Quantum computation

Andrew Childs

Department of Computer Science,
Institute for Advanced Computer Studies, and
Joint Center for Quantum Information and Computer Science (QuICS)
University of Maryland

Joint Center for Quantum Information and Computer Science

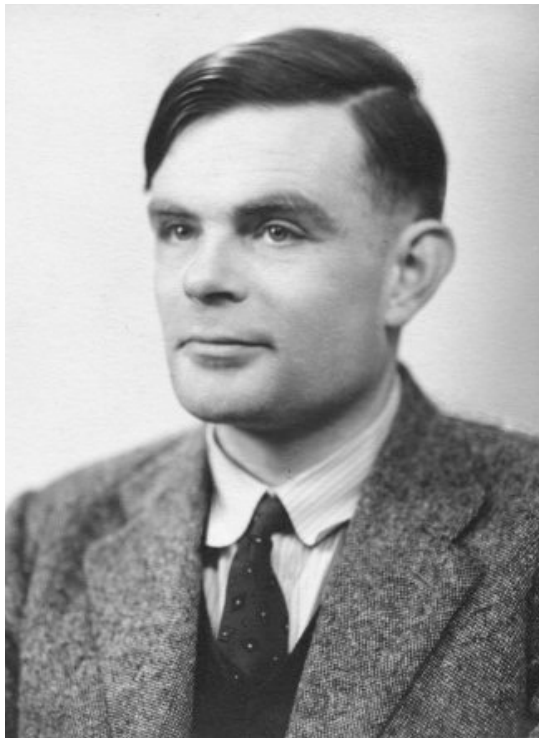
What is a computer?

A means for performing calculations by following a sequence of instructions.

Turing machines

In 1936, Alan Turing formulated what has become the standard mathematical model of computation.

The Turing machine is a mathematical abstraction of a concrete physical process.



The Church-Turing thesis

Church-Turing Thesis

Any calculation that can be performed by mechanical means can be performed by a Turing machine.

Consistent with everything we know about physics.

What about efficiency?

Strong Church-Turing Thesis

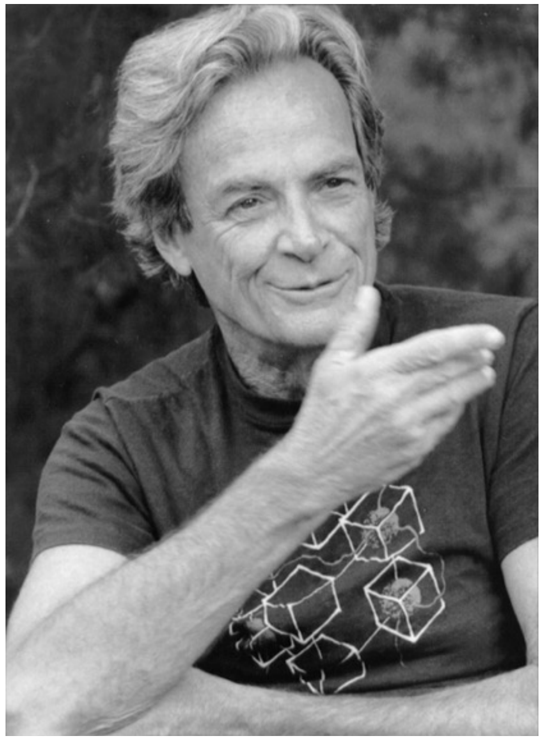
Any calculation that can be performed efficiently by mechanical means can be performed efficiently by a Turing machine.

Challenging the strong Church-Turing thesis

Quantum mechanics seems to be hard for computers to simulate.

A system of *n* quantum particles is described by 2^n complex numbers. We don't know how to predict the outcomes of experiments using less than an exponential amount of computation.

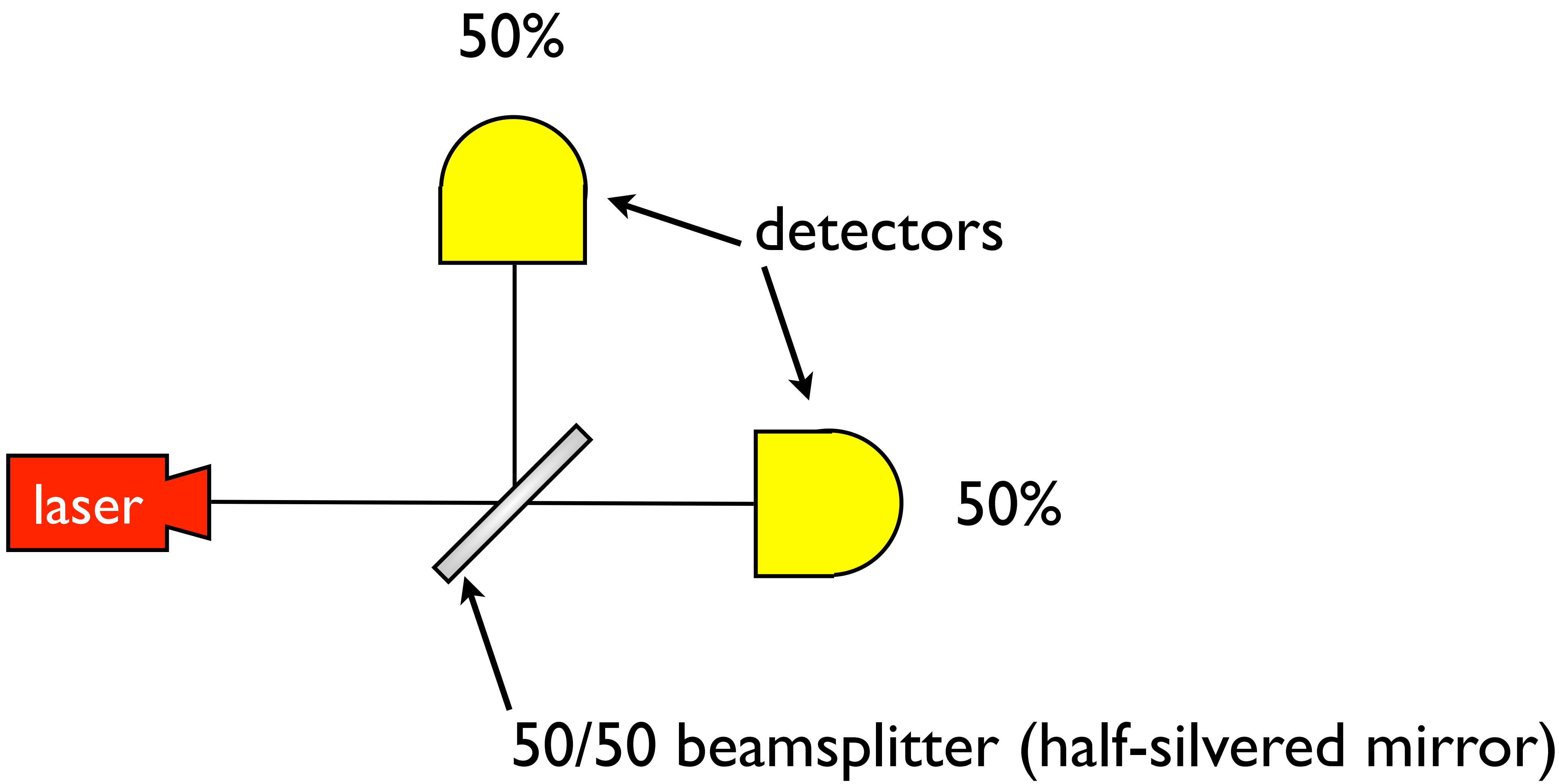
"As far as I can tell, you can simulate this with a quantum system, with quantum computer elements. It's not a Turing machine, but a machine of a different kind."



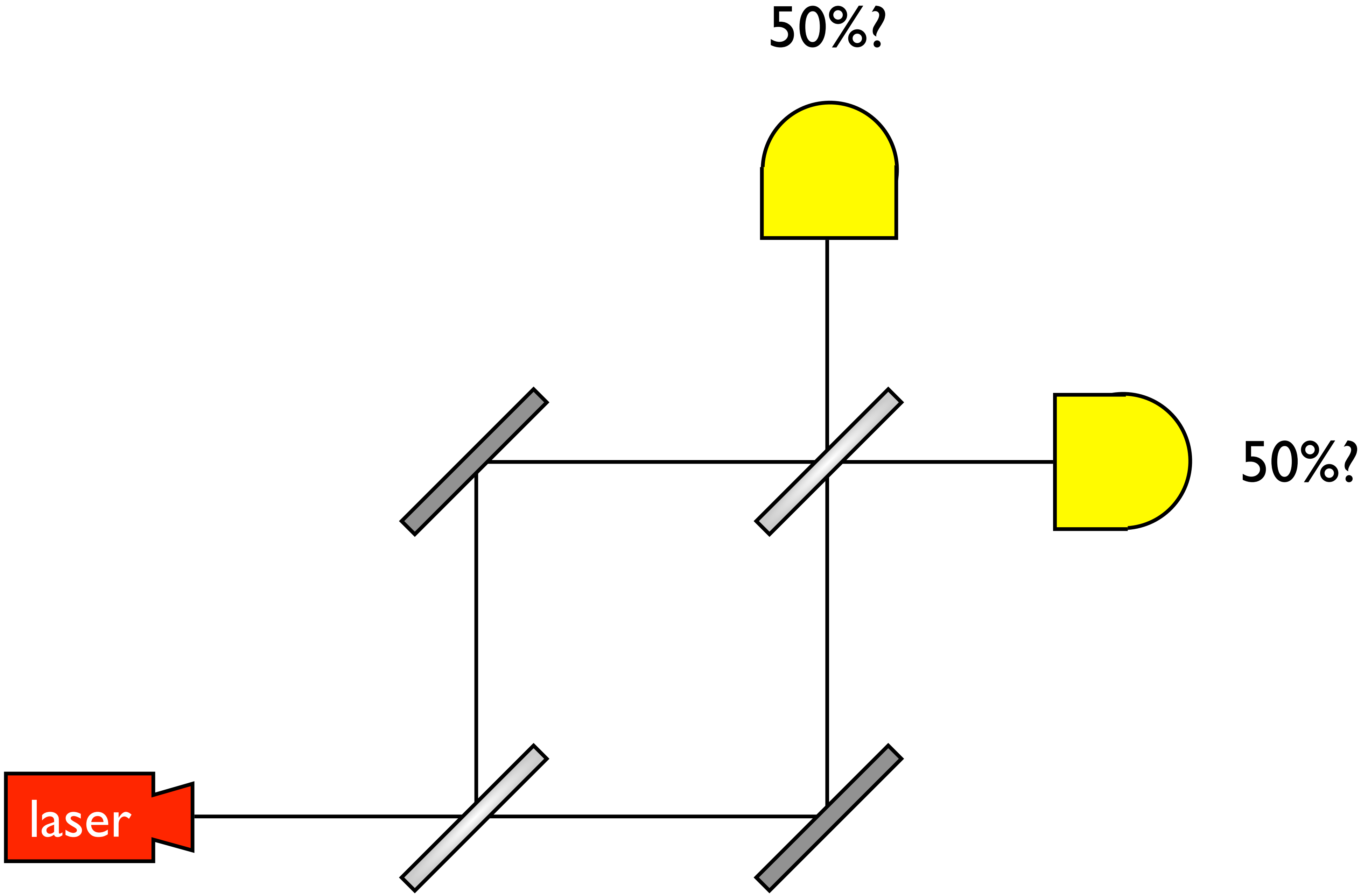
Do quantum systems naturally perform exponentially hard calculations?

Can we harness this power?

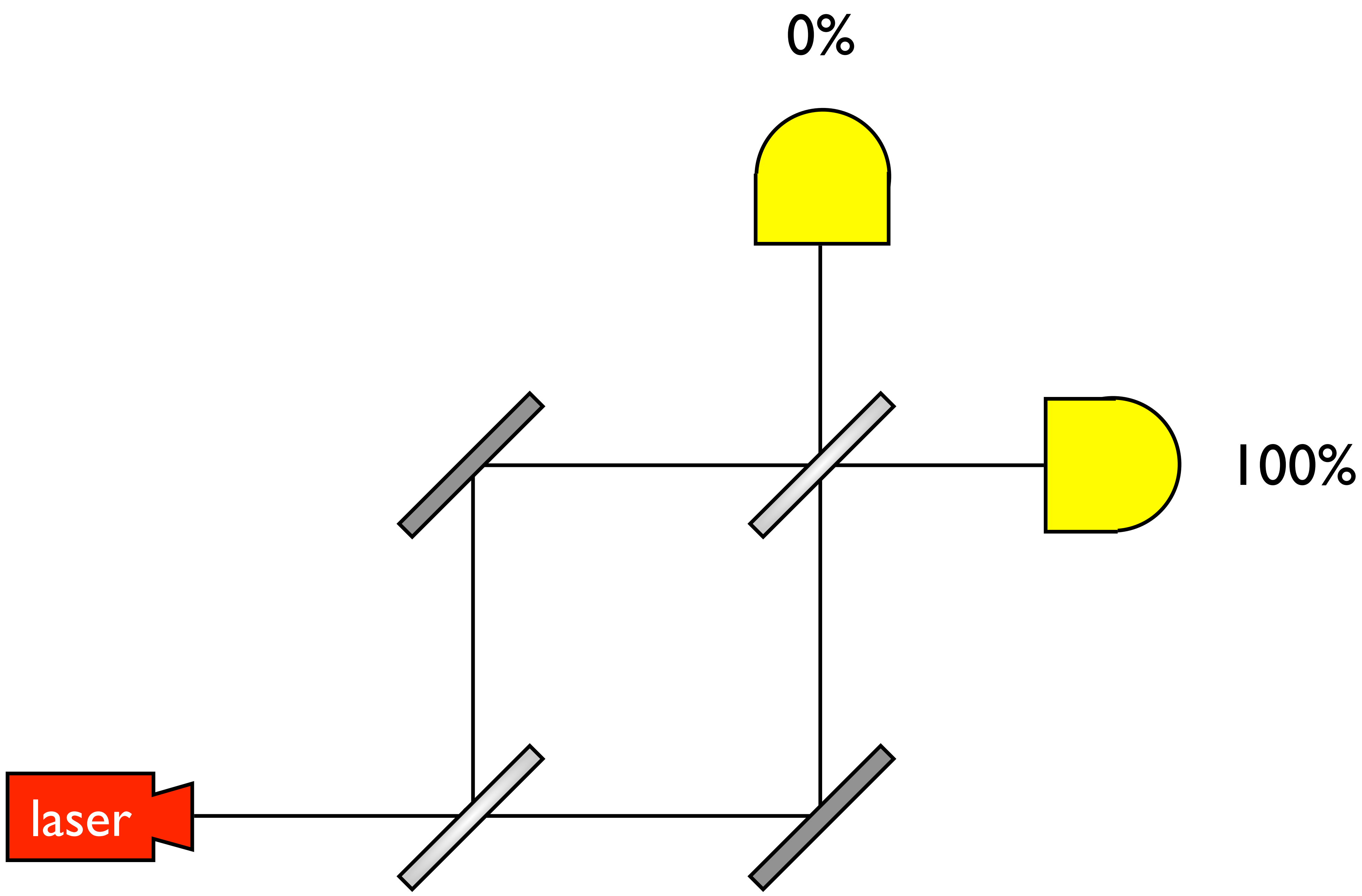
A simple experiment



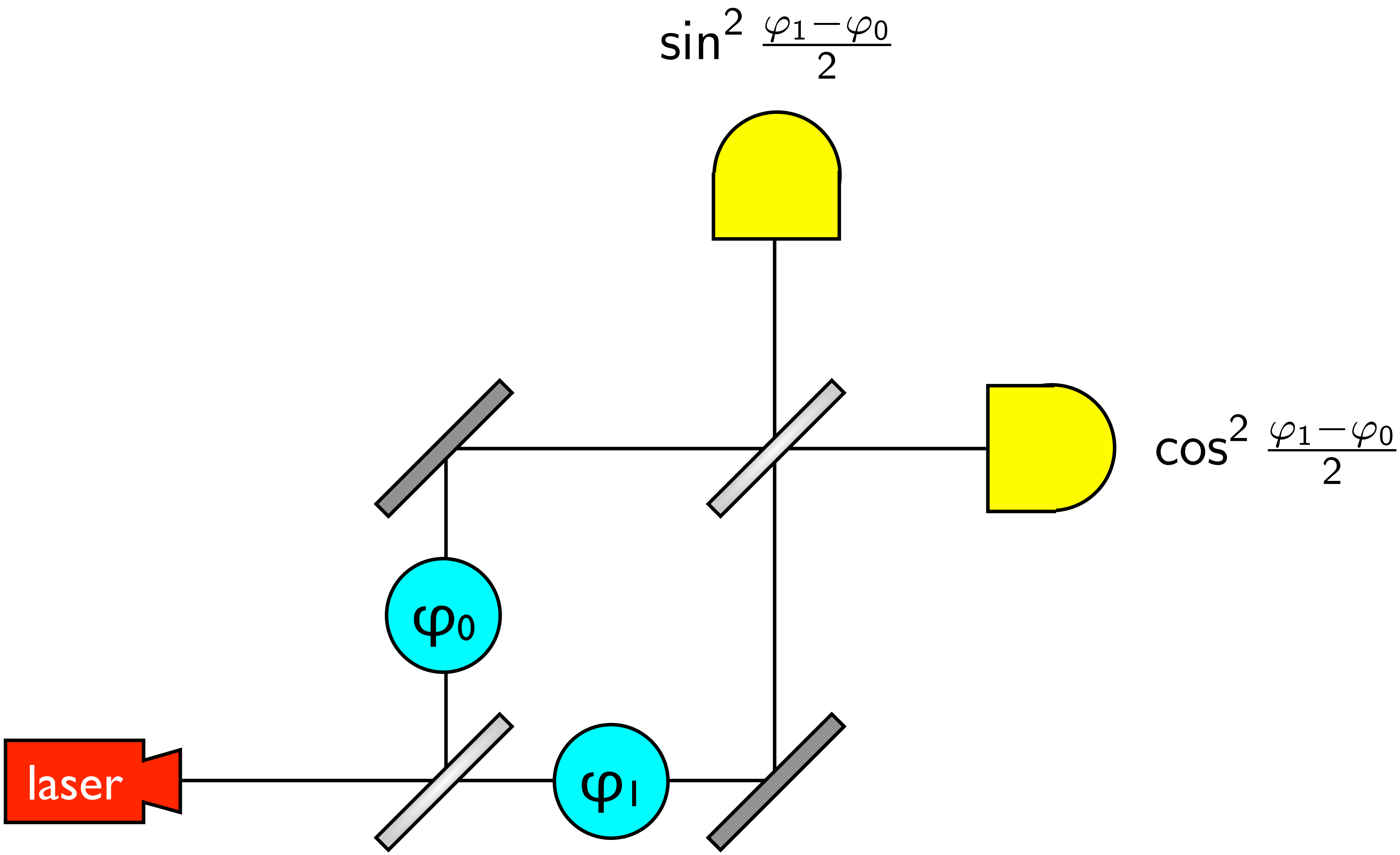
Interferometer



Interferometer



Phase shifts



Deutsch's problem

Given: A function $f: \{0,1\} \rightarrow \{0,1\}$
(As a black box: You can call the function f, but you can't read its source code.)

Task: Determine whether f is constant.

Four possible functions:

constant

x	$f_1(x)$
0	0
1	0

x	$f_2(x)$
0	1
1	1

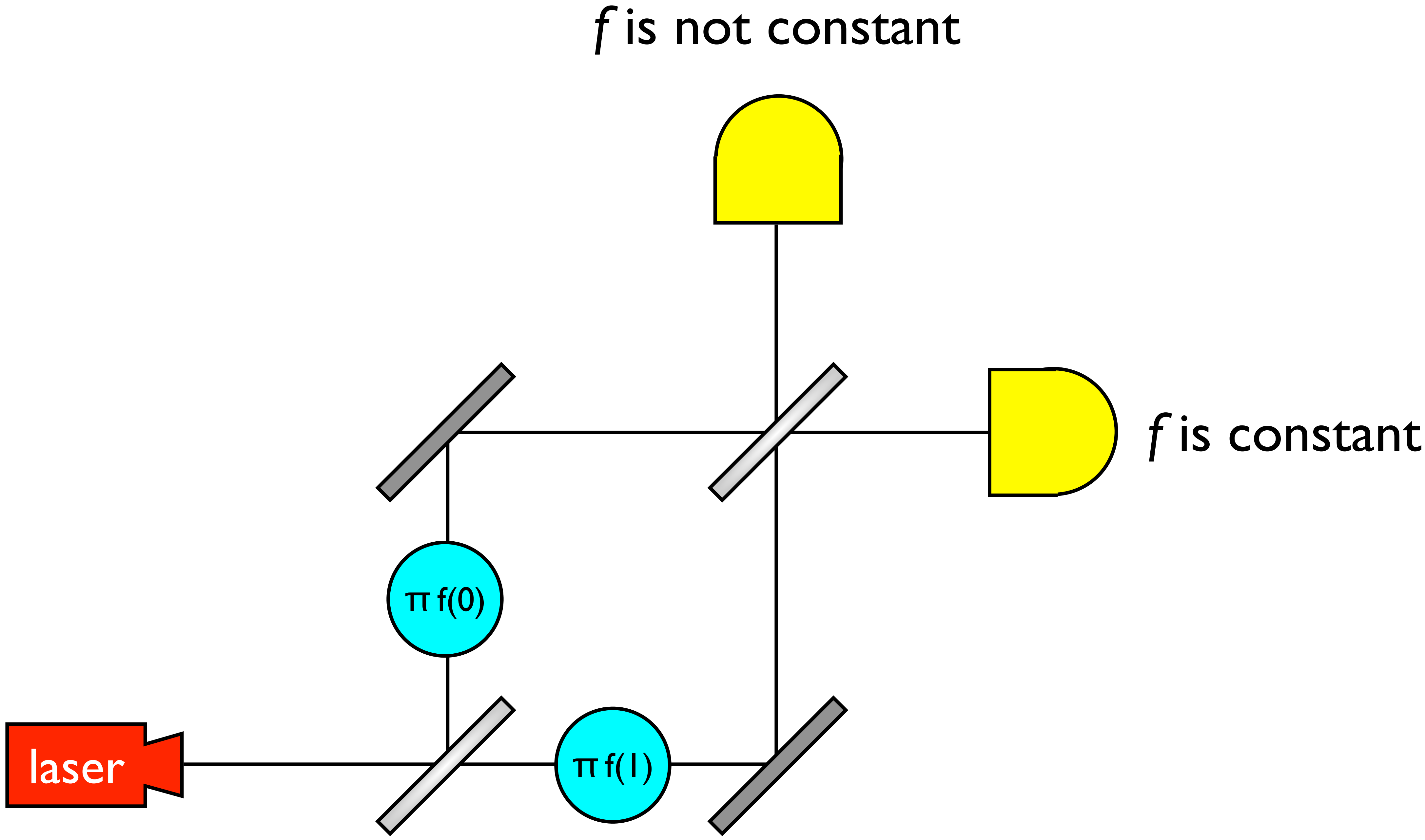
not constant

x	$f_3(x)$
0	0
1	1

x	$f_4(x)$
0	1
1	0

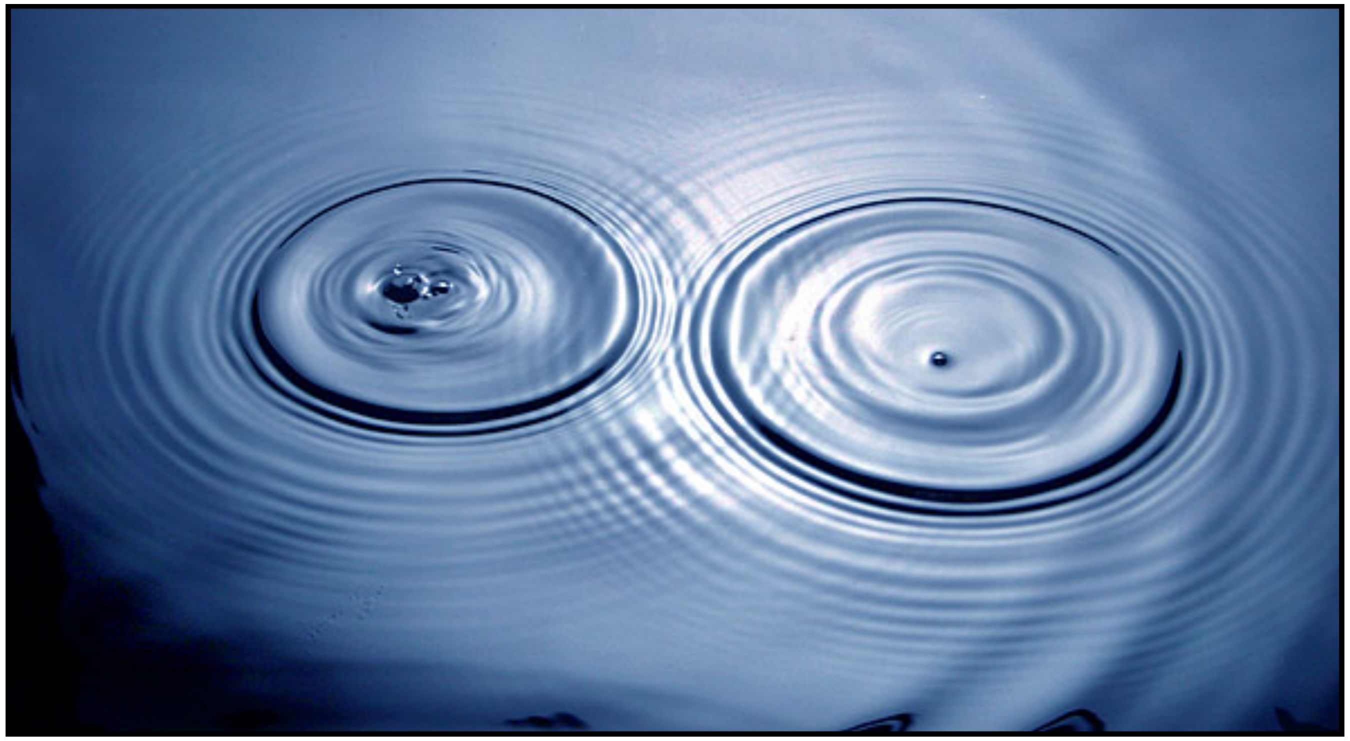
Classically, two function calls are required to solve this problem.

Deutsch's algorithm



The origin of quantum speedup

Interference between computational paths



Arrange so that
- paths to the solution interfere constructively
- paths to non-solutions interfere destructively

Quantum mechanics gives an efficient representation of high-dimensional interference phenomena

Factoring integers

Factoring integers is believed to be computationally difficult.

$$
\begin{array}{c}
3107418240490043721350750035888567930037346022842 \\
7275457201619488232064405180815045563468296717232 \\
8678243791627283803341547107310850191954852900733 \\
772482278352574238645401469173660247765234660 9
\end{array}
=
\begin{array}{c}
1634733645809253848443133883865090859841783670033 \\
092312181110852389333100104508151212118167511579 \\
\times \\
1900871281664822113126851573935413975471896789968 \\
515493666638539088027103802104498957191261465571
\end{array}
$$

The security of modern electronic commerce relies on this assumption!

In 1994, Peter Shor showed that quantum computers can efficiently factor integers.

In a nutshell: Quantum computers can efficiently detect periodicity. The periodicity of the powers of a number modulo N is closely related to the prime factorization of N.

What problems can quantum computers solve?

- Computational number theory/algebra (→ cryptanalysis)

  Factoring integers, computing discrete logarithms, decomposing Abelian groups, approximating Gauss sums, shifted Legendre symbol problem, counting points on algebraic curves, Pell's equation, computing the unit group of an algebraic number field, …

- Simulating quantum mechanics

  Computational quantum chemistry, computational materials science

- Systems of linear equations

  Differential equations, effective resistance, machine learning?

- Approximating topological invariants

  Jones polynomial, Turaev-Viro invariant

- Unstructured search and generalizations (polynomial speedup)

  Collision finding, graph problems, formula evaluation, property testing, …

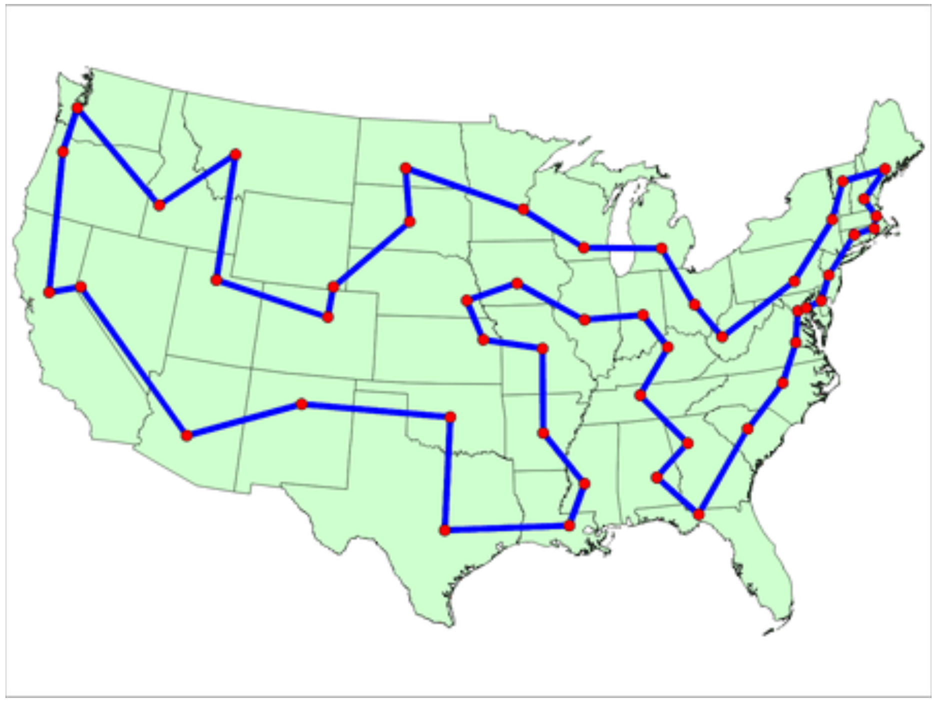
Quantum algorithm zoo: http://math.nist.gov/quantum/zoo/

What are quantum computers *not* good for?

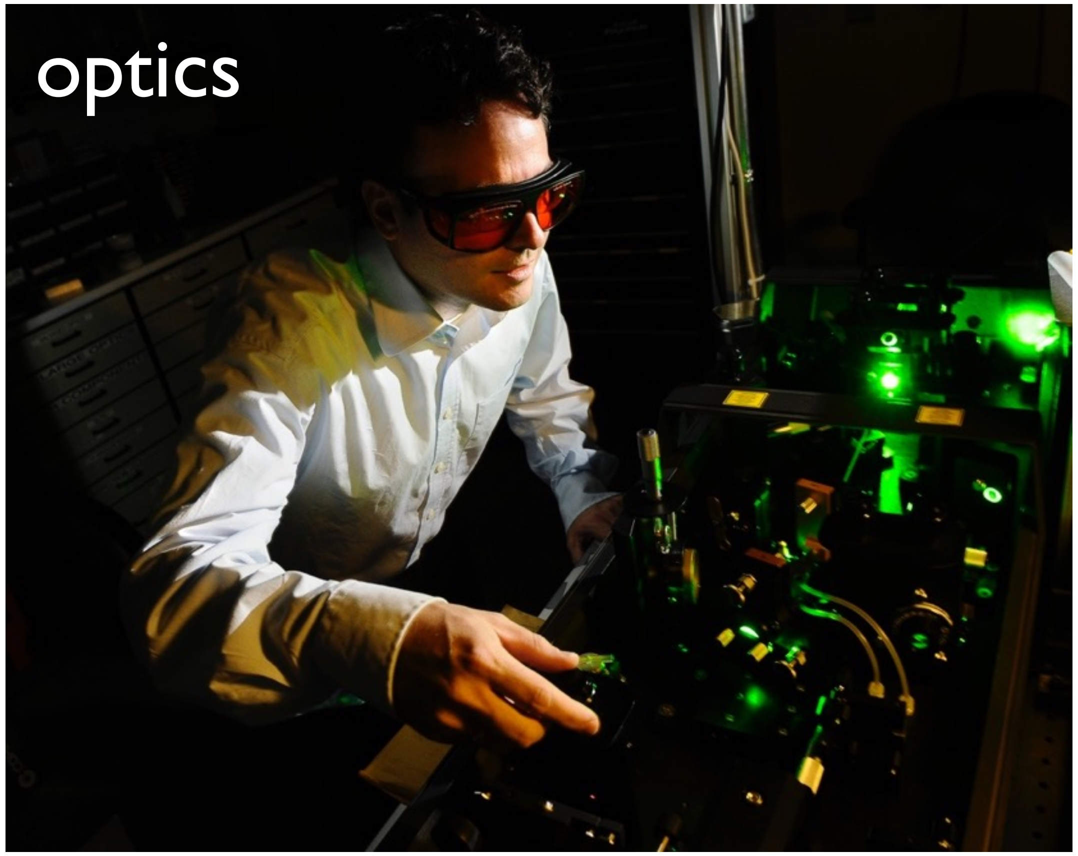
A quantum computer is not just a classical computer with a faster clock speed.

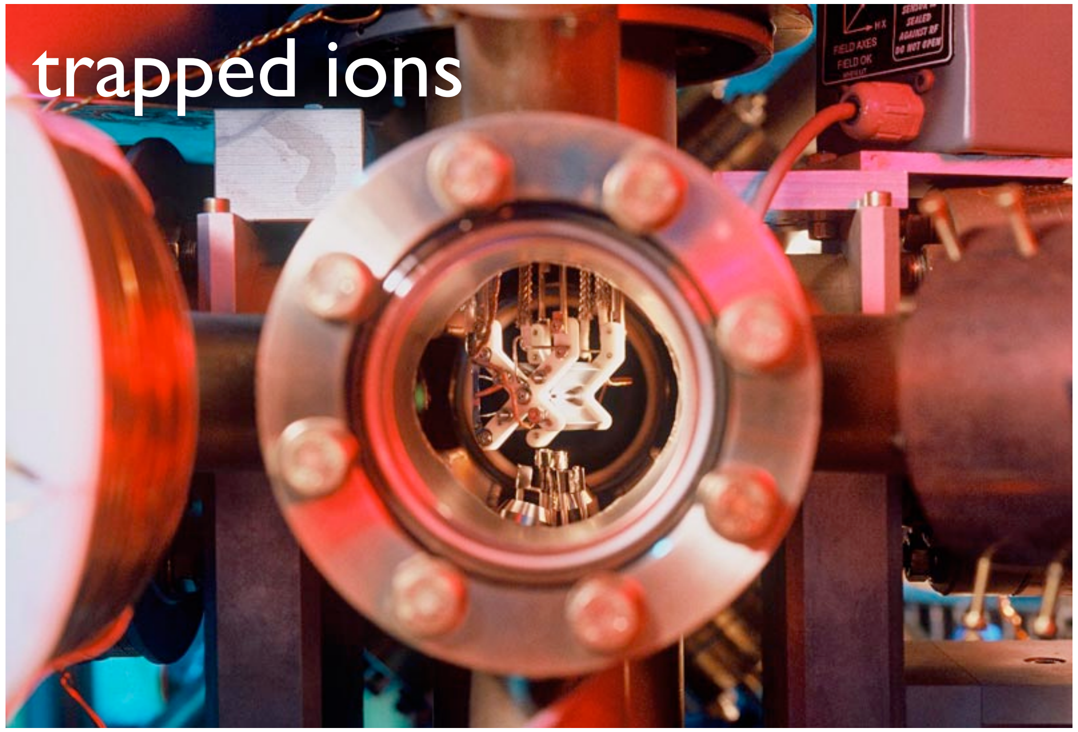
A quantum computer cannot simply try all possible solutions in parallel and find a valid one.

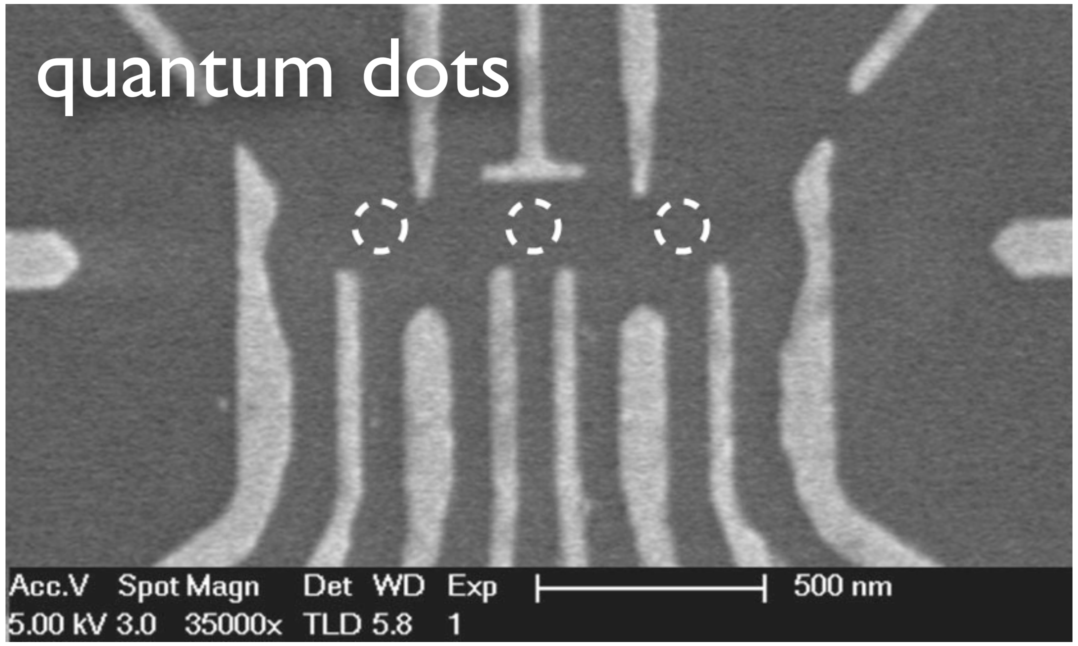
We do not expect efficient quantum algorithms for NP-hard problems.

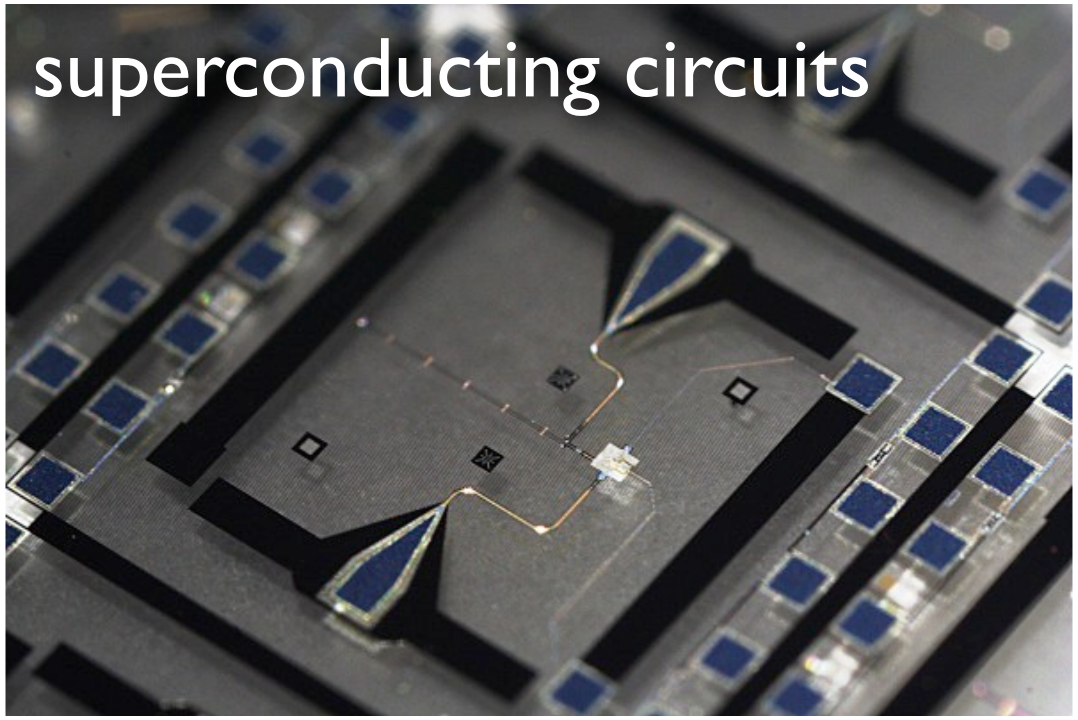


Building a quantum computer

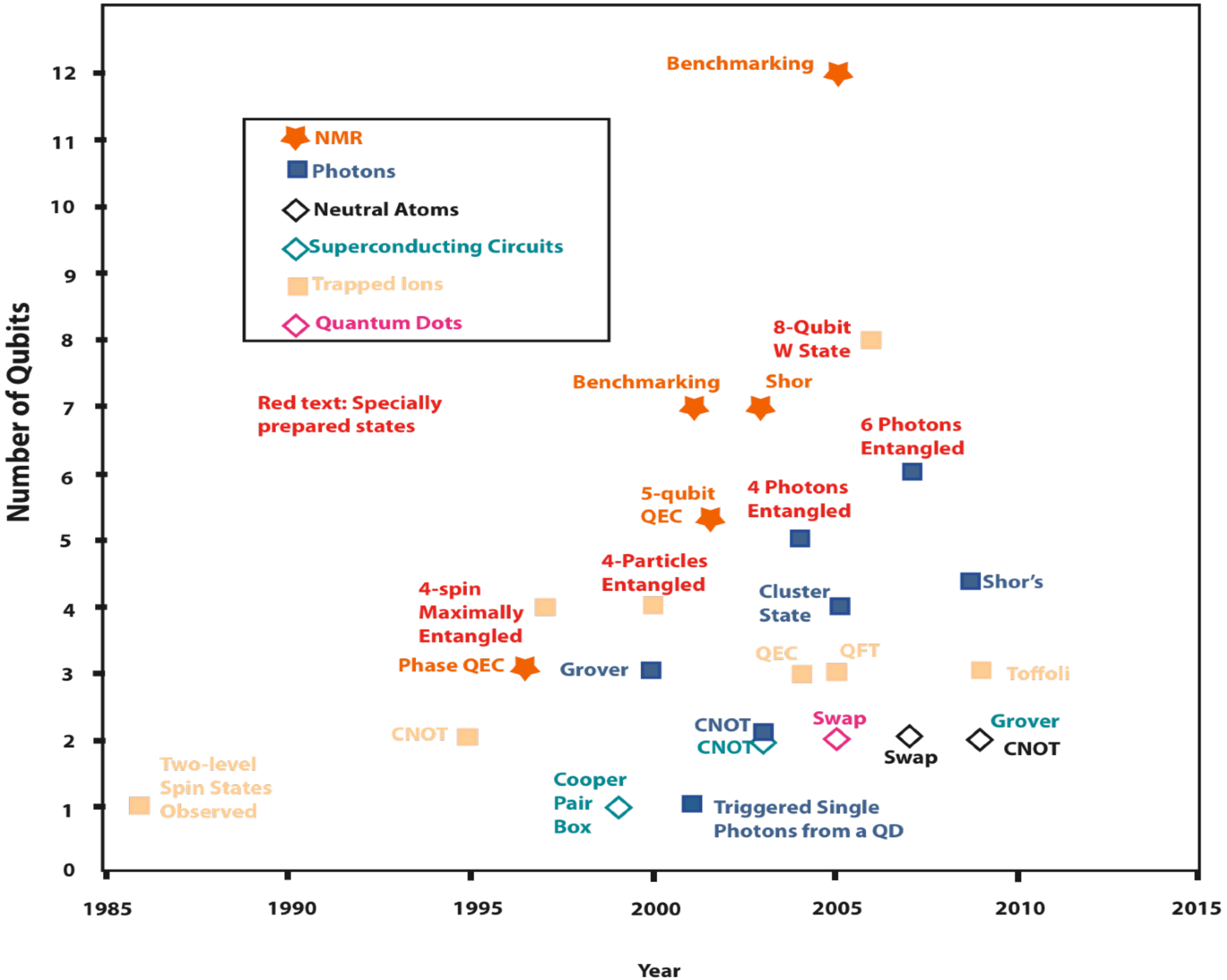








Experimental progress



Fault tolerance

Realistic quantum systems are noisy. How can we make a reliable quantum computer from unreliable components?

Main idea: Encode information in *quantum error-correcting codes*

Example (of a classical code): $0 \rightarrow 000$, $1 \rightarrow 111$

Error correction: $000 \xrightarrow{\text{bit flip error}} 010 \xrightarrow{\text{majority voting}} 000$

Make this quantum and perform logical operations fault-tolerantly

Fault-tolerance Threshold Theorem: If we can manipulate qubits sufficiently well (with constant error rate, say 10^{-4}), we can effectively make them perfect through an encoding with reasonable overhead.

The future of quantum computing

Experimental challenge: robust control of quantum systems

How can we build a scalable quantum computer?

Theoretical challenge: programming quantum computers

How can we discover new fast quantum algorithms?