

Do we need a Logic of Quantum Computation?

Matthew Leifer

Perimeter Institute for Theoretical Physics

quant-ph/0509193

Griffith University 1st November 2005

Outline

- 1) What is responsible for the power of Quantum Computing?
- 2) Logic in classical computing: models and complexity
- 3) The logic of quantum computing
- 4) Birkhoff-von Neumann Quantum Logic
- 5) Sequential Quantum Logic
- 6) Testing an SQL Proposition
- 7) Open Questions

1) The Power of QC

Which feature of quantum theory is responsible for the exponential speedup of QC w.r.t. classical computing?

- Massive Quantum Parallelism
- Superposition/Interference
- Multi-party entanglement
- The entangling power of quantum evolutions
- Nonlocality
- Contextuality
- The dimensionality of Hilbert Space
- Quantum Logic

1) The Power of QC

We can prove theorems of the form:

If a quantum algorithm does not exhibit property X, then it can be efficiently simulated on a classical computer.

We usually cannot prove:

All quantum algorithms exhibiting property X cannot be efficiently simulated on a classical computer.

Examples:

X = a particular kind of multi-party entanglement (Jozsa & Linden 2002, Vidal 2003).

The converse is blocked by the Gottesman-Knill theorem.

1) The Power of QC

What is responsible for “what is responsible for the power of QC?”?

Don't we have to know *what is* the power of QC first?

Why don't we feel the need to ask “what is responsible for the power of classical computing?”?

We certainly don't know *what is* the power of classical computing.

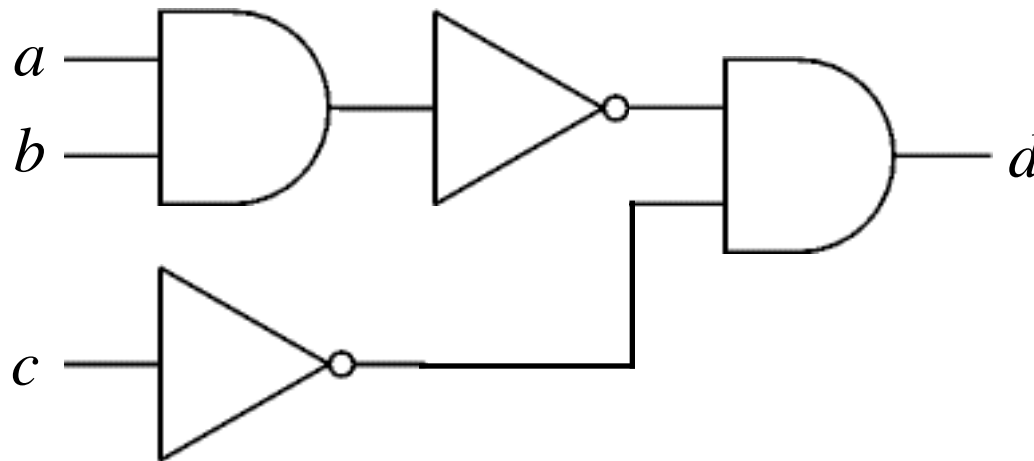
To a large extent classical logic *explains* the power of classical computing.

If we had similar results about some sort of *quantum logic* for quantum computing then the question might disappear.

We also might get some interesting algorithms and complexity results for QC.

2) Logic in Classical Computing

Classical circuit model:

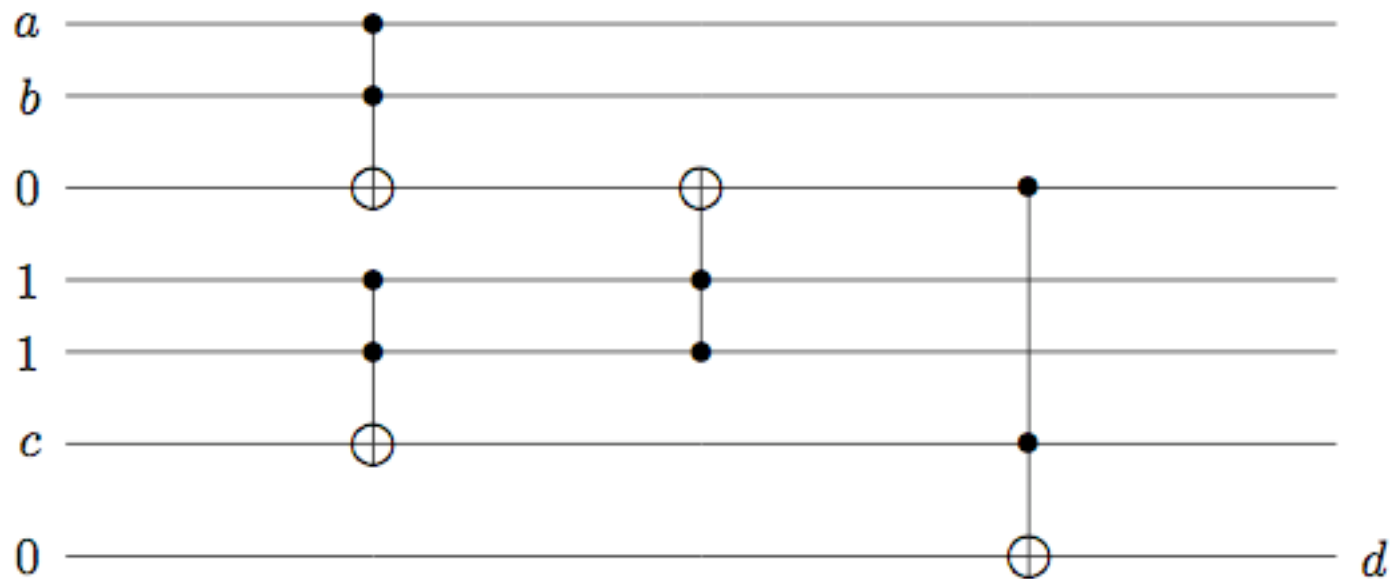


Any algorithm for solving a decision problem can be regarded as a uniform family of Boolean propositions.

$$d = \neg(a \wedge b) \wedge \neg c$$

2) Logic in Classical Computing

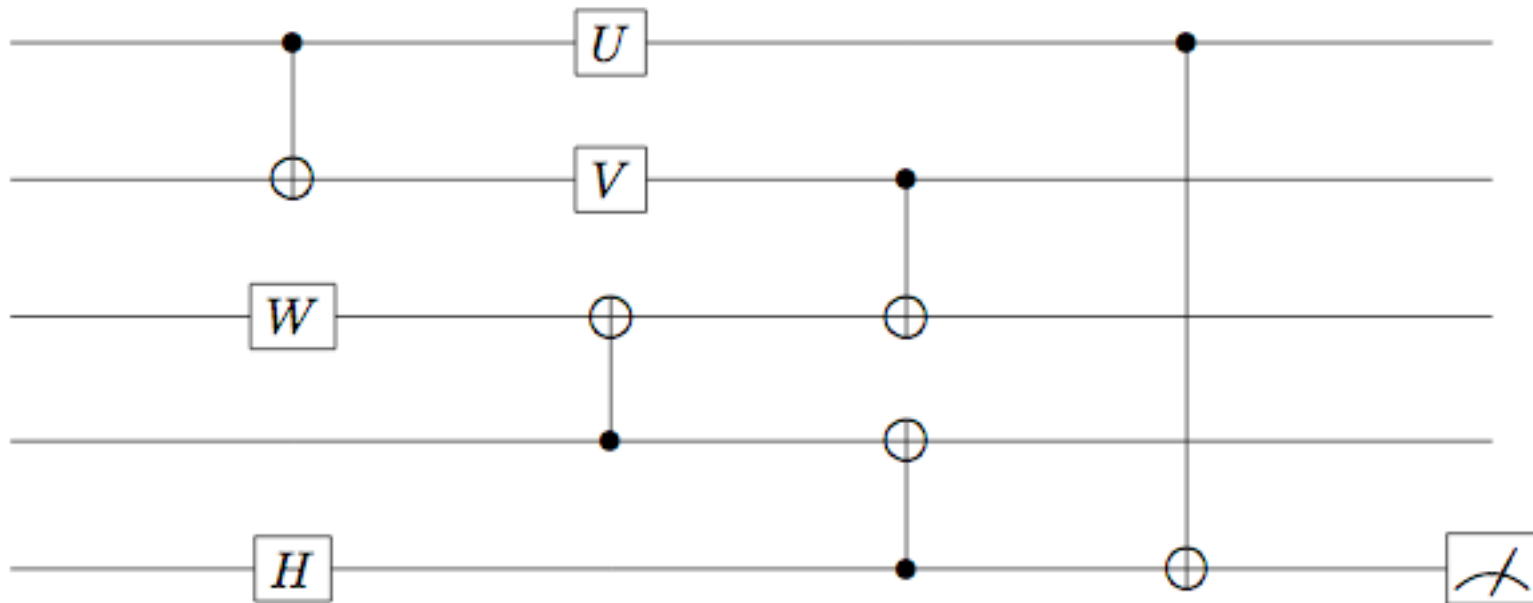
Reversible classical circuit model:



$$d = ?$$

2) Logic in Classical Computing

Quantum circuit model:



$$d = ?$$

2) Logic in Classical Computing

Classical Computational Complexity

Cook-Levin: SATISFIABILITY is NP complete.

Given a Boolean formula, decide if there are truth value assignments to the elementary propositions that make the formula true.

Quantum Computational Complexity

Kitaev: LOCAL HAMILTONIANS is QMA complete.

Let H_1, H_2, \dots, H_r be a set of positive semi-definite operators acting on $(\mathbb{C}^2)^{\otimes n}$. Each matrix comes with a specification of the n qubits, out of a total of m qubits, on which it acts and is specified to $\text{poly}(n)$ bits. Let a and b be two real numbers, specified to $\text{poly}(n)$ bits, such that $a - b = \frac{1}{\text{poly}(n)}$. Is the smallest eigenvalue of $H_1 + H_2 + \dots + H_r$ smaller than a or larger than b .

2) Logic in Classical Computing

Descriptive Complexity

Fagin's Theorem: A set of structures T is in **NP** iff there is a second order existential formula Φ such that T is the set of all finite structures satisfying Φ .

Example: 3-COLOURABLE GRAPHS

$$(\exists R)(\exists Y)(\exists B)(\forall x)((R(x) \vee Y(x) \vee B(x)) \wedge (\forall y)(E(x, y) \rightarrow \neg(R(x) \wedge R(y)) \wedge \neg(Y(x) \wedge Y(y)) \wedge \neg(B(x) \wedge B(y))))))$$

3) The logic of QC

“What is responsible for the power of QC?” is replaced by:

Can quantum computations be regarded as tests of propositions in a formal logic?

Can we prove analogs of Cook-Levin and Fagin’s theorem in such a logic?

Aside: Other proposals for a “logic of QC”

- Deutsch, Ekert and Lupacchini (1999) *Extend classical logic*
 - Formalized by dalla Chiara, Guintini and Leoprini (2003)
- Baltag and Smets (2004) *Dynamic Quantum Logic*
- Abramsky and Coecke (2004) *Categorical Quantum Logic*

4) BvN Quantum Logic

Elementary propositions \Leftrightarrow basic experimentally testable statements.

The position of the particle is between x and $x + dx$.

The momentum of the particle is between p and $p + dp$.

These correspond to closed subspaces of Hilbert Space or projectors.

Analogous to the assignment of sets to propositions in classical logic.

Define connectives \wedge, \vee, \neg to assign meaning to heretical propositions.

The position of the particle is between x and $x + dx$ **AND**
the momentum of the particle is between p and $p + dp$.

Arrive at a logic describing alternative possible measurements that can be made at a single time.

4) BvN Quantum Logic

First attempt: Define a model of computing that tests propositions in BvNQL

BvN Circuit model: Nontrivial since there are no truth values.

Turns out model can be efficiently simulated on a classical computer.

Essentially because size of Hilbert Space required only grows linearly with number of inputs.

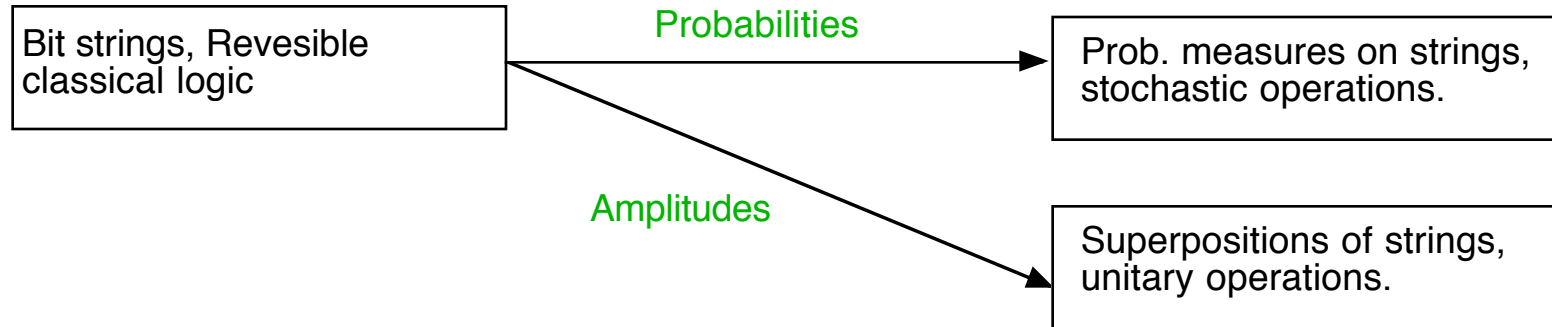
Surprisingly, BvNSAT is only NP complete.

Intuitive reason: BvNQL is about alternative possible measurements whereas computing is about sequences of operations.

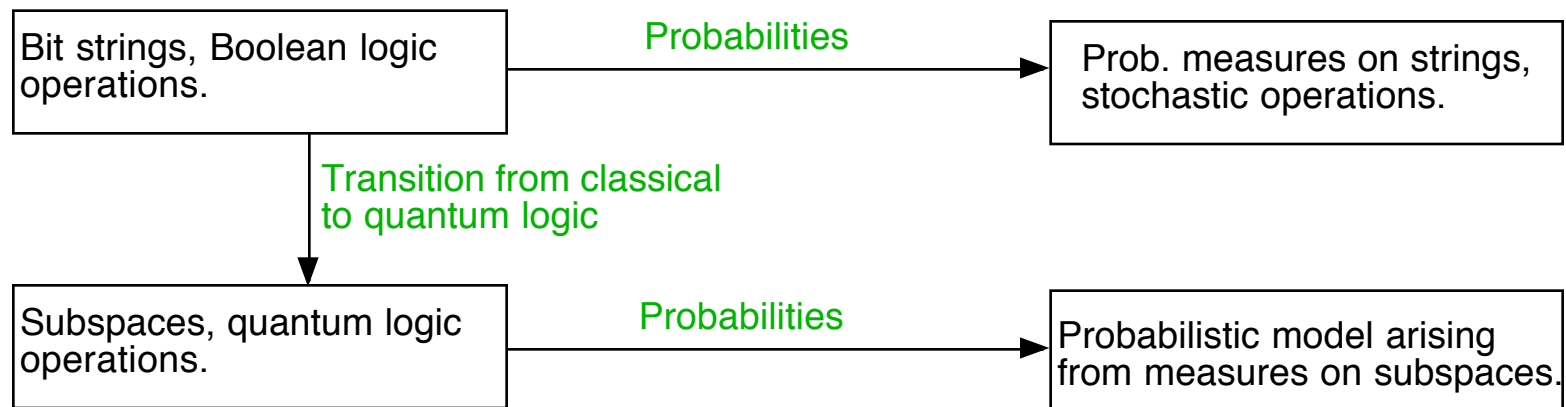
Need a logic of *processes* rather than just a logic of *properties*.

In classical logic, AND \Leftrightarrow AND THEN, since testing a proposition does not disturb the state of the system.

Traditional view of quantum computing

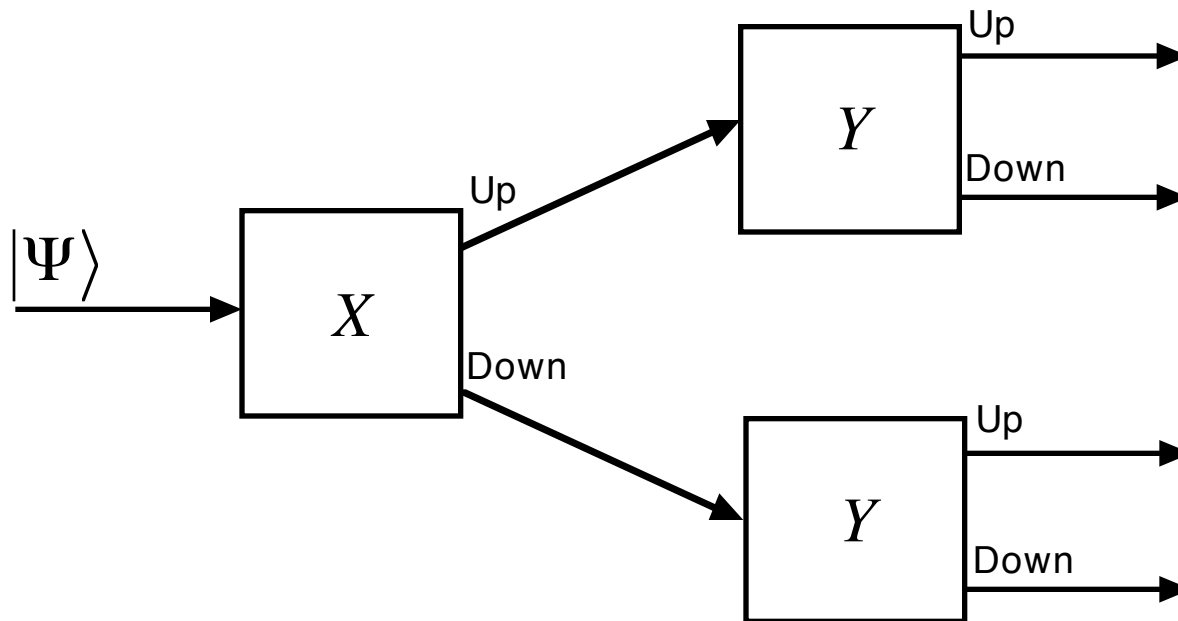


von-Neumanized view of quantum computing



5) Sequential Quantum Logic

- Developed in late 70's/early 80's by Stachow and Mittlestaedt.
- Extended by Isham et. al. in mid 90's as a possible route to QGravity.



5) SQL: Structure

$$\langle L, S, \sqcap, \neg, (,) \rangle$$

L is a set of elementary propositions and S is the set of sequential propositions given by:

- If $a \in L$ then $a \in S$.
- If $a, b \in S$ then $(a \sqcap b) \in S$.
- If $a \in S$ then $\neg a \in S$.

Not commutative: $a \sqcap b \neq b \sqcap a$

Associative: $(a \sqcap b) \sqcap c = a \sqcap (b \sqcap c) = a \sqcap b \sqcap c$

5) SQL: Hilbert Space Model

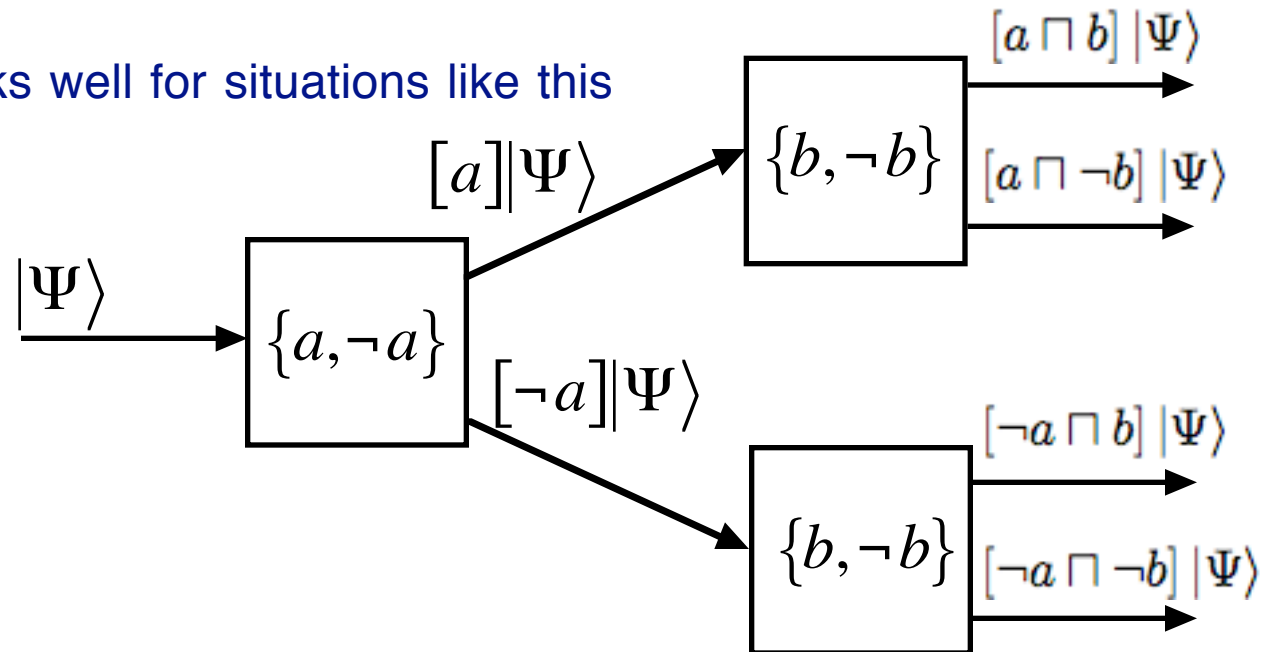
| | | |
|--------------------------|-------------------------|---|
| Hilbert space: | \mathcal{H} | |
| Elementary propositions: | $L(\mathcal{H})$ | - projection operators on \mathcal{H} . |
| Notation: | $[a]$ | - operator associated to prop. a . |
| Negation: | $[\neg a] = I - [a]$ | “NOT a ”. |
| Sequential conjunction: | $[a \sqcap b] = [b][a]$ | “ a AND THEN b ”. |

Note: Usual conjunction can be obtained by including limit propositions

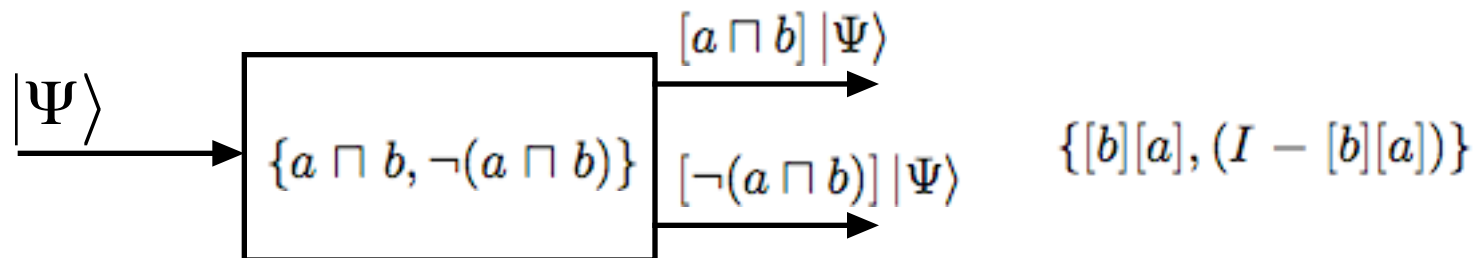
$$[a \wedge b] = \lim_{n \rightarrow \infty} ([b][a])^n$$

5) SQL: Problems

SQL works well for situations like this



But it does not handle “coarse-grainings” well:



6) Testing an SQL Proposition

The algorithm is inspired by recent QInfo inspired approaches to DMRG.

Cirac, Latorre, Rico Ortega, Verstraete, Vidal, et. al.

It has 3 main steps:

- 1) Prepare a “history” state that encodes the results of the underlying sequence of measurements.
- 2) Apply rounds of “renormalization” (coherent AND and NOT gates) to get the desired proposition.
- 3) Measure a qubit to test the proposition.

Note: 2) can only be implemented probabilistically.

6) Testing: History state

Suppose we want to test a simple SQL proposition $d = \neg(a \sqcap b) \sqcap c$.

$$x = a, b, c$$

$$[x] = |\psi_x\rangle \langle \psi_x|, \quad |\psi_x\rangle \in \mathbb{C}^2$$

Notation: $[x^0] = [\neg x], \quad [x^1] = [x]$

Define: $U_x |0\rangle = |\psi_{\neg x}\rangle, \quad U_x |1\rangle = |\psi_x\rangle$

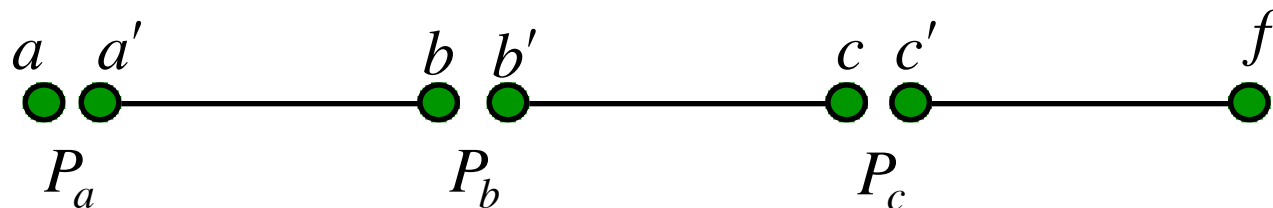
History state:
$$\sum_{j,k,m=0}^1 |j\rangle_a |k\rangle_b |m\rangle_c [c^m][b^k][a^j] |\Psi\rangle_f$$

6) Testing: History state

The history state is Matrix Product State.

Starting state:

$$|\text{start}\rangle = |\Psi\rangle_a |\Phi^+\rangle_{a'b} |\Phi^+\rangle_{b'c} |\Phi^+\rangle_{c'f} \text{ where } |\Phi^+\rangle = |00\rangle + |11\rangle$$



$$\sum_{j,k,m=0}^1 |j\rangle_a |k\rangle_b |m\rangle_c [c^m] [b^k] [a^j] |\Psi\rangle_f = P_a \otimes P_b \otimes P_c |\text{start}\rangle$$

$$P_x = \sum_{j,k,m=0}^1 [x^j]_{km} |j\rangle_x \langle km|_{xx'} \quad [x^0] = [\neg x], [x^1] = [x]$$

6) Testing: History state

How to prepare the history state:

i) Apply $U_a^\dagger \otimes U_a^T$ to qubits a and a' .

ii) Perform a parity measurement on a and a' .

$$P_0 = |00\rangle\langle 00| + |11\rangle\langle 11|, \quad P_1 = |01\rangle\langle 01| + |10\rangle\langle 10|$$

iii) Perform $\text{CNOT}_{a \rightarrow a'}$ and discard a' .

iv) If outcome P_1 occurred then perform $U_a X U_a^\dagger$ on qubit b .

v) Repeat steps i) - iv) for (b, b', c) and (c, c', f) .

Note: Equivalent to applying operators $\sum_{i,j,k} [x^i]_{jk} |i\rangle_x \langle jk|_{xx'}$.

6) Testing: Processing

$$d = \neg(a \sqcap b) \sqcap c \quad \sum_{j,k,m=0}^1 |j\rangle_a |k\rangle_b |m\rangle_c [c^m][b^k][a^j] |\Psi\rangle_f$$

i) Compute $a \sqcap b$ by applying “coherent AND” to qubits a and b .

$$A_{a,b} = |0\rangle_{a \sqcap b} (\langle 00| + \langle 01| + \langle 10|)_{ab} + |1\rangle_{a \sqcap b} \langle 11|_{ab} \\ \sum_{j,k=0}^1 |j\rangle_{a \sqcap b} |k\rangle_c [c^j][(a \sqcap b)^j] |\Psi\rangle_f$$

ii) Compute $\neg(a \sqcap b)$ by applying X to qubit $a \sqcap b$.

$$\sum_{j,k=0}^1 |j\rangle_{\neg(a \sqcap b)} |k\rangle_c [c^j][\neg(a \sqcap b)^j] |\Psi\rangle_f$$

iii) Compute d by applying $A_{\neg(a \sqcap b),c}$.

$$\sum_{j=0}^1 |j\rangle_d [d^j] |\Psi\rangle_f$$

6) Testing: Implementing cohAND

$$A_{a,b} = |0\rangle_{a \cap b} (\langle 00| + \langle 01| + \langle 10|)_{ab} + |1\rangle_{a \cap b} \langle 11|_{ab}$$

is not a directly implementable. Instead, implement measurement:

$$M_s = \frac{1}{\sqrt{3}} (|01\rangle [\langle 00| + \langle 01| + \langle 10|] + |11\rangle \langle 11|)$$
$$M_f = (I - M_s^\dagger M_s)^{1/2}$$

If M_s , discard 2nd qubit and proceed.

If M_f , abort and restart algorithm from beginning.

Note: Algorithm will succeed with exponentially small probability in no. \square gates.

5) Testing: Complexity

Let n = no. propositions in underlying sequence.

Let m = no. \square gates in formula.

Count no. 2-qubit gates:

Preparing entangled states n

Preparing history state $O(n)$

Performing \square gates $O(m)$

Generalization to d dimensional H.S.: Let $r = \lceil \log_2 d \rceil$.

Preparing entangled states rn

Preparing history state $O(n)$

Applying $U_x^\dagger \otimes U_x^T$ $O(n2^{2r})$

5) Generalizations

- Testing projectors of arbitrary rank.
- Testing props on d -dimensional Hilbert space.
 - Need upper bound on d needed to get correct probs. for all formulae of length n .
- Testing multiple propositions.
 - On disjoint subsets of an underlying sequence of propositions.
 - By copying qubits in the computational basis.

7) Open Questions

Can SQL be modified so that all sequential propositions can be tested?

Modify definition of sequential conjunction.

Is SQL the logic of an interesting model of computing?

c.f. Aaronson's QC with post-selection.

DMRG: A new paradigm for irreversible quantum computing?

Which DMRG schemes are universal for quantum computing?

Can they be understood as a kind of Sequential Quantum Logic?

Is there a natural quantum logic in which quantum analogs of Cook-Levin and Fagin's theorem can be proved?