Title: Quantum simulation logic, oracles, and the quantum advantage

Speakers: Jan-Ake Larsson

Series: Particle Physics

Date: September 10, 2019 - 3:30 PM

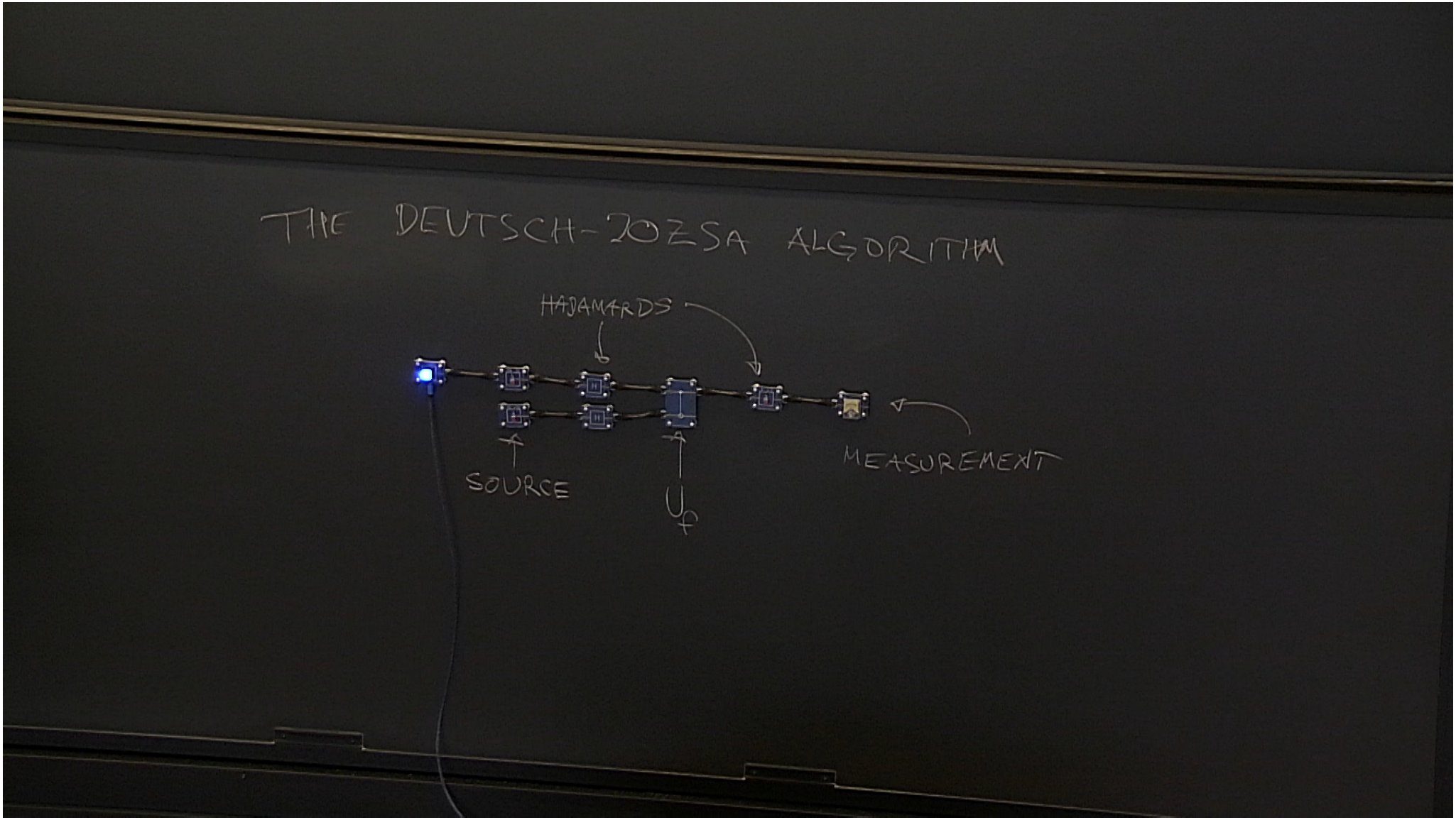
URL: http://pirsa.org/19090094

Abstract: Query complexity is a common tool for comparing quantum and classical computation, and it has produced many examples of how quantum algorithms differ from classical ones. Here we investigate in detail the role that oracles play for the advantage of quantum algorithms. We do so by using a simulation framework, Quantum Simulation Logic (QSL), to construct oracles and algorithms that solve some problems with the same success probability and number of queries as the quantum algorithms. The framework can be simulated using only classical resources at a constant overhead as compared to the quantum resources used in quantum computation. Our results clarify the assumptions made and the conditions needed when using quantum oracles. Using the same assumptions on oracles within the simulation framework we show that for some specific algorithms, such as the Deutsch-Jozsa and Simonâ€™s algorithms, there simply is no advantage in terms of query complexity. This does not detract from the fact that quantum query complexity provides examples of how a quantum computer can be expected to behave, which in turn has proved useful for finding new quantum algorithms outside of the oracle paradigm, where the most prominent example is Shorâ€™s algorithm for integer factorization.

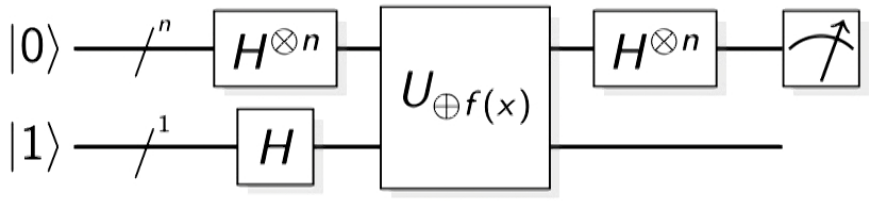
Quantum computation and the additional degrees of freedom in a physical system

Niklas Johansson, Jan-Åke Larsson

LINKÖPING UNIVERSITY


THE DEUTSCH-JOZSA ALGORITHM
HADAMARDS
SOURCE
U_f
MEASUREMENT

There are several proposed resources for quantum computational speedup

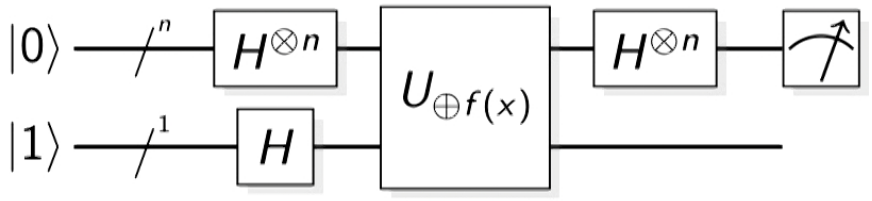


Proposed resources:

- Quantum Superposition and Interference
- Quantum Entanglement
- Quantum Nonlocality
- Quantum Contextuality
- Continuity of Quantum state-space

LINKÖPING UNIVERSITY

There are several proposed resources for quantum computational speedup

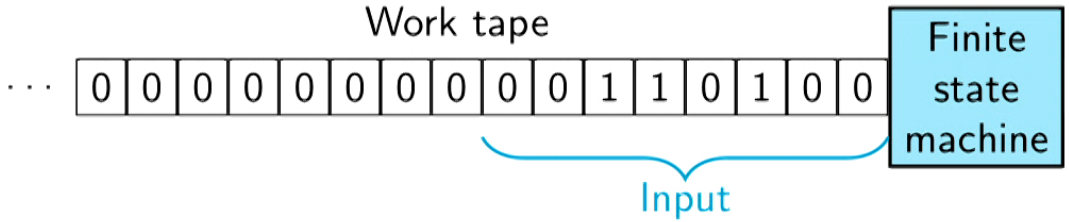


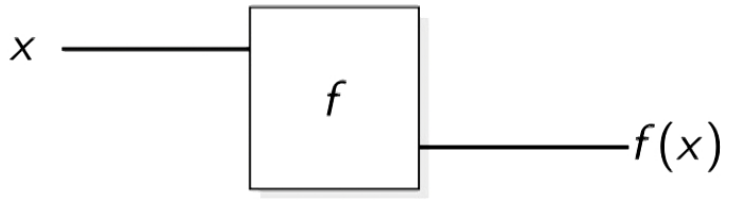
Proposed resources:

- Quantum Superposition and Interference
- Quantum Entanglement
- Quantum Nonlocality
- Quantum Contextuality
- Continuity of Quantum state-space
- R/W in additional degree of freedom

Turing machines and functions as gate arrays

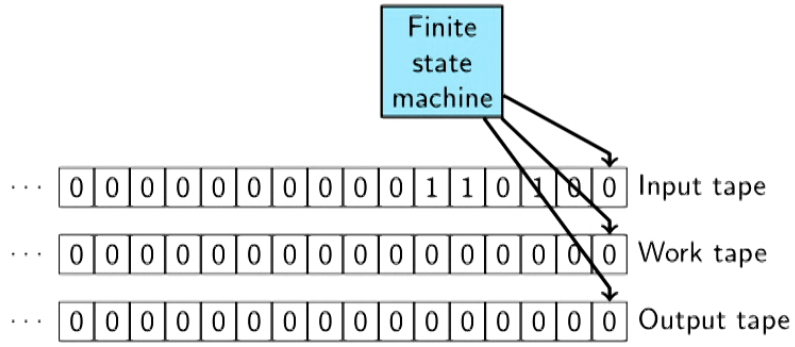
A one-tape machine corresponds to
a simple irreversible gate array

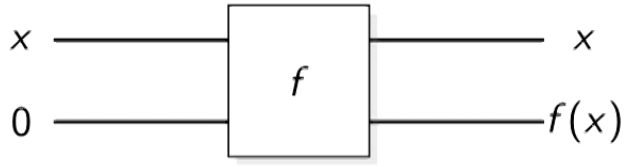




Reversible computers use a reversible gate array

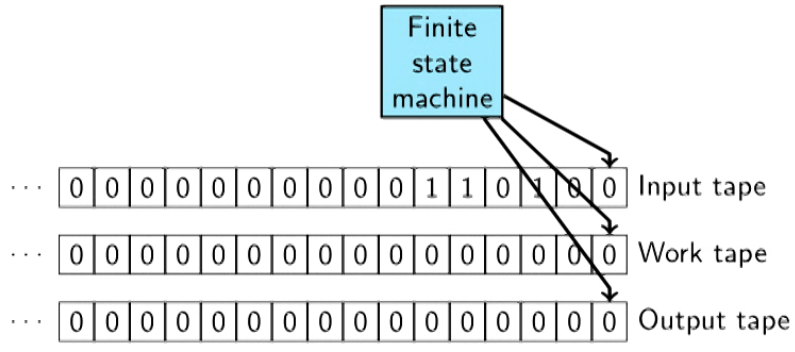
A three-tape machine doesn't quite correspond to the most common way of drawing a reversible gate array

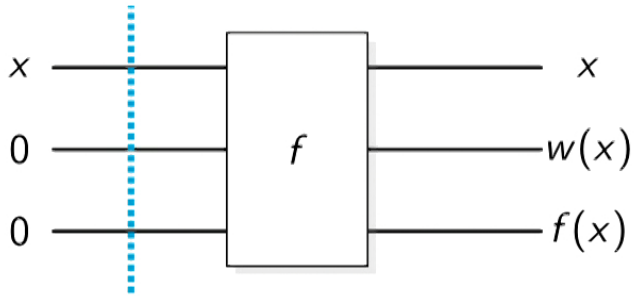




Reversible computers use a reversible gate array

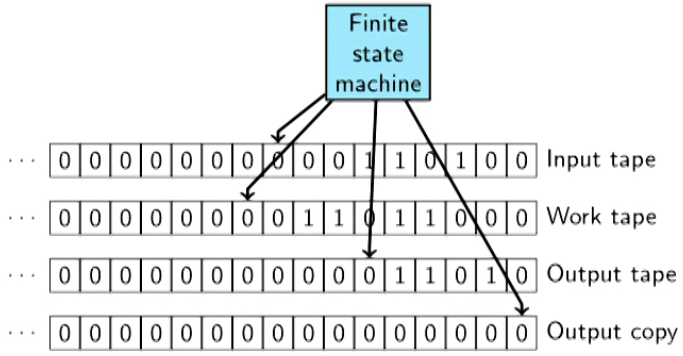
The work tape corresponds to a calculation log on ancillary bits

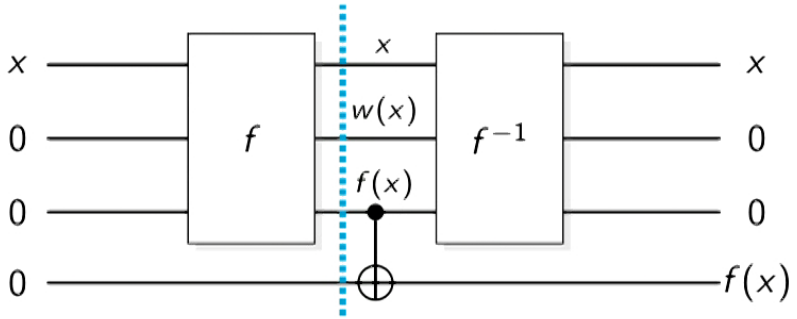




The reversible gate is often built using Bennett's trick (1973)

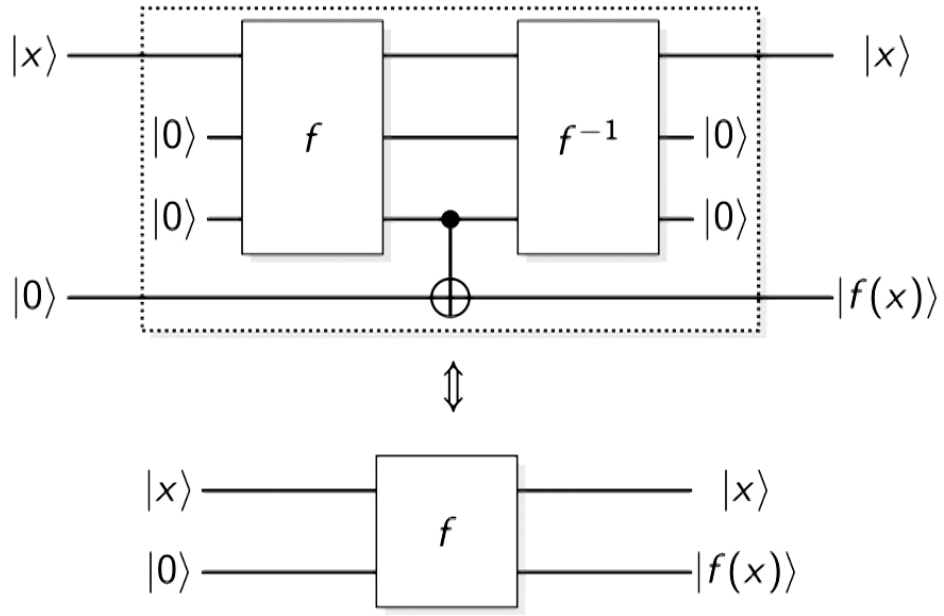
Add a tape for the copy of the result





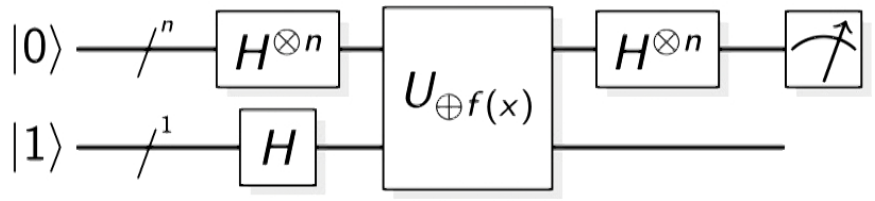
Even when not built using Bennett's trick, the gate is unitarily equivalent to one that is

This can be clarified by enclosing Bennett's trick as a composite gate



Many quantum algorithms use Hadamard transformations

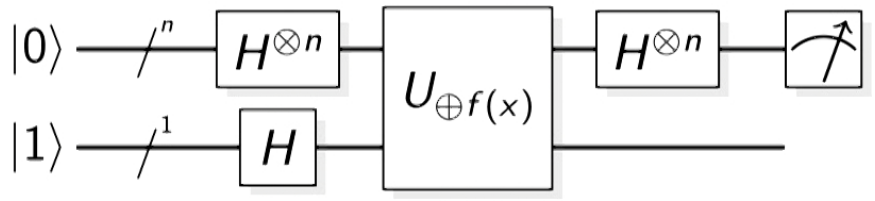
The Deutsch-Jozsa algorithm is one example



The standard explanation is that all the possible input values are input at once, the calculation is done in parallel for all of them, and then all the parallel results are combined by interference to find the result

Many quantum algorithms use Hadamard transformations

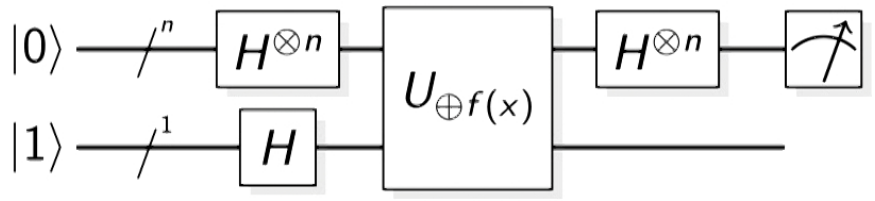
The Deutsch-Jozsa algorithm is one example



The standard explanation is that all the possible input values are input at once, the calculation is done in parallel for all of them, and then all the parallel results are combined by interference to find the result

Many quantum algorithms use Hadamard transformations

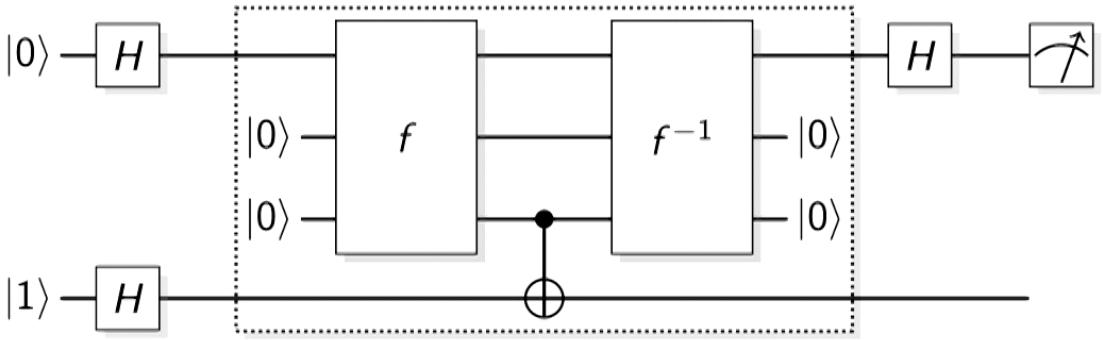
The Deutsch-Jozsa algorithm is one example



The standard explanation is that all the possible input values are input at once, the calculation is done in parallel for all of them, and then all the parallel results are combined by interference to find the result

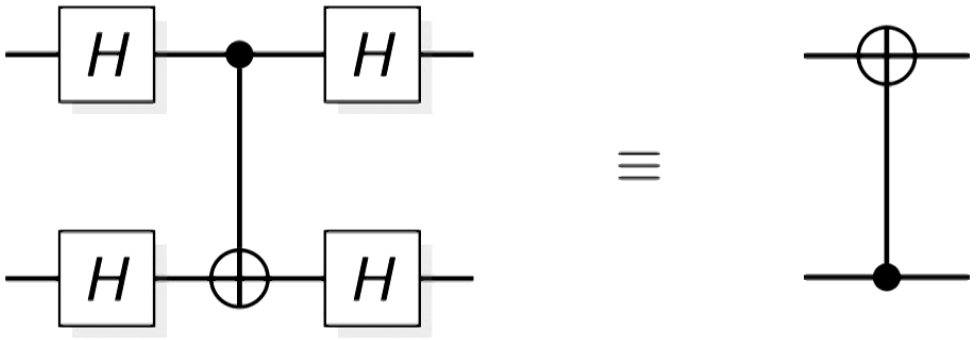
Many quantum algorithms use Hadamard transformations

This can be clarified by enclosing Bennett's trick as a composite gate

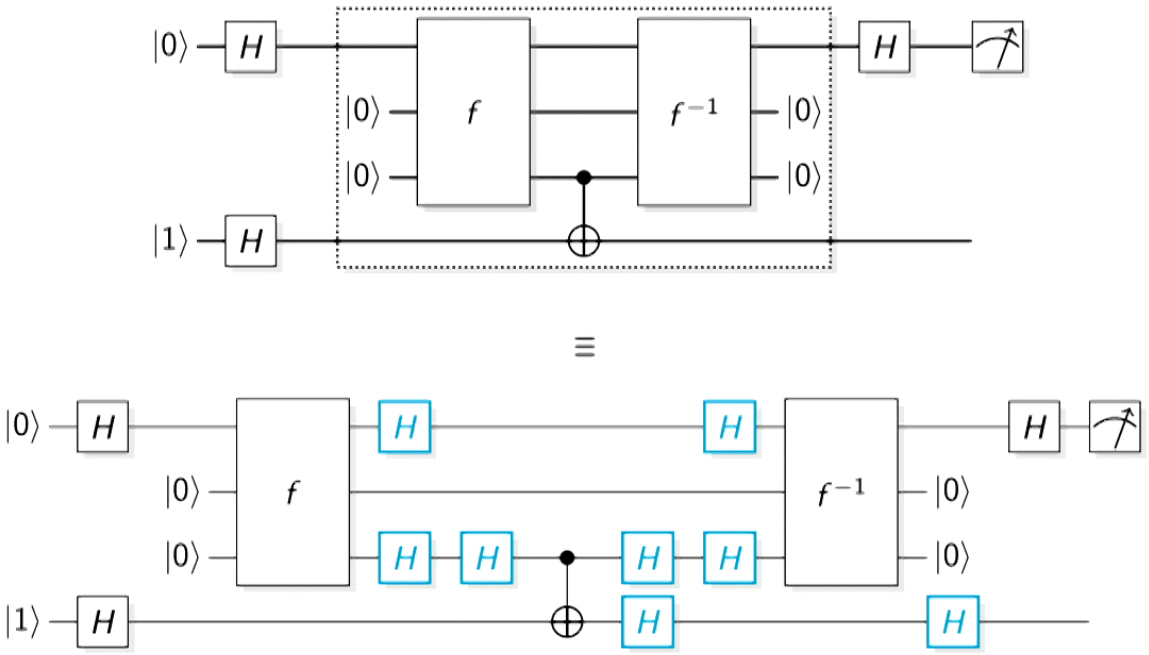


Our questions are: Is the standard explanation correct? Are the results computed in parallel? And is this the central property of the calculation?

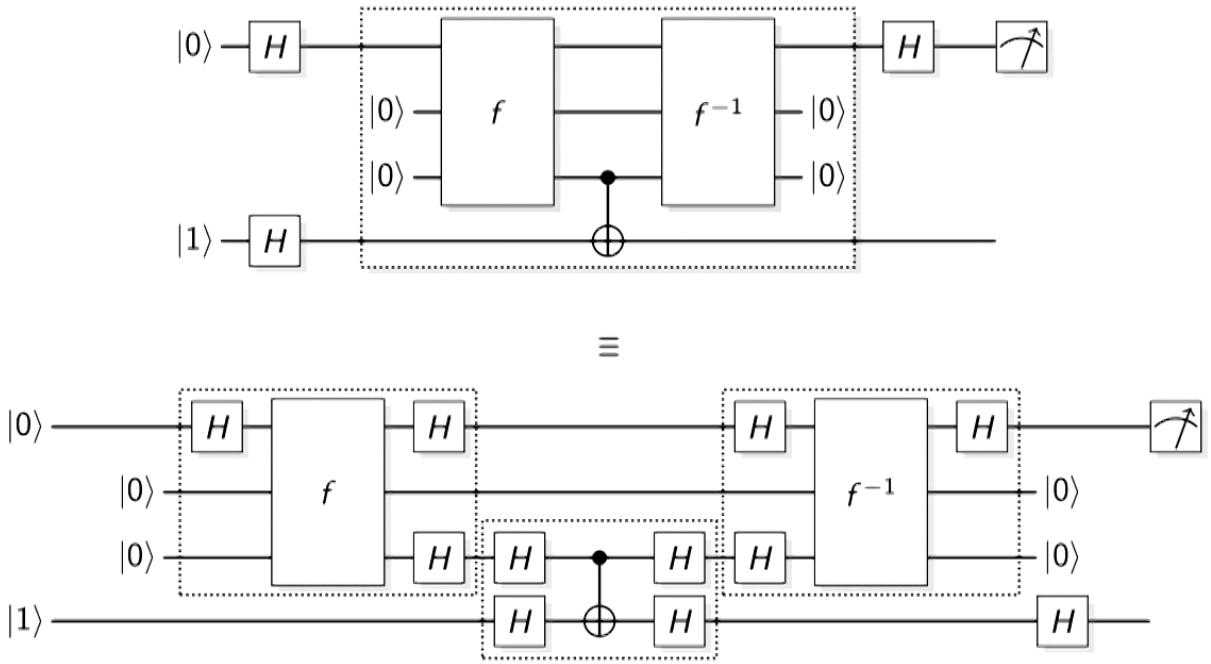
Reminder: the "Phase kick-back" identity



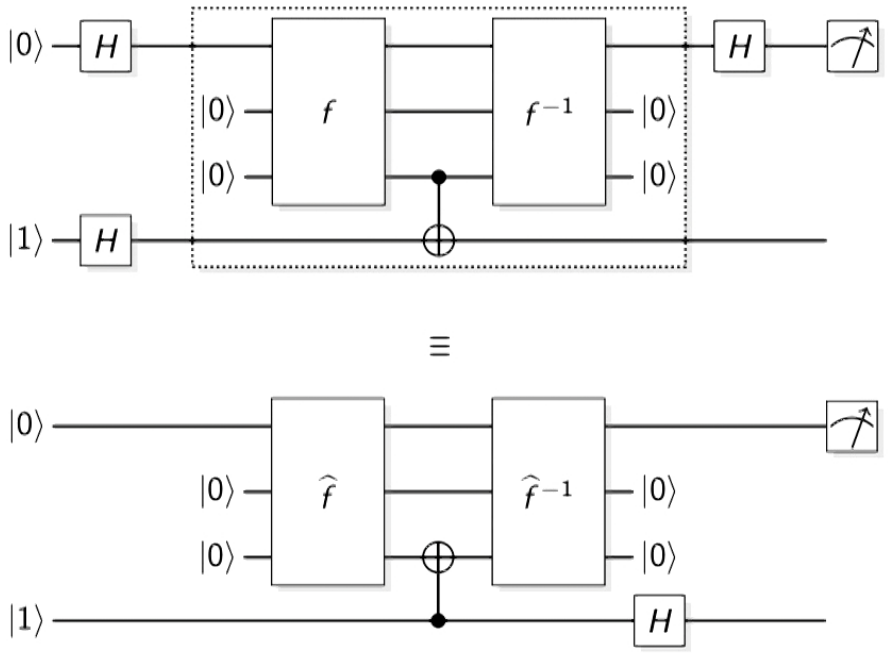
This transforms the function gate



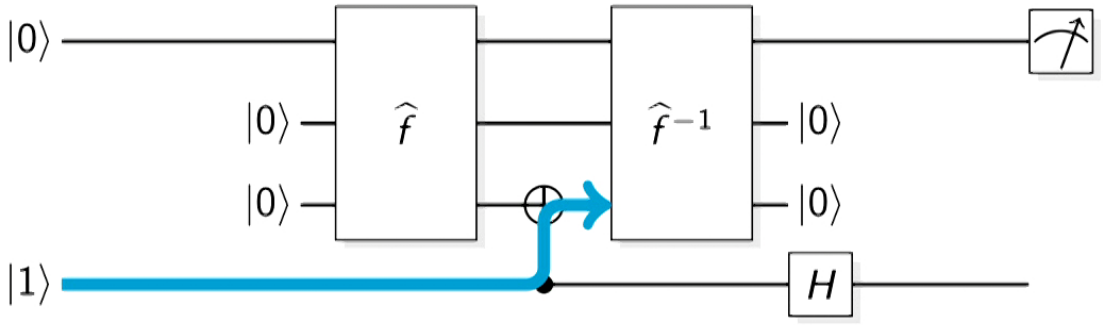
This transforms the function gate



This transforms the function gate

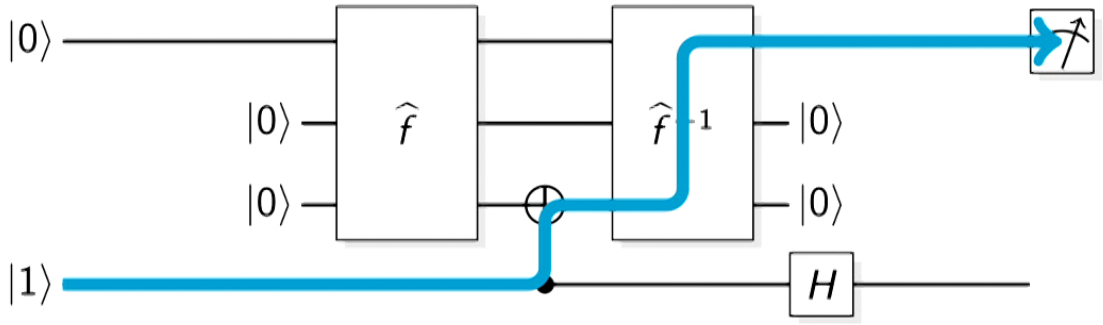


The function evaluation is viewed from a different degree of freedom



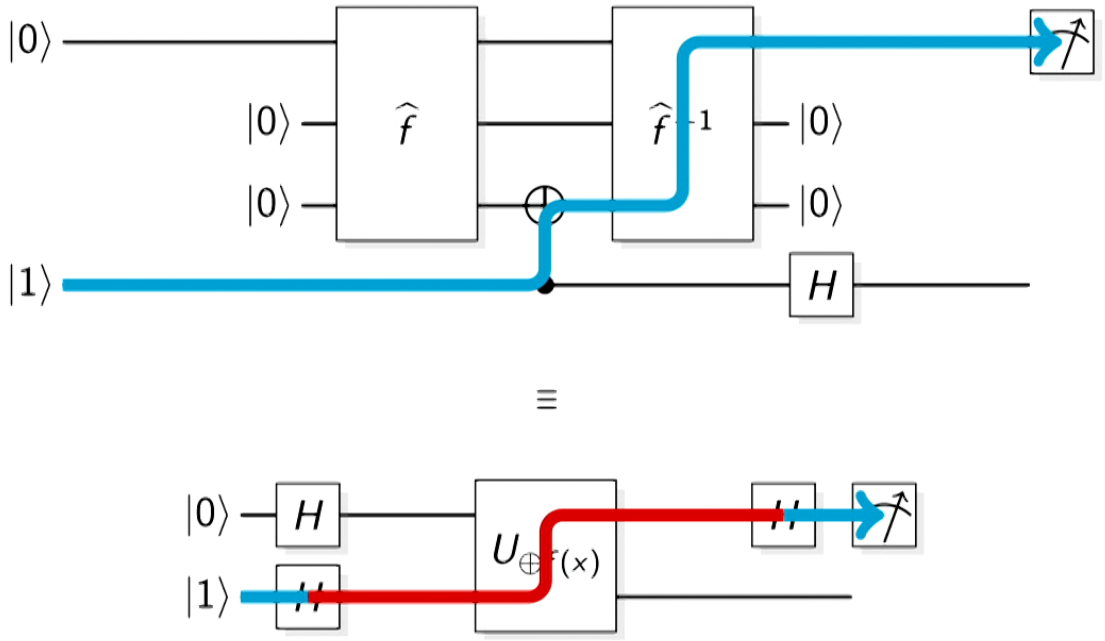
- The quantum algorithm contains the evaluation of transformed functions
- Information from the "target" system is copied into the evaluation

The function evaluation is viewed from a different degree of freedom



- The quantum algorithm contains the evaluation of transformed functions
- Information from the "target" system is copied into the evaluation
- Ancillas are zeroed, so the information (that cannot be erased) must influence the "control" output

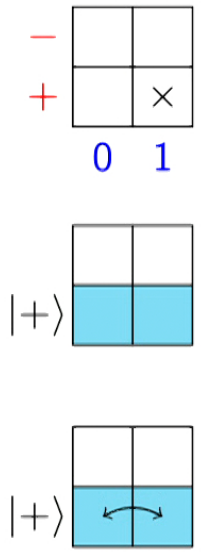
The function evaluation is viewed from a different degree of freedom



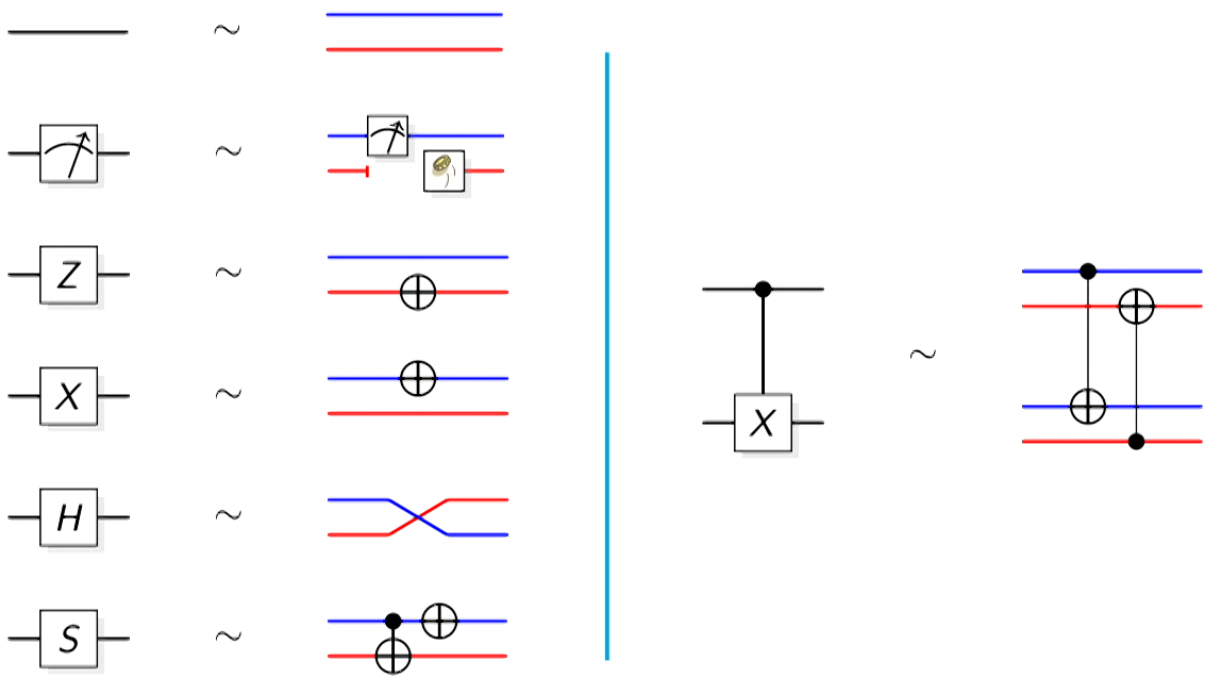
The Hadamard transform enables access to the additional degree of freedom: use the phase space

Spekkens' toy model

- Each spin-1/2 system is associated with a two-bit "ontic state" that singles out a point in the associated phase space
- Each spin-1/2 measurement outcome is associated with an "epistemic state", that plays the role of the quantum state
- The uncertainty relation is translated into the "knowledge balance principle": measuring one of the ontic bits randomizes the other



Clifford-group gates can now be represented, even on composite systems



Spekkens' toy model reproduces many of the quantum predictions

In the toy model

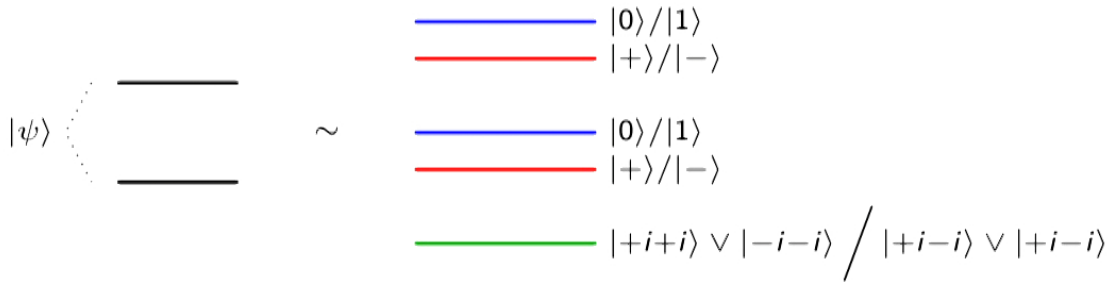
- Noncommuting measurements
- Uncertainty
- Interference
- Remote steering
- No cloning
- No broadcasting
- Mutually Unbiased Partitions
- Superdense coding
- Entanglement monogamy
- Teleportation
- Positive Operator Valued Measures
- ...

Not in the toy model

- Contextuality
- Nonlocality
- Quantum-computational speedup
- Continuum of states
- ...

Including quantum contextuality in the model requires more memory than two bits per system

- Kleinmann, Gühne, Portillo, JÅL, Cabello (2011): lower bound of 5 bits of information
  - Rewrite the model into the language of Mealy machines
  - Extend the Mealy machine to include Peres-Mermin-square contextuality



Including quantum contextuality in the model requires more memory than two bits per system

- Kleinmann, Gühne, Portillo, JÅL, Cabello (2011): lower bound of 5 bits of information
- The knowledge balance principle becomes a knowledge imbalance principle, because there is always more knowledge one lacks than one has

Including quantum contextuality in the model requires more memory than two bits per system

- Kleinmann, Gühne, Portillo, JÅL, Cabello (2011): lower bound of 5 bits of information
- The knowledge balance principle becomes a knowledge imbalance principle, because there is always more knowledge one lacks than one has
- Harrysson (2016): upper bounds for small number of qubits
- Karanjai, Wallman, Bartlett (2018): full simulation requires at least $n(n-1)/2$ bits
- Lillystone and Emerson (2019): Explicit model with $(2n+1)(n+1)$ bits
  - Model is symmetric-always-ψ-epistemic
  - Also improves lower bound to $(2n+1)(n-1)$ bits

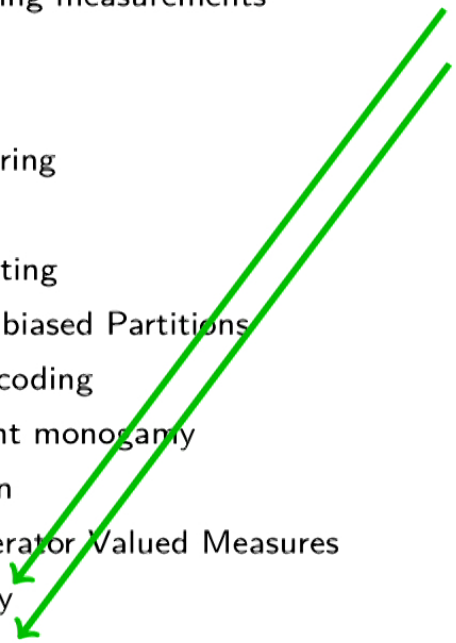
Spekkens' toy model reproduces many of the quantum predictions

In the toy model

- Noncommuting measurements
- Uncertainty
- Interference
- Remote steering
- No cloning
- No broadcasting
- Mutually Unbiased Partitions
- Superdense coding
- Entanglement monogamy
- Teleportation
- Positive Operator Valued Measures
- \+ Contextuality
- \+ Nonlocality
- . . .

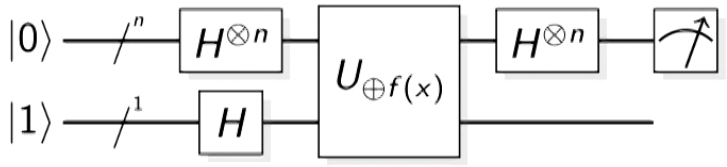
Not in the toy model

- ~~Contextuality~~
- ~~Nonlocality~~
- Quantum-computational speedup?
- Continuum of states
- . . .



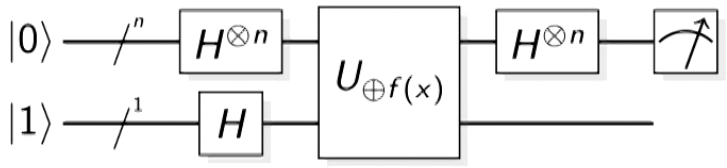
LINKÖPING UNIVERSITY

How much contextuality do you need to do Deutsch-Jozsa for 3-bit inputs?

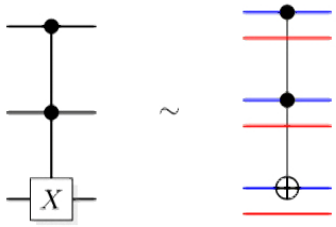


- Determine if f is balanced or constant
- One- or two-bit input function oracles are in the Clifford group, and also in Spekkens' toy model
- Three-bit input or larger function oracles are not

How much contextuality do you need to do Deutsch-Jozsa for 3-bit inputs?

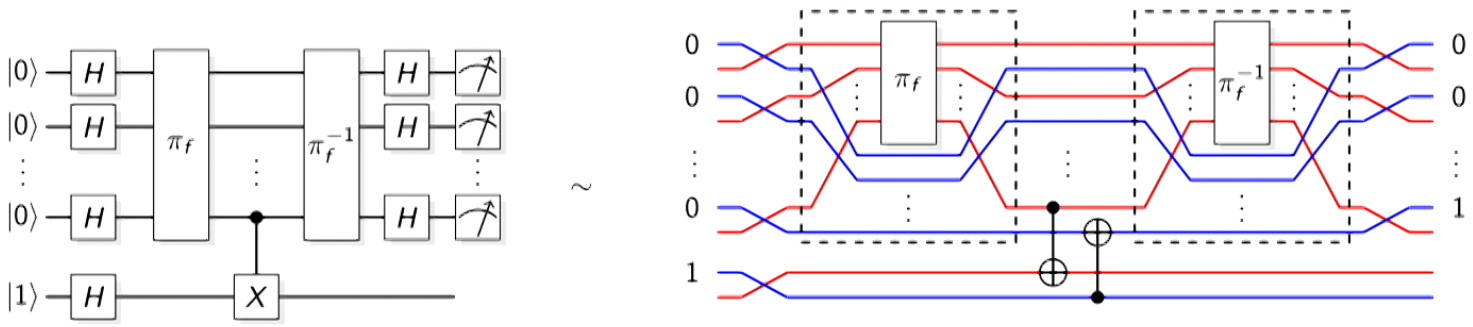


- Determine if f is balanced or constant
- One- or two-bit input function oracles are in the Clifford group, and also in Spekkens' toy model
- Three-bit input or larger function oracles are not
- No contextuality is needed but nonetheless, three-bit input oracles can be realized



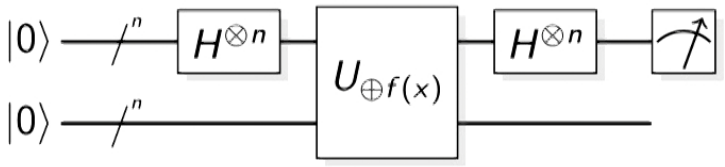
LINKÖPING UNIVERSITY

Deutsch-Jozsa in our framework

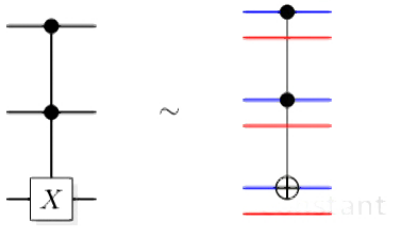


- A balanced oracle has the center CNOT, a constant oracle has not
- Measurement after the final Hadamards will reveal whether it is balanced or constant
- The overhead is constant, a factor 2

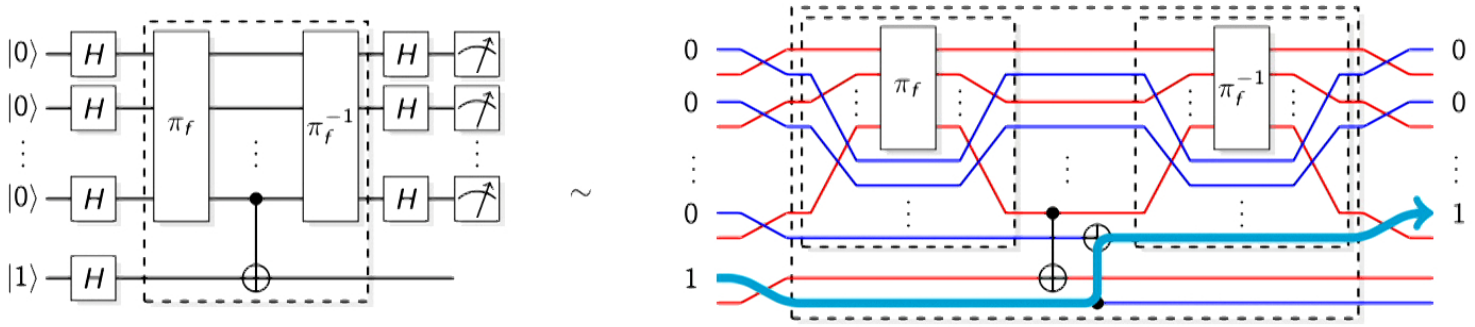
Simon's algorithm, that finds the generator of a hidden order-two group, can also be efficiently realized



- The function is such that $f(x) = f(y)$ iff $x = y \oplus s$ for a secret s
- Determine if s is nonzero (or find s)
- Needs an exponential number of classical function calls, but only $O(n)$ quantum oracle calls
- Our framework also finds s in $O(n)$ oracle calls

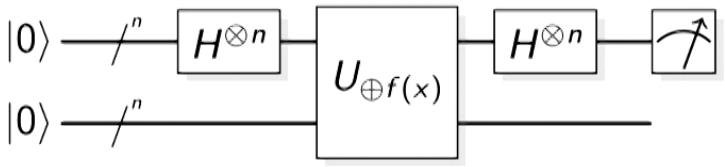


Deutsch-Jozsa in our framework

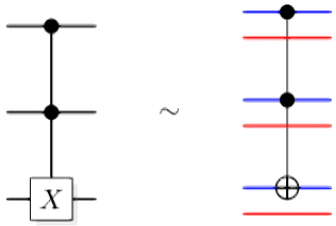


- A balanced oracle has the center CNOT, a constant oracle has not
- Measurement after the final Hadamards will reveal whether it is balanced or constant
- The overhead is constant, a factor 2

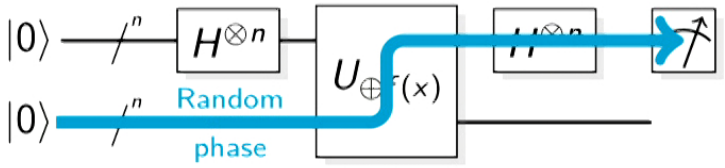
Simon's algorithm, that finds the generator of a hidden order-two group, can also be efficiently realized



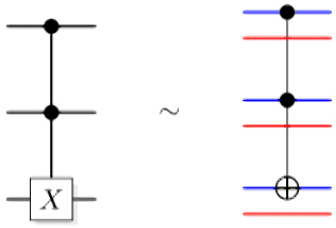
- The function is such that $f(x) = f(y)$ iff $x = y \oplus s$ for a secret s
- Determine if s is nonzero (or find s)
- Needs an exponential number of classical function calls, but only $O(n)$ quantum oracle calls
- Our framework also finds s in $O(n)$ oracle calls



Simon's algorithm, that finds the generator of a hidden order-two group, can also be efficiently realized



- The function is such that $f(x) = f(y)$ iff $x = y \oplus s$ for a secret s
- Determine if s is nonzero (or find s)
- Needs an exponential number of classical function calls, but only $O(n)$ quantum oracle calls
- Our framework also finds s in $O(n)$ oracle calls



Surely this is impossible, there are theorems!

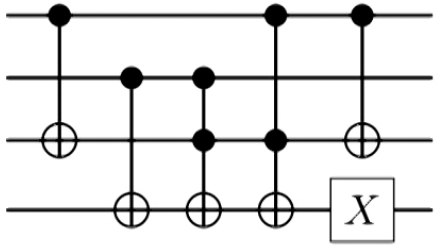
Theorem (Simon, 1997)

Let O be an oracle constructed as follows: for each n, a random n-bit string $s(n)$ and a random bit $b(n)$ are uniformly chosen from $\{0,1\}^n$ and $\{0,1\}$, respectively. If $b(n) = 0$, then the function $f_n : \{0,1\}^n \to \{0,1\}^n$ chosen for O to compute on n-bit queries is a random function uniformly distributed over permutations on $\{0,1\}^n$; otherwise, it is a random function uniformly distributed over two-to-one functions such that $f_n(x) = f_n\big(x \oplus s(n)\big)$ for all x, where $\oplus$ denotes bitwise exclusive-or. Then any PTM that queries O no more than $2^{n/4}$ times cannot correctly guess $b(n)$ with probability greater than $(1/2) + 2^{-n/2}$, over choices made in the construction of O.

The important detail here is that $f_n : \{0,1\}^n \to \{0,1\}^n$, which is not the case in quantum computation. Nor is it the case in our framework.

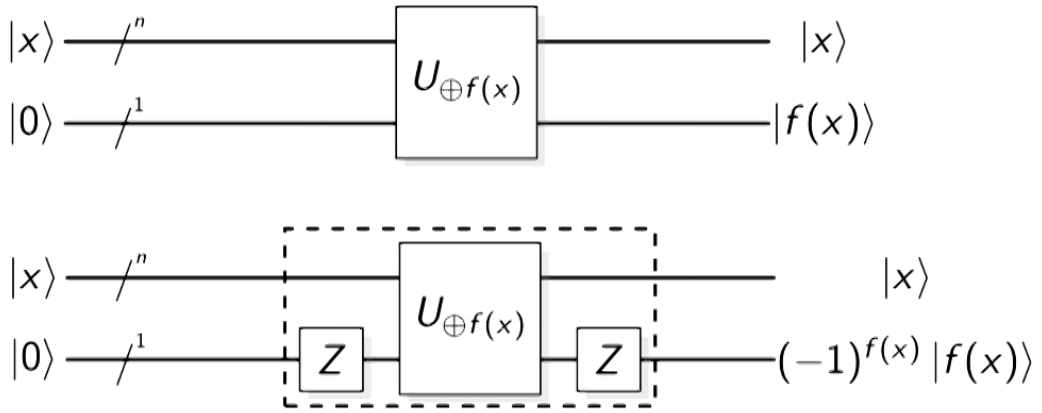
What if I give you the function as a reversible gate array?

x	$f(x)$
0	0
1	0
2	1
3	0
4	1
5	0
6	1
7	1



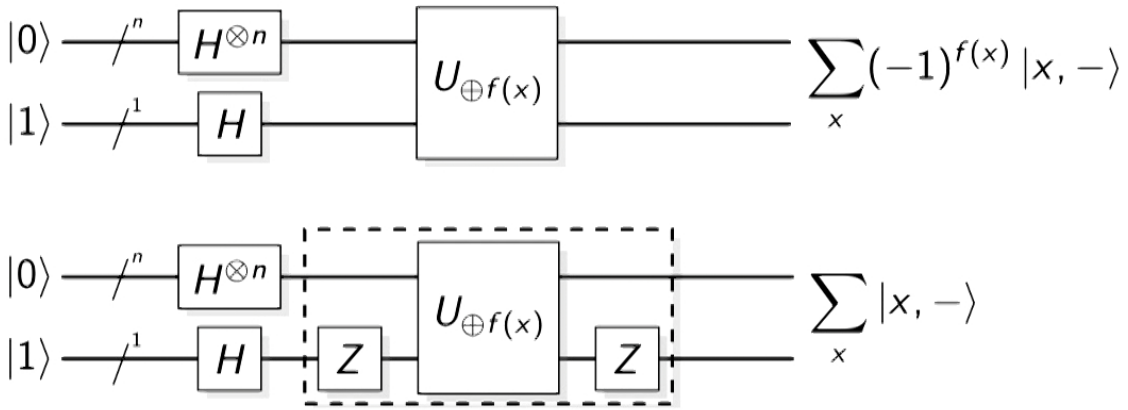
- In most cases, our simulation will work, but not always (because our Toffoli gives systematic errors)
- But you are outside the oracle paradigm
- There are $\binom{2^n}{2^{n-1}} \sim 2^{2^n}$ possible functions (for Deutsch-Jozsa, even more in Simon's), and many more possible gate arrays
- It is not even possible to transfer that much information from you to me in polynomial time

Superposition, or quantum parallelism, is not a good explanation



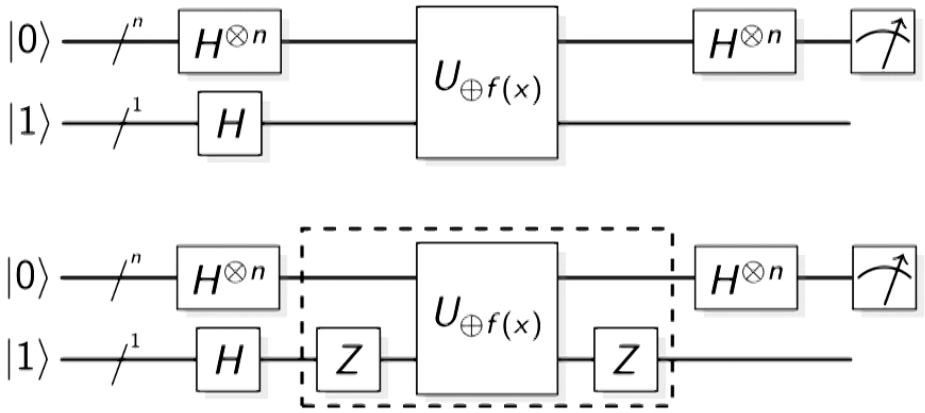
- These two realize the same function, and both are unitary

Superposition, or quantum parallelism, is not a good explanation



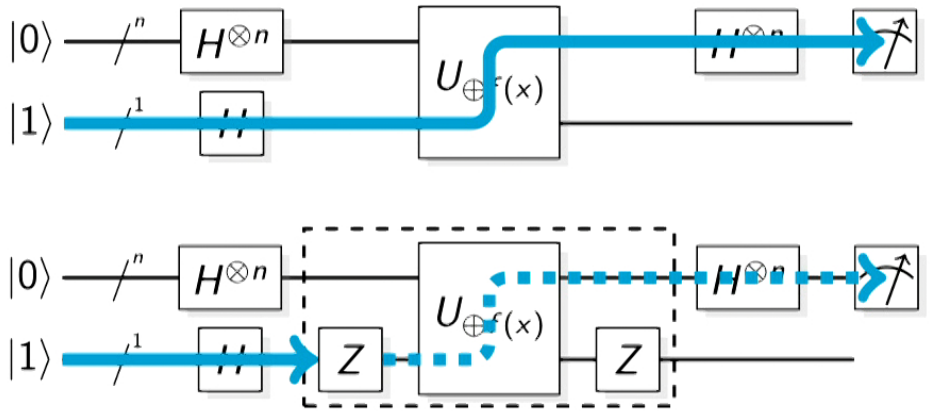
- These two realize the same function, and both are unitary
- But the quantum algorithm gives radically different outcomes

Superposition, or quantum parallelism, is not a good explanation



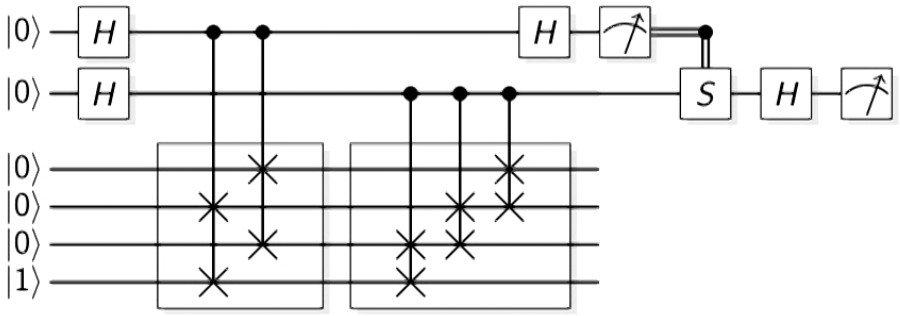
- These two realize the same function, and both are unitary
- But the quantum algorithm gives radically different outcomes
- "All the possible input values" are still superposed, but the lower quantum algorithm fails

Superposition, or quantum parallelism, is not a good explanation

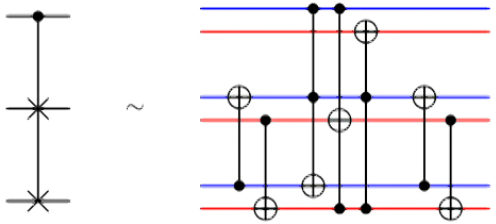


- These two realize the same function, and both are unitary
- But the quantum algorithm gives radically different outcomes
- "All the possible input values" are still superposed, but the lower quantum algorithm fails

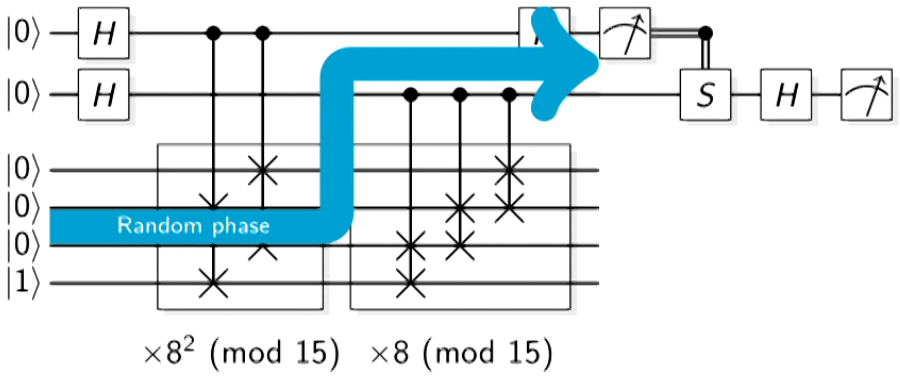
What about Shor's algorithm?



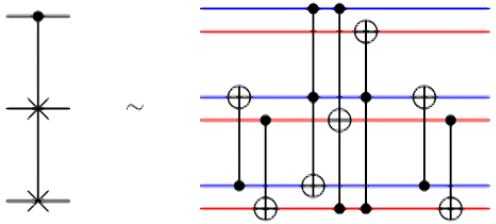
- Find period of a^x mod N
- Corresponding classical algorithm is $O(e^n)$
- Best known classical algorithm is $O(e^{\sqrt[3]{n}})$
- Quantum algorithm is $O(n^3)$
- If you do not "compile" the circuitry, even factoring 15 is outside the Clifford group



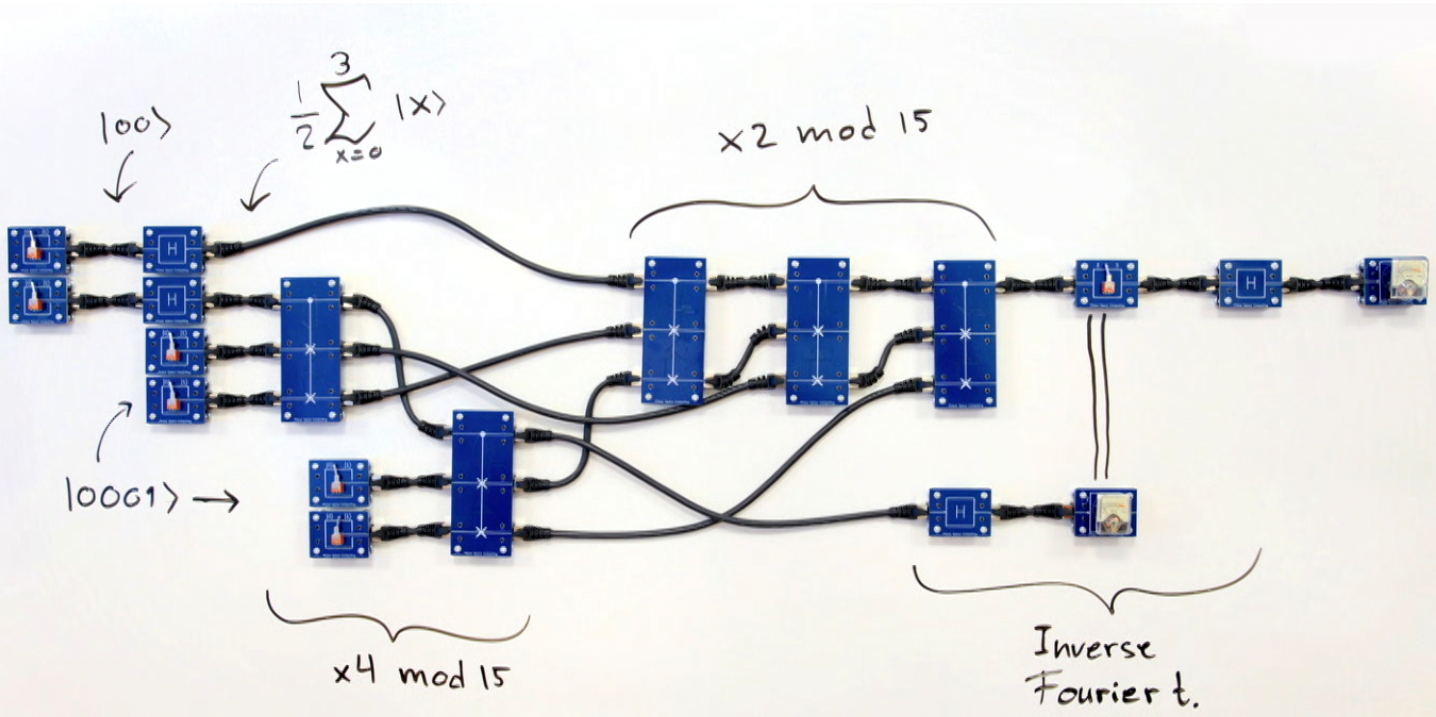
What about Shor's algorithm?



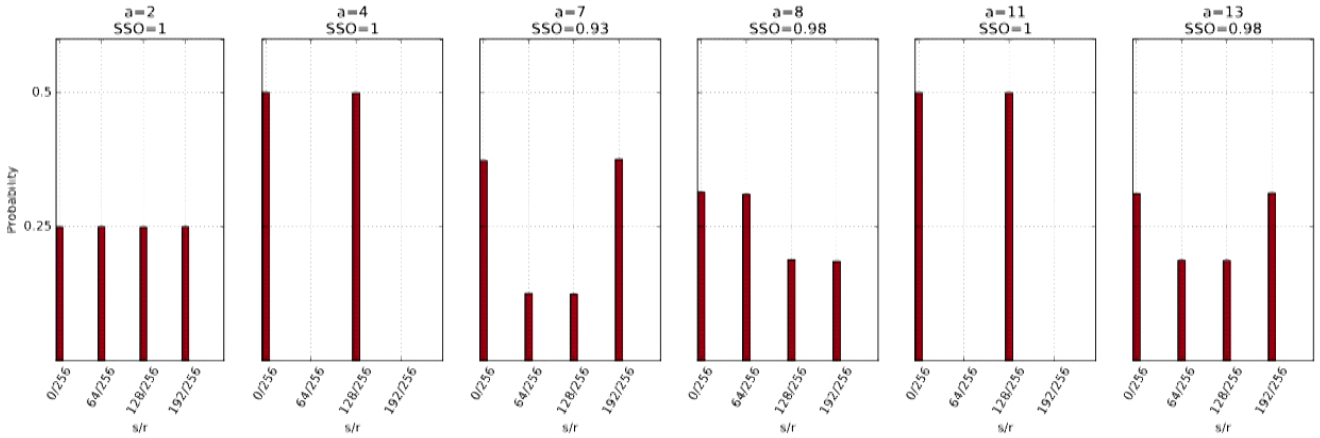
- Find period of a^x mod N
- Corresponding classical algorithm is $O(e^n)$
- Best known classical algorithm is $O(e^{\sqrt[3]{n}})$
- Quantum algorithm is $O(n^3)$
- If you do not "compile" the circuitry, even factoring 15 is outside the Clifford group



Shor's algorithm in complementary pass-transistor logic

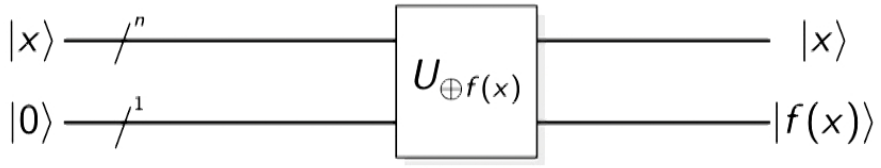


Shor's algorithm in complementary pass-transistor logic



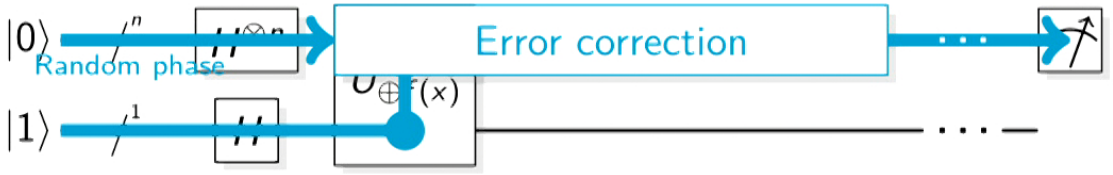
- There are systematic errors
- (. . . but they are on par with the other state-of-the-art experiments)
- Asymptotic behavior is unclear (at the moment)

Grover's algorithm



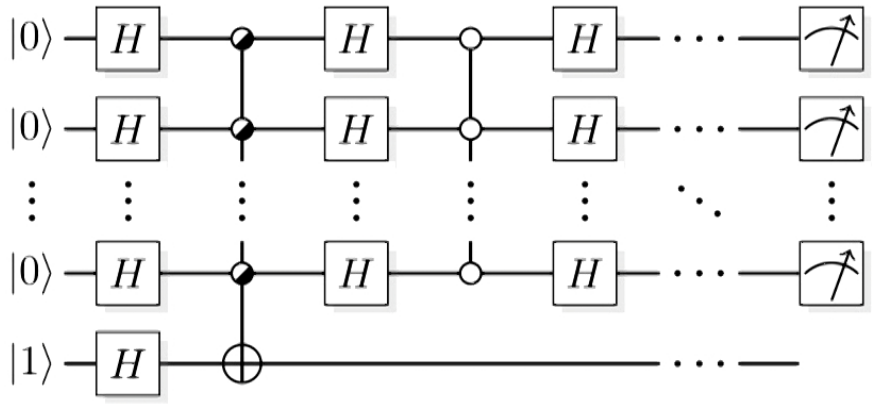
- Find x such that $f(x) = 1$
- Classical algorithm is $O(2^n)$

Grover's algorithm



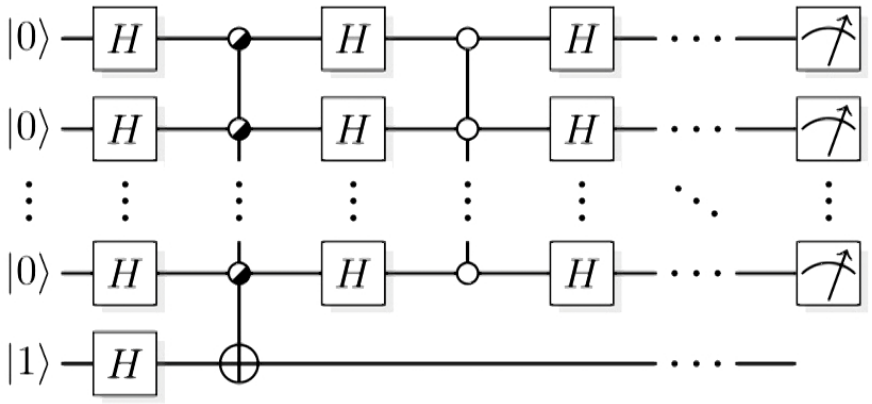
- Find x such that $f(x) = 1$
- Classical algorithm is $O(2^n)$
- Quantum algorithm is $O(2^{n/2})$

Grover's algorithm

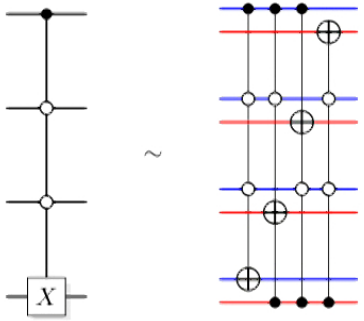


- Find x such that $f(x) = 1$
  (Deduce the color of the controls)
- Classical algorithm is $O(2^n)$
- Quantum algorithm is $O(2^{n/2})$

Grover's algorithm



- Find x such that $f(x) = 1$
  (Deduce the color of the controls)
- Classical algorithm is $O(2^n)$
- Quantum algorithm is $O(2^{n/2})$
- QSL one-shot algorithm is $O(2^n/n)$

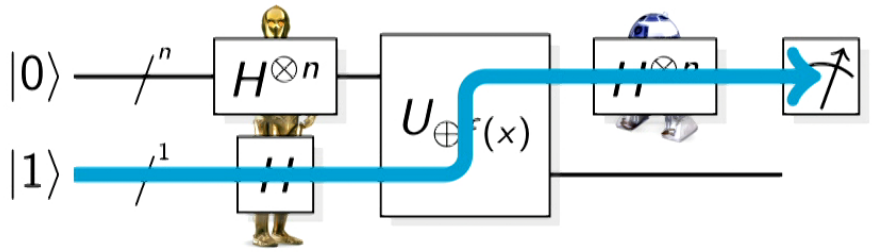


Conclusions

- Quantum algorithms *realize* the evaluation of transformed functions
- Information from the "target" system serves as input in the evaluation
- This information must influence the "control" output (~reversibility)

- You have access to an additional degree of freedom to use in your calculation
- The additional degree of freedom reveals some structure of the function
- Some care is needed when formulating separation theorems

Superposition and interference are not the resources you are looking for



Entropy 2019, 21(8), 800
https://doi.org/10.3390/e21080800

www.phasespacecomputing.com

LINKÖPING UNIVERSITY