gate can be operated with finite resources of energy.

The energy dissipation has been reduced by approximately a factor of ten every five years, so a rough extrapolation suggests that a reduction of the energy dissipation per logic operation below kT (thermal noise, that is of the order of 10^{-18} picojoule at room temperature) may become relevant in about 10 years. This issue may cause a variety of problems for classical computers, e.g., cooling may be difficult (according to current day knowledge/technology).

Irreversible operations, as the NAND gate, the binary addition $(a, b) \mapsto (a \oplus b, a \wedge b)$ (sum and carry) and the real addition $(x, y) \mapsto x + y$, dissipate energy. Is logical reversibility dissipation free?

The above irreversible operations can be easily simulated by reversible ones. A reversible version of the NAND gate[a] is, for example, Toffoli's gate

$$(a, b, c) \mapsto (a, b, c \oplus (a \wedge b)).^{\text{b}} \tag{3}$$

[a] A single NAND gate is as good as having both AND and NOT: $\neg a$ =NAND$(a, 1)$, AND(a, b) = $\neg$(NAND (a, b)) = NAND(NAND$(a, b), 1)$.

[b] Recall that $a \oplus b$ is 1 only if a and b have different values, i.e., $a = 0, b = 1$ or $a = 1, b = 0$.

Indeed, (3) is a reversible 3-bit gate that flips the third bit if the first two both take the value 1 and does nothing otherwise. Hence, the third output bit becomes the NAND of a and b in case $c = 1$. The price paid to get reversibility consisted in adding a new variable c.

Similar tricks can be used to produce reversible versions of the binary addition, $(a, b) \mapsto (a, a \oplus b, a \wedge b)$ and real addition $(x, y) \mapsto (x + y, x - y)$. In the first case we replicated the first variable a; in the second case we added a new component storing some additional value.

A computer may be fully reversible and yet dissipate energy! The important point is that the laws of physics allow for technologies to make reversible computers operate with negligible dissipation. To build a reversible computer one needs only two types of logical gates, say AND and NOT (because any other gate can be constructed from these two types–they are *universal*). Clearly, the NOT gate is reversible as its composition with itself gives the initial input. However, the AND gate is irreversible. Reason: it has two inputs and only one output, so it has to lose information (it is impossible to tell exactly what inputs must have been if all one is told is the output 0: any of the three combinations $(0,0),(0,1),(1,0)$, could have been the "real input").

To make a reversible variant of the gate AND we need to make sure that we have the same number of output lines as input ones, so, in principle, we can just add some "garbage" output lines to solve the problem. However, this may not be enough, as we want to guarantee also universality! One possibility is Toffoli's reversible 3-bit gate which uses in addition to a, b a control bit c. Input bits a and b do not change their states; the control bit, however, will change its state, but only when $a = b = 1$.

Toffoli's truth table is the following:

input			**output**		
a	b	c	a	b	c
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	1	1	1

input			**output**		
a	b	c	a	b	c
0	0	1	0	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Toffoli's gate.

Fredkin's reversible 3-bit gate also uses in addition to a, b a control bit c in the following way: a) if $c = 0$, then the values of a, b are transmitted unaltered, i.e., the output is the pair (a, b), b) if $c = 1$, then the values of a, b are switched to the opposite output, i.e., the output is the pair (b, a). Its truth table is the following:

input			**output**		
a	b	c	a	b	c
0	0	0	0	0	0
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	1	1	0

input			**output**		
a	b	c	a	b	c
0	0	1	0	0	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	1	1	1	1

Fredkin's gate.

Fredkin's gate is universal in the sense that it can be used to construct reversible variants of all Boolean gates, and satisfies one additional requirement: the number of 1s and 0s never changes. To prove universality it is enough to show that the gates NOT and AND can be represented using Fredkin's gate FREDKIN with particular inputs. It is not difficult to check that the following formulae work:

$$\text{FREDKIN } (a, b, c)$$

$$= ((((\neg c) \wedge a) \vee (c \wedge b)), ((a \wedge c) \vee (b \wedge (\neg c)), c),$$

$$\text{FREDKIN } (1, 0, c) = (\neg c, c, c), \tag{4}$$

and

$$\text{FREDKIN } (0, b, c) = (b \wedge c, b \wedge (\neg c), c), \tag{5}$$

So, both NOT and AND can be simulated by FREDKIN.

Fredkin's gate has often been used for photon based gates where a 1 represents a photon and a 0 simply denotes the absence of a photon; nonlinear optics is used to control the output of an interferometer. The number of ones cannot change as the number of photons cannot change–absorption is not allowed for reversible gates.

In both formulae (4) and (5) there are more outputs than are required for the computed functions (one for NOT and two for AND): these outputs, called *garbage* bits, are a necessary consequence of reversible logic. Consequently, one may wonder whether we have only postponed the energy cost; garbage bits can be irreversibly erased, but that would require to pay Landauer's price ...

Bennett found in 1973 that any computation can be performed using only reversible steps, and so "in principle" requires no dissipation and no power expenditure: *we do not need to erase the garbage bits!* By pointing out that a reversible computer can run forward to the end of a computation, print out a copy of the answer (a logically reversible operation) and then reverse all of its steps to return to its initial configuration, Bennett invented a procedure to remove the garbage without any energy cost. Here is a simple illustration of this technique:

$$\text{INPUT00000} \mapsto \text{ COMPUTER00000}$$

$$\mapsto \text{INPUTOUTPUT } \mapsto \text{COPYOUTPUT}$$

$$\mapsto \text{RETUPMOC } \mapsto \text{INPUT00000} \mapsto \text{OUTPUT}$$

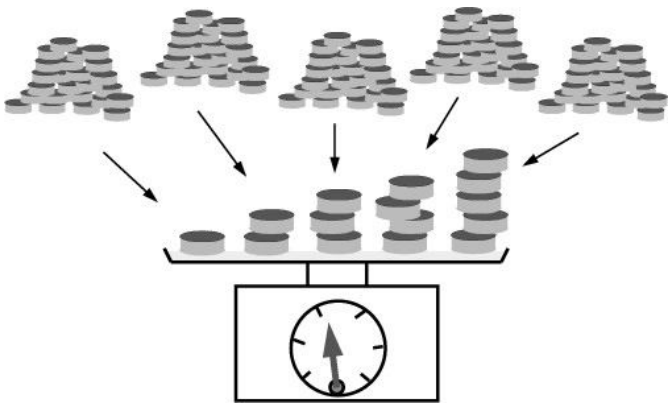
The inevitability of handling garbage bits implies the necessity to allow more input bits than are needed in an irreversible computation (INPUT $\mapsto$ INPUT00000); of course, some garbage bits may be useful only for internal computation, but still, a certain additional bits will be required.

In principle, we need not pay any "power bill" to compute reversibly. In practice, the (irreversible) computers in use today dissipate energy of orders of magnitude more than (2) per gate, so *today* Landauer's limit seems not to be an important engineering principle. But as computing hardware continues to shrink in size, it may become important to beat Landauer's limit, for example, to prevent the components from melting. Then reversible computation may be one, if not the only one, option.

The following problem illustrates the "quantum" approach to problem solving.

Merchant's Problem

A merchant learns than one of his five stacks of coins contains only false coins, 0.01 grams heavier than normal ones. Can he find the odd stack by a single "weighing"?



Coin selection

What about the case when more than one stack of coins contains false coins: can we, again with only one single weighting find all stacks containing false coins?

Solution: choose 1,2,4,8,16 coins from each stack!

- What are the limits of the above solutions?
- What about the case when we are allowed to take only just one coin from each stack?
- What about the case when we have infinitely many stacks?

Bits and Qubits

A classical **bit** (e.g., the position of gear teeth in Babbage's differential engine, a memory element or wire carrying a binary signal, in contemporary machines) is a system comprising many atoms. Typically, the system is described by one or more continuous parameters, for example, voltage. Such a parameter is used to separate the space into two well-defined regions chosen to represent 0 and 1. Manufacturing imperfections, local perturbations may affect, so signals are periodically restored toward these regions to prevent them from drifting away. An n-bit register of memory can exist in any of 2^n logical states, from $00\ldots0$ (n zeros) to $11\ldots1$ (n ones).

A quantum event in which we have two possible mutually exclusive outcomes is the elementary act of observation: all knowledge of the physical world is based upon such acts. An elementary act of observation is simultaneously like a coin-toss and not like a coin-toss. The information derived from an elementary act of observation is no more than a single bit, but *there is more on it than that.* To mark this difference Schumaker has coined the name qubit.

A quantum bit, **qubit**, is typically a microscopic system, such as an atom or nuclear spin or polarized photon. For example, the state of a spin-$\frac{1}{2}$ particle, when measured, is always found to be in one of two possible states, represented as

$$|+\frac{1}{2}\rangle \text{ (spin-up) or } |-\frac{1}{2}\rangle \text{ (spin-down).}$$

One can use one spin state to represent 0, and the other spin state to represent 1. There is nothing special about spin systems–any 2-state quantum system can be equally used to represent 0 and 1. What is really special here is the existence of a continuum of intermediate states which are superpositions of 0s and 1s. Mathematically they are just linear combinations of the basis states.

Unlike the intermediate states of a classical bit (for example, any voltages between the "standard" representations of 0 and 1) which can be distinguished from 0 and 1, but do not exist from an informational point of view, quantum intermediate states cannot be reliably distinguished, even in principle, from the basis states, but do have an informational "existence".

An n-qubit system can exist in any superposition of the form

$$\Psi = \sum_{x=00\ldots0}^{11\ldots1} c_x |x\rangle, \tag{6}$$

where c_x are (complex) numbers such that $\sum_x |c_x|^2 = 1$. The exponential "explosion" represented by formula (6) distinguishes quantum systems from classical ones: in a classical system a state is described by a number of parameters growing only linearly with the size of the system (classical systems are completely described locally, that is, via each state in part), but, as we shall see later, quantum systems may not admit such a description (because quantum states may be "entangled").

We present some rudiments of (finite dimensional) Hilbert space theory. A Hilbert space is a mathematical model for representing vectors. The state of a quantum system can be described by a column vector in a Hilbert space of wave functions; as the system evolves, its state vector rotates with its base anchored to the origin of axes. Vectors can be added and multiplied by (complex) numbers. State vectors are typically written with a special angular bracket notation, the "ket vector" $|\Psi\rangle$. The word "ket" was invented by Paul Dirac. Row vectors, such as $\langle\Psi|$, are known as "bra" vectors; when you put together a column and a bra vector, you get a bracket, that is the inner product of the two vectors, $\langle\Psi||\Psi\rangle$, also written as $\langle\Psi|\Psi\rangle$.

A simple 2-state quantum system (the basic block of a quantum memory register) can, by definition, be in one of two possible states. To model it we need the smallest non-trivial Hilbert space $\mathbf{C}^2$, a two dimensional space. Assume that a particular complete orthonormal basis, denoted by $\{|0\rangle, |1\rangle\}$, has been fixed.[a] These vectors, $|0\rangle$ and $|1\rangle$, correspond to the classical bit values 0 and 1, respectively.

[a]Here "complete" refers to the fact that every state vector in the Hilbert space can be represented in the form (7), and "orthonormal" means that vectors are perpendicular to one another, and normalized. For example, the vectors $|0\rangle$ and $|1\rangle$ may correspond to the horizontal polarization $|\rightarrow\rangle$ and the vertical polarization $|\uparrow\rangle$ of a photon, respectively.

A qubit is a unit vector in the space $\mathbf{C}^2$, so for each qubit $|x\rangle$, there are two (complex) numbers $a, b \in \mathbf{C}$ such that

$$|x\rangle = a|0\rangle + b|1\rangle = \begin{pmatrix} a \\ b \end{pmatrix}, \tag{7}$$

where

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

and $|a|^2 + |b|^2 = 1$.

The angle which a qubit makes with the vertical axis describes the relative contributions of $|0\rangle$ and $|1\rangle$. The angle through which the vector is rotated about the vertical axis induce to so-called "phase". So, different qubits may have the same proportion of $|0\rangle$ and $|1\rangle$, but with different phase factors. Phase is irrelevant for the whole states but it's crucial for "quantum interference effects".

We can perform a measurement that projects the qubit onto the basis $\{|0\rangle, |1\rangle\}$. Then we will obtain the outcome $|1\rangle$ with probability $|b|^2$, and the outcome $|0\rangle$ with probability $|a|^2$. With the exception of limit cases $a = 0$ and $b = 0$, the *measurement irrevocably disturbs the state*: If the value of the qubit is initially unknown, then there is no way to determine a and b with any conceivable measurement. However, *after* performing the measurement, the qubit has been prepared in a known state (either $|0\rangle$ or $|1\rangle$); this state is typically different from the previous state.

The above facts point out an important difference between qubits and classical bits. There is no problem in measuring a classical bit without disturbing it, so we can decode all of the information that it encodes. If we have a classical bit with a fixed, but unknown value (0 or 1), then we can only say that there is a probability that the bit has the value 0, and a probability that the bit has the value 1, and these two probabilities add up to 1. When we measure the bit, we acquire additional information; after measurement, we will know *completely* the value of the bit.

The ability of quantum systems to exist in a "blend" of all their allowed states simultaneously is known as the *Principle of Superposition*. Even though a qubit can be put in a superposition (7), it contains no more information than a classical bit, in spite of its having infinitely many states. The reason is that information can be extracted only by measurement. But, as we have argued, for any measurement of a qubit with respect to a given orthonormal basis, there are only two possible results, corresponding to the two vectors of the basis. On the other hand, it is not possible to capture more information measuring in two different bases because the measurement changes the state. Even worse, quantum states cannot be cloned, hence it's impossible to measure a qubit in two different ways (even, indirectly, by using a copy trick, that is copying and measuring the copy).

Is a qubit identical to a probabilistic classical bit? The answer is negative and an argument is that the numbers a and b in (7) encode *more* than just the probabilities of the outcomes of a measurement in the $\{|0\rangle, |1\rangle\}$ basis. For example, the relative phase of a and b is crucial.

For example, consider the qubits $|\varphi\rangle = a|0\rangle + b|1\rangle$ and $|\psi\rangle = a|0\rangle - b|1\rangle$, which are similar in amplitudes (as the relative proportions of $|0\rangle$ and $|1\rangle$ are the same), but *differ by a phase*. Consider the quantum transformation

$$\alpha|0\rangle + \beta|1\rangle \mapsto [(\alpha+\beta)\sqrt{2}/2]|0\rangle + [(\alpha-\beta)\sqrt{2}/2]|1\rangle.$$

For $a = b = \sqrt{2}/2$, $|\varphi\rangle$ produces surely $|0\rangle$ while $|\psi\rangle$ produces surely $|1\rangle$. Hence, the final result is acutely sensitive to the phase factors.

Systems of more than one qubit need a Hilbert space which captures the interaction of the qubits. A two qubit system can be represented by a unit vector in the tensor product of two copies of $\mathbf{C}^2$, i.e., the space $\mathbf{C}^2 \otimes \mathbf{C}^2$. Using Dirac notation, if $|0\rangle$ and $|1\rangle$ are the vectors of a basis in $\mathbf{C}^2$ then, the set

$$\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\}$$

$$= \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$$

is a basis in $\mathbf{C}^2 \otimes \mathbf{C}^2$.

More precisely,

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

$$|10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \qquad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

In general, a system containing exactly $n \geq 2$ qubits is represented by n copies of $\mathbf{C}^2$ tensored together. Therefore, the state space is 2^n dimensional. A natural basis for this space consists of 2^n tensor products:

$$|0\rangle \otimes |0\rangle \otimes \ldots \otimes |0\rangle,$$

$$|0\rangle \otimes |0\rangle \otimes \ldots \otimes |1\rangle,$$

$$\vdots$$

$$|1\rangle \otimes |1\rangle \otimes \ldots \otimes |1\rangle.$$

A classical string of bits $i_1 i_2 \ldots i_n$ with $i_k \in \{0, 1\}$, $1 \leq k \leq n$, corresponds to the quantum state $|i_1\rangle \otimes |i_2\rangle \otimes \ldots \otimes |i_n\rangle$ which is simply denoted by $|i_1 i_2 \ldots i_n\rangle$. If $|0\rangle$ and $|1\rangle$ are orthogonal unit vectors in $\mathbf{C}^2$, then the set

$$\{|i_1 i_2 \ldots i_n\rangle | i_k \in \{0, 1\}, 1 \leq k \leq n\}$$

is an orthonormal basis in $\mathbf{C}^2 \otimes \mathbf{C}^2 \otimes \ldots \otimes \mathbf{C}^2$.

In contrast with the classical physics, where the state of a system is completely defined by describing the state of each of its component pieces separately, in a quantum system the state cannot always be described considering only the component pieces. For instance, the state

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

cannot be decomposed into separate states for each of the two bits. This means that we cannot express this state as a tensor product of two single qubits. Indeed, let's assume for the sake of a contradiction, that there exist two kets $|x\rangle$ and $|y\rangle$ in $\mathbf{C}^2$ such that

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = |x\rangle \otimes |y\rangle.$$

Since each single qubit is in a superposition of $|0\rangle$ and $|1\rangle$, there exist four complex numbers a_1, b_1, a_2, b_2 such that

$$|x\rangle = a_1|0\rangle + b_1|1\rangle$$

and

$$|y\rangle = a_2|0\rangle + b_2|1\rangle.$$

It follows that

$$\begin{aligned} |x\rangle \otimes |y\rangle &= (a_1|0\rangle + b_1|1\rangle) \otimes (a_2|0\rangle + b_2|1\rangle) \\ &= a_1a_2|00\rangle + a_1b_2|01\rangle + b_1a_2|10\rangle \\ &\quad + b_1b_2|11\rangle, \end{aligned}$$

hence $a_1b_2 = 0$ and $a_1a_2 = \frac{1}{\sqrt{2}} = b_1b_2$, which is impossible.

A state that cannot be expressed as a tensor product is called an **entangled state**. Since the space $\mathbf{C}^2 \otimes \mathbf{C}^2$ is spanned by the set $\{|x\rangle \otimes |y\rangle| \quad x, y \in \mathbf{C}^2\}$, the existence of entangled states proves that the previous set is not a linear space. One can easily find entangled states in an n qubit system, for any integer $n \geq 2$.

Note that it would require vast resources to simulate even a small quantum system on a conventional computer, as such a simulation would require keeping track of exponentially many states: the dimension of the cartesian product of multiple classical particles grows linearly with the number of particles, while the dimension of the tensor product of quantum systems grows exponentially. A reason for the (potential) power of quantum computers is the ability of exploiting the quantum state evolution as a computational mechanism.

Qubit Evolution

The quantum evolution (quantum transformation, operator) of (on) a qubit is described by a "unitary operator", that is an operator induced by a unitary matrix.[a]

Any unitary operator $U : \mathbf{C}^2 \to \mathbf{C}^2$ can be viewed as a single qubit gate. Considering the basis $\{|0\rangle, |1\rangle\}$, the transformation is fully specified by its effect on the basis vectors. In order to obtain the associated matrix of an operator U, we put the coordinates of $U|0\rangle$ in the first column and the coordinates of $U|1\rangle$ in the second one. So, the general form of a transformation that acts on a single qubit is a 2×2 matrix

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

which transforms the qubit state $\alpha|0\rangle + \beta|1\rangle$ into the state $(\alpha a + \beta b)|0\rangle + (c\alpha + d\beta)|1\rangle$:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha a + \beta b \\ c\alpha + d\beta \end{pmatrix}.$$

[a] A quadratic matrix A of order n over $\mathbf{C}$ is *unitary* if $AA^\dagger = I$ (the identity $n \times n$ matrix); $A^\dagger$ is the transposed conjugate matrix of A.

For $\theta \in [0, 2\pi)$, the rotation R_θ is given by

$$R_\theta = \begin{pmatrix} cos\,\theta & -sin\,\theta \\ sin\,\theta & cos\,\theta \end{pmatrix}.$$

Hence, R_θ acts as follows:

$$|0\rangle \mapsto cos\,\theta|0\rangle + sin\,\theta|1\rangle,\ |1\rangle \mapsto -sin\,\theta|0\rangle + cos\,\theta|1\rangle.$$

One can easily verify that $R_\theta R_\theta^\dagger = R_\theta R_\theta^T = I$, hence R_θ is unitary. Note that in the special case $\theta = 0$ we get the identity transformation of $\mathbf{C}^2$:

$$R_0 = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

We may think of logic gates as transformations. For example, the NOT transformation which interchanges the vectors $|0\rangle$ and $|1\rangle$, is given by R_π, that is the matrix

$$\text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

It flips that state of its input,

$$\text{NOT}\,|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle,$$

and

$$\text{NOT}\,|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle.$$

The phase shift gate $Shift$ is defined as follows:
$Shift\,|0\rangle = |0\rangle, Shift\,|1\rangle = -|1\rangle$, so

$$Shift = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Since NOT $\cdot$ NOT $^{\dagger} = I$ and $Shift \cdot Shift^{\dagger} = I$, the operators NOT and $Shift$ are also unitary. The operator $Shift \cdot$ NOT is also a unitary transformation and we have:

$$Shift \cdot \text{NOT}\ |0\rangle = Shift\,|1\rangle = -|1\rangle,$$

$$Shift \cdot \text{NOT}\ |1\rangle = Shift\,|0\rangle = |0\rangle.$$

Therefore, its associated matrix is

$$R_{3\pi/2} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

The square-root of NOT (introduced by Deutsch) is the transformation

$$\sqrt{\text{NOT}}: \quad \begin{aligned} |0\rangle &\rightarrow \tfrac{1}{2}(1+i)|0\rangle + \tfrac{1}{2}(1-i)|1\rangle, \\ |1\rangle &\rightarrow \tfrac{1}{2}(1-i)|0\rangle + \tfrac{1}{2}(1+i)|1\rangle, \end{aligned}$$

$$\sqrt{\text{NOT}} = \frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}.$$

$$\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}} = \text{NOT}\ , \tag{8}$$

and

$$\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}^{\dagger}$$

$$= \frac{1}{4}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}\begin{pmatrix} 1-i & 1+i \\ 1+i & 1-i \end{pmatrix} = I.$$

The square-root of NOT is a typical "quantum" gate in the sense that *it is impossible to have a single-input/single-output classical binary logic gate that satisfies* (8). Indeed, any classical binary

$$\sqrt{\text{NOT}}_{\text{classical}}$$

gate is going to output a 0 or a 1 for each possible input 0/1. Assume that we have such a classical square-root of NOT gate acting as a pair of transformations

$$\sqrt{\text{NOT}}_{\text{classical}}(0) = 1,\ \sqrt{\text{NOT}}_{\text{classical}}(1) = 0.$$

Then, two consecutive applications of it will *not* flip the input!

Finally we consider the Hadamard transformation H is defined by

$$H: \begin{array}{rcl} |0\rangle & \rightarrow & \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ |1\rangle & \rightarrow & \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{array},$$

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

This transformation has a number of important applications. When applied to $|0\rangle$, H creates a superposition state

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

Applied to n bits individually, H generates a superposition of all 2^n possible states. To see this we need some rudiments on tensor products.

Consider two operators $A : \mathbf{C}^n \to \mathbf{C}^m$ and $B : \mathbf{C}^q \to \mathbf{C}^p$. The tensor product of A and B is the operator $A \otimes B : \mathbf{C}^n \otimes \mathbf{C}^q \to \mathbf{C}^m \otimes \mathbf{C}^p$, with the property $A \otimes B(x \otimes y) = Ax \otimes By$, for any $x \in \mathbf{C}^n$ and $y \in \mathbf{C}^q$. A convenient way is, again, to work with matrices. Let A be a $(m \times n)$ matrix and B a $(p \times q)$ matrix. The (right) Kronecker product of A and B is the $(mp \times nq)$ matrix defined as follows:

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}.$$

For example, if

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix},$$

are two 2×2 matrices, then we have:

$$A \otimes B = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{pmatrix}$$

Two important mathematical results are useful:

a) If A and B are matrices associated to the operators A and B, then the matrix associated to $A \otimes B$ is the Kronecker product of A and B.

b) The tensor products of two unitary transformations is also unitary.

Consequently, considering the tensor product of n single qubit transformations, we can obtain examples of unitary transformations acting on n qubits.

For instance, let $(|00\rangle, |01\rangle, |10\rangle, |11\rangle)$ be the basis in $\mathbf{C}^2 \otimes \mathbf{C}^2$ and consider the following transformations:

$$I \otimes I : \begin{array}{lcl} |00\rangle & \rightarrow & |00\rangle \\ |01\rangle & \rightarrow & |01\rangle \\ |10\rangle & \rightarrow & |10\rangle \\ |11\rangle & \rightarrow & |11\rangle \end{array}, \; I \otimes I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$I \otimes \mathrm{NOT} \; : \; \begin{array}{lcl} |00\rangle & \rightarrow & |01\rangle \\ |01\rangle & \rightarrow & |00\rangle \\ |10\rangle & \rightarrow & |11\rangle \\ |11\rangle & \rightarrow & |10\rangle \end{array},$$

$$I \otimes \mathrm{NOT} \; = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

$$\mathrm{NOT} \otimes I : \; \begin{array}{lcl} |00\rangle & \rightarrow & |10\rangle \\ |01\rangle & \rightarrow & |11\rangle \\ |10\rangle & \rightarrow & |00\rangle \\ |11\rangle & \rightarrow & |01\rangle \end{array},$$

$$\mathrm{NOT} \otimes I = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

$$\text{NOT} \otimes \text{NOT} \; : \; \begin{array}{lcl} |00\rangle & \rightarrow & |11\rangle \\ |01\rangle & \rightarrow & |10\rangle \\ |10\rangle & \rightarrow & |01\rangle \\ |11\rangle & \rightarrow & |00\rangle \end{array},$$

$$\text{NOT} \otimes \text{NOT} \; = \; \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

$$Shift \otimes \text{NOT} \; : \; \begin{array}{lcl} |00\rangle & \rightarrow & |01\rangle \\ |01\rangle & \rightarrow & |00\rangle \\ |10\rangle & \rightarrow & -|11\rangle \\ |11\rangle & \rightarrow & -|10\rangle \end{array},$$

$$Shift \otimes \text{NOT} \; = \; \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{pmatrix}.$$

We come back to binary representations of the numbers from 0 to $2^n - 1$ via Hadamard operator. The Walsh-Hadamard transformation is defined recursively by

$$W_n = H, \text{ if } n = 1 \text{ and } W_n = H \otimes W_{n-1},$$

for any $n \geq 2$. For example, if $n = 2$ then

$$\begin{aligned} W_2|00\rangle &= (H \otimes H)|00\rangle \\ &= H|0\rangle \otimes H|0\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle), \end{aligned}$$

$$\begin{aligned} W_2|01\rangle &= (H \otimes H)|01\rangle \\ &= H|0\rangle \otimes H|1\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle), \end{aligned}$$

$$
\begin{aligned}
W_2|10\rangle &= (H \otimes H)|10\rangle \\
&= H|1\rangle \otimes H|0\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\
&= \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle - |11\rangle),
\end{aligned}
$$

$$
\begin{aligned}
W_2|11\rangle &= (H \otimes H)|11\rangle \\
&= H|1\rangle \otimes H|1\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\
&= \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle),
\end{aligned}
$$

so the associated matrix is

$$
W_2 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.
$$

For $n = 3$, we have

$$\begin{aligned}
W_3|000\rangle &= (H \otimes W_2)|000\rangle \\
&= H|0\rangle \otimes W_2|00\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{2}(|00\rangle + |01\rangle \\
&\quad +|10\rangle + |11\rangle) \\
&= \frac{1}{2\sqrt{2}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle \\
&\quad +|100\rangle + |101\rangle + |110\rangle + |111\rangle),
\end{aligned}$$

$$\begin{aligned}
W_3|001\rangle &= (H \otimes W_2)|001\rangle \\
&= H|0\rangle \otimes W_2|01\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{2}(|00\rangle - |01\rangle \\
&\quad +|10\rangle - |11\rangle) \\
&= \frac{1}{2\sqrt{2}}(|000\rangle - |001\rangle + |010\rangle - |011\rangle \\
&\quad +|100\rangle - |101\rangle + |110\rangle - |111\rangle),
\end{aligned}$$

and so on.

The associated matrix is $W_3 = H \otimes W_2$:

$$W_3 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

$$= \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}.$$

If we apply the Walsh-Hadamard transformation to $|00\ldots0\rangle$ we get a superposition of all possible states:

$$\begin{aligned} W_n|00\ldots0\rangle &= (H_2 \otimes H_2 \otimes \ldots \otimes H_2)|00\ldots0\rangle \\ &= \frac{1}{\sqrt{2^n}}((|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \\ &\quad \ldots \otimes (|0\rangle + |1\rangle)) \\ &= \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle. \end{aligned}$$

For many quantum algorithms the state

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle \tag{9}$$

is very convenient to be an "initial state" because it contains an equal weighted distribution of all basis states. An "empty" register can be "set" in the above state by an application of the Walsh-Hadamard transformation. In this way, *using a linear number of operations we can transform one basis state into an exponentially large, equally weighted superposition of all basis states.*

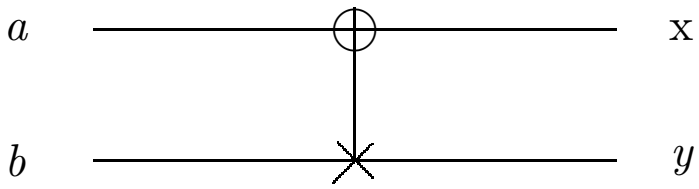
A useful transformation on $\mathbf{C}^2 \otimes \mathbf{C}^2$ is the "controlled-NOT" gate, C_{NOT} defined as follows:

$$C_{\mathrm{NOT}}: \begin{array}{lcl} |00\rangle & \rightarrow & |00\rangle \\ |01\rangle & \rightarrow & |01\rangle \\ |10\rangle & \rightarrow & |11\rangle \\ |11\rangle & \rightarrow & |10\rangle \end{array},$$

$$C_{\mathrm{NOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Given the input state $|ij\rangle$, $i, j \in \{0, 1\}$, the output state produced by $C_{\rm NOT}$ is $|ik\rangle$, where $k = i \oplus j \pmod 2$. The first bit is not disturbed (it is a control bit) and the second one interchanges 0 and 1 iff the first bit is 1, which corresponds to the logical exclusive-OR (XOR).

The controlled-NOT gate $C_{\rm NOT}$ can be represented by a circuit of the form specified in the following Figure.



The controlled-NOT gate.

The oplus indicates the control bit; the opposite symbol indicates the conditional negation of the second bit. If the input states at a and b are in bases states $|0\rangle$ or $|1\rangle$, then the output state at x is the same as the input state at a, and the output state at y is the exclusive-OR of the two input states.

The transformation C_{NOT} is unitary since

$$C_{\mathrm{NOT}}^{\dagger} = C_{\mathrm{NOT}}$$

and

$$C_{\mathrm{NOT}}^{2} = I_4$$

(the 4×4 identity matrix). On the other hand,

> C_{NOT} *cannot be written as a tensor product of two operators.*

Indeed, assume the contrary, and take two operators A, B such that $C_{\mathrm{NOT}} = A \otimes B$. Assume that the associated matrices are

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \; B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

so, the matrix $A \otimes B$ corresponds to C_{NOT} and we have

$$A \otimes B = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{pmatrix}$$

$$= C_{\mathrm{NOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Since $a_{11}b_{11} = 1$ and $a_{11}b_{12} = 0$ it follows that $a_{11} \neq 0$ and $b_{12} = 0$, which is impossible because $a_{22}b_{12} = 1$.

Similarly, one can define the "controlled-controlled-NOT" transformation, CC_{NOT}, operating on three qubits, which negates the rightmost bit if and only if the first two are both 1:

$$CC_{\mathrm{NOT}}: \begin{array}{lcl} |000\rangle & \rightarrow & |000\rangle \\ |001\rangle & \rightarrow & |001\rangle \\ |010\rangle & \rightarrow & |010\rangle \\ |011\rangle & \rightarrow & |011\rangle \\ |100\rangle & \rightarrow & |100\rangle \\ |101\rangle & \rightarrow & |101\rangle \\ |110\rangle & \rightarrow & |111\rangle \\ |111\rangle & \rightarrow & |110\rangle \end{array},$$

$$CC_{\mathrm{NOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

We have $CC_{\mathrm{NOT}} \cdot CC^{\dagger}_{\mathrm{NOT}} = CC^{\dagger}_{\mathrm{NOT}} \cdot CC_{\mathrm{NOT}} = I_8$, hence CC_{NOT} is also unitary.

Quantum states cannot be cloned, as Wooters and Zurek, and Dieks have proved as an application of the linearity of unitary transformations. It is not possible to create the state $(a|0\rangle + b|1\rangle) \otimes (a|0\rangle + b|1\rangle)$ from an *unknown state* $a|0\rangle + b|1\rangle$.

In other words, there is no unitary transformation U such that $U|\varphi 0\rangle = |\varphi\varphi\rangle$ for all quantum states $|\varphi\rangle$.

Indeed, assume the contrary and let $|\varphi\rangle$ and $|\psi\rangle$ be two orthogonal vectors in $\mathbf{C}^2$ and take $|x\rangle = \frac{1}{\sqrt{2}}(|\varphi\rangle + |\psi\rangle)$. Then, $U|\varphi 0\rangle = |\varphi\varphi\rangle$ and $U|\psi 0\rangle = |\psi\psi\rangle$. On the one hand,

$$\begin{aligned} U|x0\rangle &= |xx\rangle \\ &= \frac{1}{\sqrt{2}}(|\varphi\rangle + |\psi\rangle) \otimes \frac{1}{\sqrt{2}}(|\varphi\rangle + |\psi\rangle) \\ &= \frac{1}{2}(|\varphi\varphi\rangle + |\varphi\psi\rangle + |\psi\varphi\rangle + |\psi\psi\rangle). \end{aligned}$$

On the other hand,

$$\begin{aligned} U|x0\rangle &= U(\frac{1}{\sqrt{2}}(|\varphi 0\rangle + |\psi 0\rangle)) \\ &= \frac{1}{\sqrt{2}}(U|\varphi 0\rangle + U|\psi 0\rangle) \\ &= \frac{1}{\sqrt{2}}(|\varphi\varphi\rangle + |\psi\psi\rangle). \end{aligned}$$

Since the vectors φ and ψ are orthogonal, the vectors $|\varphi\varphi\rangle$, $|\varphi\psi\rangle$, $|\psi\varphi\rangle$, $|\psi\psi\rangle$ constitute a basis in $\mathbf{C}^2 \otimes \mathbf{C}^2$ and the vector $|xx\rangle = U|x0\rangle$ has been written in two different ways as a linear combination of this basis vectors, an impossibility.

The no cloning principle states the impossibility of reliably cloning an *unkown* quantum state: it is possible to clone a *known* quantum state. It is possible to obtain n particles in an entangled state $a|00\ldots0\rangle + b|11\ldots\rangle$ from an unknown state $a|0\rangle + b|1\rangle$. Each particles will behave in exactly the same way when measured with respect to the basis $\{|00\ldots0\rangle, |00\ldots01\rangle, \ldots |11\ldots1\rangle\}$, but *not* when measured with respect to other bases. It is not possible to create the n particle state

$$(a|0\rangle + b|1\rangle) \otimes (a|0\rangle + b|1\rangle) \otimes \ldots (a|0\rangle + b|1\rangle)$$

from an unkown state $a|0\rangle + b|1\rangle$.

In a sense, the no cloning principle seems to announce "bad news": we loose one of the most important facilities of classical computation, the unlimited possibility to copy. There are "good news" derived from this principle, for example, the possibility of unconditional secure key generation (see Section 6.2 in Gruska's book). New techniques open possibilites to produce "approximate" copies of qubits: imperfect, but very close to real copies of qubits can be produced with a "quality" no depending upon the qubits to be copied. Of course, there is price to be paid: copies produced in this way are entangled.

The measurement of one or more particles in a quantum system results in a projection of the state of the system prior to measurement onto the subspace of the state space compatible with the measured values. The amplitude of the projection is rescaled to make sure that the resulting state vector has length one. The probability that the result of the measurement is a given value is the sum of the squares of the absolute values of the amplitudes of all components compatible with that value of the measurement.

A simple example of measurement in a two qubit system will illustrate the above points. Let's fix the basis $\{|0\rangle, |1\rangle\}$, and assume that all measurements of individual qubits will be done with respect to this basis. An arbitrary state of a two qubit system can be written as

$$a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle,$$

where a, b, c and d are complex numbers such that

$$|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1.$$

When the first qubit is measured, then the probability that the result is $|0\rangle$ is $|a|^2 + |b|^2$.

Assume now that the measurement gives the first qubit exactly that value, that is, $|0\rangle$. Consequently, the state is projected onto the subspace compatible with the measurement which is the subspace spanned by $|00\rangle$ and $|01\rangle$ and the result of this projection is $a|00\rangle + b|01\rangle$. Renormalizing we get:

$$\frac{1}{\sqrt{|a|^2 + |b|^2}} \cdot (a|00\rangle + b|01\rangle).$$

In general, consider a system containing n qubits ($n \geq 2$). Any state $|x\rangle$ of the system can be expressed as

$$\sum_{i_1,i_2,\ldots,i_n=0,1} c_{i_1 i_2 \ldots i_n} |i_1 i_2 \ldots i_n\rangle,$$

where

$$\sum_{i_1,i_2,\ldots,i_n=0,1} |c_{i_1 i_2 \ldots i_n}|^2 = 1.$$

When the first qubit is measured with respect to the basis $\{|0\rangle, |1\rangle\}$, then the result $|0\rangle$ is obtained with probability

$$P = \sum_{i_2,\ldots i_n=0,1} |0 i_2 \ldots i_n\rangle|^2.^{\text{a}}$$

After rescaling, the new state obtained after the measurement is

$$\frac{1}{\sqrt{\sum_{i_2,\ldots,i_n=0,1} |c_{0 i_2 \ldots i_n}|^2}} \cdot \left(\sum_{i_2,\ldots,i_n=0,1} c_{0 i_2 \ldots i_n} |0 i_2 \ldots i_n\rangle \right).$$

[a] We used the projection onto the space spanned by $\{|0 i_2 \ldots i_n\rangle \,|\, i_k \in \{0,1\}, 2 \leq k \leq n\}$.

Similarly, the measurement gives the outcome $|1\rangle$ with the probability

$$1 - P = \sum_{i_2,\ldots,i_n=0,1} |c_{1i_2\ldots i_n}|^2,$$

and the state changes correspondingly.

What is the price of measurement? According to Landauer

> *If it [measurement] is simply information transfer, that is done all the time inside the computer, and can be done with arbitrarily little dissipation.*

There are many speculations about the "collapse of the wave function (state)" due to an irreversible interaction of the microphysical quantum system with the macroscopic measurement apparatus. Some authors (Greenberg and YaSin or Herzog, Kwiat Weinfuter and Zeilinger) have argued that it is, in fact, possible to reconstruct the state of the physical system before the measurement, that is, to "reverse the collapse of the wave function" if the process of measurement is reversible. After "reconstruction" no information about the measurement is left.

The act of measurement gives another perspective about entangled particles. Particles are not entangled if the measurement of one has no effect on the other. For instance, the state

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

is entangled since the probability that the first bit is measured to be $|0\rangle$ is $1/2$ *if* the second bit has not been measured. However, if the second bit *had been* measured, then the probability that the first bit is measured as $|0\rangle$ is different from $1/2$, it is either 1 or 0, depending on whether the second bit was measured as $|0\rangle$ or $|1\rangle$, respectively. Hence, the probability of measuring the first bit has been changed by the measurement of the second bit.

In contrast, the state

$$\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) = |0\rangle \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

is not entangled. Reason: any measure of the first qubit will produce the result $|0\rangle$ independently whether a measurement is performed or not on the second qubit, and the second qubit has probability $1/2$ to be measured to $|0\rangle$ regardless of whether the first qubit was measured or not.

In a sense, entangled states can be equivalently presented in mathematical terms (they cannot be represented as a tensor product of two states) or in physical terms (the measurement on one affects the other); however, the physical meaning is reacher than the mathematical formalism.

An important consequence of the existence of entangled states is the fact that if a quantum memory register exists in an entangled state, one can change the state of one part of the register simply by measuring another part of it. This is a unique feature of quantum physics[a] which has no parallel in classical physics. *Entanglement is one of the most important features which distinguishes Quantum from conventional Computing.*

[a]Which is crucial in many quantum algorithms, teleportation, information transmission, etc.

How to produce entangled quantum states?

One possibility is to create a source which, by construction, is such that the quantum states emerging already have the indistinguishability feature. For example, consider the decay of a spin-0 particle into two spin-1/2 particles under conservation of the internal angular momentum. The two spins of the emerging particles have to be opposite, so the emerging quantum state is

$$|\psi\rangle_{12} = \frac{1}{\sqrt{2}}(|\uparrow\rangle_1|\downarrow\rangle_2 - |\downarrow\rangle_1|\uparrow\rangle_2),$$

where $|\uparrow\rangle_1$ means particle 1 with spin up.

The above state is rotationally invariant, so the two spins are anti-parallel along whichever direction we choose to measure.

Quantum teleportation

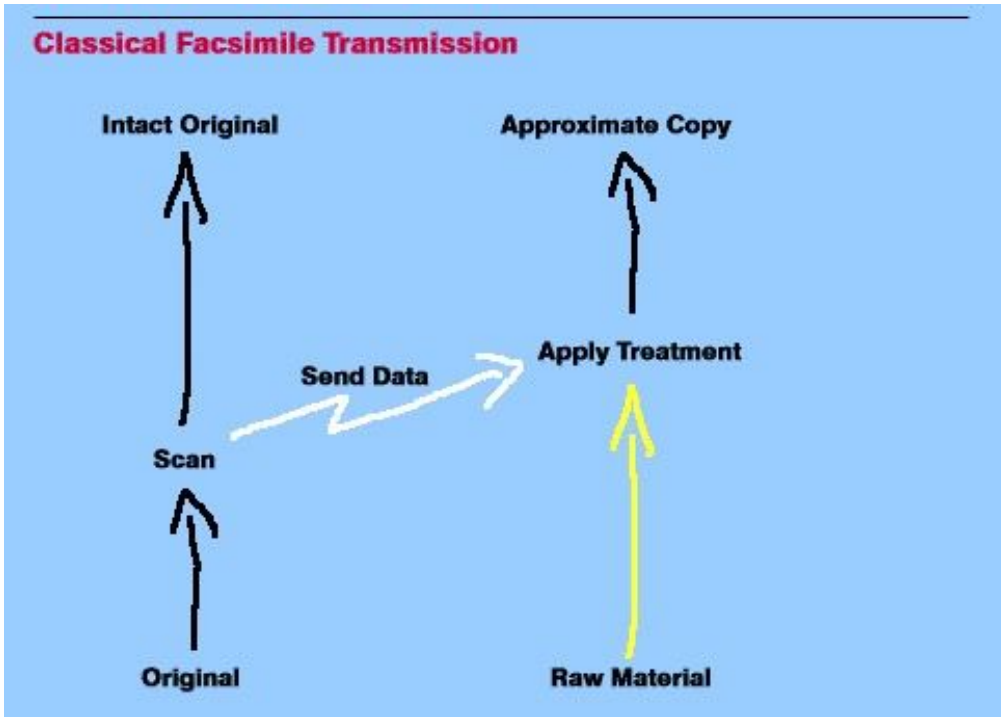
It is possible to transmit qubits without sending qubits!

What does this mean? It's pun? According to Bennett[a]

"It's a mean by which you can take apart an unknown quantum state into classical information and purely quantum information, send them through two separate channels, put them back together, and get back the original quantum state".

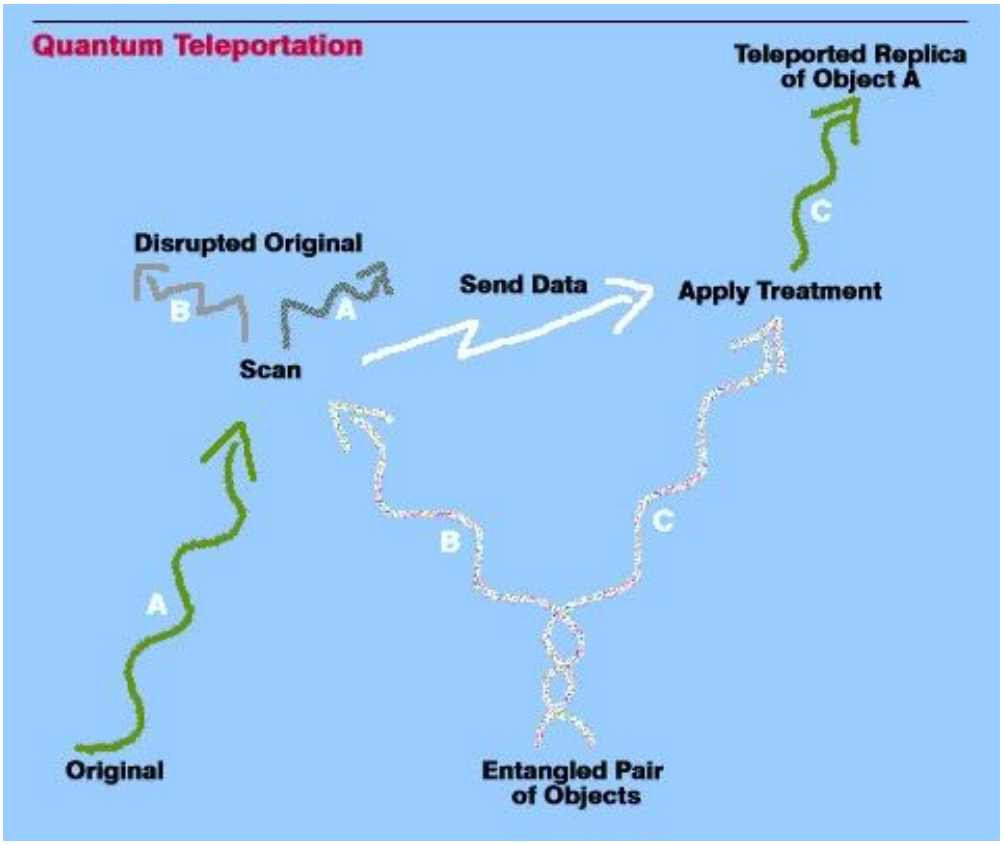
Teleportation, as it is commonly understood, is a fictional procedure of transferring an object from one location to another location in a three stage process: a) dissociation, b) information transmission, c) reconstitution. The point is that, in contrast with fax transmission–where the original object remains intact at the initial location, only an approximate replica is constructed at destination,[b] in teleportation the original object is destroyed after enough information about it has been extracted, the object is not traversing in any way the space between locations, but it is reconstructed, as an exact replica, at the destination.

[a] A co-author of a 1993 paper that proposed quantum teleportation.
[b] At the end, two "identical" versions of the original object result.



Fax transmission

Quantum teleportation allows for the transmission of quantum information to a distant location. The objective is to transmit the quantum state of a particle using classical bits and reconstruct the state at the receiver.



Quantum teleportation

Locality

Locality interaction is

- mediated by another entity (particle, field),
- propagates no faster than light,
- its strength drops off with distance.

All known forces in the universe (electromagnetic, gravitational, strong/weak nuclear) are *local*. So, what's left? The *collapse of the state vector*. Nothing explains, mediates or determines the exact mechanism of the collapse. In particular, the collapse involves *no forces* of any kind.

Let's assume that Alice wishes to communicate with Bob a single qubit in an unknown state $\varphi = a|0\rangle + b|1\rangle$; she wants to make the transmission through classical channels. Alice cannot know with certainty the state as any measurement she may perform may change it; she cannot clone it because of the no cloning result! So, it seems that the only way to send Bob the qubit is to send him the *physical qubit*, or to swap the state into another quantum system and then send Bob that system.

Alice and Bob use an entangled pair

$$\psi_0 = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

Alice controls the first half of the pair and Bob controls the second one. The input state is

$$\begin{aligned}
\varphi \otimes \psi_0 &= (a|0\rangle + b|1\rangle) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
&= \frac{1}{\sqrt{2}}(a|0\rangle \otimes |00\rangle + a|0\rangle \\
&\quad \otimes |11\rangle + b|1\rangle \otimes |00\rangle + b|1\rangle \otimes |11\rangle) \\
&= \frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle).
\end{aligned}$$

Alice now applies the transformation
$(H \otimes I \otimes I) \circ (C_{not} \otimes I)$ to this state. The third bit is left unchanged; only the first two bits belong to Alice and the rightmost one belongs to Bob.

Applying now $H \otimes I \otimes I$, we have:

$$
\begin{aligned}
& (H \otimes I \otimes I) \circ (C_{not} \otimes I)(\varphi \otimes \psi_0) \\
= \quad & \frac{1}{\sqrt{2}} H \otimes I \otimes I(a|000\rangle + a|011\rangle \\
& + b|110\rangle + b|101\rangle) \\
= \quad & \frac{1}{\sqrt{2}}(aH|0\rangle \otimes (I \otimes I)|00\rangle \\
& + aH|0\rangle \otimes (I \otimes I)|11\rangle + \\
& + bH|1\rangle \otimes (I \otimes I)|10\rangle + bH|1\rangle \\
& \otimes (I \otimes I)|01\rangle) \\
= \quad & \frac{1}{\sqrt{2}}(a\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |00\rangle + a\frac{1}{\sqrt{2}}(|0\rangle \\
& + |1\rangle) \otimes |11\rangle + b\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes |10\rangle \\
& + b\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes |01\rangle) \\
= \quad & \frac{1}{2}(a(|000\rangle + |100\rangle + |011\rangle + |111\rangle) + b(|010\rangle \\
& - |110\rangle + |001\rangle - |101\rangle)).
\end{aligned}
$$

This state may be re-written by regrouping terms:

$$
\begin{aligned}
& (H \otimes I \otimes I) \circ (C_{not} \otimes I)(\varphi \otimes \psi_0) \\
= \quad & \frac{1}{2}(|00\rangle(a|0\rangle + b|1\rangle) \\
& + |01\rangle(a|1\rangle + b|0\rangle) + |10\rangle(a|0\rangle - b|1\rangle) \\
& + |11\rangle(a|1\rangle - b|0\rangle)).
\end{aligned}
$$

Alice then measures her two qubits, obtaining four possible results: $|00\rangle$, $|01\rangle$, $|10\rangle$, or $|11\rangle$ with equal probability 1/4. Depending on the result of the measurement, the quantum state of Bob's qubit is projected to $a|0\rangle + b|1\rangle$, $a|1\rangle + b|0\rangle$, $a|0\rangle - b|1\rangle$, $a|1\rangle - b|0\rangle$, respectively. Alice sends the result of her measurement as two classical bits to Bob. He will know what has happened, and can apply the decoding transformation $T \in \{I, X, Y, Z\}$ to fix his qubit.

Received bits	State	Transformation	Result Result
00	$a\|0\rangle + b\|1\rangle$	I	$a\|0\rangle + b\|1\rangle$
01	$a\|1\rangle + b\|0\rangle$	X	$a\|0\rangle + b\|1\rangle$
10	$a\|0\rangle - b\|1\rangle$	Z	$a\|0\rangle + b\|1\rangle$
11	$a\|1\rangle - b\|0\rangle$	Y	$a\|0\rangle + b\|1\rangle$

The final output state is $\varphi = a|0\rangle + b|1\rangle$, which, as desired, is the unknown qubit that Alice wanted to send.

1. The above scheme teleports the "quantum state" not the object.
2. We cannot use the scheme for teleporting an electron, for example; rather we can teleport the "spin" orientation of one electron.
3. The scheme is limited by the classical component.
4. According to S. Braunstein, the current technology would need 100 million centuries to transmit a human body (described down to atomic structure) via a single channel!
5. So, *why teleport a quantum state?* One reason is that this type of communication my be used inside a quantum computer or between quantum computers.

Recently, important teleportation experiments have been performed in Innsbruck (A. Zeilinger), Rome (F. De Martini) and Caltech (J. Kimble). There is a lot of controversy about the nature of quantum teleportation and what criteria should be met by a successful experiment. The following criteria for evaluating a quantum teleportation procedure have been proposed:

- How well can it teleport any arbitrary quantum state it is intended to teleport? (fidelity of teleportation)
- How often does it succeed to teleport, when it is given an input state within the set of states it is designed to teleport? (efficiency of teleportation)
- If given a state the scheme is not intended to teleport, how well does it reject such a state? (cross-talk rejection efficiency)

Let us close this discussion with another controversial statement of the same Bennett:

"I think it's quite clear that anything approximating teleportation of complex living beings, even bacteria, is so far away technologically that it's not really worth thinking about it."

The old "are you sure?"

The EPR Conundrum and Bell's Theorem

According to the philosophical view called realism, *reality* exists and has definite properties irrespective of whether they are observed by some agent. Motivated by this view point, Einstein, Podolsky and Rosen suggested a classical argument to "show" that quantum mechanics is incomplete.

EPR assumed:
(a) the non-existence of action-at-a-distance,
(b) that some of the statistical predictions of quantum mechanics are correct, and
(c) a reasonable criterion defining the existence of "an element of physical reality".

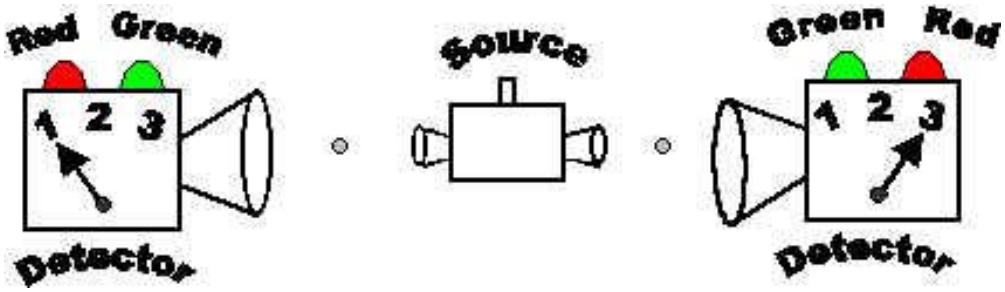
If, without in any way disturbing a system, we can predict with certainty (i.e. with probability equal to unity) the value of a physical quantity, then there exists an element of physical reality corresponding to this physica quantity.

They considered a system of two spatially separated but quantum mechanically correlated particles. A "mysterious" feature appears: By counterfactual reasoning, quantum mechanical experiments yield outcomes which cannot be predicted by quantum theory; hence the quantum mechanical description of the system is incomplete!

One possibility to complete the quantum mechanical description is to postulate additional "hidden-variables" in the hope that completeness, determinism and causality will be thus restored. But then, another conundrum occurs: Using basically the same postulates as those of EPR, Bell showed that no deterministic local hidden-variables theory can reproduce all statistical predictions of quantum mechanics.

Mermin's EPR Device

Mermin's EPR device has three "completely unconnected" parts (there are no relevant connections, neither mechanical nor electromagnetic), two detectors (D1) and (D2) and a source (S) emitting particles. The source is placed between the detectors: whenever a button is pushed on (S), shortly thereafter two particles emerge, moving off toward detectors (D1) and (D2). Each detector has a switch that can be set in one of three possible positions – labelled 1,2,3 – and a bulb that can flash a red (R) or a green (G) light. The purpose of lights is to "communicate" information to the observer. Each detector flashes either red or green whenever a particle reaches it. Because of the lack of any relevant connections between any parts of the device, the link between the emission of particles by (S), i.e. as a result of pressing a button, and the subsequent flashing of detectors, can only be provided by the passage of particles from (S) to (D1) and (D2). Additional tools can be used to check and confirm the lack of any communication.



Mermin's experiment

The device is repeatedly operated as follows:

1. the switch of either detector (D1) and (D2) is set randomly to 1 or 2 or 3, i.e. the settings or states 11, 12, 13, 21, 22, 23, 31, 32, 33 are equally likely,
2. pushing a button on (S) determines the emission toward both (D1) and (D2),
3. sometime later, (D1) and (D2) flash one of their lights, G or R,
4. every run is recorded in the form $ijXY$, meaning that (D1) was set to state i and flashed X and (D2) was set to j and flashed Y.

For example, the record $31GR$ means "(D1) was set to 3 and flashed G and (D2) was set to 1 and flashed R".

Long recorded runs show the following pattern:

(a) For records starting with ii, i.e. 11, 22, 33, both (D1) and (D2) flash the same colours, RR, GG, with equal frequency; RG and GR are never flashed.

(b) For records starting with $ij, i \neq j$, i.e. 12, 13, 21, 23, 31, 32, both (D1) and (D2) flash the same colour only 1/4 of the time (RR and GG come with equal frequencies); the other 3/4 of the time, they flash different colours (RG, GR), occurring again with equal frequencies.

Of course, the above patterns are statistical, that is they are subject to usual fluctuations expected in every statistical prediction: patterns are more and more "visible" as the number of runs becomes larger and larger.

The conundrum posed by the existence of Mermin's device reveals as soon as we notice that the seemingly simplest physical explanation of the pattern (a) is incompatible with pattern (b). Indeed, as (D1) and (D2) are unconnected there is no way for one detector to "know", at any time, the state of the other detector or which colour the other is flashing. Consequently, it seems plausible to assume that the colour flashed by detectors is determined only by some property, or group of properties of particles, say speed, size, shape, etc. What properties determine the colour does not really matter; only the fact that each particle carries a "program" which determines which colour a detector will flash in some state is important.

So, we are led to the following two hypotheses:

H1 *Particles are classified into eight categories:*

$$GGG, GGR, GRG, GRR, RGG, RGR, RRG, RRR.$$[a]

H2 *Two particles produced in a given run carry identical programs.*

According to H1–H2, if particles produced in a run are of type RGR, then both detectors will flash R in states 1 and 3; they will flash G if both are in state 2. Detectors flash the same colours when being in the same states because *particles carry the same programs.*

[a] A particle of type XYZ will cause a detector in state 1 to flash X; a detector in state 2 will flash Y and a detector in state 3 will flash Z.

It is clear that from H1–H2 it follows that *programs carried by particles do not depend in any way on the specific states of detectors*: they are properties of particles not of detectors. Consequently, both particles carry the same program whether or not detectors (D1) and (D2) are in the same states. The emitting source (S) has no knowledge about the states of (D1) and (D2) and there is no communication among any parts of the device.

We are ready to argue that

[L] For each type of particle, *in runs of type* (b) *both detectors flash the same colour at least one third of the time.*

If both particles are of types GGG or RRR, then detectors will flash the same colour all the time. For particles carrying programs containing one colour appearing once and the other colour appearing twice, only in two cases out of six possible combinations both detectors will flash the same light. For example, for particles of type RGR, both detectors will flash R if (D1) is in state 1 and (D2) is in state 3 and vice versa. In all remaining cases detectors will flash different lights. The argument remains the same for all combinations as the conclusion was solely based on the fact that one colour appears once and the other twice. So, the lights are the same one third of the time.

The conundrum reveals as a significant difference appears between the data dictated by particle programs (colours agree at least one third of the time) and the quantum mechanical prediction (colours agree only one quarter of the time):

under H1–H2, *the observed pattern (b) is incompatible with* [L].

Mermin's GHZ Device

Mermin's GHZ device is based on Greenberg, Horne and Zeilinger's version of EPR experiment. The device has a source and three widely separated detectors (A), (B), (C), each of which has only two switch settings, 1 and 2. Any detector, when triggered, flashes red (R) or green (G). Again, detectors are supposed to be far away from the source and there are no connections between the source and detectors (except those induced by a group of particles flying from the source to each detector).

The experiment runs as follows. Each detector is in a randomly chosen state (1 or 2) and then by pressing a button at the source a trio of particles are released towards detectors; each particle will reach a detector and, consequently, each detector will flash a light, green or red.

There are eight possible states, but for the argument we need to take into consideration only those for which the number of 1's is odd, i.e. 111, 122, 212, 221.

(a) If one detector is set to 1 (and the others to 2), then an *odd* number of red lights always flash, i.e.
RRR, RGG, GRG, GGR, and they are equally likely.

(b) If all detectors are set to 1, then an *odd* number of red lights is *never* flashed: GRR, RGR, RRG, GGG.

It is immediate that in case (a) knowing the colour flashed by two detectors, say (A) and (B), *determines uniquely* the colour flashed by the third detector, (C). The explanation can come only because particles are emitted by the same source (there are no connections between detectors). A similar conclusion as in the case of EPR device reveals: *particles carry programs instructing their detectors what colour to flash.* Any particle carries a program of the form XY telling its detector to flash colour X if in state 1 and colour Y if in state 2. There are four types of programs: GG, GR, RG, RR. A run in which programs carried by the trio of particles are of types (RG, GR, GG) will result in RRG if the states were 122, in GGG if the states were 212, and in GRG if the states were 221. This is an *illegal* set of programs as the number of R's is not odd (in RRG, for example).

A *legal* set of programs is (RG, GR, GR) as it produces RRR, GGR, GRG on 122, 212, 221. There are eight legal programs,

$$(RR, RR, RR), (RR, GG, GG), (GG, RR, GG),$$

$$(GG, GG, RR), (RG, GR, GR), (RG, RG, RG),$$

$$(GR, GR, RG), (GR, RG, GR)$$

out of 64 possible programs.

The conundrum reveals again as none of the above programs respects (b), i.e. it is compatible with the case 111. *A single* 111 *run suffices to prove inconsistency!*

Particle programs require an odd number of R*'s to be flashed on* 111, *but quantum mechanics prohibits this in every* 111 *run.*

Bell's Theorem

Bell showed, using basically the same postulates as those of EPR, that no deterministic local hidden-variables theory can reproduce all statistical predictions of quantum mechanics. The setting is the following. We consider two physical systems; on one two types of measurements are made (A, B), and on the other one two other types (C, D). The results are binary, so they will be denoted by "+" and "−". We will repeat these measurements to ensure statistically relevant results. *Correlations* appear when measurements give the same outcome, that is, "++" and "−−". The basic result is that in almost all cases, more "++" and "−−" (and less "+−" and "−−") coincidences are recorded than one can explain by any local classical analysis.

Let $p(x|i)$ be the probability that, by taking the measure $i \in \{A, B\}$ on the first system, the outcome will be $x \in \{+, -\}$; $p(x|ij)$ is the probability that by taking the measure $i \in \{A, B\}$ on the first system *and* the measure $j \in \{C, D\}$ on the second, the outcome of the first system *alone* will be x; $p(xy|ij)$ is the probability that by taking the measure i on the first system and measure j on the second system, the outcomes will be respectively, $x \in \{+, -\}$ and $y \in \{+, -\}$; finally, $p(x|ijy)$ is the probability that when taking the measures $i \in \{A, B\}$ on the first system and $j \in \{C, D\}$ on the second one, and having outcome y on the second, the outcome of the first will be x.

The main result can be stated as follows:

> *If the outcomes of the experiments on both systems are* independent*, that is*
>
> $$p(xy|ij) = p(x|i) \cdot p(y|j),$$
>
> *then the lack of correlation in one of the two types of measures cannot exceed the lack of correlation in the remaining types, that is, the following* quadrangular inequality *holds true:*

$$\begin{aligned} & p(+-|AC) + p(-+|AC) \\ & \qquad \leq \quad p(+-|AD) + p(-+|AD) \\ & \qquad + \quad p(+-|BD) + p(-+|BD) \\ & \qquad + \quad p(+-|BC) + p(-+|BC). \end{aligned} \tag{10}$$

It is remarkable that this inequality can be obtained with just an elementary manipulation of binary variables. To see this, let's denote $p(+|A)$ by a, $p(-|A)$ by $1-a$ (due to the bivalence nature of measurements we have $p(+|A)+p(-|A)=1$), and so on. Using the independence hypothesis, that is,

$$p(+-|AC) = p(+|A) \cdot p(-|C) = a(1-c),$$

and the like, the inequality (10) can be re-written as

$$\begin{aligned} a\,(1-c) + (1-a)\,c \quad \leq \quad & a\,(1-d) + (1-a)\,d + b\,(1-d) \\ & +(1-b)\,d + b\,(1-c) + (1-b)\,c, \end{aligned}$$

or, equivalently,

$$ab + bd + bc \leq ac + b + d,$$

where $a, b, c, d \in [0, 1]$.

To finish we consider the following three cases:

- if $b \leq a$, then $c(b-a) \leq 0$, so $ad+bd+c(b-a) \leq ad+bd$, and (10) follows as $ad \leq d$ and $bd \leq b$;
- if $d \leq c$, then $a(d-c)+bd+bc \leq b+d$, so (10) follows;
- if $a \leq b$ and $c \leq d$, then either $b \leq d$ and in this case $d(a+b)+c(b-a) \leq b+d$, or $d \leq b$ and in this case $a(d-c)+b(d+c) \leq b+d$, and in each case we deduce (10).

The probabilistic hypothesis of independence can actually be decomposed in the conjunction of two hypotheses with more physical significance:

Separability: The statistical outcomes performed on one system are *independent of the outcomes* performed on the other system:

$$p(x|ijy) = p(x|ij) \text{ and } p(y|ijx) = p(y|ij).$$

Locality: The statistical outcomes performed an experiment on one system are *independent of the types of experiments* performed on the other system:

$$p(x|ij) = p(x|i) \text{ and } p(y|ij) = p(y|j).$$

Separability says that the spatio-temporal separation between the two systems makes them reducible to individual parts, the "whole" is no more than the "sum of parts"; locality forbids any instantaneous interaction.

Separability and locality implies independence as

$$p(xy|ij) = p(xy|ijy)\cdot p(y|ij) = p(x|ij)\cdot p(y|ij) = p(x|i)\cdot p(y|j).$$

Consequently, *if the outcomes of the experiments on both systems are* separable *and* local, *then the lack of correlation in one of the two types of measures cannot exceed the lack of correlation in the remaining types.*

Probabilities can be interpreted as truth-values of elementary propositions, so the above analysis can be reformulated in the language of "classical logic". Indeed, let's write A for $p(+|A)$ and $\neg A$ for $p(-|A)$, and similarly for B, C. Further on, let's notice that the elementary operations with probabilities can be reformulated as logical operations, namely, conjunction $\wedge$ will correspond to product, disjunction $\vee$ to sum, and implication $\rightarrow$ to $\leq$.

A "logical" version of the quadrangular inequality can be deduced:

> *If the conjunction is distributive with respect to disjunction for all propositions* $A, \neg A, B, \neg B, C, \neg C$, *that is,*
>
> $$\alpha \wedge (\beta \vee \gamma) \rightarrow (\alpha \wedge \beta) \vee (\alpha \wedge \gamma),$$
>
> *then the following* quadrangular implication *holds true:*

$$\begin{aligned} (A \wedge \neg C) \vee (\neg A \wedge C) \quad &\rightarrow \quad (A \wedge \neg D) \vee (\neg A \wedge D) \\ &\vee \quad (D \wedge \neg B) \vee (\neg D \wedge B) \\ &\vee \quad (B \wedge \neg C) \vee (\neg B \wedge C). \end{aligned}$$

First, use the following weak form of distributivity

$$\alpha \wedge (\neg\beta \vee \beta) \rightarrow (\alpha \wedge \neg\beta) \vee (\alpha \wedge \beta),$$

for $\alpha = X \wedge \neg Y$, and $\beta = Z$:

$$(X \wedge \neg Y) \wedge (\neg Z \vee Z) \rightarrow (X \wedge \neg Y \wedge \neg Z) \vee (X \wedge \neg Y \wedge Z),$$

so by the law of excluded middle we get:

$$(X \wedge \neg Y) \rightarrow (X \wedge \neg Y \wedge \neg Z) \vee (X \wedge \neg Y \wedge Z).$$

Weakening the conclusion we get:

$$(X \wedge \neg Y) \rightarrow (X \wedge \neg Z) \vee (Z \wedge \neg Y). \qquad (11)$$

Using (11) for the triples $(X, Y, Z) = (A, C, D), (D, C, B)$ we get

$$(A \wedge \neg C) \rightarrow (A \wedge \neg D) \vee (D \wedge \neg C),$$

and

$$(D \wedge \neg C) \rightarrow (D \wedge \neg B) \vee (B \wedge \neg C),$$

which imply

$$(A \wedge \neg C) \rightarrow (A \wedge \neg D) \vee (D \wedge \neg B) \vee (B \wedge \neg C).$$

Similarly, we obtain the implication

$$(\neg A \wedge C) \rightarrow (\neg A \wedge D) \vee (\neg D \wedge B) \vee (\neg B \wedge C),$$

which concludes the argument.

Both quadrangular inequality and implication have been experimentally falsified, hence no theory satisfying their hypotheses can be physically correct. So, *locality* and *separability* cannot be simultaneously adopted. Quantum mechanics has chosen to drop separability. The failure of independence affects Reichenbach's *causality* principle: two correlated (non independent) events have a common cause, that there exists an event in their "past" with respect to which they are independent.

So, we arrive at the idea of *synchronicity* that has important implication for Quantum Computation:

> *there exist events which are correlated in a way which is neither casual nor causal.*

Finally, the failure of *distributivity* – the "mark" of quantum logic, has been proved to be more pervasive than the universe of quantum mechanics statements: it is excluded from any logic aiming to describe the physical world. Is any hope to rescue classical logic, which seems to be so brutally excluded ...

A Probabilistic Automaton Simulating Mermin's EPR Device

The states of the automaton are all combinations of states of detectors (D1) and (D2),
$Q = \{11, 12, 13, 21, 22, 23, 31, 32, 33\}$, the input alphabet models the lights, red and green, $\Sigma = \{G, R\}$, the output alphabet captures all combinations of lights flashed by (D1) and (D2), $O = \{GG, GR, RG, RR\}$, and the output function $f : Q \to O$, modeling all combinations of green/red lights flashed by (D1) and (D2) in all their possible states, is probabilistically defined by:

$$
\begin{aligned}
f(ii) &= XX, \text{ with probability } 1/2, \text{ for } i = 1, 2, 3, \\
&\quad X \in \{G, R\}, \\
f(ii) &= XY, \text{ with probability } 0, \text{ for } i = 1, 2, 3, \\
&\quad X, Y \in \{G, R\}, X \neq Y, \\
f(ij) &= XX, \text{ with probability } 1/8, \text{ for } i, j = 1, 2, 3, \\
&\quad i \neq j, X \in \{G, R\}, \\
f(ij) &= XY, \text{ with probability } 3/8, \text{ for } i, j = 1, 2, 3, \\
&\quad i \neq j, X, Y \in \{G, R\}, \\
&\quad X \neq Y.
\end{aligned}
$$

For example, $f(11) = RR$ with probability $1/2$, $f(11) = GR$ with probability 0, $f(11) = RG$ with probability 0, $f(11) = RR$ with probability $1/2$, $f(12) = GG$ with probability $1/8$, $f(12) = GR$ with probability $3/8$, $f(12) = RG$ with probability $3/8$, $f(12) = RR$ with probability $1/8$, etc.

The automaton transition $\delta : Q \times \Sigma \to Q$ is *not specified.* In fact, varying all transition functions δ we get a class of Mermin EPR automata:

$$\mathcal{M}\ (EPR) = (Q, \Sigma, O, \delta, (p_{ij}^{XY}, i, j = 1, 2, 3, X, Y \in \{G, R\})),$$

where $p_{ii}^{XX} = 1/2,\ p_{ii}^{XY} = 0,\ X \neq Y,\ p_{ij}^{XX} = 1/8,\ p_{ij}^{XY} = 3/8,\ X \neq Y.$

Are there two identical, spatially separated, probabilistic automata with identical initial states, whose direct product "simulates" a Mermin's EPR automaton? More formally, are there two probabilistic automata

$$\mathcal{M}_i = (\{1,2,3\}, \{G,R\}, \{G,R\}, \delta_i, (\alpha^X_{i,j}, j = 1,2,3, X \in \{G,R\}))$$

such that their direct product $\mathcal{M}_1 \otimes \mathcal{M}_2$ is isomorphic to a Mermin's automaton $\mathcal{M}$ (EPR), i.e.,

$\delta(ij, X) = \delta_1(i, X)\delta_2(j, X)$, and $p^{XY}_{ij} = \alpha^X_{1,i}\alpha^Y_{2,j}$, for all $j = 1,2,3, X, Y \in \{G,R\}$?

The answer is *negative*. In fact, a stronger result is true:

> *no single state of any Mermin's EPR probabilistic automaton $\mathcal{M}$ (EPR) can be simulated by the product of the corresponding states of any probabilistic automata $\mathcal{M}_i$.*

Indeed, $\alpha^G_{i,j} = 1 - \alpha^R_{i,j}$. For a state ii we get the following contradictory relations:

$$\alpha^G_{1,i}\ \alpha^G_{2,i} = (1 - \alpha^G_{1,i})(1 - \alpha^G_{2,i}) = 1/2,$$

$$\alpha^G_{1,i}\ (1 - \alpha^G_{2,i}) = (1 - \alpha^G_{1,i})\ \alpha^G_{2,i} = 0.$$

For a state kl with $k \neq l$ we, again, get two contradictory relations:

$$\alpha^G_{1,k}\ \alpha^G_{2,l} = (1 - \alpha^G_{1,k})(1 - \alpha^G_{2,l}) = 1/8,$$

$$\alpha^G_{1,k}\ (1 - \alpha^G_{2,l}) = (1 - \alpha^G_{1,k})\ \alpha^G_{2,l} = 3/8.$$

Every Mermin's EPR probabilistic automaton $\mathcal{M}$ (EPR) has strong correlations preventing it from being decomposed as a direct product of two independent probabilistic automata, no matter what transitions and output functions.

Let's turn our attention to Mermin's GHZ device and to this aim consider a probabilistic automaton simulating Mermin's GHZ device. The states of the Mermin's GHZ automaton are all combinations of states of detectors (A), (B) and (C),

$Q = \{111, 112, 121, 122, 211, 212, 221, 222\}$,

the input alphabet models the lights, red and green, $\Sigma = \{G, R\}$, the output alphabet captures all combinations of lights flashed by (A), (B) and (C),

$O = \{GGG, GGR, GRG, GRR, RGG, RGR, RRG, RRR\}$,

and the output function $f : Q \to O$, modeling all combinations of green/red lights flashed by (A), (B) and (C), is determined by the following conditions. (Note that the following conditions do not determine uniquely the output function.)

$$
\begin{aligned}
f(ijk) &= XYZ, \text{ with probability } 1/4, \text{ for} \\
&\quad ijk \in \{122, 212, 221\}, \\
&\quad XYZ \in \{RRR, RGG, GRG, GGR\}, \\
f(ijk) &= XYZ, \text{ with probability } 0, \text{ for} \\
&\quad ijk \in \{122, 212, 221\}, \\
&\quad XYZ \in \{GRR, RGR, RRG, GGG\}, \\
f(111) &= XYZ, \text{ with probability } 0, \text{ for} \\
&\quad XYZ \in \{RRR, RGG, GRG, GGR\}, \\
f(111) &= XYZ, \text{ with probability } 1/4, \text{ for} \\
&\quad XYZ \in \{GRR, RGR, RRG, GGG\}.
\end{aligned}
$$

Again, the transition function $\delta : Q \times \Sigma \rightarrow Q$ is not specified. We get a class of Mermin GHZ automata

$$\mathcal{M}\ (GHZ) = (Q, \Sigma, O, \delta, (p_{ijk}^{XYZ}, i, j, k = 1, 2, X, Y, Z \in \{G, R\})),$$

where

$p_{ijk}^{XYZ} = 1/4$, for $ijk \in \{122, 212, 221\}$, $XYZ \in \{RRR, RGG, GRG, GGR\}$ or $i = j = k = 1$, $XYZ \in \{GRR, RGR, RRG, GGG\}$,

and

$p_{ijk}^{XYZ} = 0$, for $ijk \in \{122, 212, 221\}$, $XYZ \in \{GRR, RGR, RRG, GGG\}$ or $i = j = k = 1$, $XYZ \in \{RRR, RGG, GRG, GGR\}$.

Is there any Mermin's GHZ automaton which can be decomposed into three identical, spatially separated, probabilistic automata with identical initial values? Rephrased, are there three probabilistic automata

$$\mathcal{M}_i = (\{1,2\}, \{G,R\}, \{G,R\}, \delta_i, (\alpha^X_{i,j}, j = 1,2, X \in \{G,R\}))$$

such that their direct product $\mathcal{M}_1 \otimes \mathcal{M}_2 \otimes \mathcal{M}_3$ is isomorphic to a Mermin's automaton $\mathcal{M}$ (GHZ):

$\delta(ijk, XYZ) = \delta_1(i,X)\delta_2(j,Y)\delta_3(k,Z)$ and
$p^{XYZ}_{ijk} = \alpha^X_{1,i}\alpha^Y_{2,j}\alpha^Z_{3,k}$, for all $j = 1,2,\ X,Y \in \{G,R\}$?

The answer is again *negative:*

> *no single state of any Mermin's GHZ probabilistic automaton* $\mathcal{M}$ (GHZ) *can be simulated by the product of the corresponding states of any probabilistic automata* $\mathcal{M}_i$.

We have $\alpha^G_{i,j} = 1 - \alpha^R_{i,j}$. Take the output $XYZ = GGR$. As $p^{GGR}_{111} = 0$ we deduce that

$$\alpha^G_{1,1}\alpha^G_{i2,1}(1 - \alpha^G_{3,1}) = 0,$$

which contradicts the system of equalities

$$p^{GGR}_{122} = p^{GGR}_{212} = p^{GGR}_{221} = 1/4,$$

and the same conclusion can be derived for any output.

Again, due to strong correlations, every Mermin's GHZ probabilistic automaton $\mathcal{M}$ (EPR) cannot be decomposed as a direct product of three independent probabilistic automata, no matter what transitions and output functions.

We continue with an analysis using Meal automata.

First we deal with Mermin EPR device. To this aim we discuss a configuration in which two identical deterministic Mealy automata. (recall that in a Mealy automaton the output function depends both on the current state and input letter.) $\mathcal{M}_1$ and $\mathcal{M}_2$ with unknown but identical initial states are detected in (D1) and (D2), respectively.

More precisely, let us assume that each automaton $\mathcal{M}_j$, $j = 1, 2$, has three states $Q = \{1, 2, 3\}$, the input alphabet $\Sigma = \{1, 2, 3\}$, the output alphabet $O = \{G, R\}$, as well as a(n) (irreversible, i.e., many-to-one) transition function $\delta_j(q, i) = i$ and output function $\lambda_j(q, i) = G$, if $q = i$ and $\lambda_j(q, i) = R$, otherwise; $q \in Q$ and $i \in \Sigma$. Let us further assume that there is an equidistribution of initial states, i.e., each one occurs with equal probability 1/3.

We can construct a joint output function by the Cartesian product $\lambda : Q \times \Sigma \to O \times O$, $\lambda(q,i) = (\lambda_1(q,i), \lambda_2(q,i))$.

Since both $\mathcal{M}_1$ and $\mathcal{M}_2$ are in an identical initial value, there are just three allowed categories GRR, RGR, RRG out of the conceivable ones
$GGG, GGR, GRG, GRR, RGG, RGR, RRG, RRR$.

A straightforward combinatorial argument shows that with these assumptions one obtains the following probabilities:

$$
\begin{aligned}
\lambda(i,i) &= GG, \text{ with probability } 1/3, \text{ for } i = 1,2,3,\\
\lambda(i,i) &= RR, \text{ with probability } 2/3, \text{ for } i = 1,2,3,\\
\lambda(i,i) &= XY, \text{ with probability } 0, \text{ for } i = 1,2,3,\\
&\quad X, Y \in \{G,R\}, X \neq Y,\\
\lambda(i,j) &= GG, \text{ with probability } 0, \text{ for } i,j = 1,2,3, i \neq j,\\
\lambda(i,j) &= GR, \text{ with probability } 1/3, \text{ for } i,j = 1,2,3,\\
&\quad i \neq j,\\
\lambda(i,j) &= RG, \text{ with probability } 1/3, \text{ for } i,j = 1,2,3,\\
&\quad i \neq j,\\
\lambda(i,j) &= RR, \text{ with probability } 1/3, \text{ for } i,j = 1,2,3,\\
&\quad i \neq j.
\end{aligned}
$$

The automata flash the same colour (red) 1/3 of the time and different colours 2/3 of the time. This is not exactly the classical case as discussed by Mermin, but it comes close to it in terms of classicality and locality of the automata arrangement. To understand why, let us define the notion of *correlation function* in the automaton context. Assume again two output symbols, say R and G, and three input symbols, say $1, 2$ and 3.

Associate the numbers $n_t(i, \mathcal{M}_j) = +1$ and $n_t(i, \mathcal{M}_j) = -1$ with the outcomes R and G of the experiment with input i at discrete time t, respectively. In analogy to physical correlation functions we can define a correlation function C as the weighted average over the product of the numbers associated with the outcomes of the first and second automata $\mathcal{M}_1, \mathcal{M}_2$, i.e.,

$$C(i, j) = \tfrac{1}{N} \sum_{t=1}^{N} n_t(i, \mathcal{M}_1) \cdot n_t(j, \mathcal{M}_2).$$

We always get $-1 \leq C(i,j) \leq +1$. In the above case, for identical inputs, $C(i,i) = 1$, $i = 1,2,3$. For nonidentical input $i \neq j$, $C(ij) = -1/3$. The "Bell inequality" is considered a measure for classicality and locality; in particular

$$|C(1,2) - C(1,3)| \leq 1 + C(2,3). \tag{12}$$

is always satisfied for classical systems. The automaton correlation functions always satisfy this inequality and the others obtained by permuting the inputs. This is an indication (although no sufficient condition) that the corresponding classical system behaves locally in the sense used in physics. That is, no causal influence such as a light signal originating from a measurement on one particle can influence the measurement on the other particle and vice versa. This comes as no surprise, because the way the two-automaton setup was conceived, both automata are causally separated in a classical sense.

These results are independent of the particular transition function δ involved, provided it is not a permutation (one-to-one).

An automaton realization which comes close to Mermin's treatment of the GHZ experiment can be given by three identical automata $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ with identical initial value, given by the following table ($q \in Q$, $i \in \Sigma$, $o \in O$):

$q/i, o$	1	2	1	2
1	1	1	R	R
2	1	1	R	G

Here, in configurations like 122, there always occurs an odd number of R's, whereas for 111, only a single result RRR emerges, which has an odd number of R's and is distinct from the quantum mechanical result containing an even number of R's.

Again, the argument is independent of the transition function as long as it is not a permutation.

Deutsch's problem

The simplest way to illustrate the power of quantum computing is to solve the so-called *Deutsch's problem*. Consider a Boolean function $f : \{0,1\} \to \{0,1\}$ and suppose that we have a black box to compute it. We would like to know whether f is constant (that is, $f(0) = f(1)$) or balanced ($f(0) \neq f(1)$). To make this test classically, we need two computations of f, $f(0)$ and $f(1)$ and one comparison. Is it possible to do it better? The answer is affirmative, and here is a possible solution.

Suppose that we have a quantum black box to compute f. Consider the transformation U_f which applies to two qubits, $|x\rangle$ and $|y\rangle$ and produces $|x\rangle|y \oplus f(x)\rangle$.[a] This transformation flips the second qubit if f acting on the first qubit is 1, and does nothing if f acting on the first qubit is 0.

[a] By $\oplus$ we denote, as usual, the sum modulo 2.

The black box is "quantum", so we can chose the input state to be a superposition of $|0\rangle$ and $|1\rangle$. Assume first that the second qubit is initially prepared in the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Then,

$$\begin{aligned} & U_f \left(|x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \right) \\ = \quad & |x\rangle \frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle) \\ = \quad & (-1)^{f(x)} |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned}$$

Next take the first qubit to be $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The black box will produce

$$\begin{aligned} & U_f \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \right) \\ & = \frac{1}{\sqrt{2}}((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ & = \frac{1}{2}(-1)^{f(0)}(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle)(|0\rangle - |1\rangle). \end{aligned}$$

Next will perform a measurement that projects the first qubit onto the basis $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$: we will obtain $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ if the function f is balanced and $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ in the opposite case. So, Deutsch's problem was solved with only one computation of f. The explanation consists in the ability of a quantum computer to be in a blend of states: we can compute $f(0)$ and $f(1)$, but also, and more importantly, we can extract some information about f which tells us whether $f(0)$ is equal or not to $f(1)$.

Can any function $f : \{0,1\} \to \{0,1\}$ be implemented by a quantum gate array U_f? The answer is affirmative. Identifying the values 0 and 1 with the kets $|0\rangle$ respectively $|1\rangle$, U_f may be defined as the linear operator $U_f : \mathbf{C}^4 \to \mathbf{C}^4$, which satisfies, for any $x, y \in \{0,1\}$, the equality

$$U_f|x,y\rangle = |x, y \oplus f(x)\rangle. \tag{13}$$

To compute $f(x)$ we apply U_f to $|x0\rangle$. Graphically, the transformation U_f is presented in the next Figure. We shall argue that

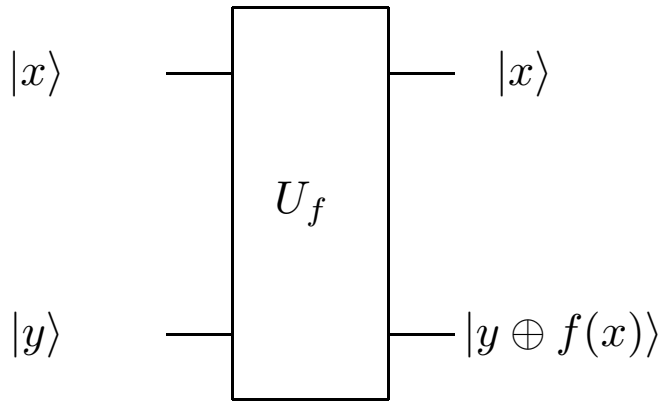
> *for any function $f : \{0,1\} \to \{0,1\}$, U_f is a unitary transformation.*

We have

$$U_f U_f|x,y\rangle = U_f|x, y \oplus f(x)\rangle$$

$$= |x, (y \oplus f(x)) \oplus f(x)\rangle = |x,y\rangle,$$

hence, in view of the equality $U_f U_f = I$, it suffices to prove that $U_f^\dagger = U_f$.



Quantum gate array U_f.

The function f can be defined in four ways: 1. $f(0) = f(1) = 0$, 2. $f(0) = 0$, $f(1) = 1$, 3. $f(0) = 1$, $f(1) = 0$, and 4. $f(0) = f(1) = 1$.

We will investigate the matrix U_f in each situation, taking into account the correspondences:

$$0 \to |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \; 1 \to |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

In the first case, we have $U_f|x, y\rangle = |x, y \oplus 0\rangle = |x, y\rangle$, hence $U_f = I = U_f^\dagger$. In the second case, $U_f|00\rangle = |00\rangle$, $U_f|01\rangle = |01\rangle$, $U_f|10\rangle = |11\rangle$, $U_f|11\rangle = |10\rangle$, so it follows that

$$U_f = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = U_f^\dagger.$$

A direct computation shows that in the third case, $U_f|00\rangle = |01\rangle$, $U_f|01\rangle = |00\rangle$, $U_f|10\rangle = |10\rangle$ and $U_f|11\rangle = |11\rangle$, therefore,

$$U_f = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = U_f^\dagger.$$

Finally, $U_f|x, y\rangle = |x, y \oplus 1\rangle$, i.e., $U_f|x0\rangle = |x1\rangle$ and $U_f|x1\rangle = |x0\rangle$, hence

$$U_f = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = U_f^\dagger.$$

Zeno Machines

A Zeno machine is a Turing machine which computes with "increased speed". Two time scales act simultaneously: the *intrinsic time scale of the process of computation* approaches the infinity in a finite *extrinsic (or proper) time of some outside observer*. As a consequence, certain uncomputable functions (i.e., functions which cannot be computed by any Turing machine) become Zeno computable. For example, the halting problem–the most notorious unsolvable problem in classical computation theory is Zeno solvable.

Zeno machines have been introduced by Weyl. He wrote:

Yet, if the segment of length 1 really consists of infinitely many subsegments of length $1/2, 1/4, 1/8, \ldots$*, as of 'chopped-off' wholes, then it is incompatible with the character of the infinite as the 'incompletable' that Achilles should have been able to traverse them all. If one admits this possibility, then there is no reason why a machine should not be capable of completing an infinite sequence of distinct acts of decision within a finite amount of time; say, by supplying the first result after* $1/2$ *minute, the second after another* $1/4$ *minute, the third* $1/8$ *minute later than the second, etc. In this way it would be possible, provided the receptive power of the brain would function similarly, to achieve a traversal of all natural numbers and thereby a sure yes-or-no decision regarding any existential question about natural numbers!*

Is it kinematically feasible for a machine to carry out an infinite sequence of operations in a finite time?

A possible construction of a Zeno machine starts with a normal Turing machine and considers two time scales, τ and t as follows:

- The *proper time* τ measures the physical system time by clocks in an usual way.
- A discrete cycle time $t = 0, 1, 2, \ldots$ characterizes an "intrinsic" time scale for a process running on the machine.
- For some unspecified reason we assume that the machine allows us to "squeeze" its intrinsic time t with respect to the proper time τ by a geometric progression. For $k < 1$ we let any time cycle of t, if measured in terms of τ, to be "squeezed" by a factor of k with respect to the foregoing time cycle.

More precisely,

$$\tau_0 = 0,\ \tau_1 = k,\ \tau_{t+1} - \tau_t = k(\tau_t - \tau_{t-1}),$$

that is

$$\begin{aligned} \tau_{t+1} - \tau_t &= k(\tau_t - \tau_{t-1}) \\ &= k^2(\tau_{t-1} - \tau_{t-2}) \\ &\dots \\ &= k^t(\tau_1 - \tau_0) \\ &= k^{t+1} \end{aligned}$$

so

$$\tau_{t+1} = k\frac{k^{t+1} - 1}{k - 1}.$$

In the limit when t approaches the infinity, the proper time τ_∞ approaches $k/(1-k)$, so it *remains finite.*

There is no commonly accepted classical physical principle which would, a priori, forbid such a behaviour.[a] One might argue that such an "oracle" would require a geometric energy increase resulting in an infinite consumption of energy. Yet, no currently accepted classical physical principle excludes us from assuming that every geometric decrease in cycle time could be associated with a geometric progression in energy consumption, at least up to some limiting (e.g., Planck) scale.

So, classical physics doesn't forbid the existence of Zeno machines. However, *classical logic does*. A simple diagonalization argument, which mimics the undecidability of the halting problem, shows that Zeno machines are *logically* impossible. Consider an arbitrary algorithm $B(x)$ whose input is a binary string x. Assume, for the sake of a contradiction, that there exists an effective halting algorithm `HALT`, implementable on a Zeno machine, which is able to decide whether B eventually stops on x or not.

[a] Classical mechanics postulates space and time continua as a foundational principle.

Using HALT($B(x)$) we shall construct another Zeno machine $A(B)$, which has as an input a program B and which proceeds as follows: Upon reading the program B as an input, A makes a copy of it.[a]

In the next step, our machine uses the code $\#(B)$ as an input string for B *itself*, that is, A forms $B(\#(B))$, henceforth denoted by $B(B)$. The machine hands $B(B)$ over to its subroutine HALT. Then, $A(B)$ proceeds as follows:

- if HALT($B(B)$) decides that $B(B)$ eventually halts, then A does not halt,[b]
- if HALT($B(B)$) decides that $B(B)$ never halts, then A halts.

[a] This can be readily achieved, since the program B is presented to A in some encoded form $\#(B)$, i.e., as a string of symbols.

[b] This can be realized by an infinite loop.

What about using A on its own code as input? Notice that B is arbitrary, so there is no restriction to prevent us for doing this! Consequently, A, which is representable by its code $\#(A)$ will be applied to itself.

Assume that classically A *is restricted to classical bits* of information. Then, whenever $A(A)$ halts, $\texttt{HALT}(A(A))$ forces $A(A)$ not to halt, and conversely, whenever $A(A)$ does not halt, then $\texttt{HALT}(A(A))$ steers $A(A)$ into the halting state. In both cases one arrives at a contradiction, therefore, Zeno machines are logically inconsistent.

What about the case when *A is allowed a qubit* of information. Assume that $|0\rangle$ and $|1\rangle$ are the halting and nonhalting states, respectively. The computation can be performed if A receives as an input a qubit corresponding to the fixed point state $|\star\rangle$ of the NOT operator:

$$\text{NOT} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

$$\text{NOT}|\star\rangle = |\star\rangle.$$

A simple computation shows that

$$|\star\rangle = |\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\rangle.$$

The qubit solution $|\star\rangle$ proves the impossibility of A to control the output as the probability to reach a halting (nonhalting) state is exactly one half. At the level of probability amplitudes, quantum theory permits Zeno machines, but at the level of observable probabilities, this super-power is nullified, as the result of the computation appears to be random.

Simon's problem

Deutsch and Jozsa (1992) imagined a simple "promise problem" that can be solved "efficiently" without error on a quantum Turing machine, but, classically, one can perform very "inefficiently". Unfortunately, this problem, as well as some other related ones suggested by various authors, **can** be efficiently solved by classical probabilistic Turing machines with exponential small error probability.

In 1994 Simon imagined a simple problem that can be solved in polynomial time on a quantum Turing machine, but cannot be solved in polynomial time on *any* probabilistic Turing machine. Here is Simon's problem:

> Suppose we are given a function $f : \{0,1\}^n \to \{0,1\}^n$and we are promised that either f is one-to-one or there exists a non-trivial n-bit string s such that for all distinct n-bit strings $x,\ x'$ we have
>
> $$f(x) = f(x') \text{ iff } x' = x \oplus s,$$
>
> that is, $f(x) = f(x')$ iff the bits of x and x' differ in exactly those positions where the bits of s are 1.
>
> We wish to determine which of these two conditions holds for f, and, in the second case, to find s.

Let $a = (a_1, \ldots, a_n), b = (b_1, \ldots, b_n)$ be two n-bit strings regarded as n-bit vectors in the $\mathbf{Z}_2^n = \{0,1\}^n$. We say that $a < b$ in case

$$\sum_{i=1}^{n} a_i 2^{i-1} > \sum_{i=1}^{n} b_i 2^{i-1}.$$

The inner product of a, b is

$$a \cdot b = \sum_{i=1}^{n} a_i b_i \pmod 2.$$

A set $B \subset \{0,1\}^n$ is linearly independent if for ever $b \in B$ and every subset $B' \subset B \setminus \{b\}$ we have

$$(0, \ldots, 0) \neq \bigoplus_{b' \in B'} b'.$$

Recall the quantum gate array U_f and the Walsh-Hadamard transformation W:

$$U_f(|x, y\rangle) = |x, f(x) \oplus y\rangle,$$

$$W(|x\rangle) = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle.$$

Simon's solution is the following. Use two quantum registers, both with n qubits and the initial states $|0,\ldots,0\rangle, |0,\ldots,0\rangle$. Then, apply the Walsh-Hadamard transformation on the first register, then apply U_f, then again the Walsh-Hadamard transformation on the first register, and, finally, observe the resulting pair of states to get a pair $(y, f(x))$. More formally, the algorithm can presented in the following form:

$$
\begin{aligned}
& |(0,\ldots,0),(0,\ldots,0)\rangle \\
\xrightarrow{W} \quad & \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x,(0,\ldots,0)\rangle \\
\xrightarrow{U_f} \quad & \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x,f(x)\rangle \\
\xrightarrow{W} \quad & \left(\frac{1}{\sqrt{2^n}}\right)^2 \sum_{x,y\in\{0,1\}^n} (-1)^{x\cdot y}|y,f(x)\rangle \\
& = \frac{1}{2^n} \sum_{x,y\in\{0,1\}^n} (-1)^{x\cdot y}|y,f(x)\rangle.
\end{aligned}
$$

If f is one-to-one, then f is bijective so all possible states $|y, f(x)\rangle$ are *distinct*, so the result of applying the above scheme $n-1$ times consists of $n-1$ pairs $(y_1, f(x_1)), \ldots, (y_{n-1}, f(x_{n-1}))$, uniformly distributed over all pairs $(y, f(x))$.

However, if there is a non-trivial n-bit string s such that $f(x) = f(x')$ iff $x' = x \oplus s$, for all $x \neq x'$, then for each y, x we have

$$|y, f(x)\rangle = |y, f(x \oplus s)\rangle.$$

In this case we have:

$$\begin{aligned}
&\frac{1}{2^n} \sum_{x,y \in \{0,1\}^n} (-1)^{x \cdot y} |y, f(x)\rangle \\
&= \frac{1}{2^{n+1}} \sum_{x,y \in \{0,1\}^n} \left((-1)^{x \cdot y} + (-1)^{(x \oplus s) \cdot y} \right) \\
&\quad |y, f(x)\rangle \\
&= \frac{1}{2^{n+1}} \sum_{x,y \in \{0,1\}^n} (-1)^{x \cdot y} \left(1 + (-1)^{s \cdot y}\right) |y, f(x)\rangle
\end{aligned}$$

Consequently, after $n-1$ independent applications of the scheme we will get $n-1$ independent pairs $(y_1, f(x_1)), \ldots, (y_{n-1}, f(x_{n-1}))$, such that for every $1 \leq i \leq n-1$, $y_i \cdot s \equiv 0 \pmod 2$.

In both cases, after $n-1$ repetitions of the scheme we will get $n-1$ vectors y_i, $i = 1, 2, \ldots n-1$. There are two possibilities according to whether the set $\{y_i \mid 1 \leq i \leq n-1\}$ is linearly independent or not.

In the first case, the linear system of $n-1$ equations $y_i \cdot s = 0$ can be solved in $\mathbf{Z}_2$ to obtain s. There are two cases:

- if f is one-to-one, then the solution s is irrelevant.
- if f is two-to-one, then the solution s is the one required by Simon's problem.

To distinguish between these two cases we need to compute and compare the values of $f(0, \ldots, 0)$ and $f(s)$.

In the second case, that is when the set $\{y_i \mid 1 \leq i \leq n-1\}$ is not linearly independent, we have to repeat the whole process. However, with probability at least $\frac{1}{4}$, the set of vectors $\{y_i \mid 1 \leq i \leq n-1\}$ is linearly independent.[a] So, after an expected $\mathrm{O}(n)$ repetitions, sufficiently many linearly independent vectors y_i will have been collected such that s is uniquely determined.

[a] If u is a non-zero n-bit vector, then by choosing $n-1$ uniformly distributed n-bit vectors y_i, $1 \leq i \leq n$, such that $y_i \cdot u \equiv 0 \pmod 2$ we obtain a linearly independent set of vectors $\{y_i \mid 1 \leq i \leq n\}$ with probability at least $\frac{1}{4}$.

Consequently, the time of the computation is $O(nT_f(n) + G(n))$, where $T_f(n)$ is the time required to compute f on an n-bit string and $G(n)$ is the time required to solve an $n \times n$ linear system of equations in $\mathbf{Z}_2$. Can we do it equally better with a probabilistic Turing machine? The answer is *negative*.

To prove this let construct an oracle, corresponding to a "hard probability distribution", as follows: for each n uniformly generated two bit strings $s(n) \in \{0,1\}^n, b(n) \in \{0,1\}$. If $b(n) = 0$, then the function $f_n : \{0,1\}^n \to \{0,1\}^n$ is uniformly generated from the set of all permutations of $\{0,1\}^n$; if $b(n) = 1$, then f_n is uniformly generated from the set of two-to-one functions such that $f_n(x) = f_n(x \oplus s(n))$, for all n-bit strings x. Then, any probabilistic Turing machine that queries the above oracle no more than $2^{-n/4}$ times cannot correctly guess $b(n)$ with probability greater than $\frac{1}{2} + 2^{-n/2}$. So,

> *any probabilistic Turing machine needs an exponential time to solve Simon's problem on infinitely many inputs.*

A recent paper by Hemaspaandra, Hemaspaandra, Zimand shows that a variant of the Simon's problem that is still solvable in quantum polynomial time needs on a classical machine an exponential time on almost every input.

Simon's quantum algorithm works in polynomial time in the *expected time*[a] and there is no upper bound for the time required in the worst case. Brassard and Høyer improved Simon's algorithm (using Grover's database search algorithm) and showed that Simon's problem can be solved in polynomial time in the worst case. We will follow Mihara and Sung to present a simpler polynomial time algorithm (in the worst case) for Simon's problem.

[a]Recall, we need an expected $O(n)$ repetitions to collect the linearly independent vectors y_i.

First let f be as in Simon's problem and let g be a non-zero n-bit vector different from s. The following quantum algorithm returns an n-bit vector y such that

$$s \cdot y = 0, \text{ and } g \cdot y = 1.$$

The idea is to use Simon's algorithm with U_F instead of U_f, where F is an appropriately constructed function.

To define F we construct two functions

$$\phi_g(x) = \max\{f(x), f(x \oplus g)\},$$

$$\psi_g(x) = \begin{cases} 0, & \text{if } f(x) > f(x \oplus g), \\ 1, & \text{otherwise,} \end{cases}$$

and we put

$$F(x, y) = (-1)^{\psi_g(x)} |x, \phi_g(x) \oplus y\rangle.$$

It is seen that $\phi_g(x) = \phi_g(x \oplus g)$, and $f(x) \neq f(x \oplus g)$ iff $\psi_g(x) \neq \psi_g(x \oplus g)$.

$$
\begin{aligned}
& |(0,\ldots,0),(0,\ldots,0)\rangle \\
\xrightarrow{W} \quad & \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x,(0,\ldots,0)\rangle \\
\xrightarrow{U_F} \quad & \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{\psi_g(x)} |y,\phi_g(x)\rangle \\
\xrightarrow{W} \quad & \left(\frac{1}{\sqrt{2^n}}\right)^2 \sum_{x,y\in\{0,1\}^n} (-1)^{x\cdot y}(-1)^{\psi_g(x)} |y,\phi_g(x)\rangle \\
& = \frac{1}{2^{n+2}} \sum_{x,y\in\{0,1\}^n} (-1)^{x\cdot y}(-1)^{\psi_g(x)}((-1)^{x\cdot y} \\
& \quad + (-1)^{(x\oplus s)\cdot y} - (-1)^{(x\oplus g)\cdot y} \\
& \quad + (-1)^{(x\oplus g\oplus s)\cdot y})|y,\phi_g(x)\rangle \\
& = \frac{1}{2^{n+2}} \sum_{x,y\in\{0,1\}^n} (-1)^{x\cdot y}(1+(-1)^{s\cdot y} \\
& \quad - (-1)^{g\cdot y} - (-1)^{(g\oplus s)\cdot y})|y,\phi_g(x)\rangle \\
& = \frac{1}{2^{n+2}} \sum_{x,y\in\{0,1\}^n} (-1)^{x\cdot y}(1+(-1)^{s\cdot y})(1- \\
& \quad (-1)^{g\cdot y})|y,\phi_g(x)\rangle.
\end{aligned}
$$

Hence, $(1+(-1)^{s\cdot y})(1-(-1)^{g\cdot y}) \neq 0$ iff $s \cdot y = 0$ and $g \cdot y = 1$. So, by measuring the first register we can obtain y such that $s \cdot y = 0$ and $g \cdot y = 1$. The total running time of the algorithm is $O(n + T_f(n))$, where $T_f(n)$ is the time required to compute f on an n-bit string.

We could solve Simon's problem if one could produce enough y's such that $s \cdot y = 0$. The above algorithm shows how produce such an y when we use a different, non-zero g.

We next show that with *certitude* we can find a g for obtaining y. To this aim we need two simple mathematical facts:

1) There exists a polynomial time algorithm that for every linearly independent set $B \subset \{0,1\}^n$, returns a non-zero n-bit string $g \in \{0,1\}^n$ such that $g \cdot y = 0$, for every $y \in B$. If B has exactly $n-1$ elements, then g is unique.

2) Let B be a linearly independent set and $g \in \{0,1\}^n$ such that $g \cdot y = 0$, for every $y \in B$. Then, for every y' such that $g \cdot y' = 1$, the set $B \cup \{y'\}$ is linearly independent.

Using the above facts we can write the following quantum algorithm:

- Put $B = \emptyset$.
- Select a non-zero n-bit vector g.
- Repeat the following steps while $g \neq s$:
- Use the above quantum algorithm to find a non-zero n-bit vector y such that $s \cdot y = 0$ and $g \cdot y = 1$ and put $B = B \cup \{y\}$.
- Use 1) to find a non-zero g' such that $g' \cdot y = 0$, for every $y \in B$ and set $g = g'$.
- Return g.

In view of 2), the set B is linearly independent, therefore we can find a non-zero s such that $s \cdot y = 0$, for all $y \in B$. The running time of the algorithm is $\mathrm{O}(n^2 + nT_f(n) + nG(n))$ in the worst case: here $T_f(n)$ is the time required to compute f on an n-bit string and $G(n)$ is the time required to produce g as in 1), a polynomial in n.

Complexity

A computation on a quantum Turing machine QTM as described by Deutsch can be represented by a tree much in the same way we did it for probabilistic computation. The major change required by quantum mechanics is to replace probabilities with amplitudes. For complexity issues it is enough to consider only real amplitudes in the interval $[-1, 1]$. The amplitude of a node is the product of the amplitudes of the edges on the path from the root to that node. The amplitude of a configuration at any step in the computation is the sum of the amplitudes of all nodes corresponding to that configuration at the level in the tree corresponding to that step.

When an observation is made, the probability associated with each configuration is not the configuration's amplitude in the superposition, but rather the squared magnitude of its amplitude. Hence, the probability of a configuration at any step is the square of its amplitude. For example, the probability of a configuration is the square of the sum of the amplitudes of all leaf nodes corresponding to that configuration. Some specific properties follow. For instance, a particular configuration c may correspond to two leaf nodes with conjugate amplitudes, α and $-\alpha$, and the probability of c being the final configuration will be zero. Still the parent nodes of these two nodes might both have non-zero probabilities.

The computation produces c with probability α^2 if the configuration of one leaf is different from the other. If both leaves have amplitude α, then the probability of c being the final configuration is $4\alpha^2$ (not just $2\alpha^2$). This mutual influence between branches is a consequence of quantum interference.

A quantum computation tree must obey the property that the sum of the probabilities of configurations at any level add up to one. Note that it is not enough to ask that for each node the sum of the squares of the amplitudes on edges leading to its children is one! Computation steps should be unitary, so reversible. A quantum computation results, in just one single step, in a superposition of all its branches tree simultaneously.

A classical probabilistic computation tree has to be well-defined and local, with probabilities adding up to one. A quantum computation tree has to be well-defined, local, and unitary.

Quantum variations of time and complexity classes have been intensively studied. For time complexity, one-tape multitrack QTM are considered; for space complexity, off-line multitape QTM with one-way, read-only, input tape, a working tape, and one-way, write-only, out-tape are used. For time complexity it is enough to consider computations in which the measurement is done only after the machine halts; to study space complexity, a measurement is done each time a symbol is written on the output tape. Many variations of models and approaches have been considered.

There are no essential differences between conventional and quantum computation as concerns the space efficiency. Things are different for time complexity. Quantum versions of classes P and BPP are classes EQP and BQP. The class BQP, which is regarded as the class of languages (problems) that can be decided efficiently on QTMs, is defined as the family of languages L such that there exists a QTM that can decide, with probability at least $2/3$, for each string x whether $x \in L$.

The following basic relations hold true:

$$\mathrm{P} \subseteq \mathrm{EQP} \subseteq \mathrm{BQP},$$

and

$$\mathrm{BPP} \subseteq \mathrm{BQP} \subseteq \mathrm{PP} \subseteq \mathrm{PSPACE}.$$

It is an open problem to decide which inclusion is proper. Quantum complexity classes are intimately related to conventional complexity classes; in particular, showing that QTM's are more powerful than PTMs needs a breakthrough result in classical complexity theory.

Randomness and Quantum Computing

Randomness is at the very heart of quantum physics. When a physical state that is in a superposition of states is measured, then it collapses into one of its possible states in a completely unpredictable way–we can only evaluate the probability of obtaining various possible outcomes. An extreme view is to claim with Peres that

> *in a strict sense quantum theory is a set of rules allowing the computation of probabilities for the outcomes of tests which follow specific preparations.*

According to Milburn, a quantum priciple is

> *physical reality is irreducible random.*

We are talking about "true" randomness, not the "randomness" which, at times, nature appears to exhibit and for which classical physics blames our ignorance: meteorologists cannot predict accurately the path of a hurricane,[a] for example.

[a]The explanation is not difficult to obtain: the equations governing the motion of the atmosphere are nonlinear and tiny errors in the initial conditions can immensily amplify. This behaviour is known as "deterministic chaos".

A mathematical definition of randomness is provided by algorithmic information theory. The idea is to define (algorithmic) randomness as incompressibility. The length of the smallest program (say, for a universal Turing machine[a]) generating a binary string is the *program-size complexity* of the string. This idea can be extended in an appropriate way to infinite sequences. A random string/sequence is incompressible as the smallest program for generating it is the string/sequence itself!

[a]For technical reasons, we use self-delimiting Turing machines, machines having a "prefix-free" domain: no proper extension of a program that eventually halts has that property.

Strings/sequences that can be generated by small programs are deemed to be less random than those requiring longer programs. For example, the digits of $\pi(3.1415926...)$ can be computed one by one; nonetheless, if examined locally, without being aware of their provenance, they appear "random". People have calculated π out to a billion or more digits. A reason for doing this is the question of whether each digit occurs the same number of times, a sympton of randomness. It seems, but remains unproven, that the digits 0 through 9 each occur 10% of the time in a decimal expansion of π. If this turns out to be true, then π would be a so-called simply normal real number. But although π may be random in so far as it's "normal", it is far from (algoritmic) random, because its infinity of digits can be compressed into a concise program for calculating them.

The numbers generated by the so-called logistic map,

$$x_{n+1} = rx_n(1 - x_n), \tag{14}$$

where r is an arbitrary constant and the process starts at some state $x_0 = c$, may appear "random" for some values, say $x_0 = 0.1$ and $r = 3.98$; however, they are not, because of the succintness of the rule (14) describing them. In general, a long string of pseudo-random bits produced by a program may pass all practical statistical tests for randomness, but it is not (algorithmic) random: its program-size complexity is bounded by the size of the generating program plus a few extra bits which specify the random number seed.

Similarly, a long string of binary bits produced by any classical physical system, of which a Turing machine or a Java program is just an instance, is not (algorithmic) random. The program-size complexity of such a string is bounded by the size of the program generating it, that is, the physical law which governs its evolution, plus the size of the initial conditions on which the law acts. Any classical computer can only feign randomness; thinking otherwise is not only wrong, but as von Neumann said,

> *Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.*

Note that human beings are not doing a better job in generating "random" bits as Shannon has argued. Biases observed in people's preferences for popular lottery numbers are manifest.

Are there any physical system that can generate arbitrarily long (algorithmic) random strings?

It is not difficult to distroy randomness. For example, start with a random sequence $x_1x_2 \ldots x_n \ldots$ over the alphabet $\{0, 1\}$ and define a new sequence $y_1y_2 \ldots y_n \ldots$, over the alphabet $\{0, 1, 2\}$, by

$$y_1 = x_1, y_n = x_{n-1} + x_n, \ n \geq 2.$$

Then, the new sequence *is not random.* The motivation is simple: the strings 02 and 20 (and, infinitely many more others) never appear, so the sequence has clear regularities (which can, actually, be detected by simple statistical randomness tests).

It is much more demanding to "generate" a truly random long string starting from an initial state with a simple description. Note that the condition of simplicity of the initial state is *crucial*: starting from a random string one can generate, in a pure algorithmic way, many other random strings. For example, if $x_1 x_2 \dots x_{2n-1} x_{2n}$ is a random binary string, then break the string into pairs and then code 00, 01, 10, 11 by a, b, c, d: the result is again a random sequence. So, the problem is to start from an initial state which can be precisely controlled and has a low program-size complexity and produce measurements of unbounded program-size complexity out its natural dynamical evolution.

Quantum mechanics seems capable to produce, with probability one, truly random strings. Here is a way to do it. Consider the operator

$$R_\theta = \begin{pmatrix} cos\,\theta & -sin\,\theta \\ sin\,\theta & cos\,\theta \end{pmatrix},$$

and recall that it rotates a qubit $a|0\rangle + b|1\rangle$ through an angle θ. In particular, $R_{\frac{\pi}{4}}$ transforms that state $|0\rangle$ into an equally weighted superposition of 0 and 1:

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle. \tag{15}$$

So, to make a quantum device to produce random bits one needs to place a 2-state quantum system in the $|0\rangle$ state, apply the operator $R_{\frac{\pi}{4}}$ to rotate the state into the superposition (15), and the observe the superposition. The act of observation produces the collapse into either $|0\rangle$ or $|1\rangle$, with equal chances. Consequently, one can use the quantum superposition and indeterminism to simulate, with probability one, a "fair" coin toss. Random digits produced with quantum random generators of the type described above are, with probability one, free of subtle correlations that haunt classical pseudo-random number generators.

Of course, the problem of producing algorithmic random strings is still open. Indeed, let's assume that we have a classical silicon computer that simulates, using a high-quality pseudo-random generator, the quantum mechanics dynamics and quantum measurement of a 2-state quantum system. The simulated world will be statistically almost identical (up to some degree) with the "real" quantum system. However, all simulated bits will be, in the long run, highly compressible. How can we be sure that the "real" quantum system is not just a superpowerful pseudo-random generator?

The Merchant's Problem Revised: The Finite Case

We have N stacks of coins and we know that *at most one stack may contain false coins.* We are allowed to take just one coin from each stack and we want to see whether all coins are true or there is a stack of false coins. Can we solve this problem with just one "weighting"?

Assume that a true coin has $\Gamma = 1$ grams and a false coin has $\Gamma + \gamma$ grams ($0 < \gamma < 1$). Consider as quantum space the space $H_N = \mathbf{R}^N$, a real Hilbert space of dimension N. The elements of $\mathbf{R}^N$ are vectors $\mathbf{x} = (x_1, x_2, \ldots, x_N)$. The scalar product of $\mathbf{x}$ and $\mathbf{y}$ is defined by $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{N} x_i y_i$. The norm of the vector $\mathbf{x}$ is defined by $\| \mathbf{x} \| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$. Let $0 < n < N$, and consider $\Omega^n \subset \mathbf{R}^n$. A set $X \subset \mathbf{R}^N$ is called *cylindrical* if $X = \Omega^n \times \mathbf{R}^{N-n}$. Let us denote by μ^k the Lebesgue measure in $\mathbf{R}^k$. If $\Omega^n \subset \mathbf{R}^n$ is measurable, then the cylinder $X = \Omega^n \times \mathbf{R}^{N-n}$ is measurable and $\mu^N(X) = \mu^n(\Omega^n)$.

Next we consider the standard basis $(e_i)_{i=1,N}$ and the projections $\mathbf{P}_i : \mathbf{R}^N \to \mathbf{R}^N$, $\mathbf{P}_i(\mathbf{x}) = (0, 0, \ldots, x_i, 0, \ldots, 0)$. Denote by q_i the weight of a coin in the i-th stack; if the i-th stack contains true coins, then $q_i = \Gamma = 1$, otherwise, $q_i = \Gamma + \gamma = 1 + \gamma$.

Consider the operator $\mathbf{Q} = \sum_{i=1}^{N} q_i \mathbf{P}_i$. For every vector $\mathbf{x} \in \mathbf{R}^N$,

$$\mathbf{Q}(\mathbf{x}) = (q_1 \mathbf{P}_1, \ldots, q_N \mathbf{P}_N)(\mathbf{x}) = (q_1 x_1, \ldots, q_N x_N).$$

The t-th ($t > 1$) iteration of the operator $\mathbf{Q}$ can be used to distinguish the case in which all coins are true from the case in which one stack contains false coins: we construct the quadratic form $\langle \mathbf{Q}^t(\mathbf{x}), \mathbf{x} \rangle$ and consider its dynamics. In case all coins are true $\langle \mathbf{Q}^t(\mathbf{x}), \mathbf{x} \rangle = \| \mathbf{x} \|^2$, for all $\mathbf{x} \in \mathbf{R}^N$; if there are false coins in some stack, for some $\mathbf{x} \in \mathbf{R}^N$, $\langle \mathbf{Q}^t(\mathbf{x}), \mathbf{x} \rangle > \| \mathbf{x} \|^2$, and the value increases with every new iteration. Different operators can be considered, e.g. $\mathbf{Q}(x) = \sum_i^N 2^{(q_i - \Gamma)} \mathbf{P}_i$. To speed-up the computation one can accelerate the iterations of $\mathbf{Q}$, for example by considering the quadratic form $\langle \mathbf{Q}^{2^t}(\mathbf{x}), \mathbf{x} \rangle$ instead of $\langle \mathbf{Q}^t(\mathbf{x}), \mathbf{x} \rangle$.

Now we can introduce a "weighted Lebesgue measure" with proper non-negative continuous density ρ. For example, this can be achieved with the density equal to the Gaussian distribution

$$\rho(\mathbf{x}) = \frac{1}{\pi^{N/2}}\ e^{-\sum_{s=1}^{N} |x_s|^2},$$

a function which will be used in what follows.

We can interpret the measure generated by the density as the probability distribution corresponding to the standard $Normal\,(N; 0, \frac{1}{2}\mathbf{I})$. Hence the probability of the event $\{\mathbf{x} \mid x_1 \in \Omega\}$ is the integral $\text{Prob}(\Omega) = \int_{\Omega \times \mathbf{R}^{N-1}} \rho dm$.

Then, because of the continuity of the density, we deduce that the probability of any "low-dimensional event" is equal to zero. In particular, the event $\{\mathbf{x} \mid x_s = 0\}$ has probability zero, that is, with probability one all components of a randomly chosen normalized vector $\mathbf{x}$ are non-zero.

To solve our problem we assume that time is discrete, $t = 1, 2, \ldots$. The procedure will be *probabilistic:* it will indicate a method to decide, with a probability as close to one as we want, whether there exist any false coins.

Fix a computable real $\eta \in (0, 1)$ as probability threshold. Assume that both η and γ are computable reals. Choose a "test" vector $\mathbf{x} \in \mathbf{R}^N$. Assume that we have a quantum "device" which measures the quadratic form and clicks at time T on $\mathbf{x}$ when

$$\langle \mathbf{Q}^T(\mathbf{x}), \mathbf{x} \rangle > (1 + \varepsilon) \parallel \mathbf{x} \parallel^2 . \tag{16}$$

In this case we say that the quantum "device" has sensitivity ε. In what follows we will assume that $\varepsilon > 0$ is a positive computable real.

Two cases may appear. If for some $T > 0$, $\langle \mathbf{Q}^T(\mathbf{x}), \mathbf{x} \rangle > (1 + \varepsilon) \parallel \mathbf{x} \parallel^2$, then the "device" has clicked and we know for *sure* that there exist false coins in the system.

However, it is possible that at some time $T > 0$ the "device" hasn't (yet?) clicked because $\langle \mathbf{Q}^T(\mathbf{x}), \mathbf{x} \rangle \leq (1 + \varepsilon) \parallel \mathbf{x} \parallel^2$. This may happen because either all coins are true, i.e., $\langle \mathbf{Q}^t(\mathbf{x}), \mathbf{x} \rangle = \parallel \mathbf{x} \parallel^2$, for all $t > 0$, or because at time T the growth of $\langle \mathbf{Q}^T(\mathbf{x}), \mathbf{x} \rangle$ hasn't yet reached the threshold $(1 + \varepsilon) \parallel \mathbf{x} \parallel^2$.

In the first case the "device" will *never* click, so at each stage t the test-vector $\mathbf{x}$ produces "true" information; we can call $\mathbf{x}$ a "true" vector.

In the second case, the test-vector $\mathbf{x}$ is "lying" at time T as we *do* have false coins in the system, but they were not detected at time T; we say that $\mathbf{x}$ produces "false" information at time T.

Hence, the "true" vector has non-zero coordinates corresponding to stacks of false coins (if any); a vector "lying" at time T may have zero or small coordinates corresponding to stacks of false coins. For instance, the null vector produces "false" information at any time. If the system has false coins and they are located in the j-th stack, then each test vector $\mathbf{x}$ whose j-th coordinate is 0 produces "false" information at any time. If the system has false coins and they are located in the j-th stack, $x_j \neq 0$, but

$$\| \mathbf{x} \|^2 + ((1+\gamma)^T - 1)|x_j|^2 \leq (1+\varepsilon) \| \mathbf{x} \|^2,$$

then $\mathbf{x}$ produces "false" information at time T. If $|x_j| \neq 0$, then $\mathbf{x}$ produces "false" information only a finite period of time, that is, only for

$$T \leq \log_{1+\gamma} \left(1 + \frac{\varepsilon \| \mathbf{x} \|^2}{|x_j|^2} \right);$$

after this time the quantum "device" starts clicking.

The major problem is to distinguish between the presence/absence of false coins in the system. We will show how to compute the time T such that when presented a randomly chosen test-vector $\mathbf{x} \in \mathbf{R}^N \setminus \{\mathbf{0}\}$ to a quantum "device" with sensitivity ε that fails to click in time T, then the system doesn't contain false coins with probability larger than $1 - \eta$.

Consider now the *indistinguishable set at time t*

$$\mathcal{F}_{\varepsilon,t} = \{\mathbf{x} \in \mathbf{R}^N \mid \langle \mathbf{Q}^t(\mathbf{x}), \mathbf{x} \rangle \leq (1+\varepsilon) \parallel \mathbf{x} \parallel^2\}.$$

If the system contains only true coins, then $\mathcal{F}_{\varepsilon,t} = \mathbf{R}^N$, for all $\varepsilon > 0, t \geq 1$. If there is one stack (say, the j-th one) containing false coins, then $\mathcal{F}_{\varepsilon,t}$ is a cone $\mathcal{F}_{\varepsilon,t,j}$ centered at the "false" plane $x_j = 0$:

$$((1+\gamma)^t - 1)\,|x_j|^2 \leq \parallel x \parallel^2 .$$

A direct calculation shows that

$$\begin{aligned} &\text{Prob}(\mathcal{F}_{\varepsilon,t}) \\ &= \text{Prob}(\mathcal{F}_{\varepsilon,t,j}) \leq \frac{2M\sqrt{\varepsilon}}{\sqrt{\pi}\sqrt{(1+\gamma)^t - 1}} + \frac{N\sqrt{N}}{M\sqrt{\pi}}\, e^{-\frac{M^2}{N}}. \end{aligned} \quad (17)$$

Selecting

$$M = N^{3/4} \cdot \left(\frac{1+\gamma)^t - 1}{\varepsilon}\right)^{1/4},$$

in (17) we get

$$\text{Prob}(\mathcal{F}_{\varepsilon,t}) \leq \frac{3N^{3/4}\varepsilon^{1/4}}{\sqrt{\pi}((1+\gamma)^t - 1)^{1/4}} \quad (18)$$

hence,

$$\lim_{t\to\infty} \text{Prob}(\mathcal{F}_{\varepsilon,t}) = 0. \quad (19)$$

The above limit is *constructive*, that is, from (18) and every computable $\eta \in (0,1)$ we can construct the computable bound

$$T_\eta = \log_{1+\gamma}\left(\frac{3^4 N^3 \varepsilon}{\eta^4 \pi^2} + 1\right) \tag{20}$$

such that *assuming that the system contains false coins, if* $t \geq T_\eta$, *then we get*

$$\text{Prob}(\mathcal{F}_{\varepsilon,t}) \leq \eta.$$

Recall that we have a finite system of N stacks in which at most one stack contains false coins. So, if we assume that there are $N+1$ equiprobable possibilities, then either all coins are true or only the first stack contains false coins, or only the second stack contains false coins, or only the Nth stack contains false coins. Let us now denote by $\mathcal{N}$ the event "the system contains no false coins" and by $\mathcal{Y}$ the event "the system contains false coins". By $P(\mathcal{N})$ ($P(\mathcal{Y})$) we denote the *a priori* probability that the system contains no false coins (the system contains false coins).

In the simplest case $P(\mathcal{Y}) = \frac{N}{N+1}, P(\mathcal{N}) = 1 - P(\mathcal{Y}) = \frac{1}{N+1}$. We can use Bayes' formula to obtain the *a posteriori probability that the system contains only true coins when at time t the quantum "device" didn't click*:

$$P_{\text{non-click}}(\mathcal{N}) = \frac{P(\mathcal{N})}{P(\mathcal{N}) + (1 - P(\mathcal{N}))\text{Prob}(\mathcal{F}_{\varepsilon,t})}$$

$$\geq 1 - N \cdot \text{Prob}(\mathcal{F}_{\varepsilon,t}).$$

When $t \to \infty$, $\text{Prob}(\Omega_{\varepsilon,t})$ goes to 0, so $P_{\text{non-click}}(\mathcal{N})$ goes to 1. More precisely, if $t \geq T_\eta$, as in (20), then

$$P_{\text{non-click}}(\mathcal{N}) \geq 1 - \eta N.$$

In conclusion,

for every computable $\eta \in (0,1)$ we can construct a computable time T_η such that picking up at random a test-vector $\mathbf{x} \in \mathbf{R}^N \setminus \{\mathbf{0}\}$ and using a quantum "device" with sensitivity ε up to time T_η either

- ◇ *we get a click at some time $t \leq T_\eta$, so the system contains false coins, or*
- ◇ *we don't get a click in time T_η, so with probability greater than $1 - \eta N$ all coins are true.*

The Merchant's Problem: The Infinite Variant

Let us assume that we have now a countable number of stacks, all of them, except at most one, containing true coins only. Can we determine whether there is a stack containing false coins? It is not difficult to recognize that the infinite variant of the Merchant's Problem is equivalent to the Halting Problem: Decide whether an arbitrary program (Turing machine, probabilistic Turing machine, Java program, etc.) eventually halts. This problem is undecidable, i.e., no Turing machine can solve it.

Let's have a program which we want to check whether it stops or not. Given the program and a clock, we make a stack of coins at each tick of the clock. The stack consists of false coins only if the machine was running at the previous tick and stopped at the present tick. Otherwise, the stack of coins is from true coins. This process is, of course, infinite and cannot be completed *classically*. Further on, our quantum algorithm needs the whole, infinite sequence. Do we get a problem? Yes. Can we solve it? Yes, and here is an oversimplified way to explain the solution (for details see the paper). Instead of trying to construct the infinite sequence we pick an infinite random test-vector in the Hilbert space and work with this vector. We fix also the accuracy we want to achieve. We will then compute "classically" the finite time T which will be used by our quantum device to run the experiment on the random test-vector. This is possible because of a similar relation as (19) which says that the set of "lying test-vectors" has constructive measure zero. If the device clicks in time T, then the answer is "the programs stops"; otherwise, the answer is "the program doesn't stop with the pre-given accuracy".

DNA Computing

The DNA computing features:

- DNA has a well-known structure, of a clear syntactic type, already tested by nature and known by genetic engineers;
- DNA is one of the most efficient data storage, with the possibility of encoding a bit at the level of a molecule;
- DNA makes possible a huge parallelism, in a small test tube one can put billions of molecules;
- the biochemical reactions are very efficient from an energetic point of view and they are reversible;
- the biochemical reactions have a high degree of nondeterminism, which, suitably exploited (together with the parallelism), can lead to very efficient computations.

The nondeterminism of biochemical reactions is, at least in this moment, also a *bad* feature, because the result we get is reliable only with a certain probability.

Other drawbacks of DNA Computing are the fact that we have to carefully handle the errors and the many exceptions customary in biochemistry.

Biochemistry inspired Computing (DNA Computing and Membrane Computing) uses specific

- *data structures*,
- *operations* with these data structures.

DNA has a potentially gigantic memory capacity (in reasonable concentrations, a liter of DNA solution can store up to 10^{22} bits of information), and biochemical operations are massively parallel. So DNA has a "built-in" computational power.

The power of DNA computing comes from the memory capacity and parallel processing. If forced to behave sequentially, DNA loses its appeal. For example, let's look at the read and write rate of DNA. In bacteria, DNA can be replicated at a rate of about 500 base pairs a second. Biologically this is quite fast (10 times faster than human cells) and considering the low error rates, an impressive achievement. But this is only 1000 bits/sec, which is a snail's pace when compared to the data throughput of an average hard drive.

But look what happens if you allow many copies of the replication enzymes to work on DNA in parallel. First of all, the replication enzymes can start on the second replicated strand of DNA even before they are finished copying the first one. So already the data rate jumps to 2000 bits/sec. But look what happens after each replication is finished – the number of DNA strands increases exponentially (2^n after n iterations). With each additional strand, the data rate increases by 1000 bits/sec. So after 10 iterations, the DNA is being replicated at a rate of about 1Mbit/sec; after 30 iterations it increases to 1000 Gbits/sec. This is beyond the sustained data rates of the fastest hard drives.

The familiar double helix of DNA arises by the bonding of two separate polymer chains known as DNA strands. These chains, composed of the four DNA bases A (adenine), G (guanine), T (thymine), and C (cytosine), obey the Watson-Crick complementarity rule:

A *bonds with* T and C *bonds with* G.

This restriction means that a DNA chain can pair with another chain only when their sequences of bases are mutually complementary. Thus, fundamental information is available "for free": knowing one member of a bond means automatically knowing the other member.

Complementarity Yields Universality

Consider the set of all possible words (sequences) that can be obtained from two given words by *shuffling* them without changing the order of letters. Then collect all shufflings of all pairs of complementary words into the so-called *twin-shuffle language:* TS. For instance, from AG get the complement TC, and by shuffling produce

AGTC, ATGC, ATCG, TCAG, TAGC, TACG.

All these words belong to TS; AGCT is not in TS.

There is a simple way to go from a DNA double strand to a word in the twin-shuffle language and back.

Universality follows from the fact that any Turing computation can be performed by using an appropriate finite automaton to filter the (fixed) twin-shuffle language:

> *For every computably enumerable language L, we can effectively construct a gsm-mapping g (i.e. a mapping computed by a finite deterministic automaton) such that*
>
> $$L = g(TS).$$

Actually, we can also work with usual strings: by heating a solution which contains DNA molecules, the two strands are separated (one says that DNA is *denaturated*). By cooling the solution, the single strands will again glue together, observing the complementarity of nucleotides and forming double stranded molecules (one says that DNA is *renaturated*; the operation is also called *annealing*).

We have thus already obtained two operations: denaturation and annealing. Another important one is *recombination* (*crossing-over*), the basic one used in Genetic Algorithms. We consider it in the form formalized by Head, in 1987, under the mane of *splicing*,

Consider the following two DNA molecules

CCCCCTCGACCCCC

GGGGGAGCTGGGGG

AAAAAGCGCAAAAA

TTTTTCGCGTTTTT

as well as the restriction enzymes *Taq*I and *Sci*NI, which recognize, respectively, the following patterns

T|C G A G|C G C
A G C|T C G C|G

which indicates the way of cutting the DNA molecules.

When acting on the two molecules mentioned above, these enzymes will produce the following four fragments:

CCCCCT CGACCCCC
GGGGGAGC TGGGGG

AAAAAG CGCAAAAA
TTTTTCGC GTTTTT

We have obtained molecules with identical sticky ends, therefore the four fragments can be bound together, either restoring the initial molecules, or producing new molecules by recombination. The recombination gives the following new molecules

CCCCCTCGCAAAAA
GGGGGAGCGTTTTT

AAAAAGCGACCCCC
TTTTTCGCTGGGGG

This operation (cut at certain places and recombine the fragments which have matching sticky ends) is now used as the basic ingredient of a large class of computing mechanisms, known under the name of *H systems*.

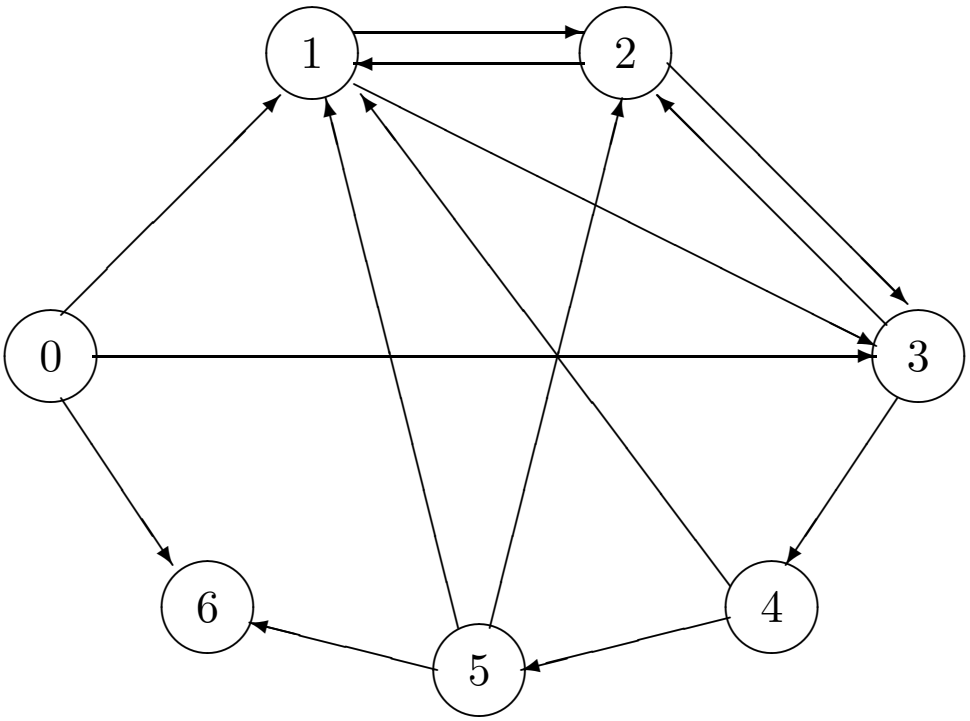
Although the theory of H systems is much developed and in spite of the fact that H systems of various types are capable to perform universal computations (they can simulate any Turing Machine), nothing is known in this moment about the *complexity* of computations performed by H systems (whether or not hard problems can be solved in this framework in an efficient way).

Adleman Experiment

Adleman experiment uses as basic operations the denaturation and annealing (plus filtering DNA molecules according to various criteria, amplification with selective primers, gel electrophoresis and solves in linear time an NP-complete problem – whether or not a given directed graph contains any Hamiltonian path starting in a given node and ending in a given node.

Following the terminology of Hartmanis, this was a convincing *demo*, which has proved that genetic engineering materials and techniques constitute a possible new framework for computability. Adleman's experiment is important not only because it was the first of this type, but also by the way it was conducted.

The graph considered by Adleman is pictured in the next figure. We have seven vertices and fourteen arcs. The question is whether or not there is a path from vertex 0 to vertex 6 which passes exactly once through each of the other vertices.



The graph in Adleman's experiment.

By a simple examination of the graph, we see that there is such a path, namely that following the numbering of the vertices: 0, 1, 2, 3, 4, 5, 6.

However, the problem is a hard one, among the hardest which can be solved in polynomial time by non-deterministic algorithms: it is an NP- complete problem. Otherwise stated, all known deterministic algorithms for solving this problem are essentially equally complex as the exhaustive search.

However, we can trade time for space, and this is exactly what Adleman has done, making use of the massive parallelism of DNA (in some sense, the massive parallelism can simulate non-determi- nism, and in this way non-deterministic algorith -ms can be implemented).

The algorithm used by Adleman was the following:

Input: A directed graph G with n vertices, among which there are two designated vertices v_{in} and v_{out}.

Step 1: Generate paths in G randomly in large quantities.

Step 2: Remove all paths that do not begin with v_{in} or do not end in v_{out}.

Step 3: Remove all paths that do not involve exactly n vertices.

Step 4: For each of the n vertices v, remove all paths that do not involve v.

Output: "Yes" if any path remains, "No" otherwise.

If we assume that each of the operations in the algorithm takes exactly one time unit, then the solution is obtained in a number of time units which is linear in n, the number of vertices in the graph: steps 1, 2, 3 need a constant number of time units (say, 4: generate all paths, select the paths starting with v_{in}, select the paths ending with v_{out}, select the paths of length n), while step 4 takes n time units (check for each vertex its presence in the currently non-rejected paths).

The main difficulty lies in step 1, where we have to generate a large number of paths in the graph, as large as possible, in order to reach with a high enough probability the Hamiltonian paths, if any.

Of course, when the graph also contains cycles, the set of paths is infinite. In a graph without cycles, the set of paths is finite, but it can be of an exponential cardinality with respect to the number of vertices. This is the point where the massive parallelism and the non-determinism of chemical reactions were cleverly used by Adleman in such a way that this step was performed in a time practically independent of the size of the graph.

The biochemical implementation of the above described algorithm was the following.

Each vertex of the graph was encoded by a single stranded sequence of nucleotides, namely of length 20. These codes were constructed at random; the length 20 is enough in order to ensure that the codes are "sufficiently different". A huge number of these oligonucleotides (amplified by PCR) were placed in a test tube. In the same test tube are then added (a huge number of) codes of the graph edges, of the following form: if there is an edge from vertex i to vertex j and the codes of these vertices are $s_i = u_i v_i, s_j = u_j v_j$, where u_i, v_i, u_j, v_j are sequences of length 10, then the edge $i \to j$ is encoded by the Watson-Crick complement of the sequence $v_i u_j$.

For instance, for the codes of vertices 2, 3, 4 specified below

$$\begin{aligned} s_2 &= \text{TATCGGATCGGTATATCCGA}, \\ s_3 &= \text{GCTATTCGAGCTTAAAGCTA}, \\ s_4 &= \text{GGCTAGGTACCAGCATGCTT}, \end{aligned}$$

the edges $2 \to 3, 3 \to 2$ and $3 \to 4$ were encoded by

$$\begin{aligned} e_{2\to 3} &= - \text{CATATAGGCTCGATAAGCTC}, \\ e_{3\to 2} &= - \text{GAATTTCGATATAGCCTAGC}, \\ e_{3\to 4} &= = \text{GAATTTCGATCCGATCCATG}. \end{aligned}$$

By annealing, the codes of the vertices act as splits with respect to codes of edges and longer molecules are obtained, encoding paths in the graph. One can easily see how the molecules specified above will lead to sequences encoding the paths $2 \to 3 \to 4$, $3 \to 2 \to 3 \to 4$, etc.

Adleman has let the process to proceed four hours, in order to be sure that all ligation operations take place. What we obtain is a solution containing a lot of non-Hamiltonian paths (short paths, cycles, paths passing twice through the same vertex). The rest of the procedure consists of checking whether or not at least a molecule exists which encodes a Hamiltonian path which starts in 0 and ends in 6.

A very important point: The difficult step of the computation was carried out "automatically" by the DNA molecules, making use of the parallelism and the Watson–Crick complementarity. In this way, we get a large set of *candidate solutions*, (hopefully) both molecules which encode paths which are looked for, but also many molecules which should be filtered out, a sort of "garbage" to be rejected.

This second part of the procedure, of filtering the result of step 1 in order to see whether a solution to our problem exists, was carried out by Adleman in about seven days of laboratory work, by performing the following operations: By a PCR amplification with primers representing the input and the output vertices (0 and 6), only paths starting in 0 and ending in 6 were preserved.

After that, by gel electrophoresis there have been extracted the molecules of the proper length: 140, because we have 7 vertices encoded by 20 nucleotides each. Thus, at the end of step 3 we have a set of molecules which encode paths in the graph which start in 0, end in 6, and pass through 7 vertices. (It is worth noting that this does not ensure that such a path is Hamiltonian in our graph: 0, 3, 2, 3, 4, 5, 6 is a path which visits seven vertices, but it passes twice through 3 and never through 1.) Roughly speaking, step 4 is performed by repeating for each vertex i the following operations: melt the result of step 3, add the complement of the code s_i of vertex i and let to anneal; remove all molecules which do not anneal.

If any molecule survives step 4, then it encodes a Hamiltonian path in our graph, namely one which starts in vertex 0 and ends in vertex 6.

The practical details of this procedure are not very important for the present discussion. They depend on the present day laboratory possibilities and can be performed also by other techniques; furthermore, the algorithm itself can be changed, improved or completely replaced by another one. What is important here is the

proof that such a computation is possible.

Purely biochemical means were used in order to solve a hard problem, actually an intractable one, in a linear time as the number of lab operations. These operations, in an abstract formulation, are another main output of this experiment and of the thought about it, leading to a sort of programming language based on test tubes and DNA molecules manipulation.

Such a "test tube programming language" was proposed by Lipton in 1995: the well-known SAT Problem, probably the most used NP-complete problem, was solved in linear time by a procedure which extends Adleman's algorithm.

Hartmanis has proved that in order to handle in this manner graphs with 200 nodes, graphs with a size of practical importance, easily handled by conventional computers, we need $3 \cdot 10^{25}$ Kg of DNA, which is more than the weight of the Earth!) In short, Adleman's procedure is elegant, copes in a nice way with the errors, but it cannot be scaled-up.

The Adleman–Lipton paradigm, which has the root in Adleman's experiment, solves combinatorial optimization problems, such as NP-complete problems, using DNA molecules. Each solution candidate of a problem is represented by a DNA molecule. The paradigm consists of the following two steps.

1. Solution candidates are randomly generated.
2. Real solutions are selected from among the generated candidates.

In the first step, the ability of DNA molecules to hybridize with one another through Watson-Crick base-pairing is exploited. In Adleman's experiment, vertices and edges in a directed graph are represented by single-stranded DNA molecules. They hybridize together to form a double-stranded DNA molecule that represents a path in the graph.

In the second step, real solutions are selected by molecular biology operations. Each operation is applied to all the candidates in a test tube. This step is a typical data-parallel computation.

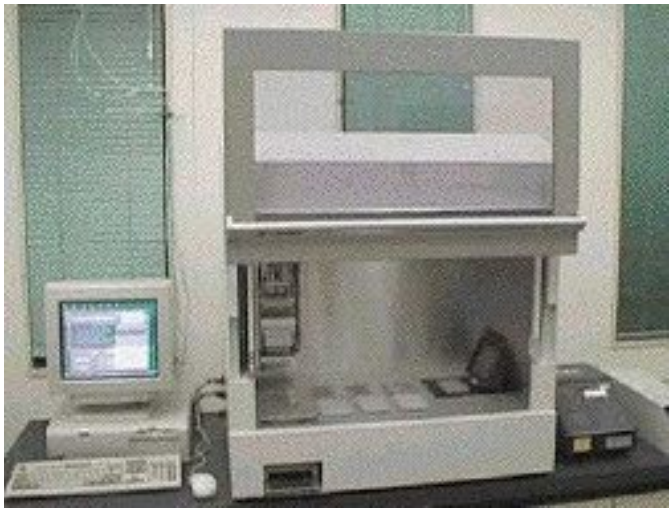
The complexity of these molecular computations can be measured by the required time and the number of necessary molecules. Since the Adleman–Lipton paradigm reduces the time by sacrificing the number of molecules, the size of problems solvable by this method is *limited* by the number of molecules required for computations. Hence, the crucial point of the paradigm is how to *reduce the number of potential candidates.*

Many improvements upon the Adleman–Lipton paradigm have been proposed. Morimoto, Arita and Suyama (1999) have proposed an iterative approach. Rather than generate all of the random candidates prior to selection, the generation and selection of partial solutions is repeated. In this approach, once candidates of partial solutions are identified, those that cannot be extended to a total solution are immediately removed. The remaining partial solutions are then extended, and the process is reapplied to these extended partial solutions.

Using this approach, an algorithm for solving the satisfiability problem for 3-literal clausal formulas (the 3-SAT problem) was designed using DNA molecules. The algorithm does not generate assignments of all the variables at once. Rather it extends partial assignments for one variable at a time. Those partial assignments that cannot be extended to complete satisfying assignments are removed.

The algorithm by Yoshida and Suyama (1999) succeeded in solving a 4-variable 10-clause instance of the 3-SAT problem. Currently, they are developing a DNA computing robot with which they plan to solve a 30-variable 100-clause instance of the problem.

According to Suyama's estimation, solving a 100-variable instance is not impossible. However, even if a 100-variable instance of the 3-SAT problem could be solved, DNA computers would not outperform electronic computers. *Breakthroughs are required both in experimental technologies and in algorithms, before molecular computers based on the Adleman–Lipton paradigm can solve problems that are beyond the capability of electronic computers.*



Suyama robot

Adleman–Lipton Model

A (test) tube is a set of molecules of DNA, i.e. a multi-set of finite strings over the alphabet $\{A, C, G, T\}$. (A multi-set is a classical set in which elements appears with different multiplicities.)

Given a tube, one can perform the following operations:

- **Separate.** Given a tube T and a string S over $\{A, C, G, T\}$, produce two tubes

  $$+(T, S),\ \text{-}(T, S),$$

  where $+(T, S)$ consists all of the molecules of DNA in T which contain the consecutive substring S and $\text{-}(T, S)$ is all of the molecules of DNA in T which do not contain the consecutive substring S.

- **Merge.** Given tubes T_1, T_2, produce

$$\cup(T_1, T_2) = T_1 \cup T_2, .$$

- **Detect.** Given a tube T, say *yes* if T contains at least one DNA molecule, and say *no* if it contains none.
- **Amplify.** Given a tube T produce two tubes $T'(T)$ and $T''(T)$ such that

$$T = T'(T) = T''(T).$$

Example: Two DNA Programs

A Simple Test

(1) Input(T)
(2) T_1 = -(T, C)
(3) T_2 = -(T_1, G)
(4) T_3 = -(T_2, T)
(5) Output(**Detect**(T_3))

On input a tube, it returns *yes* if the tube contains a DNA molecule which is composed entirely of As and otherwise returns *no*.

A Test with Output

On input a tube, the program

(1) Input(T)
(2) $T_1 =$ -(T, C)
(3) $T_2 =$ -(T_1, G)
(4) $T_3 =$ -(T_2, T)
(5) Output(T_3)

returns the tube containing exactly those DNA molecules from the input tube which are composed entirely of As.

3–Colorability Complete Problem

Decide for an undirected graph $G = < V, E >$ whether each vertex can be colored either red, blue, or green in such a way that, after coloring, no two vertices which are connected by an edge have the same color. This is an NP–complete problem.

Given an n vertex graph G with edges $e_1, e_2, \ldots, e_z$, (an edge is a pair $< i, j >$ of vertices) let

$$\Sigma = \{r_1, b_1, g_1, r_2, b_2, g_2, \ldots, r_n, b_n, g_n\},$$

and consider the following program (using only the first three types of operations, i.e. no amplification is used) on input

$$T = \{\alpha \mid \alpha \subset \Sigma,\ \alpha = \{c_1, c_2, \ldots, c_n\},$$

$$\text{for all } 1 \leq i \leq n,\ c_i = r_i \text{ or } c_i = b_i \text{ or } c_i = g_i\} :$$

(1) Input(T)

(2) For $k = 1$ to z do:

(a) $T_{red} = +(T, r_i)$ and ($T_{blue} =$ -(T, r_i) or $T_{green} =$ -(T, r_i)).

(b) ($T_{blue} = +(T_{blue}, b_i)$ or
$T_{blue} = +(T_{green}, b_i)$) and
($T_{green} =$ -(T_{blue}, b_i) or
$T_{green} =$ -(T_{green}, b_i)).

(c) $T_{red}^{good} =$ -(T_{red}, r_j).

(d) $T_{blue}^{good} =$ -(T_{blue}, b_j).

(e) $T_{green}^{good} =$ -(T_{green}, g_j).

(f) $T' = \cup(T_{red}^{good}, T_{blue}^{good})$.

(g) $T = \cup(T_{green}^{good}, T')$.

(3) Output(**Detect**(T)).

The elements in the tube T are in a bijective correspondence with the possible ways to assign colors to the vertices of G. Step (2) creates from T a new tube from which all aggregates α with $c_i = c_j$ have been removed, i.e. we assign the same color to vertex i and j. After $5k$ separations, $2k$ merges and 1 detect, the program will output *yes* if G is 3-colorable and *no* otherwise. This program works in *linear time* in the number of edges. The price paid for this performance consists in the possibility to realize the huge input consisting of no less than 3^n inputs simultaneously.

Several improvements of Adleman's procedure were proposed. The most promising approach is to use an evolutionary computing strategy: instead of constructing a "complete data pool" from which the solutions are then filtered out, only "good" candidates are produced, which saves a lot of the needed DNA. Still, no problem of a practical size was reported to be solved.

Adleman's experiment suggests a general (somewhat unusual) strategy of computing, proposed in G. Păun under the name of *computing by carving*: generate a set of candidate solutions and then "carve" it, removing sets of non-solutions, iteratively, until a solution is obtained. This mode of "working on the complement" proves to be very powerful (even sets which are Turing non-computable can be "computed" in this way) and it is expected to be also useful from the complexity point of view.

New Computational Paradigms

A number of new computational paradigms are emerging. These include:

1. DNA computing
2. molecular computing
3. chemical computing
4. aqueous computing
5. crystal computing
6. cell computing
7. gel computing
8. amorphous computing
9. membrane computing

These computational paradigms can be classified within the context of molecular computing into four kinds of computation:

(1) computing inside a single molecule,
(2) computing by interactions among molecules,
(3) computing with membrane,
(4) computing with geometry.

Note that each group is a source of great computational power, and there are connections between these groups.

Computing Inside a Single Molecule

Hairpin Engines are typical examples of computation inside a single molecule. Protein folding is a typical example of structural formation by a single molecule. Fraenkel was the first to explicitly state the relationship between protein folding and computation. He showed that protein folding is NP-complete by reducing the energy minimization problem of spin glasses, which was known to be NP-complete, into the protein folding problem. This result, however, is purely theoretical and there is a big gap between Fraenkel's model and an actual protein.

Head and Yamamura showed that even write-once memory has high computational power. They proposed Aqueous Computing, in which write-once memory is used to solve NP-complete problems such as the max clique problem. Write-once memory is implemented by a plasmid containing sequences called stations, each corresponding to one bit. A station is surrounded by the recognition sites of the same restriction enzyme, and writing on a station is implemented by removing the station from the plasmid through sequential restriction endonuclease digestion and ligation.