What quantum computing can do for you

What quantum computing can do for you

Inaugural lecture

delivered on the appointment to the
chair of Theoretical Computer Science
at the University of Amsterdam
on Thursday, 20 September 2012

by

Ronald de Wolf

Vossiuspers UvA

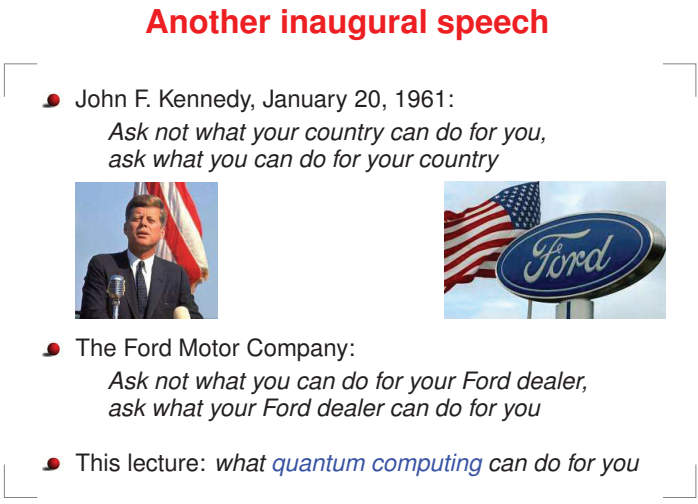
This is inaugural lecture 444, published in this series of the University of Amsterdam.

Lay-out: JAPES, Amsterdam

Mevrouw de Rector Magnificus,
Meneer de Decaan,
Geachte aanwezigen,

I will speak in English, which is probably the most common language that we have here today. Thank you all very much for coming to this lecture, which is to mark the occasion of my appointment as professor at this university for one day in the week, as of March 1 2011. The title of this talk is 'What quantum computing can do for you'. Quantum computing is my field of research and what I will try to do in these 45 minutes is to give a non-technical introduction to this area, with a little bit about my own work near the end.

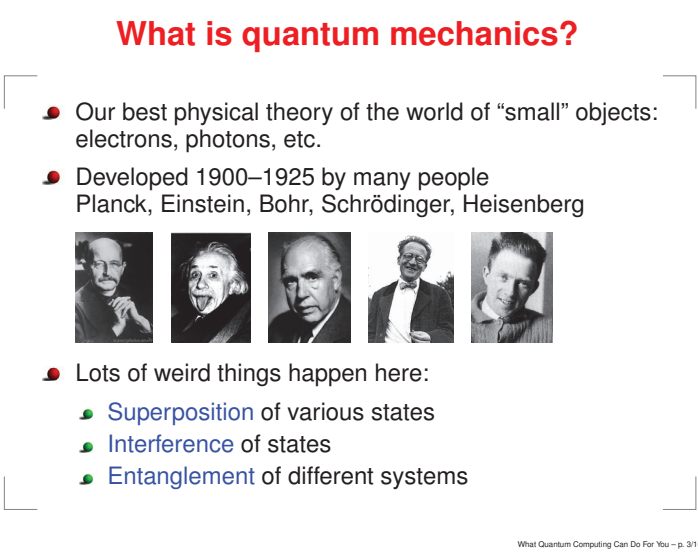
Figure 1



Let me start by explaining where the title comes from. It comes from another inaugural lecture, namely the one of John F. Kennedy in 1961 when he was

elected as president of the United States. He gave a very memorable speech, full of memorable lines, and one of the most memorable was the line 'Ask not what your country can do for you, ask what you can do for your country'. In other words, he was exhorting his fellow countrymen to make sacrifices for their country; to not be selfish, to not take from the country, but to actually give to the country. Very soon afterwards, the Ford Motor Company came up with a very nice advertisement, with the slogan 'Ask not what you can do for your Ford dealer, ask what your Ford dealer can do for you'. So they basically turned Kennedy's line around, offering service instead of asking for sacrifice. This of course is the modern, service-minded attitude that is also being demanded of scientists these days. We should prove our usefulness, our worth. That is really the goal of this lecture: to answer the question what quantum computing can do for you.

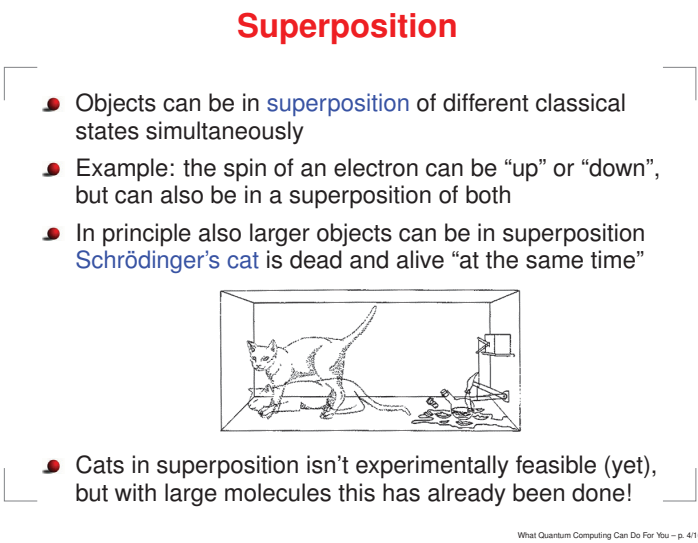
Figure 2



What is quantum computing? It is the use of quantum mechanics for the purposes of computing. In order to be able to talk about that, I will first have to say a little bit about quantum mechanics. Now I am not a physicist, not an expert in physics, far from it, but I think I do understand the basics of quantum mechanics fairly well, and that's basically what I want to explain here. I am not going to use any mathematics, so unavoidably some parts of the expla-

nation will be on the level of metaphor, but I hope you will get something out of it nonetheless. What is quantum mechanics? It is one of our two great physical theories. The other is of course relativity, but I'm not going to talk about that here. Quantum mechanics is in particular our best physical theory when it comes to the world of small objects, and when I mean 'small' I mean really really small, so things like electrons, photons and other elementary particles. Unlike relativity theory, which was basically a one-man show, set up by Albert Einstein between 1905 and 1915, quantum mechanics was developed by many people, roughly in the first quarter of the 20th century. Let me just mention a few of these people. The first was Max Planck who, in the year 1900, solved the problem of blackbody radiation by postulating that energy only comes in certain discrete 'chunks', called 'quanta'. He was very reluctant to make that assumption, he never really reconciled himself with the new quantum theory. He never really liked it, despite the fact that he started it. The second one who contributed was Einstein, who used a similar quantization assumption to explain something called the photoelectric effect. Like Planck, Einstein never really reconciled himself with quantum effects; he basically didn't like the theory and thought it was philosophically unacceptable. We'll return to that topic a little bit later. Then we get to Niels Bohr who in many ways was the central figure of the development of quantum theory. He was a bit older than the people who came after him, and was a bit like a guru of quantum theory; he also pronounced on various philosophical questions coming out of quantum mechanics. He was both an inspiration and a father-figure for the younger people who worked on quantum mechanics. Two of those that I want to mention are Erwin Schrödinger and Werner Heisenberg, both of whom we'll see again later in this talk. The funny thing about quantum mechanics is that many weird effects take place here. Let me mention three buzzwords that I'll explain later on in the talk, at least to some extent. There's the *superposition* principle, so things can exist in a superposition. There's something called *interference*, and there's something called *entanglement*. At this point these three buzzwords don't mean very much, so I'll try to explain them a bit more as we go along.

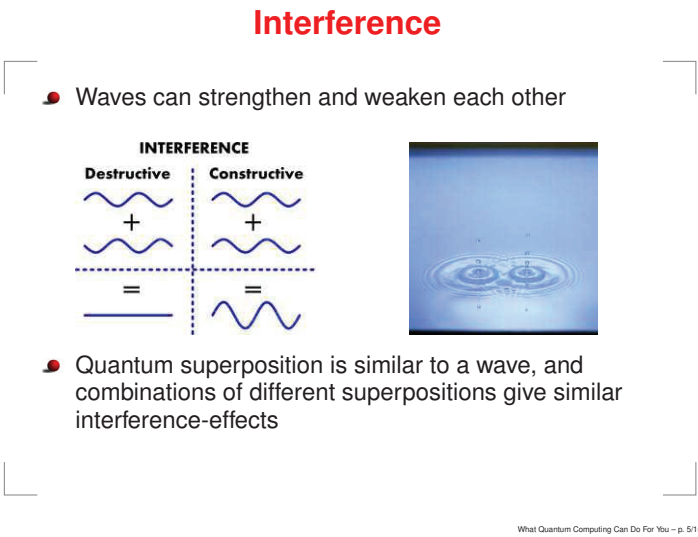
Figure 3



Let me start by explaining the superposition principle. In a way this the principle that's easiest to state, though it's not the principle that's easiest to understand. The idea here is that, while a classical system will have a whole bunch of mutually exclusive possible states, a quantum system can actually be in a superposition of all those states simultaneously. This is of course a metaphor, I am trying to avoid mathematics here, but up to a point it's a reasonable way to think about a superposition as several different alternatives existing side-by-side. So things that would exclude each other in the classical world can exist simultaneously in the quantum world. A good example of this is the ‘spin’ of an electron. An electron is a tiny charged particle. Classically you can think of it as having a certain direction of angular momentum, called ‘spin’, which can be either up or down. In a classical world these two things would exclude each other, the spin of the electron would be either up or down, but in a quantum world it can be both of these things simultaneously: it can be in a superposition of ‘spin up’ and ‘spin down’. We will see this again later. As I already said, quantum mechanics is our best theory of small objects, but there is no fundamental reason why it wouldn't also apply to larger objects. In fact it should: in principle also large objects could be in a superposition. A very famous thought experiment to illustrate this is the Schrödinger cat example. The idea is that we have an extremely well-insulated box. In the box is a cat, initially alive. Also

inside the box is an atom that may or may not decay, in fact it is in a superposition of decaying and not decaying. If the atom decays, it triggers a small flask of poison to break, releasing the poison inside the box. Of course, if the poison is released, the cat dies. In that way a microscopic superposition, namely of the atom decaying or not decaying, can actually be magnified to a macroscopic superposition of a cat that is dead and alive 'at the same time'. Now the funny thing is that as long as nobody touches the box, both the atom, which can decay or not decay, and the cat itself, which can be dead or alive, will live in a superposition. So they live in two paths of the world simultaneously (of course this is still a metaphor): in one path the atom has decayed and the cat is dead, and in the other the atom has not decayed and the cat is alive. Of course, such experiments are just way too big and complicated to do with live cats, but they have actually been done with large molecules, specifically with buckyballs. To some extent such large molecules are still very small objects, but they are already much bigger than elementary particles such as electrons. So quantum mechanics, in principle, applies to all objects, not just the smallest ones; it's just much easier to see the effects of quantum mechanics in tiny objects, because bigger objects tend to interact much more with their environment, it's much harder to isolate a big object. This is basically what I wanted to say about superpositions, and the short summary is that quantum systems can exist in superpositions of different states that would exclude each other in the classical world.

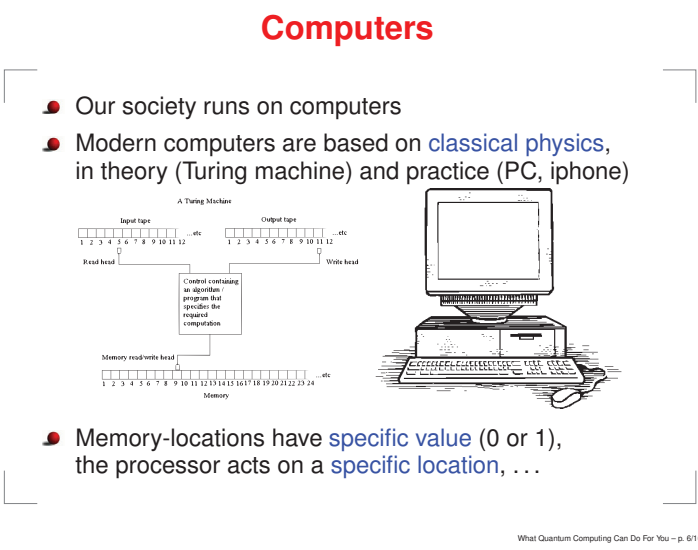
Figure 4



That brings me to an explanation of the second buzzword, which is 'interference'. Again, I think the best way to explain this is not to throw some mathematical formulas at you, but the use a metaphor, namely waves. In fact it's a bit more than a metaphor, because in many ways quantum states do behave like waves. We all know that waves can either strengthen or weaken each other. If you have two waves with opposite phases, so they go up and down at opposite times, and you put those two waves on top of each other, then the result is that the two waves cancel each other out, and the net effect is that there will be no wave left. This is called destructive interference. But if you have two waves with equal phase, so that go up and down at the same time, and you put those on top of each other, then the two waves add to each other, and the result is one stronger wave with the same phase. This is called constructive interference. This may sound a bit abstract, but we have all seen this happening for instance in water. If you have a tranquil water surface and you drop two water drops or two small stones in them, then you will get two concentric waves that interact with each other, giving complicated and beautiful water landscape with patterns of constructive and destructive interference. The funny thing about about quantum superpositions is that in many ways they behave like waves, and that means that if you can combine different superpositions then you can get similar interference effects. The whole trick of quantum comput-

ing, as we will see later, is to somehow steer or engineer these interference effects to work for you. Roughly speaking you want the good stuff to strengthen each other and you want the bad stuff to cancel each other out, so that at the end of the days there is only good stuff left.

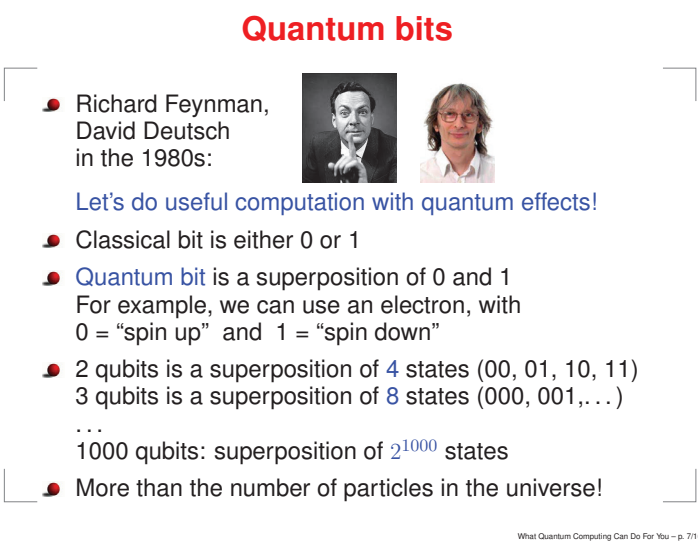
Figure 5



So let's talk about computers. Really I'm a computer scientist, not a physicist – when I'm talking about physics like I did just now, I'm merely an amateur. But computers is something I know a little bit more about, at least in theory. In fact we all know a little more about them because, unlike quantum effects, computers these days are everywhere in daily life. It is no exaggeration to say that our society runs on computers. Now, if you look at modern computers, then these are based on ideas from classical physics: the bits that make up a computer's memory are supposed to be 0 or 1, not both simultaneously, and the instructions that make up a computer's program are supposed to be well-defined rules for taking certain specific bit-values to other specific bit-values. So everything inside a classical computer is supposed to have specific, well-defined properties. This holds for the theoretical model of computers, which is the Turing machine, introduced by Alan Turing in 1936. In fact this year is the Alan Turing year, because he was born 100 years ago. And it also holds for practical computers, like PCs or iPhones. For example if you're storing a pic-

ture in your computer's memory you want all its pixels to have definite values, and not some fuzzy superposition of values, otherwise your picture will be blurred. Before you correct me, I should add a caveat here: in a practical sense actual computers are based on quantum mechanics. If you look at how transistors are built these days, then these are such small objects that they are bound to exhibit some quantum effects. However, for modern chip-designers these quantum effects are actually a nuisance. They try to suppress such effects as much as possible, to ensure that the bits and operations actually do behave as they should. In particular, even when implemented as a tiny object based on some quantum effects, they still want a bit to take either the value 0 or the value 1, and not anything in between.

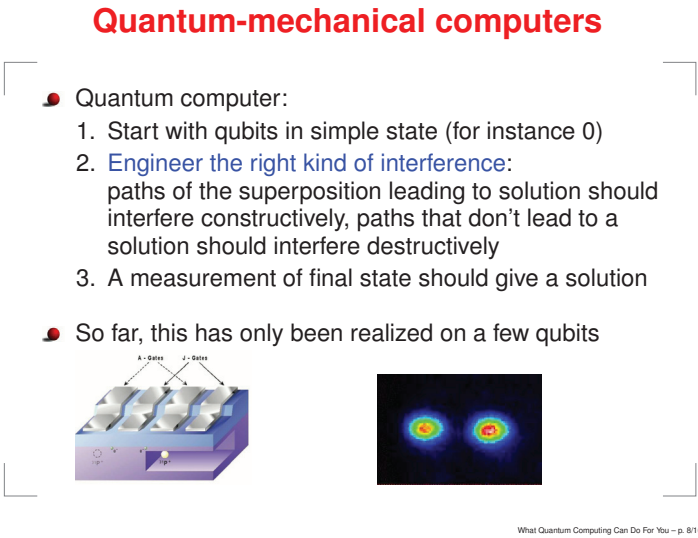
Figure 6



We can however, try to go beyond suppressing these quantum effects to build classical computers, and actually try to use these quantum effects to their full extent, to use them for something stronger, something that would be impossible to do with classical computers. This is the core idea of quantum computing, and was first introduced by Richard Feynman and David Deutsch in the early 1980s. It's actually surprising that it took so long – in a way Turing could already have done this, since quantum mechanics was basically complete in 1925, and when Turing did his work in the 1930s he had some knowledge and

interest in quantum mechanics. However, Turing's perspective was a bit different: for him, a 'computer' was a human being performing calculations, and he wanted to examine the ultimate limits of such a computer, which he quite naturally treated as a classical system. Let me now say a bit more about quantum bits, also called 'qubits'. A classical bit is some object, it doesn't really matter how it is implemented in physics, that takes value either 0 or 1. And if you have a lot of these bits then you have a large computer memory. Quantum bits is where the superposition principle comes in. A quantum bit is something whose state is a superposition of the values 0 and 1. So roughly speaking, a quantum bit can be 0 and 1 simultaneously: the values 0 and 1 would exclude each other in the classical world, but they can exist together in the quantum world, as the two possible basic states of a superposition. If you want to think about a physical implementation of this, you can think of the spin of an electron that I mentioned before: an electron can have a spin up or a spin down, and it can in fact have any superposition of those two values. This is in fact one of the most common ways to experimentally realize quantum bits in a physical system. One qubit takes you only so far. Things only become really interesting if you have many qubits, because the number of basis states in the superposition goes up very fast, namely exponentially with the number of qubits. One qubit has two basic states, 0 and 1; two qubits together have 4 basic states, 00, 01, 10, 11; three qubits have 8 possible basic states, and so on. Every qubit that you add, doubles the number of possible basis states. And if you have a system of a 1000 qubits, it can be in a superposition of 2^1000 basic states, which is a truly colossal number: much more than the number of particles in the universe. This enormous state space is one of the reasons for the power of quantum computers. It is certainly not the only reason. We cannot just do whatever we want in this enormous state space; quantum mechanics very much constrains what we can do. Nevertheless, the fact that we have all this room available is one of the key aspects of quantum computers.

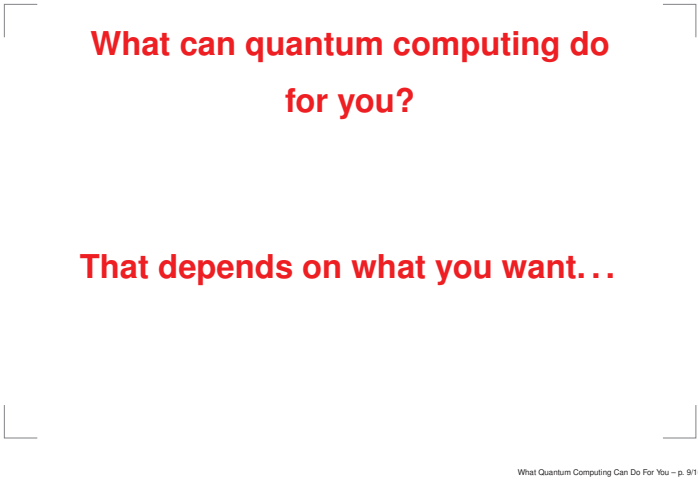
Figure 7



Let us go from qubits, which are the basic objects on which a quantum computer can act, to those quantum computers themselves. A high-level way to describe a quantum computer is as a three-step process. First, you start with some qubits that have been prepared in a very simple state, for instance they could all initially have the classical value 0. Second, the trick (and I'm hiding an enormous amount of technical complication here) is then to engineer the right kind of interference. Quantum mechanics is very constraining in the kinds of interference it allows. The intuition you should have here is that we have this enormous, exponential number of paths in a superposition. Some of them will lead to good outcomes of the computation and some of them to bad outcomes, and the trick is to somehow have constructive interference so that the good outcomes strengthen each other, and to have destructive interference so that the bad outcomes cancel each other out. This is certainly not always possible. In fact the number of computational problems where something like this really gives benefits over classical computers is quite limited; we will see some examples later on in this lecture. Then thirdly, after having engineered the right kind of interference, hopefully at the end of the computation you will have a superposition left where the good computational paths, the ones leading to the correct outcome, have most of the weight. At that point, if you just try to measure this final superposition you will probably see a solution to your

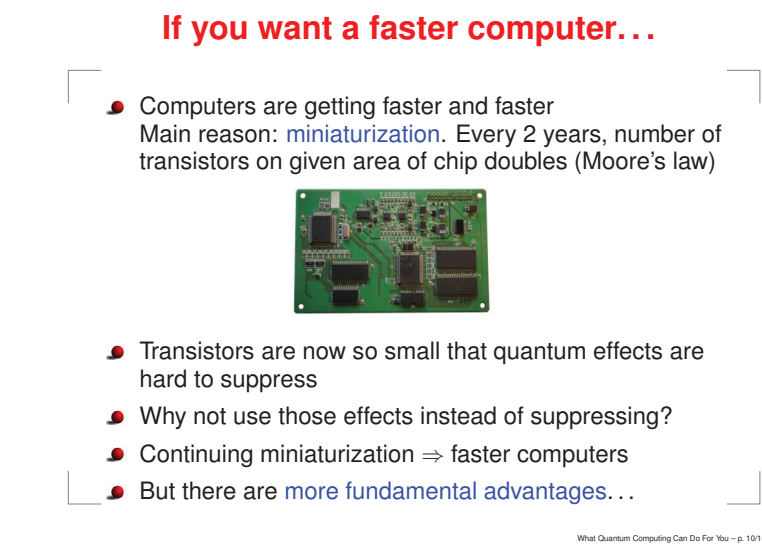
computational problem. In steps 2 and 3 there are actually some deep mysteries hidden, both mathematical and philosophical, that I won't get into here. Of course everybody interested in quantum computers always asks to what extent these things have actually already been built. A lot of different physical systems have already been tried for building qubits, for example an electron attached to a phosphorus atom embedded in silicon, or ions caught in electromagnetic traps, or tiny superconducting circuits where a current can flow in one of two directions. So individual qubits can definitely be built. However, so far only very small quantum computers have been built, acting on only a few quantum bits. The reason is that this is an extremely hard engineering problem. It is extremely hard to manipulate these tiny particles (photons, electrons) very precisely and at the same time to isolate them from their environment to prevent too much noise coming into the system. There doesn't seem to be a fundamental reason why this cannot be done, but in practice it's a very hard problem.

Figure 8



This brings me back to the main question of this lecture: what can quantum computing do for you. The answer of course depends on who you are and on what you want. As we will see, lots of different groups can benefit from quantum computing: thieves, people who have something to hide, people who have lost something, philosophers, mathematicians. The next few slides I will go over different examples of what quantum computing can do for you.

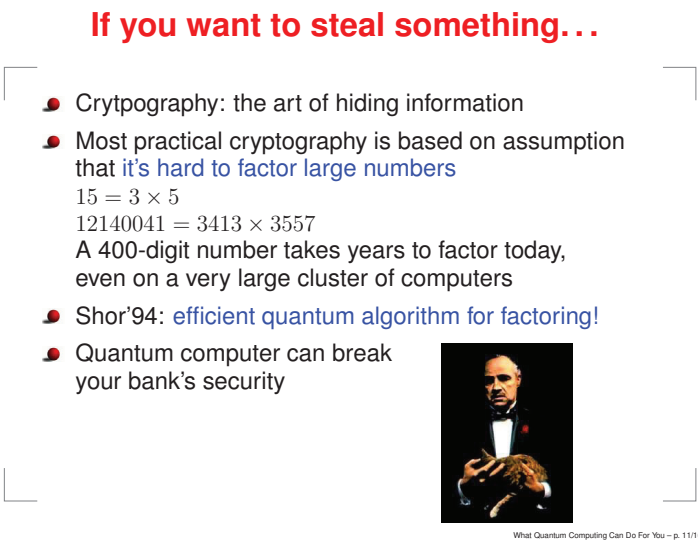
Figure 9



First of all, when you read about quantum computers in the press, they are just presented as these superfast computers. I think this is very misleading, but let me briefly talk about it anyway. Suppose you want a faster computer. Until now, basically the only thing you had to do was to wait a bit. Computers are getting faster and faster at an amazing rate. The main reason this is happening is miniaturization: things are getting smaller and smaller. In the 1960s Gordon Moore, one of the founders of Intel, posited what is now called Moore's law. This says that every two years the number of transistors that you can put on a fixed area of chip, doubles. Transistors are the basic objects from which computers are made, you can think of them as basic switches. Saying that you can put more and more of these on the same area is of course is the same as saying that these transistors themselves are shrinking by a constant factor every two years. This is an amazing feat of engineering, and more and more costly too: whenever Intel builds a new factory for the next generation of chips, it's several times more expensive than the previous factory. Every new chips-factory costs billions to set up. So far, people have indeed been sustaining Moore's law, and computers are still getting faster and faster. However, the problem here is that we have now miniaturized things so far, that current-day transistors are really very tiny objects, so tiny that quantum effects have become very hard to suppress. As I said before, quantum effects are already present in today's compu-

ters, but they are being suppressed as much as possible in order to have classical bits with well-defined properties. People are working with quantum objects, but trying to make them behave as classical as possible. This is getting harder and harder, because the smaller you go, the more dominant and harder to suppress the quantum effects become. A very obvious question, certainly in retrospect, is: why not try to work with those quantum effects. Instead of suppressing them to make systems behave as classically as possible, why not try to benefit from them? As I already mentioned, this is the idea of the whole area of quantum computing. One obvious benefit of this is that you may be able to continue the ongoing miniaturization for a bit longer, sustaining Moore's law for a few more iterations. This is true, but from a fundamental perspective this is not really the interesting thing. More interesting is the fundamental advantage that you can get by using these quantum effects – superposition, interference, entanglement – to do something useful, in fact to do things that are impossible to do with classical bits and classical operations. So faster computer are a somewhat superficial potential benefit of quantum computers, but much more interesting would be to figure out things that can be done fundamentally better if we use quantum effects.

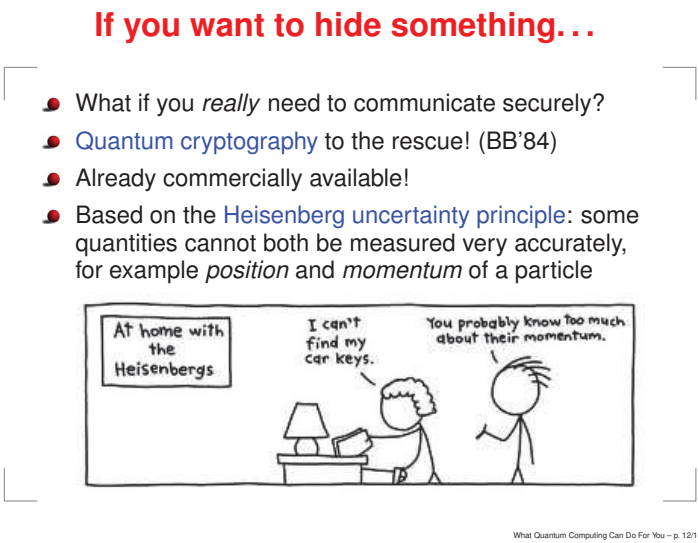
Figure 10



One thing you can do fundamentally better with a quantum computer is to steal things. Nowadays if you want to steal something, you could try to pick somebody's wallet, and you might gain 50 Euros or so. But if you want to gain much more, you could try to break something on the internet, because that is where the big money is these days, in electronic transactions. Of course there are people trying to prevent this, in fact there is a whole field of research called 'cryptography' that is trying to find ways to hide and protect information. This is used a lot: if you buy something from Amazon or eBay, cryptography is used to protect your transaction, to prevent outside parties from messing with your transaction or stealing your money. Most practical cryptography in use today is based on a seemingly esoteric assumption, namely that it is hard to factor large integers into their prime factors. Every number can be written uniquely as a multiplication of prime numbers (a number is prime if its only divisors are 1 and the number itself). For example 15 can be written as the product of 3 and 5. That one was easy. It gets harder if the number you want to factor gets larger. For example what are the prime factors of the 8-digit number 12140041? This takes a bit longer, but still with a bit of work you can figure out that its factors are 3413 and 3557. The real trouble comes when you want to factor much larger numbers, for example ones that have 400 digits, or a 1000 bits or some such. In practice this is impossible on today's computers, even on a clus-

ter of today's computers – it just takes way too much time to find the factors of such a number; it would take ages and ages for the computation to finish even if we use the best algorithms we know. So factoring large numbers is too hard to do in practice, and this hardness can be actually be turned into a benefit, because the hardness of this problem can be used to protect information via an idea called public-key cryptography. I'm not going to explain in detail how that works, but the main idea is that somebody who holds the prime factors of a number (the secret key) can do more things than somebody who only holds the number itself without knowing its factorization. Now the punchline here is that a quantum computer could do these things efficiently. One of the breakthroughs of quantum computers was Peter Shor's algorithm from 1994, which efficiently can find the prime factors even of very large numbers. As a consequence, a quantum computer could break your bank's security. That's bad news for some and good news for others. Of course it's very good news for the mafia who can now steal your money, and also good news for organizations like the NSA, the American National Security Agency, which can now read your email. This discovery was really a big thing, and it caused a surge of interest in the area; the number of people working in quantum computing really exploded in the mid- and late 1990s after the discovery of Shor's algorithm. I myself was one of the people who joined the field only a few years after Shor's algorithm.

Figure 11



Shor's algorithm is very bad news for people who want to hide something. So suppose you do really need to hide something, or you really need to communicate securely. Clearly, you shouldn't base this anymore on the supposed hardness of factoring and related problems, because all those can be broken by a quantum computer. Fortunately there is an alternative to classical cryptography, which is called quantum cryptography. This was invented by Bennett and Brassard in 1984, so already about a decade before Shor's algorithm. Unlike real quantum computers this is already implemented. You can already buy quantum cryptographic devices today, there are companies selling these things. The reason is that quantum cryptography requires much less engineering than a real large-scale quantum computer. So why does quantum cryptography work, why can it be shown to be secure even against an enemy who himself has a quantum computer? A very famous principle of quantum mechanics, actually more of a corollary than a principle, is the Heisenberg uncertainty principle. Basically what this says is that some quantities cannot be known with high accuracy simultaneously. The canonical example in physics are the position and momentum of a particle. The position is just the particle's location in space, and for the purposes of this talk you can think of the momentum as the direction in which the particle is moving. Quantum mechanics implies that you cannot know both of these things very well simultaneously: if

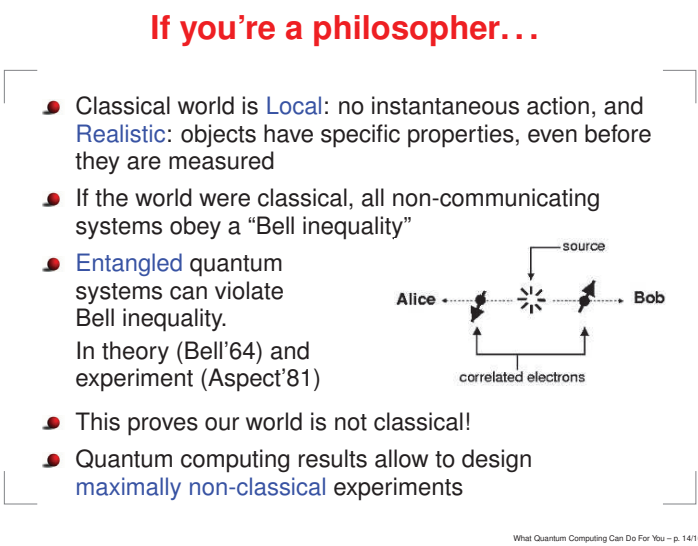
you know a particle's momentum extremely precisely, you will necessarily have a lot of uncertainty about its location, in other words you don't really know where it is. This can lead to lots of problems as illustrated on this cartoon, where Heisenberg's wife is complaining that she can't find her keys, and Heisenberg himself conjectures that she probably knows too much about the momentum of the keys – because by the uncertainty principle, if you know too much about the keys' momentum, then you can't know their location very well. This of course is only a cartoon-explanation, in reality quantum cryptography is a large and complicated field of research, but in principle it's all based on these kinds of uncertainty principles, which don't exist in the classical world. In the classical world objects have specific properties that you can all measure as precisely as you want. Classically there are no limits on your knowledge about the properties of a particles, while in the quantum world there are such limits and we can actually employ them for something useful, in this case secure quantum communication.

Figure 12

Now suppose you are looking for something, for example for your lost keys, like Heisenberg's wife in the previous cartoon. There could be a whole list of locations where those keys could be, you could trace in your memory the places where you've been, where you may have lost the keys. You could try to

go to all of those places and see if your keys are there. Suppose there are N possible locations where your keys could be, then in the classical world it would typically take you on average roughly N/2 steps before you found your keys. If you're lucky you might already find it after a smaller number of tries, but on average you would expect to have to inspect about half of the locations before you have a decent probability of having recovered your keys. The good thing about a quantum computer is that there is an algorithm that solves this task a lot faster. In 1996 Lov Grover discovered his quantum search algorithm, which can solve this search problem in an amount of time that is roughly the square-root of N. So for instance if N is a million then a classical search requires on the order of a million look-ups, while a quantum algorithm finds the looked-for object in roughly a thousand steps. Roughly speaking, what the quantum search algorithm does is that in each step it looks at all N locations in superposition and moves some of the weight in the superposition towards the right location. You can show that after about square-root of N many such steps, almost all the weight is concentrated on the right location. If you then measure the final superposition, you will probably learn what the right location is. So a quantum computer can find the proverbial 'needle in a haystack' much faster than any classical computer can. This search problem is very important because search is a basic subroutine in many different computer science tasks, and all of those tasks can be improved by using Grover's algorithm to do the search. So unlike Shor's algorithm for factoring large integers, Grover's search algorithm has a ton of other applications. For example, you can also use it to find the shortest route between two cities much faster than you could on a classical computer. This could be useful for navigation software.

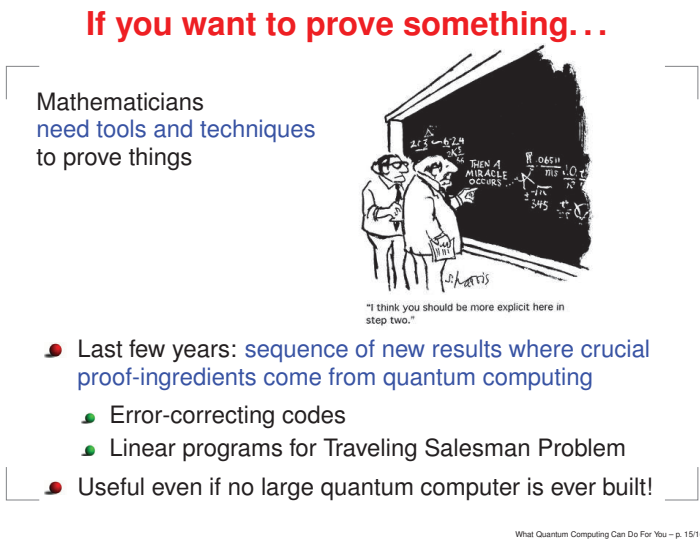
Figure 13



This brings me to a more high-level topic, namely philosophy. I know that some of you here are philosophers – I once was too. If you're a philosopher then what you try to do is to study and understand the world. Most philosophers, certainly until 1900 or so, had a very common-sense idea of the world, assuming the world has some very reasonable properties. One of those properties is 'locality', which says that there is no instantaneous action. If I do something here, that is not going to have an immediate, instantaneous effect on the other end of the universe. It takes a bit of time for signals to travel. Locality is a very reasonable, acceptable principle, which is also adopted for instance in the theory of relativity. Another very reasonable, acceptable principle is something called 'realism'. This is the idea, posited by philosophers until very recently, that the world basically consists of objects, and that those objects have properties. This is also sometimes known as the pincushion idea of matter: an object is like a pincushion, and the properties that it has are like needles that you stick into the cushion. These properties are all very well-defined: we may not know everything about an object, but its properties are all very specific and well-defined. This is a very classical and intuitive view, that the world is both local and realistic. If the world would indeed be consistent with local realism, then you can look at the behavior of non-communicating systems and the correlations between them, and you can prove that all those correlations will

obey something that's called a 'Bell inequality'. I won't really explain what this is, it's some mathematical constraint on the kinds of correlations you can have between two different systems. Now the punchline here is that if you have two different non-communication systems that are entangled – this is my third buzzword, entanglement, which refers to some very strong quantum correlations – then these systems can exhibit correlations that violate a Bell inequality. This was first described theoretically by Jon Bell in the 1960s, in response to Einstein's scepticism about quantum mechanics. It was later experimentally realized by Alain Aspect in the early 1980s. I think that from a philosophical perspective this is probably the most important thing that has come out of 20th century science: you can actually prove by experiment, or at least make extremely plausible by experiment, that this classical worldview is just false. You cannot simultaneously believe in locality and in realism. At this point you might ask why I am putting this in this talk – the theory of Bell inequalities dates from the 1960s and the first experiments from the early 1980s, predating the invention of the quantum computer by Feynman and Deutsch. What does this have to do with quantum computing? The reason I'm talking about this is that you can use the much more recent theory of quantum computing to devise much stronger experiments than the ones Bell came up with. In particular, you can use recent results from the area of quantum communication complexity to design experiments that are in some sense maximally non-classical, that are as quantum as possible, as far-removed from local realism as possible. So this is my main message to the philosophers: there is some really deep stuff that comes out of a combination of theory and experiment, which philosophers should study – and of course, nowadays many philosophers are interested in quantum mechanics.

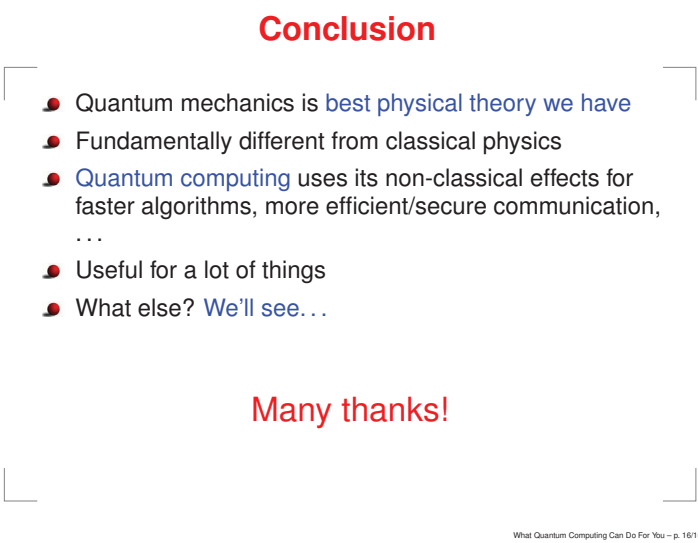
Figure 14



That brings me to my last slide before the conclusion. This is a topic that is very dear to my heart, I spent a lot of time on it. Suppose you want to prove something. I'm a theoretical computer scientist, and in many ways a theoretical computer scientist is an applied mathematician. Proving things is what mathematicians do, and in order to do that they need tools and techniques. The more powerful the tools and techniques, the more interesting things mathematicians can prove. In fact a large part of what mathematicians do is building up their toolbox, and it takes decades of experience and sustained learning to acquire a really strong mathematical toolbox. What is interesting for this lecture is that in the last few years a new toolbox, or at least a new section within the toolbox, has come about, namely all the techniques that have been developed within the area of quantum computing. Initially these tools were designed to find new algorithms and to analyze quantum computers in various ways, but we are discovering that these tools are also quite useful elsewhere. Quantum computing is a unique mixture of a number of different areas of mathematics (linear algebra, probability theory, combinatorics, group theory, information theory etc.), and this combination of various areas has given rise to new mathematical techniques that can be applied in non-quantum areas. There have been a number of new results in mathematics and classical computer science that crucially relied in some way or other on the

tools developed within the area of quantum computing and quantum information. For those of you who know about complex numbers, you can think of the analogy between real and complex numbers: sometimes the easiest way to prove something about real numbers is to go to the larger field of complex numbers and to prove a more general statement there, which you can then translate back to a statement just about real numbers. Something similar happens in the line of research I'm talking about here: sometimes the easiest way to prove something about certain classical structures is to translate them to the quantum domain, prove something in the quantum domain, and then translate the result back to the classical structures that you really care about. Just to mention two examples of this (there are many more): you can prove new limitations on certain classical error-correcting codes using tools from quantum information theory, and some results in quantum communication complexity recently inspired the solution to an old open problem about the size of so-called linear programs for hard computational problems such as the Traveling Salesman Problem. From a high-level perspective, this application of quantum computing as a proof-tool is particularly interesting because it can be used even if nobody ever builds a large-scale quantum computer. As I mentioned before, building a large quantum computer is a very hard engineering problem, it is not at all clear whether this is going to succeed – of course I hope it will. But this application of quantum computing is something that can be used today, even without an actual quantum computer, because it only uses quantum mechanics as a mathematical framework for proofs. You don't need to build a quantum computer for this, in fact even if you don't believe that quantum mechanics is true, you can still use it as a tool for coming up with new mathematical proofs.

Figure 15



That brings me to my conclusion. What is quantum mechanics? It is the best physical theory we have today. It may or may not be true, of course a lot of physical theories have gone out of the door in previous centuries and the same might happen with quantum mechanics some day, but currently it is the best theory we have, it has never been contradicted by experiment. It is fundamentally different from classical physics, remember the three buzzwords I tried to explain along the way: superposition, interference, and entanglement. These are three things that make quantum mechanics very different from classical physics. Quantum computing is the area that uses these non-classical effects to get something better: to get faster algorithms, more efficient communication, more secure communication, and a lot of other things that I didn't have time to talk about. What else will come out of this? We will see. There is really division of labour in this area: some people try to build a quantum computer, I very much hope they will succeed, and then there are people like me who try to figure out the theory of these things, so to figure out what you could do with them. So my job as a professor at the university of Amsterdam is to see what other good things we could do with a quantum computer if we had one.

To end this lecture, let me thank a number of people. Last January, I was here in this church at the celebration of the university's birthday, the dies

natalis. Four people were given honorary doctorates, and one of them, I think it was the philosopher Daniel Dennett, in his acceptance speech at some point looked at the lines of professors sitting there (I was one of them, sitting in the back) and mentioned that he himself and all of them were all lottery winners. And that's definitely true for me as well, along the way to this appointment there were a number of chance events where I was just very lucky to be put on my current path. As I'm thanking a number of people, I will highlight two of such lottery tickets. The first winning lottery ticket was given to me by Shan-Hwei Nienhuys-Cheng. When I was a Master's student at the Erasmus University of Rotterdam I didn't have a very clear idea of what I wanted to do after graduating. Then Shan-Hwei asked me to write a Master's thesis and even a book with her about the topic of Inductive Logic Programming, a theoretical topic in the area of machine learning. Writing this book took us 2.5 years and a lot of struggle, but it was a great learning experience for me. Before doing this I didn't know what to do with my life, but when I finished I had become a scientist and knew I wanted to continue being a scientist. So I'm deeply grateful to Shan-Hwei for making me a scientist. I'm also grateful to Krzysztof Apt for writing the foreword to that book, which in a way was my first contact with the CWI, the Centre for Mathematics and Computer Science. Then Paul Vitanyi gave me a position as PhD student at the UvA and CWI, initially shared with Krzysztof Apt and Johan van Benthem. I am very grateful to Paul for that position, for lots of good advice along the way, and for being a great example of no-nonsense focus on science. As a PhD student I was supposed to work on roughly the same topic I already had worked on with Shan-Hwei, namely machine learning. However, and this is the second winning lottery ticket, at the same time in Paul's group Harry Buhrman was working on something completely different, namely quantum computing. Harry involved me in his research in my first few months as a PhD student. This was 1997, a great time to enter quantum computing, right after the discovery of the algorithms of Shor and Grover, and I haven't looked back since. For that I am very grateful to Harry, as well as for all the good advice and ideas later on. Those lottery tickets eventually brought me here, so let me thank a few people who got me this appointment. First of all I want to thank Leen Torenvliet for organizing the whole appointment procedure, for first asking me if I was interested in this and then for running the process to get it done. Then I want to thank the dean Bart Noordam for nominating me, and the board of the university for the confidence they showed in me by this appointment. I want to thank the pedel for guiding me through the ceremony today. I thank Yde Venema for directing the Institute for Logic Language and Computation, the university

institute where I'm appointed, and I want to thank Jos Baeten for directing the CWI, my other employer for four days in the week. Finally, I want to thank my father for always having been there and for always having supported me.

Ik heb gezegd.