Quantum Inf Process (2017) 16:233
DOI 10.1007/s11128-017-1679-7

Efficient classical simulation of the Deutsch–Jozsa and Simon's algorithms

Niklas Johansson[1] · **Jan-Åke Larsson**[1]

Received: 10 February 2017 / Accepted: 26 July 2017 / Published online: 12 August 2017

Abstract A long-standing aim of quantum information research is to understand what gives quantum computers their advantage. This requires separating problems that need genuinely quantum resources from those for which classical resources are enough. Two examples of quantum speed-up are the Deutsch–Jozsa and Simon's problem, both efficiently solvable on a quantum Turing machine, and both believed to lack efficient classical solutions. Here we present a framework that can simulate both quantum algorithms efficiently, solving the Deutsch–Jozsa problem with probability 1 using only one oracle query, and Simon's problem using linearly many oracle queries, just as expected of an ideal quantum computer. The presented simulation framework is in turn efficiently simulatable in a classical probabilistic Turing machine. This shows that the Deutsch–Jozsa and Simon's problem do not require any genuinely quantum resources, and that the quantum algorithms show no speed-up when compared with their corresponding classical simulation. Finally, this gives insight into what properties are needed in the two algorithms and calls for further study of oracle separation between quantum and classical computation.

1 Introduction

Quantum computational speed-up has motivated much research to build quantum computers, to find new algorithms, to quantify the speed-up, and to separate classical from quantum computation. One important goal is to understand the reason for quan-

✉ Jan-Åke Larsson
jan-ake.larsson@liu.se

Niklas Johansson
niklas.johansson@liu.se

[1] Linköping University, Linköping, Sweden

tum computational speed-up, to understand what resources are needed to do quantum computation. Some candidates for such necessary resources include superposition and interference [1], entanglement [2], nonlocality [3], contextuality [4–6], and the continuity of state-space [7]. Here we will attempt to point out one such resource, sufficient for some of the known algorithms, and we will do this by simulation of the quantum algorithms in a classical Turing machine. Simulation of small quantum-mechanical systems is routinely done on supercomputers world-wide, but classical Turing machine simulations of this kind cannot be expected to be efficient. The complexity of a typical such simulation increases exponentially with the quantum system size. In other words, the complete quantum-mechanical behaviour of a quantum computation cannot be simulated efficiently [1].

In this paper we consider the possibility that it might not be necessary to reproduce the complete quantum-mechanical behaviour to efficiently simulate some quantum computation algorithms. In particular, we look at two so-called *oracle* problems: the Deutsch–Jozsa problem [8,9] and Simon's problem [10,11], thought to show that quantum computation is more powerful than classical computation. There are quantum algorithms that solve these problems efficiently, and here we present a framework that can simulate these quantum algorithms, *Quantum Simulation Logic* (QSL), that itself can be efficiently simulated on a classical probabilistic Turing machine. This shows that no genuinely quantum resources are needed to solve these problems efficiently, but also tells us which properties are actually needed: the possibility to choose between two aspects of the same system within which to store, process, and retrieve information.

The quantum algorithm for the Deutsch–Jozsa problem has been used extensively for illustrating experimental realizations of a quantum computer, while Simon's algorithm served as inspiration for the later Shor's algorithm [12]. Both problems involve finding certain properties of a function f, and this can be done efficiently in a quantum computer given a quantum gate that implements the function, known as an *oracle*. When using a classical-function oracle (an oracle that is only allowed to act on classical bits as input and giving classical bits as output), the Deutsch–Jozsa problem has an efficient solution on a classical probabilistic Turing machine [8,9], but for Simon's problem it has been proven [10,11] that no efficient solution exists. Our QSL simulation of Simon's algorithm may seem to contradict this theorem, since it runs on a classical probabilistic Turing machine, but there is no contradiction, because the function implemented by the QSL oracle is not a map from n classical bits to n classical bits—which the theorem requires—but a map from n QSL systems to n QSL systems. Thus, the theorem does not apply. This is exactly as in the quantum case, where the quantum oracle is a map from quantum systems to quantum systems, also avoiding the prerequisites of the theorem. Both the quantum and QSL oracles are richer procedures than the classical-function oracle, and the main point of this paper is to clarify exactly how they are richer.

The paper is organized as follows: we first present the Deutsch–Jozsa problem and the quantum algorithm. We then construct a simulation of the quantum algorithm, in classical reversible logic. This is followed by a brief discussion of the black-box oracle paradigm, stressing that the simulation completely reproduces the behaviour of the Deutsch–Jozsa quantum algorithm in this paradigm. The final part of the paper

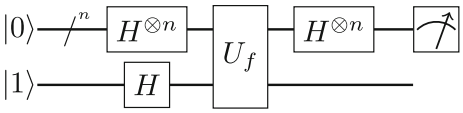

Fig. 1 Quantum circuit for the Deutsch–Jozsa algorithm. This circuit uses an n-qubit query-register prepared in the state $|0\rangle$, and an answer-register prepared in $|1\rangle$. It proceeds to apply Hadamard transformations to each qubit. The function f is embedded in an oracle U_f, and this is followed by another Hadamard transformation on each query-register qubit. The measurement at the end will test positive for $|0\rangle$ if f was constant, and negative if f was balanced

is devoted to Simon's problem, quantum algorithm, and simulation, followed by the conclusions of the paper.

2 The Deutsch–Jozsa problem and quantum algorithm

The Deutsch–Jozsa problem is the following: Suppose that you are given a Boolean function $f : \{0, 1\}^n \to \{0, 1\}$ with the promise that it is either constant or balanced. The function is constant if it gives the same output (1 or 0) for all possible inputs, and it is balanced if it gives the output 0 for half of the possible inputs, and 1 for the other half. Your task is now to distinguish between these two cases [9,13]. Given such a function, a classical Turing machine can solve this problem by checking the output for $2^{n-1}+1$ values of the input; if all are the same, the function is constant, and otherwise balanced. A stochastic algorithm with k randomized function queries gives a bounded-error probability [9] less than 2^{1-k}, showing that the problem is in the complexity class **BPP** (Bounded-error Probabilistic Polynomial time solvable problems).

A quantum computer obtains the function as a unitary that implements the function, a quantum oracle. The requirements on the oracle are simple: that it adds the function output onto the answer-register qubit for computational basis inputs, and that the quantum phase is preserved in the process, so that

$$|x\rangle\,|y\rangle \overset{U_f}{\mapsto} |x\rangle\,|y \oplus f(x)\rangle\,. \tag{1}$$

This corresponds to the classical single-input function query: if the answer-register was flipped, then the function value is 1 for that input x. Given this quantum oracle the Deutsch–Jozsa algorithm [9,13] can solve the problem with a single query (see Fig. 1). In this algorithm Hadamards are applied to the query- and answer-register states, creating a quantum superposition of all possible inputs to the oracle. When the oracle is applied, this superposition is unchanged (modulo global phase) if the function is constant, and changed to a different superposition if the function is balanced,

$$\begin{aligned}\sum_x |x\rangle\left(|0\rangle - |1\rangle\right) &\overset{U_f}{\mapsto} \sum_x |x\rangle\left(|f(x)\rangle - |1 \oplus f(x)\rangle\right)\\ &= \sum_x (-1)^{f(x)}\,|x\rangle\left(|0\rangle - |1\rangle\right).\end{aligned} \tag{2}$$

This important phenomenon is sometimes called "phase kick-back" [13]. A change can be detected by again applying Hadamards on the query-register and measuring

in the computational basis. If the function is constant, the query-register state after the Hadamards is $|0\rangle$, and otherwise the query-register state is orthogonal to $|0\rangle$. A computational basis measurement will reveal this, in the ideal case with zero error probability, showing that the problem is in **EQP** (Exact or Error-free Quantum Polynomial time solvable problems [15]). Thus, instead of an exponential number of calls to the function from a deterministic classical Turing machine, a single query of the quantum oracle is sufficient.

3 Quantum Simulation Logic

Having studied the Deutsch–Jozsa algorithm in some detail, we will now proceed to construct Quantum Simulation Logic (QSL), an approximate simulation of the quantum states and gates.

To simulate qubits we will use pairs (b_z, b_x) containing two classical bits, a "computational" bit b_z and a "phase" bit b_x. These constitute the elementary systems of our QSL framework, the "QSL bits". State preparation of a computational single qubit state $|k\rangle$ is associated with preparation of (k, X) where X is a random evenly distributed bit. Measurement in the computational basis is associated with readout of the computational bit followed by randomization of the phase bit, somewhat like the uncertainty relation or really measurement disturbance as seen in quantum mechanics. Accordingly, measurement of the phase bit will be followed by a randomization of the computational bit. These constructions of state preparation and measurement prohibits exact preparation and readout of the system; the upper limit is one bit of information per QSL bit (b_z, b_x).

It is relatively simple to simulate the single qubit gates used in the quantum algorithms: an X gate inverts the computational bit (preserving the value of the phase bit b_x), a Z gate inverts the phase bit (preserving b_z), and a H gate switches the computational and phase bits, see Fig. 2b–d. These are constructed so that the resulting gates obey the quantum identities $XX = ZZ = HH = I$ and $HZH = X$. It is also

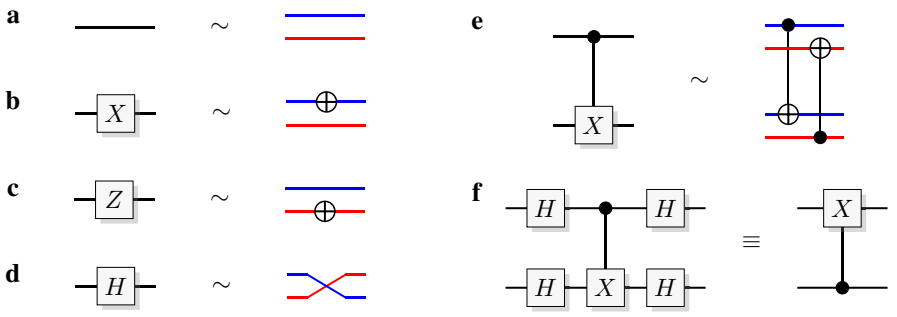

Fig. 2 QSL circuitry. **a** Identity operation, a qubit is simulated by a pair of classical bits (computational bit in blue, phase bit in red). **b** QSL construction of a NOT gate. **c** QSL construction of a Z-gate. **d** QSL construction of a Hadamard gate. **e** QSL construction of a CNOT gate. **f** Identity valid both for qubit and QSL CNOT gates

possible to define QSL Y and "phase" gates, but we will not discuss these and their associated identities here as they are not needed for the present task. A simple circuit is to prepare the state $|0\rangle$, apply a H gate, and measure in the computational basis. This will give a random evenly distributed bit as output.

A system of several qubits can now be associated with a system of several QSL bits containing two internal bits each. A QSL simulation of the quantum Controlled-NOT (CNOT) gate can be constructed from two classical reversible logic CNOTs. One classical CNOT connects the computational bits in the "forward" direction, while the other connects the phase bits in the "reverse" direction, directly implementing the phase kick-back, see Fig. 2e. This again is constructed to enable use of the same identities as apply for the quantum CNOT, see Fig. 2f.

Using QSL to simulate a generic qubit circuit is done as follows:

1. **State preparation.** Prepare n QSL systems (pairs of classical bits) in states $|0\rangle$ or $|1\rangle$ (using a random value for the phase bit).
2. **Transformation.** Apply QSL gates (as described above) in an array of the same form as the quantum circuitry.
3. **Measurement.** Measure in the computational basis (readout the computational bit value and randomize the phase bit).

The QSL framework is inspired by a construction of Spekkens [16] that captures many, but not all, properties of quantum mechanics. So far, the presented framework is completely equivalent to Spekkens' model (the H gate appears in a paper by Pusey [17]). Both frameworks are efficiently simulatable on a classical Turing machine (see above and Ref. [16]). Furthermore, both have a close relation to stabilizer states and Clifford group operations [17], and perhaps more interestingly, both frameworks capture some properties of entanglement, enabling protocols like super-dense coding and quantum-like teleportation, but cannot give all consequences of entanglement, most importantly, cannot give a Bell inequality violation.

4 Simulation of the Deutsch–Jozsa algorithm in Quantum Simulation Logic

The QSL versions of the quantum algorithm will use the same setup as in Fig. 1. For $n = 1$ the oracle for the balanced function $f(x) = x$ is a CNOT, which means that Fig. 2f contains the whole Deutsch–Jozsa algorithm when using the prescribed initial state $|0\rangle|1\rangle$. With this initial state, the query-register measurement result would be 1, so that this oracle gives the same output from the QSL algorithm as from the quantum algorithm. Indeed, all oracles for $n = 1$ and $n = 2$ only use CNOT and X gates, so their simulation will give the same behaviour as the quantum oracles. For $n = 1$ and $n = 2$, the Deutsch–Jozsa algorithm is already known to have an implementation that does not rely on quantum resources [18], and also, the gates used so far can be found within the stabilizer formalism, and here the Gottesmann–Knill theorem tells us that they can be simulated (efficiently) [19].

However, for $n \geq 3$, the Deutsch–Jozsa oracle needs the Toffoli gate, and since the Toffoli gate cannot be efficiently simulated using the stabilizer formalism [19], nor is

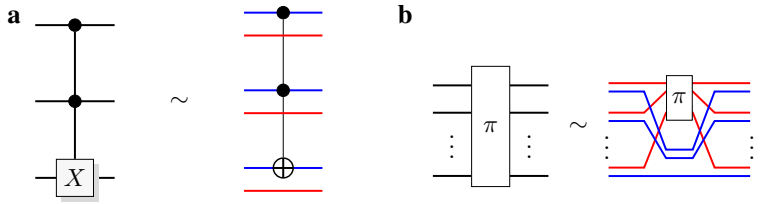

Fig. 3 QSL circuitry. **a** QSL construction of a Toffoli gate. **b** QSL construction of a general permutation

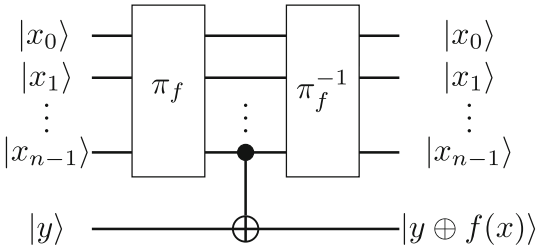

Fig. 4 Oracle example for balanced functions. The arbitrary permutation $\pi = \sum_x |\pi(x)\rangle\langle x|$ of the computational basis states is constructed from CNOT and Toffoli gates [14] ($\pi^{-1} = \pi^\dagger$). At the center is a CNOT gate from the most significant qubit $|x_{n-1}\rangle$ to the answer-register $|y\rangle$. With π as the identity permutation, the oracle performs the balanced function f' that is 1 for all inputs with the most significant bit set. Any other balanced function $f(x) = f'(\pi_f(x))$ can now be generated by choosing a suitable computational basis permutation π_f. For a function that is constant zero, the CNOT gate is omitted, while the constant one function should replace the CNOT with a Pauli-X gate acting on the answer-register

present in Spekkens' framework [16,20], it has so far been believed that the Deutsch–Jozsa algorithm does not have an efficient classical simulation. Here, we need to point out that our task is not to create exact Toffoli gate equivalents, or even simulate the full quantum-mechanical system as such. It suffices to give a working efficient QSL version of the Deutsch–Jozsa algorithm. We therefore choose not to represent Toffoli gates exactly, but design the gate so that it implements a classical Toffoli over the computational bits, and some other gate in the phase bits. For simplicity we choose the identity map for the phase bits, see Fig. 3a.

Building a general computational-state permutation π from QSL Toffoli gates will result in a mapping that only changes the computational bits, leaving the phase bits unchanged, see Fig. 3b. The quantum oracle of Fig. 4 can immediately be translated into an oracle within the QSL framework, see Fig. 5. The balanced function oracle leaves the phase bits unchanged except for the one belonging to the most significant QSL bit in the query-register. A constant function oracle leaves all phase bits unchanged. Since Hadamard gates are included before and after the oracle, measurement of the computational bits will reveal the oracle's effect on the phase bits of the query-register, solving the problem by only one oracle query.

This QSL algorithm uses equally many QSL bits and QSL gates as the quantum algorithm uses qubits and quantum gates. The time and space complexity are therefore identical to the complexity of the quantum algorithm. The QSL algorithm can be efficiently simulated on a classical probabilistic Turing machine, using two classical

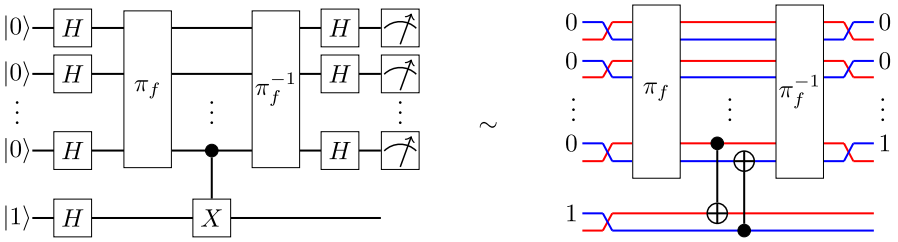

Fig. 5 Translation of the Deutsch–Jozsa quantum algorithm for a balanced function into the QSL framework. The phase kick-back of the CNOT changes the most significant phase bit of the query-register, while the π gates does not change the phase bits at all. The final Hadamard gate will make the computational base measurement reveal the changed value

bits for each simulated qubit and at most twice as many classical reversible gates as quantum gates. The error probability is 0, and in fact, the same correct output is generated for all possible random values in the probabilistic machine, so that the machine's random value could be replaced with a constant value without destroying the simulation. Therefore, this QSL algorithm can be simulated on a classical *deterministic* Turing machine, showing that, relative to the oracle, the Deutsch–Jozsa problem is in **P** (Polynomial time solvable problems).

The problem is solvable in one QSL oracle query because of the oracle's effect on the phase bits of the query-register, reproducing the quantum-mechanical phase kick-back. The phase bits of the query-register reveals the inner structure of the oracle. One may question if it is fair to compare our construction to a *classical function*, since it is clear that this construction results in an oracle that allows one to retrieve either function output or function structure. But then one must also question the comparison of the quantum algorithm to the classical-function algorithm, since the quantum oracle also allows one to retrieve either function output or function structure. On the other hand, the comparison between the QSL simulation and the *quantum algorithm* really *is* a fair comparison since the choice of what to reveal is present in both oracles; both oracles actually result in the same algorithm. And in this comparison, there is no quantum advantage with respect to the classical simulation of QSL.

5 Equal requirements on quantum and QSL oracles

It has been suggested that the construction of the QSL oracle invalidates the simulation: the above-used simple construction of the QSL oracle contains a single CNOT between query- and answer-registers for a balanced function, and does not for a constant function. It seems simple to check for presence/absence of the CNOT in the gate array. With the present design of QSL, CNOT presence or absence is in fact required for the algorithm to work. There are two points to make here.

First, we must stress that the Deutsch–Jozsa problem is stated in the *black-box oracle paradigm* [9,13]. The quantum speed-up itself is only obtained in the black-box oracle paradigm: if the explicit gate construction of the function is available (in a "white-" or "transparent-box paradigm"), the same simple solution may be usable in the fully

quantum case, for example in the oracle construction of Fig. 4. Oracles built from this design, using classical reversible logic, quantum, or QSL gates will *all* indicate that the function is balanced or constant from presence or absence of the CNOT, without the need for any function query. For this choice of (transparent-box) "oracle" and many others including most or even all quantum-experimental implementations to date, there will be no quantum speed-up when comparing to this simple "solution" of the problem. However, the appropriate comparison between quantum and classical algorithms is that between algorithms that use only the inputs and outputs of the function or oracle, not the structure of the gate array at hand [9,13], corresponding to the black-box oracle paradigm. Only in this paradigm does the generic quantum speed-up remain. And in this paradigm, the QSL simulation reproduces the quantum behaviour.

Second, the *essential requirement* on these black-box oracles is not structural, neither in quantum computation nor in QSL simulation. The essential requirement is presence of phase kick-back. For a quantum oracle, the standard way to enable phase kick-back is to enforce phase preservation as hinted in Eq. (1), or more explicitly

$$\sum_{x,y} c_{x,y} \left|x\right\rangle \left|y\right\rangle \overset{U_f}{\mapsto} \sum_{x,y} c_{x,y} \left|x\right\rangle \left|y \oplus f(x)\right\rangle . \tag{3}$$

It should be stressed that unitarity is not enough of a requirement to preserve phase in the sense of Eqs. (1)–(3), even if the functional map from query- to answer-register is enforced. There are many unitary maps that obey the functional relation in the computational basis, but do not preserve the phase relation. One simple but important example is the unitary U_f' defined by

$$\sum_{x,y} c_{x,y} \left|x\right\rangle \left|y\right\rangle \overset{U_f'}{\mapsto} \sum_{x,y} (-1)^{f(x)} c_{x,y} \left|x\right\rangle \left|y \oplus f(x)\right\rangle . \tag{4}$$

This can easily be constructed from U_f by adding Z gates on both sides on the answer-register system, corresponding to adding identity gates in classical reversible logic. The U_f' unitary gives the correct map from function input to output in the computational basis, but if used as the quantum function oracle, the Deutsch–Jozsa algorithm input would give

$$\begin{aligned} &\sum_{x} \left|x\right\rangle \left(\left|0\right\rangle - \left|1\right\rangle \right) \overset{U_f}{\mapsto} \sum_{x} (-1)^{f(x)} \left|x\right\rangle \left(\left|f(x)\right\rangle - \left|1 \oplus f(x)\right\rangle \right) \\ &= \sum_{x} \left|x\right\rangle \left(\left|0\right\rangle - \left|1\right\rangle \right), \end{aligned} \tag{5}$$

resulting in no change to the state used in the algorithm. Here, the output of the algorithm is 0 independent of the properties of the function. In other words, there is no phase kick-back. With this oracle, the quantum algorithm cannot distinguish between constant and balanced functions, and one will need to resort to the classical algorithms. It may be tempting to try to devise modified quantum algorithms that adjust to the type of oracle received, but alas, there are many unitary maps that obey the functional relation but not the phase relation required in the Deutsch–Jozsa algorithm. Testing

for the type of oracle is nontrivial and even with a discrete set of phases, the set of possible unitaries is exponential in size. The phase relation that enables the quantum speed-up restricts this set to a minimum; the remaining oracle unitaries all enable phase kick-back.

The QSL framework presented above is deliberately chosen as simple as possible, to make it clear what properties of quantum mechanics is really needed for the two algorithms under study (Simon's algorithm follows below). A direct consequence of this simplicity is that the phase relation requirement of quantum computation, that enables phase kick-back, is translated into a simple requirement on the QSL oracles, presence or absence of a CNOT between query- and answer-register. In quantum computation, the restriction that enables phase kick-back selects a small subset of all oracle constructions that give the correct functional map for the computational basis, and in QSL, exactly in the same way, the restriction that enables phase kick-back selects a small subset of all QSL constructions that give the correct functional map for the computational bits.

The requirements are exactly the same on quantum and QSL oracles alike: a black-box oracle that enables phase kick-back. The above example proves existence of black-box oracles that enable phase kick-back for the full range of possible functions, both in the quantum and QSL frameworks. And in the black-box oracle paradigm, the central issue is *existence* of oracles with the appropriate properties, not their internal structure, the latter not being available to the algorithms in question.

6 Simon's problem

Another more complicated problem that can be solved faster with quantum computation is Simon's problem, to decide whether a function $f : \{0,1\}^n \rightarrow \{0,1\}^n$ is one-to-one or two-to-one invariant under the bit-wise exclusive-OR operation with a secret string s (or equivalent, invariant under a nontrivial XOR-mask s [10,11]). In both cases, $f(x) = f(x')$ if and only if $x' = x \oplus s$ (bit-wise addition modulo 2); if $s = 0$ then f is one-to-one and otherwise f is two-to-one. When using the classical-function as an oracle, a classical probabilistic Turing machine needs an exponential number of oracle evaluations for this task due to a theorem by Simon [10,11], showing that relative to this oracle, Simon's problem is not in **BPP**.

Simon's quantum algorithm can distinguish these two cases in an expected linear number of oracle queries [10,11], showing that the problem is in **BQP** (Bounded-error Quantum Polynomial time [15]). A circuit diagram of the quantum subroutine and one realization of the oracle used in Simon's algorithm is shown in Fig. 6a, b. An explicit construction of the gate U_s can be obtained by choosing a basis $\{v^k\}$ for the part of the binary vector space orthogonal to s. If $s = 0$, the basis consists of n vectors, otherwise $n - 1$ vectors. For every entry $v^k_j = 1$, connect a CNOT from query-register bit j to ancilla-register k (in realizations, it is beneficial to choose v^k to minimize the number of CNOTs [21]). The permutation after U_s enables all possible functions. See Fig. 7 for a simple example.

The quantum subroutine output is a random bit vector y such that $y \cdot s = 0 \bmod 2$ (bit-wise dot product), uniformly distributed over all such bit vectors y. After an

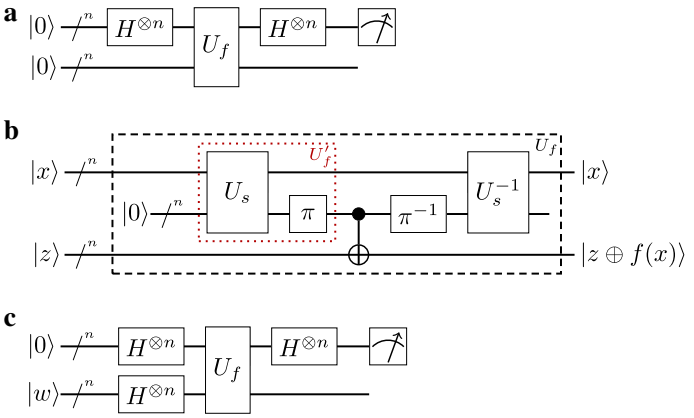

Fig. 6 Circuits for the quantum subroutine of Simon's algorithm. **a** The subroutine itself. The n-qubit query- and answer-registers are prepared in the state $|0\rangle|0\rangle$. Apply Hadamard transformations to the query-register, the function f embedded in an oracle U_f, and finally another Hadamard. The measurement at the end will give a random bit sequence y such that $y \cdot s = 0 \pmod 2$. **b** Example of oracle construction. The oracle U_f uses an internal n-qubit ancilla, and the CNOT denotes a string of bit-wise CNOTs from each ancilla bit to the corresponding answer-register bit. The unitary U_s is constructed from CNOTs [21] (see the main text) so that it implements one representative function for each specific secret string s, and the permutation π gives access to all functions f. The dotted gate combination U'_f implements $|x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle$ as required in Simon [10,11], and the additional circuitry in U_f is there to map $|x\rangle|z\rangle$ to $|x\rangle|z \oplus f(x)\rangle$ and reset the ancilla [22]. **c** The modified subroutine for the deterministic algorithm. The n-qubit query- and answer-register are prepared in the state $|0\rangle|w\rangle$. The circuit applies Hadamard transformations to the query- *and answer-registers*, the function f embedded in an oracle U_f, and finally another Hadamard before measurement

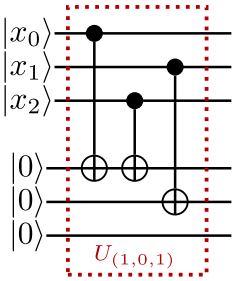

Fig. 7 Example construction of U_s when $s = (1, 0, 1)$, using the basis $\{v^k\} = \{(1, 0, 1), (0, 1, 0)\}$ which spans the binary vector space orthogonal to s (note that the scalar product is defined mod 2). The construction uses three CNOTs, one from the first query-register qubit to the first ancilla qubit, one from the third query-register qubit to the first ancilla qubit, and one from the second query-register qubit to the second ancilla qubit

expected number of iterations that is linear in n, the subroutine will have produced $n - 1$ linearly independent values of y [10,11], so that the resulting linear system of equations for s can be solved, giving one nontrivial solution s^*. If the function is one-to-one, this solution is just a random bit sequence, but if the function is two-to-one the solution is the actual secret string s. Querying the function for $f(0)$ and $f(s^*)$ will give equal values if the function is two-to-one, and unequal values if the function is

one-to-one, solving Simon's problem. Note that as a consequence of this method of solving Simon's problem one also obtains the secret string s.

The QSL versions of the subroutine and oracle is again obtained by substituting the quantum gates with their corresponding QSL gates. Inserting the QSL CNOT into U_s gives the explicit map $(x, p), (0, a) \overset{U_s}{\mapsto} \big(x, p \oplus g'(a)\big), \big(f'(x), a\big)$ where

$$\begin{aligned} f'_j(x) &= x \cdot v^j \pmod 2 = f'_j(x \oplus s), \\ g'_j(a) &= \sum_k a_k v_j^k \pmod 2. \end{aligned} \tag{6}$$

The permutations π and π^{-1} do not influence the phase, so the complete oracle U_f is the map $(x, p), (z, w) \mapsto \big(x, p \oplus g(w)\big), \big(z \oplus f(x), w\big)$ where

$$\begin{aligned} f(x) &= \pi(f'(x)) = f(x \oplus s), \\ g(w) &= g'(a) \oplus g'(w \oplus a) = g'(w) \end{aligned} \tag{7}$$

By the use of Hadamard gates, the simulation of Simon's subroutine sets the query-register phase entering the oracle to $p = 0^n$, while the answer-register phase w will be uniformly distributed. Measurement of the computational bits after the final Hadamard will therefore give a random bit vector $y = g(w)$ that is orthogonal to s, uniformly distributed over the possible values. This reproduces the quantum predictions exactly, showing that the simulation will work with this subroutine to solve Simon's problem with the same expected linear number of iterations in n. Again, efficient simulation of the QSL framework on a classical probabilistic Turing machine shows that, relative to the oracle, Simon's problem is in **BPP**.

This seems to contradict the above-mentioned theorem of Simon [10,11] that states that relative to a classical-function oracle, Simon's problem is not in **BPP**. However, there is no contradiction, because the QSL oracle is not a map from n classical bits to n classical bits, which the theorem requires, but a map from n QSL systems to n QSL systems. Thus, the theorem does not apply, and there is no contradiction: relative to the new oracle, Simon's problem is not stopped from being in **BPP**. This is because of the additional resource available in QSL systems, the choice between two aspects of the same system, the "computational" or the "phase" bit, within which to store, process, and retrieve information. It is of course crucial that the QSL framework itself can be efficiently simulated on a classical probabilistic Turing machine.

The derandomized quantum algorithm for Simon's problem [23,24] (with zero error probability), that shows that Simon's problem relative to the quantum oracle is in **EQP**, is not immediately usable because it incorporates a modified Grover search [25] which is not known to have an implementation within the QSL framework. The quantum algorithm is intended to remove already seen y and, crucially, prevent measuring $y = 0$ which would be a failed iteration of the algorithm. The modified Grover search is presented [23] as applied to the output of U_f, but since the final step of the Grover search is again U_f, the whole combination can be viewed as U_f preceded by another quantum algorithm. This other quantum algorithm, in a sense, finds an input to U_f such that the coefficient of the term $|0\rangle$ of the output is 0.

Since we do not have a Grover equivalent in our framework, we would like a simpler solution. One way to achieve this is to prepare the initial answer-register state as the Hadamard transformation of a chosen state $|w\rangle$, see Fig. 6c. Iterating $|w\rangle$ over a basis for the bit vector space will produce n values of y that span the bit vector space orthogonal to s. An example is to use the standard basis, for which the outputs can be calculated through Eqs. (6–7) to be the individual v_i, in order. Either this spans the whole space ($s = 0$), or gives a linear system of equations that has one nontrivial solution, equal to s. This QSL algorithm gives deterministic correct outputs just as the QSL Deutsch–Jozsa algorithm, and can therefore also be efficiently simulated on a classical deterministic Turing machine, showing that relative to the oracle, Simon's problem actually lies within **P**.

This linear search procedure works in our setting because the QSL permutation gate π does not influence the phase bits. In the quantum-mechanical case, the quantum gate implementing π does change the phase information, e.g. in the simple case when the quantum π is a CNOT. A more convoluted example that does not correspond to a simple modification of the phase information is when the quantum π is a (non-stabilizer) Toffoli gate. Thus, the derandomized quantum algorithm [23] requires the more advanced modified Grover's search algorithm. However, note that if the oracle is the smaller gate combination U'_f in Fig. 6b that obeys only the original [10,11] requirement $|x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle$ (for $z = 0$), then the quantum version of the simpler algorithm proposed above will work as well, avoiding the modified Grover search.

Also here the simulation makes it clear that Simon's quantum oracle is a richer procedure than the classical-function oracle, since it allows for the retrieval of more information than just the function output. Choosing to view the oracle operation from the computational basis reveals the function output, while choosing the phase basis reveals function structure, in the form of output bit vectors orthogonal to s. Also here, the comparison between the QSL simulation and the quantum algorithm *is* a fair comparison since the choice of what to reveal is present in both oracles; both oracles actually result in the same algorithm. And also in this comparison, there is no quantum advantage with respect to the classical simulation of QSL.

7 Conclusions

In conclusion, we have devised QSL equivalents of the Deutsch–Jozsa and Simon's quantum algorithms. In the quantum algorithm, you are given a black-box quantum oracle, a unitary gate that implements f by acting on qubits, your task is to distinguish two or more families of functions. In the presented QSL algorithms, you are given a black-box QSL gate that implements f by acting on QSL systems, and the same task. Using the QSL framework, we obtain the same success probability and the same time and space complexity as for the quantum algorithms, and these QSL algorithms are in turn efficiently simulatable in classical Turing machines. (Implementations that can be run with very large inputs can be constructed, given that the permutations are implemented as random permutations.) Therefore, the Deutsch–Jozsa and Simon's algorithms cannot any more be used as evidence for a relativized oracle separation

between **EQP** and **BPP**, or even **P**. Both problems are, relative to the oracles, in fact in **P**.

Apparently, the resource that gives these quantum algorithms their power is also available in QSL, but not when using an classical-function oracle. In QSL, each qubit is simulated by two classical bits in a classical probabilistic Turing machine, and in the above oracles one of these bits is used for computing the function, while the other is changed systematically when U_f is applied. This change of phase information can then be revealed by choosing to insert Hadamards before and after the oracle. This choice is present in both quantum systems and our simulation and enables revealing either function output or function structure, as desired. Our conclusion is that the needed resource is the possibility to choose between two aspects of the same system within which to store, process, and retrieve information.

While we still believe that quantum computers *are* more powerful than classical computers, the question arises whether oracle separation really can distinguish quantum computation from classical computation. There are many other examples of quantum-classical oracle separation, [26–28] and some simple cases can be conjectured to have efficient simulations in the QSL framework (e.g. 3-level systems [16] for the three-valued Deutsch–Jozsa problem [26]), but in general the question needs further study. It *is* possible that the technique can be used for direct computation using other quantum algorithms [12,25], but this will take us out of the oracle paradigm. Also, general quantum computation has been conjectured to use genuinely quantum properties such as the continuum of quantum states, or contextuality [5,6], which both are missing from the QSL framework [16]. In any case, neither the Deutsch–Jozsa nor Simon's algorithm needs genuinely quantum mechanical resources.

Acknowledgements The authors would like to thank Peter Jonsson, Chris Fuchs, Markus Grassl, and David Mermin for discussions on the subject. The project has been supported by the Foundational Questions Institute (FQXi) through the Silicon Valley Community Foundation.

References

1. Feynman, Richard P.: Simulating physics with computers. Int. J. Theor. Phys. **21**, 467–488 (1982). doi:10.1007/BF02650179
2. Einstein, A., Podolsky, B., Rosen, N.: Can quantum-mechanical description of physical reality be considered complete? Phys. Rev. **47**, 777–780 (1935). doi:10.1103/PhysRev.47.777
3. Bell, J.S.: On the Einstein–Podolsky–Rosen paradox. In: Physics (Long Island City, NY.) 1, pp. 195–200 (1964). http://philoscience.unibe.ch/documents/TexteHS10/bell1964epr.pdf
4. Kochen, S., Specker, E.P.: The problem of hidden variables in quantum mechanics. J. Math. Mech. **17**, 59–87 (1967)
5. Kleinmann, M., Gühne, O., Portillo, J.R., Larsson, J.-Å., Cabello, A.: Memory cost of quantum contextuality. New J. Phys. **13**, 113011 (2011). doi:10.1088/1367-2630/13/11/113011
6. Howard, Mark, et al.: Contextuality supplies the magic for quantum computation. Nature **510**, 351–355 (2014). doi:10.1038/nature13460
7. Shor, P.W.: Fault-tolerant quantum computation. In: Proceedings of 37th Annual Symposium on Foundations of Computer Science, pp. 56-65 (1996). doi:10.1109/SFCS.1996.548464

8. Deutsch, D.: Quantum theory, the Church–Turing principle and the universal quantum computer. Proc. R. Soc. Lond. A **400**, 97–117 (1985). doi:10.1098/rspa.1985.0070
9. Deutsch, David, Jozsa, Richard: Rapid solution of problems by quantum computation. Proc. R. Soc. Lond. A **439**, 553–558 (1992). doi:10.1098/rspa.1992.0167
10. Simon, D.R.: On the power of quantum computation. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 116–123 (1994). doi:10.1109/SFCS.1994.365701
11. Simon, D.: On the power of quantum computation. SIAM J. Comput. **26**, 1474–1483 (1997). doi:10.1137/S0097539796298637
12. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 124–134 . IEEE Computer Society Press, Santa Fe, NM (1994)
13. Cleve, R., et al.: Quantum algorithms revisited. Proc. R. Soc. Lond. A **454**, 339–354 (1998). doi:10.1098/rspa.1998.0164
14. Nielsen, Michael A., Chuang, Isaac L.: Quantum computation and quantum information. 10th Anniversary Ed. Cambridge University Press, New York, USA (2011). ISBN: 1107002176
15. Bernstein, E., Vazirani, U.: Quantum complexity theory. SIAM J. Comput. **26**, 1411–1473 (1997). doi:10.1137/S0097539796300921
16. Spekkens, Robert W.: Evidence for the epistemic view of quantum states: a toy theory. Phys. Rev. A **75**, 032110 (2007). doi:10.1103/PhysRevA.75.032110
17. Pusey, Matthew F.: Stabilizer notation for Spekkens' toy theory. Found. Phys. **42**, 688–708 (2012). doi:10.1007/s10701-012-9639-7
18. Collins, David, Kim, K.W., Holton, W.C.: Deutsch–Jozsa algorithm as a test of quantum computation. Phys. Rev. A **58**, R1633–R1636 (1998). doi:10.1103/PhysRevA.58.R1633
19. Gottesman, D.: The Heisenberg representation of quantum computers. (1998). arXiv:quant-ph/9807006
20. Johansson, N.: Efficient simulation of Deutsch–Jozsa algorithm. Linköping University, The Institute of Technology, 2015. isrn: LiTHIFM- A-EX–15/2992–SE
21. Tame, M.S., et al.: Experimental realization of a one-way quantum computer algorithm solving Simon's problem. Phys. Rev. Lett. **113**, 200501 (2014). doi:10.1103/PhysRevLett.113.200501
22. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**, 525–532 (1973). doi:10.1147/rd.176.0525
23. Brassard, G., Hoyer, P.: An exact quantum polynomial-time algorithm for Simon's problem. In: IEEE Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems 1999, pp. 12–23 (1997). http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=595153
24. Mihara, T., Sung, S.C.: Deterministic polynomial-time quantum algorithms for Simon's problem. Comput. Complex. **12**, 162–175 (2003). doi:10.1007/s00037-003-0181-z
25. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing. STOC '96. New York, NY, USA: ACM, pp. 212–219 (1996). doi:10.1145/237814.237866
26. Fan, Yale.: A generalization of the Deutsch–Jozsa algorithm to multi-valued quantum logic. In: IEEE 44th International Symposium on Multiple-Valued Logic. Vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, p. 12. (2007). doi:10.1109/ISMVL.2007.3
27. Alagic, G., Moore, C., Russell, A.: Quantum algorithms for Simon's problem over general groups. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, pp. 1217–1224 (2007). isbn: 978-0-89871-624-5. http://dl.acm.org/citation.cfmid=1283383.1283514
28. Oracular Quantum algorithms, Quantum Algorithm Zoo. http://math.nist.gov/quantum/zoo/#oracular. 20