Turing Days'04: Classical & Quantum Computing

QCL

Bernhard Ömer

30th May 2004

E-mail: Bernhard.Oemer@arcs.ac.at
Homepage: http://tph.tuwien.ac.at/~oemer

I. Quantum Computing

Classical and Quantum Computation

Classical Computation

- Computation is the mechanical manipulation of symbols.
- A problem is computable if there exists an *effective procedure* to solve it.
- In- and output parameters are encoded as strings over some finite alphabet.
- The computation is time-discrete *(computational steps)*.
- The state of the computation is observable and traceable.

Quantum Computation

- Computation is a physical process and follows the rules of quantum mechanic.
- A problem is computable if (in practice or principle) a machine can be built to solve it.
- Input is encoded as physical states *(state preparation)*.
- Output is determined by (potentially destructive) measurement.
- The computation is continuous.
- The state of the computation is not observable.

Motivation

- **Universality:** Since the basic concepts of quantum computing directly result from the most fundamental postulates of quantum mechanics, the resulting theory of computability is the most general possible in a quantum mechanical universe.
- **Quantum Parallelism:** The possibility of superpositions of up to 2^n classical states in a quantum register of length n allows the efficient computations of problems for which no polynomial classical algorithms exists.
- **Security:** The principal impossibility to copy or even undestructively observe quantum states allows the development of provably secure communication protocols without prior key-exchange.
- **Miniaturization:** Even today, the integration of logical circuits is at a scale where quantum effects can no longer be ignored. By extrapolating Moore's, law the atomic scale should be reached in less than 2 decades.
- **Energy Consumption:** As quantum computing is based on unitary and therefor reversible operations, there exists no lower thermodynamical bound for energy consumption and waste heat.

Notation

Hilbert Space *A complete vector space $\mathcal{H}$ with an inner product $\langle\cdot|\cdot\rangle$ and the corresponding norm $\|\psi\| = \sqrt{\langle\psi|\psi\rangle}$ is called Hilbert space.*

Notation	Description
$\|\psi\rangle$	general "ket" vector, e.g. $\|\psi\rangle = (c_0, c_1, \ldots)^T$
$\langle\psi\|$	dual "bra" vector to $\|\psi\rangle$, e.g. $\langle\psi\| = (c_0^*, c_1^*, \ldots)$
$\|n\rangle$	basis vector, $\|0\rangle = (1, 0, 0\ldots)^T, \|1\rangle = (0, 1, 0, \ldots)^T$, etc.
$\langle\phi\|\psi\rangle$	inner (scalar) product of $\|\phi\rangle$ and $\|\psi\rangle$
$\|\phi\rangle \otimes \|\psi\rangle$	tensor product of $\|\phi\rangle$ and $\|\psi\rangle$
$\|\phi\rangle\|\psi\rangle$	abbreviated tensor product $\|\phi\rangle \otimes \|\psi\rangle$
$\|i, j\rangle$	abbreviated tensor product of the basis vectors $\|i\rangle$ and $\|j\rangle$
$M^\dagger$	adjoint operator (matrix) $M^\dagger = (M^T)^*$
$\langle\phi\|M\|\psi\rangle$	inner product of $\|\phi\rangle$ and $M\|\psi\rangle$
$\|\psi\|$	abbreviated norm $\||\psi\rangle\|$

Table 1: *Dirac Notation*

A linear operator (matrix) A on $\mathcal{H}$ is called

- *self-adjoined* or *Hermitian* iff $A^\dagger = A$,
- *unitary* iff $A^\dagger A = I$, with I being the identity operator,
- *idempotent* iff $A^2 = A$,
- *(orthogonal) projection* iff A is self-adjoined and idempotent,
- *self-inverse* iff $A^2 = I$.

Quantum States

Postulate 1 *Associated to any physical system is a complex Hilbert space $\mathcal{H}$ know as the state space of the system. The state of the system is completely described by a unit vector $|\psi\rangle \in \mathcal{H}$ with $\|\psi\| = 1$, which is called state vector. Two state vectors $|\psi\rangle$ and $|\psi'\rangle$ are equivalent ($|\psi\rangle \simeq |\psi'\rangle$) iff $|\psi'\rangle = \mathrm{e}^{\mathrm{i}\phi}|\psi\rangle$.*

- In quantum computing we require that $\mathcal{H}$ is separable, so $\mathcal{H}$ is isomorph and isometric to either $\mathbf{C}^n$ or $\mathbf{l}_2$.
- A general quantum state $|\psi\rangle \in \mathcal{H}$ can be written as

$$|\psi\rangle = \sum_{k=0}^{n-1} c_k|k\rangle \quad \text{or} \quad |\psi\rangle = \sum_{k=0}^{\infty} c_k|k\rangle \quad \text{with} \quad \sum_k c_k {c_k}^* = 1.$$

- A system $\mathbf{q}$ with the state space $\mathcal{B} = \mathbf{C}^2$ is called a *quantum bit* or *qubit*.
- The general state of a qubit is a *superposition* of $|0\rangle$ and $|1\rangle$.

$$|\psi\rangle = a|0\rangle + b|1\rangle \quad \text{with} \quad |a|^2 + |b|^2 = 1$$

- The vectors $|0\rangle$ and $|1\rangle$ are called *computational basis*.

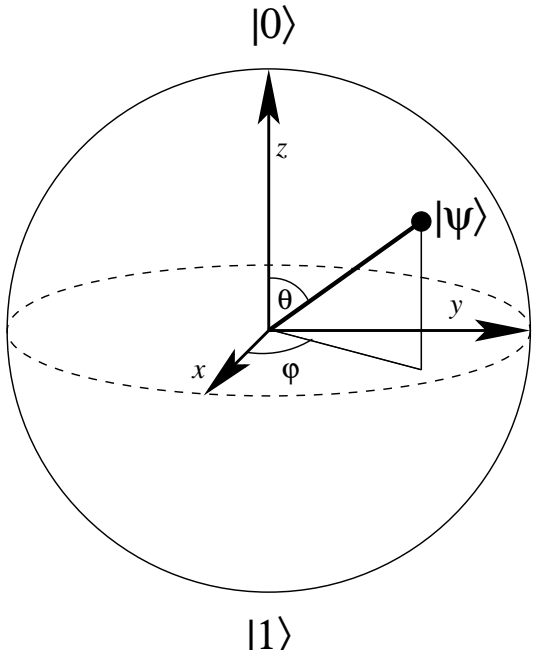
In QCL, qubits are *quantum registers* of length 1:

```
qcl> qureg q[1];                  // allocate qubit
qcl> RotX(pi/3,q);                // prepare superposition
0.86603 |0> - 0.5i |1>
```

The Bloch Sphere

Ignoring an irrelevant overall phase factor, the general state of a qubit can be written as

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + \mathrm{e}^{\mathrm{i}\varphi}\sin\frac{\theta}{2}|1\rangle.$$



- Every qubit state has a unique representation as a point on the three-dimensional unit sphere, also known as *Bloch sphere*.
- The unit-vector $\hat{r} = \hat{r}_\psi$ is called *Bloch vector* of $|\psi\rangle$.
- Bloch vectors have the property that

$$\hat{r}_\phi = -\hat{r}_\chi \iff \langle\phi|\chi\rangle = 0.$$

- Unitary qubit operations correspond to rotations in the Bloch sphere.

Unitary Evolution

Postulate 2 *The temporal evolution of the state of a closed quantum system is described by the Schrödinger equation*

$$\mathrm{i}\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H|\psi(t)\rangle$$

with the (experimental) Plank constant $\hbar$ and a self-adjoint operator H on the state space $\mathcal{H}$ known as the Hamiltonian of the system.

- We define $\hbar \equiv 1$, so the Schrödinger equation can be written in the dimensionless form $\mathrm{i}|\dot{\psi}\rangle = H|\psi\rangle$.
- The Hamiltonian H completely describes the dynamics of a closed quantum system.
- The unitary operator

  $$U(t) = \mathrm{e}^{-\mathrm{i}Ht} = \sum_{n=0}^{\infty} \frac{1}{n!} (-\mathrm{i})^n t^n H^n$$

  is called *operator of temporal evolution.*
- For the initial value $|\psi(0)\rangle = |\psi_0\rangle$ the solution of the Schrödinger equation is given as

  $$|\psi(t)\rangle = U(t)\,|\psi_0\rangle.$$

- Since $U^\dagger(t) = U(-t)$, the evolution is *reversible.*

Since H is uniquely defined by $U(t)$, the 2^{nd} postulate can be reformulated in a discrete-time version as

Postulate 2' *The temporal evolution of a closed quantum system from the state $|\psi\rangle$ at time t_1 to state $|\psi'\rangle$ at time t_2 can be described by a unitary operator $U = U(t_2 - t_1)$ such that $|\psi'\rangle = U|\psi\rangle$.*

Single Qubit Operations

The single qubit rotations $R_x(\theta), R_y(\theta), R_z(\theta)$ about the X-, Y- and Z-axis of the Bloch sphere are given as

$$\begin{pmatrix} \cos\frac{\theta}{2} & -\mathrm{i}\sin\frac{\theta}{2} \\ -\mathrm{i}\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad \begin{pmatrix} \mathrm{e}^{-\mathrm{i}\theta/2} & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\theta/2} \end{pmatrix}$$

- **Not-gate** (π rotation about the X-axis)

$$\texttt{Not} = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- **Hadamard-gate** (π rotation about the XZ-diagonal)

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

- **(Controlled) Phase-gates** (rotations about the Z-axis)

$$V(\varphi) = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\varphi} \end{pmatrix}$$

  which is equivalent to $R_z(\varphi)$. Special cases are

$$Z = V(\pi), \qquad S = V(\frac{\pi}{2}), \qquad T = V(\frac{\pi}{4})$$

Universal Sets of Operators A set of qubit operators is *universal* if any unitary transformation on $\mathcal{B}$ can be approximated to arbitrary precision.

- Any pair of rotations $R_{\hat{v}}(a\pi)$ and $R_{\hat{w}}(b\pi)$ is universal if $\hat{v} \neq \pm\hat{w}$ and a, b are irrational.
- This is the case for HT and TH so the set $\{H, T\}$ is universal.

Measurements

Postulate 3 *A measurement is described by a self-adjoint operator M, called observable, with the spectral decomposition $M = \sum_m mP_m$, where P_m is the projector onto the eigenspace of the eigenvalue m.*

The eigenvalues m of M correspond to the possible outcomes of the measurement. Measuring $|\psi\rangle$ will give the result m with probability $p(m) = \langle\psi|P_m|\psi\rangle$, thereby reducing $|\psi\rangle$ to the post-measurement state

$$|\psi'\rangle = \frac{1}{\sqrt{p(m)}} P_m|\psi\rangle.$$

- Quantum measurements are destructive *(unobservability of states)*
- The *standard observable* is the self adjoint operator

$$N = \sum_k k|k\rangle\langle k|.$$

- N describes measurements in the computational basis.
- For a general state

$$|\psi\rangle = \sum_{k=0}^{n-1} c_k|k\rangle$$

the probability to measure m is $p(m) = |c_m|^2$ with the post-measurement state $|\psi'\rangle = |m\rangle$.

```
qcl> qureg q[2];                        // allocate 2 qubits
qcl> int m;                             // classical variable
qcl> H(q);                              // prepare superposition
[2/32] 0.5 |00> + 0.5 |01> + 0.5 |10> + 0.5 |11>
qcl> measure q[0],m;                    // measure lower qubit
[2/32] 0.70711 |01> + 0.70711 |11> // reduced state, m=1
```

Composite Systems

Postulate 4 *The state space $\mathcal{H}$ of a composite physical system is the tensor product of the state spaces $\mathcal{H}_i$ of its components. Moreover, if the subsystems are in the states $|\psi_i\rangle \in \mathcal{H}_i$, then the joint state $|\Psi\rangle \in \mathcal{H}$ of the composite system is $|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots \otimes |\psi_n\rangle$.*

- For two systems $\mathbf{p}$ and $\mathbf{q}$ with the state spaces $\mathcal{H}_\mathbf{p} = \mathbf{C}^n$ and $\mathcal{H}_\mathbf{q} = \mathbf{C}^m$ and the bases $B_\mathbf{p}$ and $B_\mathbf{q}$, the state space of the joint system $\mathbf{p} \circ \mathbf{q}$ is given by the tensor product

$$\mathcal{H}_\mathbf{p} \otimes \mathcal{H}_\mathbf{q} = \mathbf{C}^{nm} \quad \text{with the basis} \quad B = B_\mathbf{p} \times B_\mathbf{q} = \{|i,j\rangle\}$$

- A linear operator U on $\mathbf{p}$ is equivalent to the operator $U(\mathbf{p}) = U \otimes I$ on $\mathcal{H}_\mathbf{p} \otimes \mathcal{H}_\mathbf{q}$.

- The *product state* of $|\psi_1\rangle = \sum_i a_i|i\rangle \in \mathcal{H}_\mathbf{p}$ and $|\psi_2\rangle = \sum_j b_j|j\rangle \in \mathcal{H}_\mathbf{q}$ is given as

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \sum_{ij} a_i b_j |i,j\rangle$$

- In product states, unitary transformations or measurements performed on one system, do not affect the state of the other system.

The following QCL code prepares the 2-qubit product state

$$|\Psi\rangle = (H\,|0\rangle) \otimes (R_y(\frac{\pi}{3})\,|0\rangle) = \left(H \otimes R_y(\frac{\pi}{3})\right)\,|0,0\rangle$$

```
qcl> qureg p[1]; qureg q[1]; // allocate 2 qubits p and q
qcl> H(p);                   // apply Hadamard gate on q
0.707 |0,0> + 0.707 |1,0>
qcl> RotY(pi/3,q);           // 120 deg. Y-rotation of p
0.612 |0,0> + 0.612 |1,0> + 0.353 |0,1> + 0.353 |1,1>
```

Entanglement

Not any joint state is a product state. A state

$$|\Psi\rangle = \sum_i \sum_j c_{ij}|i,j\rangle \quad \text{with} \quad \sum_{i,j} |c_{ij}|^2 = 1$$

whose coefficients c_{ij} cannot be written as $c_{ij} = a_i b_i$ is called *entangled*.

Consider the following joint states of two qubits

$$|\Psi_A\rangle = \frac{1}{2}|0,0\rangle + \frac{1}{2}|1,0\rangle + \frac{1}{2}|0,1\rangle + \frac{1}{2}|1,1\rangle \quad \text{and}$$

$$|\Psi_B\rangle = \frac{1}{\sqrt{2}}|0,0\rangle + \frac{1}{\sqrt{2}}|1,1\rangle$$

A single measurement of either qubit will give 0 or 1 with equal probability $p = 1/2$. Assuming that a measurement of the first qubit gave the result m, the respective post measurement states are

$$|\Psi'_A\rangle = \frac{1}{\sqrt{2}}|m,0\rangle + \frac{1}{\sqrt{2}}|m,1\rangle \quad \text{and}$$

$$|\Psi'_B\rangle = |m,m\rangle$$

A measurement of the second qubit of $|\Psi'_A\rangle$ will still give a random result, while in the case of $|\Psi'_B\rangle$, the outcome is correlated to the previous measurement and will always produce m. $|\Psi_B\rangle$ is also known as *Bell state*.

```
qcl> qureg p[1]; qureg q[1]; // allocate 2 qubits p and q
qcl> H(q);                   // prepare even superposition
0.70711 |0,0> + 0.70711 |1,0>
qcl> CNot(p,q);              // entangle states with CNot
0.70711 |0,0> + 0.70711 |1,1>
```

Gates

Quantum Computing usually deals with composite systems of qubits:

- Finite composition of n qubits

$$\mathcal{H} = \mathcal{B}^{\otimes n} = \mathbf{C}^{2^n}$$

- Infinite number of qubits with zero-tail state condition

$$\mathcal{H} = \mathcal{B}^{\star} = \bigcup_{n=1}^{\infty} \mathcal{B}^{\otimes n} \otimes \mathbf{C}^{\otimes \omega} = \left\{ |\psi\rangle \otimes |00...\rangle \;\middle|\; |\psi\rangle \in \mathbf{C}^{2^n} \right\}$$

- Dual ended tape of qubits with zero-tail state condition

$$\mathcal{H} = \mathcal{T} = \mathcal{B}^{\star} \otimes \mathcal{B}^{\star} = \mathcal{L}\left\{ |...b_{-2}b_{-1}b_0b_1b_2b_3...\rangle \;\middle|\; \sum_i b_i < \infty \right\}$$

Definition 1 *Let* $\mathbf{s}$ *be a composite system of the qubits* $\mathbf{s}_i$*. A unitary operator* U *on* $\mathcal{B}^{\otimes k} = \mathbf{C}^{2^k}$ *which can be applied to an arbitrary* k*-qubit subsystem (register)* $\mathbf{s}_{j_0} \circ \ldots \circ \mathbf{s}_{j_{k-1}}$ *of* $\mathbf{s}$ *is called a* k*-qubit gate.*

Universal Sets of Gates A set of gates is *universal* if any unitary transformation U on a finite number of qubits can be approximated to arbitrary precision.

- Single qubit operations and `CNOT` are universal.
- The *controlled-not gate* is defined as

$$\texttt{CNOT} = \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix}$$

  and can be generalized to control registers of arbitrary size

$$\texttt{CNot} : |x\rangle|y_0 \ldots y_{n-1}\rangle \rightarrow |x \oplus y_0 \wedge \ldots \wedge y_{n-1}\rangle|y_0 \ldots y_{n-1}\rangle$$

- The *standard set* $\{H, T, \texttt{CNOT}\}$ is universal.

Quantum Circuits

Gates can be connected to *quantum circuits* with the following restrictions:

- Only feed-forward networks are allowed (no loops).
- Only n-to-n gates are allowed.
- Forking of inputs is forbidden, as

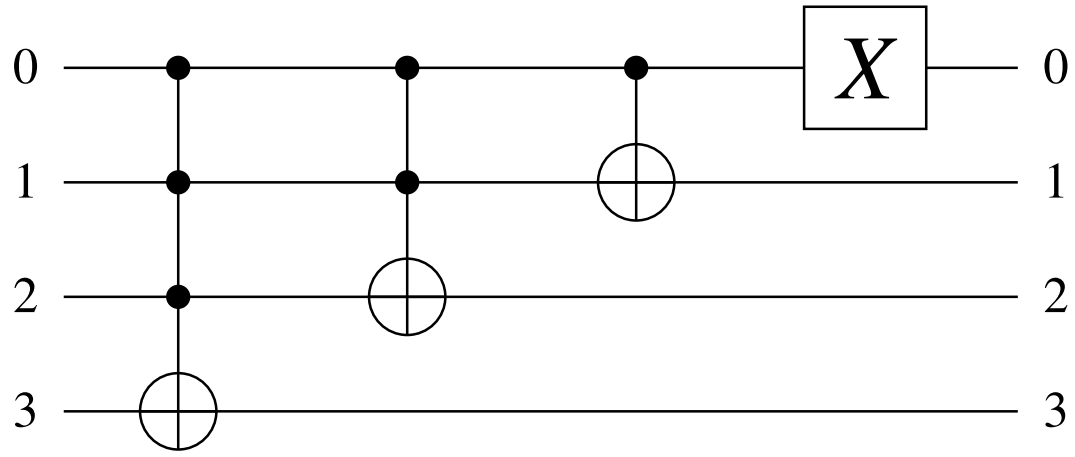
  $$\text{Copy}\,|\psi\rangle|0\rangle \rightarrow |\psi\rangle|\psi\rangle \quad \text{with} \quad |\psi\rangle \in \mathbf{C}^2$$

  is not a unitary operation.
- "dead ends" are forbidden as

  $$\text{Erase}\,|\psi\rangle \rightarrow |0\rangle \quad \text{with} \quad |\psi\rangle \in \mathbf{C}^2$$

  is not a unitary operation.

4-qubit Increment Operator



$$\texttt{inc} : |n\rangle \rightarrow |n + 1 \bmod 16\rangle$$

Quantum Computers

Definition 2 *A machine $\mathcal{M}$ is a 5-tuple $(\mathbf{S}, O, T, \delta, \beta)$ where*

(i) $\mathbf{S}$ *is a set of of computational states*

(ii) $O = \{f_i : \mathbf{S} \mapsto \mathbf{S}\}$ *is an enumerable set of memory commands*

(iii) $T = \{t_i : \mathbf{S} \mapsto \mathbf{S} \times \mathbf{B}\}$ *is an enumerable set of test commands*

(iv) $\delta : \mathcal{I} \to \mathbf{S}$ *is an input function from the enumerable input set $\mathcal{I}$*

(v) $\beta : \mathbf{S} \mapsto \mathcal{O}$ *is an (potentially probabilistic) output function to the enumerable output set $\mathcal{O}$*

In the most general case, a quantum computer $\mathcal{M} = (\mathcal{H}, O, T, \delta, \beta)$ is a probabilistic machine where

- $\mathcal{H}$ is the state space of the quantum system operated on. Typically, $\mathcal{H}$ is qubit-based so $\mathcal{H} = \mathcal{B}^{\otimes n}$ (quantum circuits), $\mathcal{H} = \mathcal{B}^\star$ (UGM) or $\mathcal{H} = \mathcal{T}$ (SQTM) or a combination thereof (QTM).

- O a set of (deterministic) unitary transformations. This can e.g. be a complete set of gates applied to arbitrary qubits (quantum circuits), or a single step-operator (QTM). A single qubit-measurement would be described by the mapping

$$\mu(\mathbf{q}) : \alpha|0\rangle_{\mathbf{q}}|\psi_0\rangle_{\mathbf{p}} + \beta|1\rangle_{\mathbf{q}}|\psi_1\rangle_{\mathbf{p}} \mapsto \begin{cases} (|0\rangle_{\mathbf{q}}|\psi_0\rangle_{\mathbf{p}}, 0), & p = |\alpha|^2 \\ (|1\rangle_{\mathbf{q}}|\psi_1\rangle_{\mathbf{p}}, 1), & p = |\beta|^2 \end{cases}$$

- T a set of (probabilistic) measurement commands. This can be empty (quantum circuits, QTM).

- δ is the initialization operator which can also be constant (e.g. if the input is provided via an oracle-operator).

- β describes the final measurement.

Example: Deutsch's Algorithm

The following quantum algorithm allows to compute $f(0) \oplus f(1)$ for an arbitrary oracle oracle-operator $F : |x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$.

1. Prepare initial 2-qubit state $|\Psi_0\rangle = |0\rangle_{\mathbf{x}}|1\rangle_{\mathbf{y}}$.

2. Apply $H(\mathbf{x} \circ \mathbf{y})$ to set up the search space **superposition**

$$|\Psi_1\rangle = \frac{1}{2}\Big(|0\rangle_{\mathbf{x}} + |1\rangle_{\mathbf{x}}\Big)\Big(|0\rangle_{\mathbf{y}} - |1\rangle_{\mathbf{y}}\Big) = |+\rangle_{\mathbf{x}}|-\rangle_{\mathbf{y}}$$

$$\text{with} \quad |\pm\rangle = \frac{1}{\sqrt{2}}\Big(|0\rangle \pm |1\rangle\Big) \quad \textit{(dual basis).}$$

3. Apply the oracle operator F, giving **quantum parallelism**

$$|\Psi_2\rangle = \frac{1}{\sqrt{2}} F\,|0\rangle_{\mathbf{x}}|-\rangle_{\mathbf{y}} + \frac{1}{\sqrt{2}} F\,|1\rangle_{\mathbf{x}}|-\rangle_{\mathbf{y}} =$$

$$\frac{(-1)^{f(0)}}{\sqrt{2}}|0\rangle_{\mathbf{x}}|-\rangle_{\mathbf{y}} + \frac{(-1)^{f(1)}}{\sqrt{2}}|1\rangle_{\mathbf{x}}|-\rangle_{\mathbf{y}} =$$

$$\pm\frac{1}{2}\,|\pm\rangle_{\mathbf{x}}|-\rangle_{\mathbf{y}}$$

4. Retransform with $H(\mathbf{x})$, resulting in **interference**

$$|\Psi_3\rangle = \frac{1}{\sqrt{2}}\,|b\rangle_{\mathbf{x}}\Big(|0\rangle_{\mathbf{y}} - |1\rangle_{\mathbf{y}}\Big) \quad \text{with} \quad b = \begin{cases} 0 & \text{for} \quad f(0) = f(1) \\ 1 & \text{for} \quad f(0) \neq f(1). \end{cases}$$

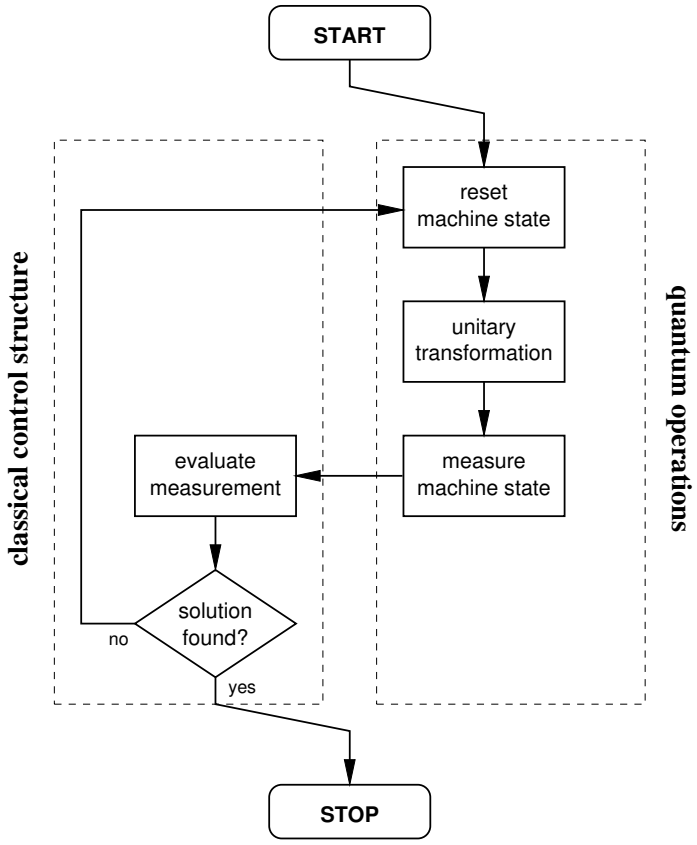
5. Measure $\mathbf{x}$. The measured value $b = f(0) \oplus f(1)$.

```
qufunct F(quconst x,quvoid y) {           // dummy oracle F
  if f(false) xor f(true) { CNot(y,x); }  // f(x) = not x
  if f(false) { Not(y); }                 // F|x,y> = |x,~x^y>
}

procedure deutsch2() {        // improved Deutsch algorithm
  qureg x[1]; qureg y[1];
  int m;
  Not(y);                     // 1. init state
  H(x & y);                   // 2. prepare superposition
  F(x,y);                     // 3. quantum parallelism
  H(x);                       // 4. interference
  measure x,m;                // 5. measure result
  print "f(0) xor f(1) =",m;
}

oemer@linux:~/qcl> qcl deutsch2.qcl -i -l1 -L1 -fb
QCL Quantum Computation Language (32 qubits, seed 1085736488)
[0/32] 1 |0>
qcl> print f(false),f(true);              // f(x) = not x
: true false
qcl> deutsch2();                          // start alg.
@ NOT(qureg q=<1>)
1 |10>
@ H(qureg q=<0,1>)
0.5 |00> + 0.5 |01> - 0.5 |10> - 0.5 |11>
@ CNOT(qureg q=<1>,quconst c=<0>)         //
0.5 |00> - 0.5 |01> - 0.5 |10> + 0.5 |11>  // oracle call
@ NOT(qureg q=<1>)                        //
-0.5 |00> + 0.5 |01> + 0.5 |10> - 0.5 |11>
@ H(qureg q=<0>)
-0.70711 |01> + 0.70711 |11>
@ MEASURE <0>
: f(0) xor f(1) = 1                       // result
[0/32] -0.70711 |1> + 0.70711 |11>
```

II. Quantum Programming in QCL

Quantum Algorithms

Due to the stochastic nature of measurements, quantum algorithms are often probabilistic.



Classical Elements

- control structure
- evaluation of measurements
- probabilistic computation

Quantum Elements

- Unitary Transformations
- Initialization
- Measurements

Quantum Programming Languages

Quantum Programming is about extending classical universal programming languages for quantum computing.

Unlike the mathematical formalism, QTMs or quantum circuits, quantum programming languages (QPLs) are at the same time

- **universal** (unlike quantum circuits),
- **constructive** (unlike the mathematical formalism),
- **hardware independent** (unlike QTMs).

Moreover, QPLs

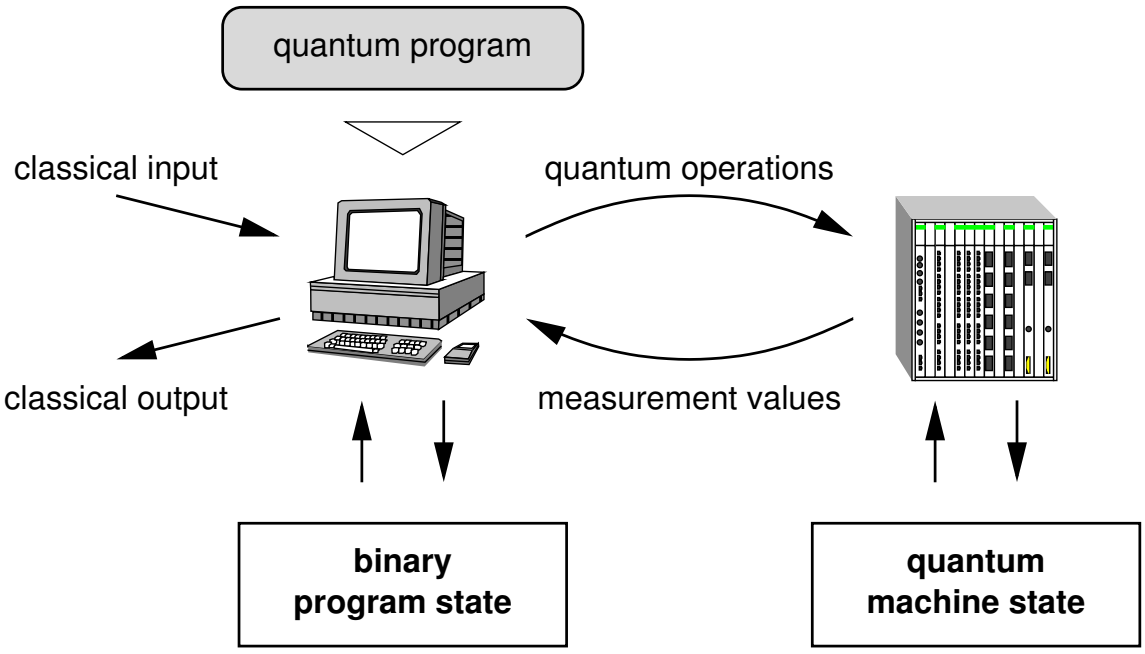
- provide arbitrary levels of abstraction,
- allow input of arbitrary sizes,
- do not require global unitary transformations,
- provide well defined space and time complexity measures and
- allow the implementation of the classical control structure.

A quantum programming language is called

- **imperative** if it provides quantum registers *(quantum variables)*, elementary gates and single qubit measurements.
- **procedural** if it additionally supports unitary subroutines and reverse execution to derive the adjoint operator.
- **structured** if it additionally allows the use of qubits and boolean expression of qubits *(quantum conditions)* in structured flow-control statements *(quantum if-statement)*.

Hybrid Architecture

The target architecture of quantum programming is a universal classical computer with a quantum oracle.



- **program state:** values of variables, execution stack, etc.
- **machine state:** common quantum state of all qubits

$$|\Psi\rangle = \sum_{(d_0 \ldots d_{n-1}) \in \mathbf{B}^n} c_{d_0 \ldots d_{n-1}} |d_0 \ldots d_{n-1}\rangle = \sum_{k=0}^{2^n-1} c_k |k\rangle$$

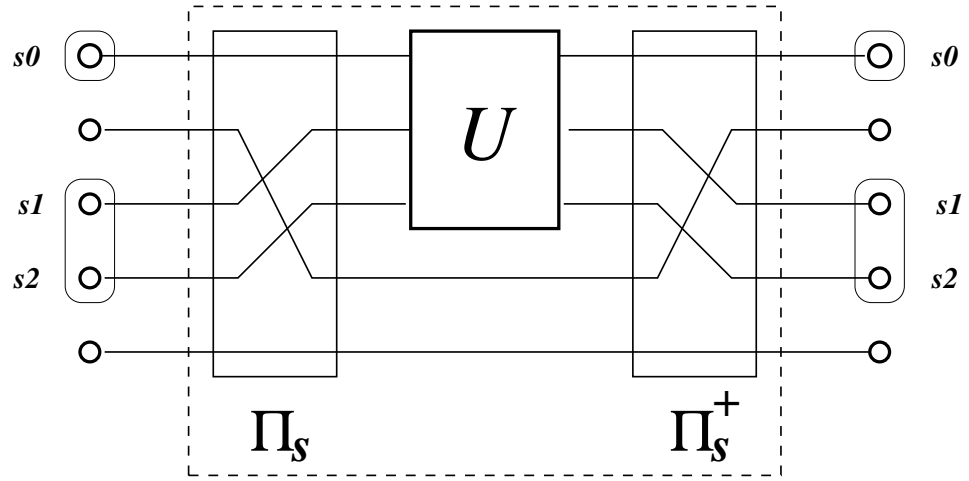
- **classical frontend:** I/O, flow control, generation of quantum OPs, evaluation of measurements, random numbers
- **quantum oracle:** elementary qubit gates, reset and measurement operations

Quantum Registers

Definition 3 *An m qubit quantum register* $\mathbf{s}$ *is a sequence of mutually different qubit positions* $\langle s_0, s_1 \ldots s_{m-1} \rangle$ *of some machine state* $|\Psi\rangle$

Definition 4 *The register operator* $U(\mathbf{s})$ *for an m-qubit unitary operator U and an m-qubit quantum register* $\mathbf{s}$ *on an n-qubit quantum computer is*

$$U(\mathbf{s}) = \Pi_{\mathbf{s}}^{\dagger} \, (U \otimes I) \, \Pi_s$$



- Registers are the formal interface to the quantum oracle.
- An n-qubit quantum oracle allows for $\lfloor en! \rfloor$ different registers.
- Operator subroutines take the size of the register (i.e. the number of qubits) as an implicit parameter.
- Operators have mathematical semantics, this esp. excludes
  - dependencies on the program state (e.g. global variables)
  - side effects on the program state
  - user input and calls to `random()`
- Non-unitary quantum operations (i.e. calls to `measure` and `reset`) are prohibited.

Example: Quantum Fourier Transform

For an N dimensional vector $|\psi\rangle$, the discrete Fourier transform is defined as

$$DFT : |x\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \mathrm{e}^{\frac{2\pi \mathrm{i}}{N} xy} |y\rangle$$

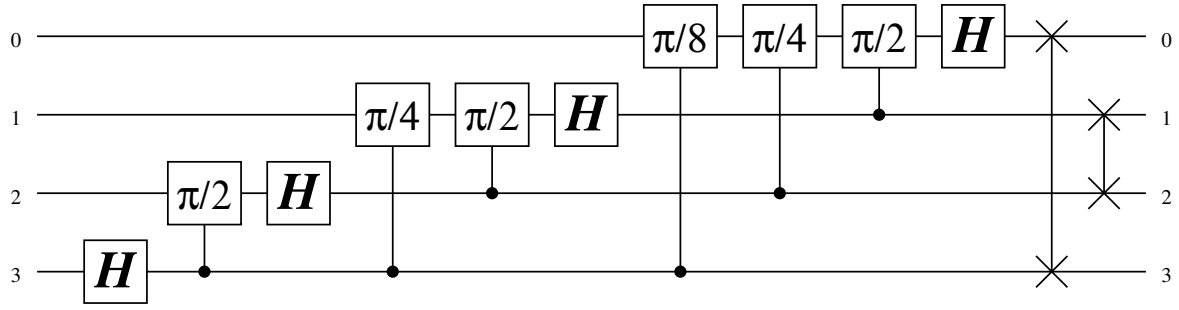
In analogy to the classical FFT, Coppersmith suggested the following quantum implementation (indices are qubit positions):

$$DFT' = \prod_{i=2}^{n} \left(H_{n-i} \prod_{j=1}^{i-1} V_{n-i,n-j}(\frac{\pi}{2^{i-j}}) \right) H_{n-1}$$

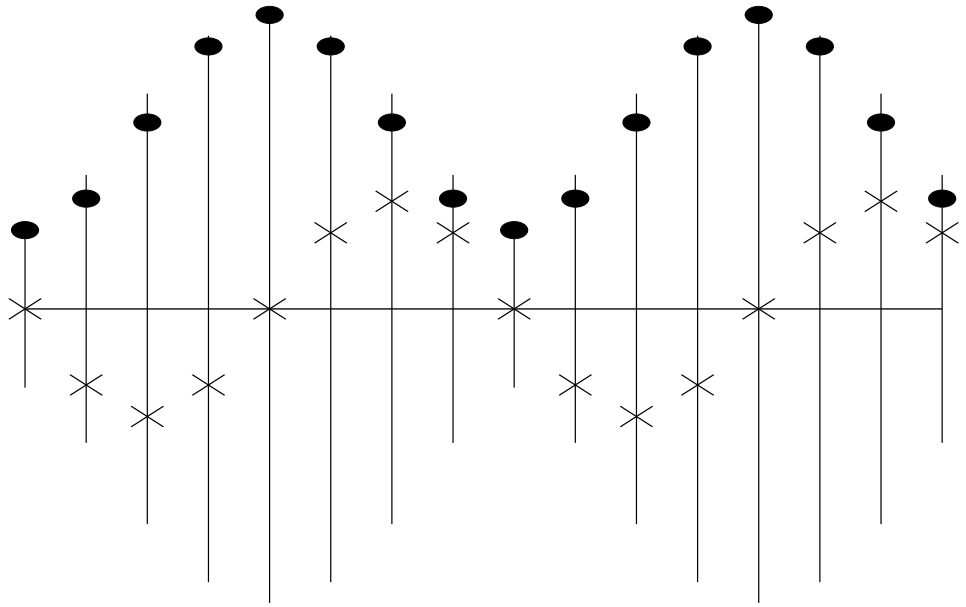
$$\text{with} \quad V(\phi) : |\epsilon\rangle \rightarrow \begin{cases} \mathrm{e}^{\mathrm{i}\phi} |\epsilon\rangle & \text{if } \epsilon = 11\ldots \\ |\epsilon\rangle & \text{otherwise} \end{cases}$$

$$\text{and} \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

```
operator dft(qureg q) { // main operator
  const n=#q;           // set n to length of input
  int i; int j;         // declare loop counters
  for i=1 to n {
    for j=1 to i-1 {    // apply conditional phase gates
      if q[n-i] and q[n-j] { Phase(pi/2^(i-j)); }
    }
    H(q[n-i]);          // qubit rotation
  }
  flip(q);              // swap bit order of the output
}
```




```
qcl> qureg q[4];          // allocate 4 qubit register
qcl> Rot(pi/3,q[1]);      // rotate 2nd qubit
0.866 |0> - 0.5 |2>
qcl> dft(q);              // perform Fourier transform
0.0915 |0> + ... + (0.1281+0.0884i) |15> (16 terms)
qcl> plot;
: PLOT STATE
```




```
qcl> !dft(q);             // inverse Fourier transform
0.866 |0> - 0.5 |2>
```

Quantum Data Types

- **General register (qureg):** general purpose register without access restrictions
- **Constant register (quconst):** register has to be invariant to all operator calls within its scope
- **Target register (quvoid):** register has to be empty when the uninverted operator is called (parameter type)
- **Scratch register (quscratch):** register is declared as scratch space (parameter or local type)

Definition 5 (Invariance of Registers) *A quantum register* $\mathbf{c}$ *is considered invariant to a register operator* $U(\mathbf{s}, \mathbf{c})$ *if* U *meets the condition*

$$U : |i, j\rangle = |i\rangle_{\mathbf{s}} |j\rangle_{\mathbf{c}} \rightarrow (U_j |i\rangle_{\mathbf{s}}) |j\rangle_{\mathbf{c}}$$

An operator $U(\mathbf{s}, \mathbf{c})$ with a constant n-qubit argument register $\mathbf{c}$ is called *selection operator* and can be written as

$$U = \mathrm{diag}(U_1, U_2 \ldots U_{2^n}).$$

Definition 6 (Empty Registers) *A quantum register* $\mathbf{s}$ *is empty iff*

$$P_0(\mathbf{s}) |\Psi\rangle = |\Psi\rangle \quad \text{with} \quad P_0 = |0\rangle\langle 0|$$

- At startup or after the `reset` command, the whole machine state is empty, thus $|\Psi\rangle = |0\rangle$.
- Newly allocated registers are empty.

Basis Permutations

Definition 7 *A n-qubit basis permutation is a unitary operator of the form* $U : |i\rangle \rightarrow |\pi_i\rangle$ *with some permutation* π *over* $\mathbf{Z}_{2^n}$.

- Basis permutations implement reversible boolean functions.
- The implementation of basis permutations (`qufunct`) must not contain calls to general unitary operators (`operator`).
- When applied to superpositions, basis permutations allow for massive parallel computation (quantum parallelism)

```
qufunct inc(qureg x) {      // increment register
  int i;
  for i = #x-1 to 1 step -1 {
    CNot(x[i],x[0:i-1]);
  }
  Not(x[0]);
}

qcl> qureg q[4];             // generate superposition
qcl> H(q[3] & q[1]);         // of |0>, |2>, |8> and |10>
0.5 |0> + 0.5 |2> + 0.5 |8> + 0.5 |10>
qcl> inc(q);                 // increment basis vectors
0.5 |1> + 0.5 |3> + 0.5 |9> + 0.5 |11>
qcl> inc(q);
0.5 |2> + 0.5 |4> + 0.5 |10> + 0.5 |12>
qcl> !inc(q);                // decrement basis vectors
0.5 |1> + 0.5 |3> + 0.5 |9> + 0.5 |11>
```

Quantum Functions

Definition 8 *For any function* $f : \mathbf{B}^n \rightarrow \mathbf{B}^m$ *(or equivalently* $f : \mathbf{Z}_{2^n} \rightarrow \mathbf{Z}_{2^m}$*) there exists a class of basis permutations* $F : \mathbf{C}^{2^{n+m}} \rightarrow \mathbf{C}^{2^{n+m}}$ *working on an n-qubits input and an m-qubits output register with* $F|x, 0\rangle = |x, f(x)\rangle$*. Operators of that kind are referred to as quantum functions.*

- Quantum functions allow the calculation of non-reversible boolean functions.
- Quantum functions have at least one constant (`quconst`) argument and one empty (`quvoid`) target register argument.
- The behavior of a quantum functions for non-empty target registers is unspecified.
- For any boolean function $f : \mathbf{B}^n \rightarrow \mathbf{B}^m$ there exist $2^n(2^m - 1)!$ different quantum functions F.
- Local scratch registers are automatically reclaimed if declared as `quscratch` and don't need to be explicitely uncomputed.

```
qufunct parity(quconst x,quvoid y) {
  int i;
  for i = 0 to #x-1 {
    CNot(y,x[i]);
  }
}

qcl> qureg x[2]; qureg y[1]; H(x);
0.5 |0,0> + 0.5 |1,0> + 0.5 |2,0> + 0.5 |3,0>
qcl> parity(x,y);
0.5 |0,0> + 0.5 |3,0> + 0.5 |1,1> + 0.5 |2,1>
```

Scratch Space Management

Let F be a quantum function with the argument register $\mathbf{x}$ (`quconst`), the target register $\mathbf{y}$ (`quvoid`) and the scratch register $\mathbf{s}$ (`quscratch`)

$$F(\mathbf{x},\mathbf{y},\mathbf{s}) : |i\rangle_{\mathbf{x}}|0\rangle_{\mathbf{y}}|0\rangle_{\mathbf{s}} \rightarrow |i\rangle_{\mathbf{x}}|f(i)\rangle_{\mathbf{y}}|j(i)\rangle_{\mathbf{s}}$$

F fills the register $\mathbf{s}$ with the temporary junk bits $j(i)$. To reclaim $\mathbf{s}$, QCL allocates an auxiliary register $\mathbf{t}$ and translates F into an operator F' which is defined as

$$F'(\mathbf{x},\mathbf{y},\mathbf{s},\mathbf{t}) = F^{\dagger}(\mathbf{x},\mathbf{t},\mathbf{s})\,\mathtt{Fanout}(\mathbf{t},\mathbf{y})\,F(\mathbf{x},\mathbf{t},\mathbf{s})$$

The *fanout* operator is a quantum function defined as

$$\mathtt{Fanout} : |i\rangle|0\rangle \rightarrow |i\rangle|i\rangle$$

The application of F' restores the scratch register $\mathbf{s}$ and the auxiliary register $\mathbf{t}$ to $|0\rangle$ while preserving the function value in the target register $\mathbf{y}$:

$$|i,0,0,0\rangle \rightarrow |i,0,j(i),f(i)\rangle \rightarrow |i,f(i),j(i),f(i)\rangle \rightarrow |i,f(i),0,0\rangle$$

```
qufunct parity2(quconst x1,quconst x2,quvoid y) {
  quscratch s[2];       // allocate 2 scratch qubits
  parity(x1,s[0]);      // store parity of x1 in s[0]
  parity(x2,s[1]);      // store parity of x2 in s[1]
  CNot(y,s);            // set y if x1 and x2 have odd
}                       // parity

qcl> qureg a[3]; qureg b[3]; qureg y[1];
qcl> H(a[2] & b[0]); Not(b[1]);
0.5 |0,2,0> + 0.5 |4,2,0> + 0.5 |0,3,0> + 0.5 |4,3,0>
qcl> parity2(a,b,y);
0.5 |0,2,0> + 0.5 |0,3,0> + 0.5 |4,3,0> + 0.5 |4,2,1>
```

Hierarchy of Subroutines

routine type	program state	machine state	inv.
`procedure`	all	all	no
`operator`	none	unitary	yes
`qufunct`	none	basis perm.	yes
functions	none	none	no

- **General subroutine (procedure):**
  - may depend on I/O, random numbers and measurements
  - used for classical and irreversible parts of quantum algorithms
- **General unitary operator (operator):**
  - effect may only depend on declared parameters
  - allows inverse execution
- **Basis Permutation (qufunct):**
  - allows scratch-space management
- **Classical function:**
  - no side-effects allowed

Example: Grover's Database Search

- Many problems in classical CS can be be formulated as search problems over bitstrings
- If a unique solution to $C(x) = 1, x \in \mathbf{B}^n$ is known to exist, this solution can be found in $O(\sqrt{n})$ steps (as opposed to $O(n/2)$ for a classical brute-force search
- For each search problem, a query operator can be defined as

$$\texttt{query} : |x, 0\rangle \rightarrow |x, C(x)\rangle \quad \text{with} \quad x \in \mathbf{B}^n \quad \text{and} \quad C : \mathbf{B}^n \rightarrow \mathbf{B}$$

1. Set up a search space by performing a n-qubit Hadamard transform

$$H : |i\rangle \rightarrow 2^{-\frac{n}{2}} \sum_{j \in \mathbf{B}^n} (-1)^{(i,j)} |j\rangle$$

2. The main loop of the algorithm consists of two steps

   (a) Perform a conditional phase shift which rotates the phase of all basis vectors which match the condition C by π radians.

$$Q : |i\rangle \rightarrow \begin{cases} -|i\rangle & \text{if } C(i) \\ |i\rangle & \text{if } \neg C(i) \end{cases}$$

   (b) Apply a diffusion operator

$$D = \sum_{ij} |i\rangle d_{ij} \langle j| \quad \text{with} \quad d_{ij} = \begin{cases} \frac{2}{N} - 1 & \text{if } i = j \\ \frac{2}{N} & \text{if } i \neq j \end{cases}$$

3. Perform measurement when the amplitude for the solution $|i_0\rangle$ (i.e. $C(i_0) = \texttt{true}$) reaches its maximum ($p > \frac{N-1}{N}$ after $\lfloor \frac{\pi}{4} \sqrt{N} \rfloor$ iterations)

```
boolean C(int x) { return x==SOLUTION; }

qufunct query(qureg x,quvoid f) {
  int i;
  for i=0 to #x-1 {     // x -> NOT (x XOR n)
    if not bit(SOLUTION,i) { Not(x[i]); }
  }
  CNot(f,x);            // flip f if x=1111..
  for i=0 to #x-1 {     // x <- NOT (x XOR n)
    if not bit(SOLUTION,i) { !Not(x[i]); }
  }
}

operator diffuse(qureg q) {       // diffusion operator
  Mix(q);               // Hadamard Transform
  Not(q);               // invert q
  if q { Phase(pi) };   // invert sign if q=1111..
  !Not(q);              // undo inversion
  !Mix(q);              // undo Hadamard Transform
}

procedure grover(int n) {         // n is upper limit for x
  int l=floor(log(n,2))+1;        // no. of qubits
  int m=ceil(pi/8*sqrt(2^l));     // no. of iterations
  int x; int i;
  qureg q[l]; qureg f[1];
  {
    reset;
    Mix(q);             // prepare superposition
    for i=1 to m {      // main loop
      query(q,f);       // calculate C(q)
      CPhase(pi,f);     // negate |x>
      !query(q,f);      // undo C(q)
      diffuse(q);       // diffusion operator
    }
    measure q,x;        // measurement
    print "measured",x;
  } until C(x);
}
```

Quantum Flow Control

Conditional Execution Classical programming languages allow the conditional execution of code depending on a boolean variable.

- **Goal:** conditional execution of code depending on qubits
- **Problem:** unobservability of qubits, unitary evolution
- **Solution:** conditional operators

If-Statements In structured languages, conditional branching is realized by if-statements with well defined entry and exit points.

- **Goal:** conditional branching on qubits
- **Problem:** The classical implementation by executing either the if- or the else-branch is impossible.
- **Solution:** conditional composition
- **Problem:** Assignments to classical variables may lead to inconsistent program states
- **Solution:** threaded execution *(quantum forking)*

Boolean Expression Classical languages allow the use of complex boolean expressions.

- **Goal:** boolean formulas of qubits *(quantum condition)*
- **Problem:** The boolean operators $\wedge$ and $\vee$ are not reversible.
- **Solution:** transformation into the XNF, transparent use of scratch qubits

Conditional Operators

Definition 9 *A conditional operator $U_{[[\mathbf{e}]]}$ with the enable register $\mathbf{e}$ is a unitary operator of the form*

$$U_{[[\mathbf{e}]]} : |i, \epsilon\rangle = |i\rangle|\epsilon\rangle_{\mathbf{e}} \rightarrow \begin{cases} (U\,|i\rangle)\,|\epsilon\rangle_{\mathbf{e}} & \text{if } \epsilon = 111\ldots \\ |i\rangle|\epsilon\rangle_{\mathbf{e}} & \text{otherwise} \end{cases}$$

- Conditional Operators are selection operators of the form

$$U_{[[\mathbf{e}]]} = C^{|\mathbf{e}|}[U] = \mathrm{diag}(I \ldots I, U)$$

- Informally, the effect of $U_{[[\mathbf{e}]]}(\mathbf{s})$ can be described as "if $\mathbf{e}$ is set then apply $U(\mathbf{s})$".

Definition 10 *An n-qubit basis permutation $P : |k\rangle \rightarrow |p(k)\rangle$ is called cyclic iff it generates the computational basis such that for any $|k\rangle$*

$$\bigcup_{i=0}^{2^n-1} P^i|k\rangle = \{|j\rangle \mid j \in \mathbf{Z}_{2^n}\}$$

Examples for P are `Not` and `inc`. The following rules apply:

- **Orthogonal Enable Registers:** Let U and V be arbitrary unitary operators and P cyclic basis permutation, then

$$\Big[U_{[[\mathbf{e}]]}(\mathbf{s})\,P(\mathbf{e}), V_{[[\mathbf{e}]]}(\mathbf{s})\,P^\dagger(\mathbf{e})\Big] = 0.$$

- **Conditional Composition:** For any selection operator

$$U = \mathrm{diag}\,(U_1, U_2 \ldots U_{2^m}) \quad \text{on} \quad \mathcal{B}^{\otimes n+m}$$

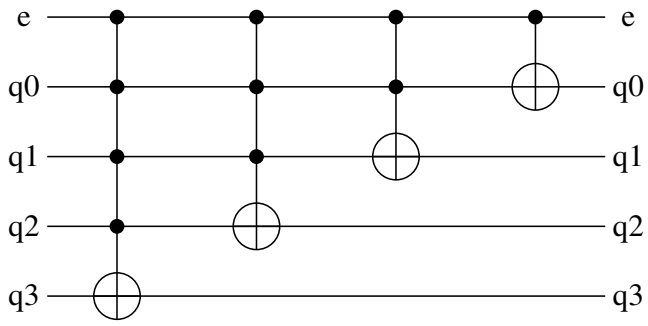
there exists an m-qubit basis permutation P such that

$$U(\mathbf{s}, \mathbf{e}) = \prod_{i=1}^{2^m} U_{i\,[[\mathbf{e}]]}(\mathbf{s})\,P(\mathbf{e})$$

Conditional Subroutines

- Operator subroutines can be declared as conditional (`cond`).
- The enable register is passed as an implicit parameter and can be set by a *quantum if-statement*.
- When an enable register is set, a circuit for the conditional operator is automatically constructed.

```
cond qufunct inc(qureg x) {          // Conditional increment
  int i;                             // as conditional
  for i = #x-1 to 0 step -1 {        // subroutine
    CNot(x[i],x[0::i]);
  }
}
qufunct cinc(qureg x,quconst e) {    // Equivalent
  int i;                             // implementation
  for i = #x-1 to 0 step -1 {        // as selection
    CNot(x[i],x[0::i] & e);          // operator
  }
}
```




```
qcl> qureg q[4]; qureg e[1];
qcl> if e { inc(q); }          // equivalent to cinc(q,e);
```

Quantum Conditions

Definition 11 *A quantum condition is a boolean formula of qubits combined by the operators* $\{\neg, \wedge, \vee, \oplus\}$.

Definition 12 *Let* S *be a set of qubits. A quantum condition*

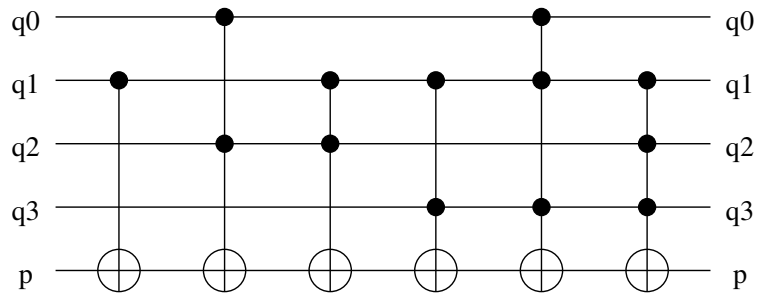
$$\mathbf{C} = \bigoplus_{i=1}^{n} \bigwedge_{j=1}^{m_i} \mathbf{p}_{ij} \quad \text{with} \quad \mathbf{p}_{ij} \in S \quad \text{and} \quad \{\mathbf{p}_{kj}\} = \{\mathbf{p}_{lj}\} \Leftrightarrow k = l$$

is in exclusive disjunctive normal form (XNF).

Any boolean formula can be transformed into XFN by recursively applying the following rules ($\texttt{false} \equiv \bigoplus_{i=1}^{0}$ and $\texttt{true} \equiv \bigwedge_{j=1}^{0}$):

$$\begin{aligned}
a \oplus a &\Longrightarrow \texttt{false} \\
a \wedge a &\Longrightarrow a \\
\neg a &\Longrightarrow \texttt{true} \oplus a \\
a \vee b &\Longrightarrow a \oplus a \wedge b \oplus b \\
a \wedge \bigoplus_i b_i &\Longrightarrow \bigoplus_i a \wedge b_i
\end{aligned}$$

- Quantum conditions in XNF can be evaluated by `CNot`-gates.
- Classical boolean expressions and qubits can be mixed.
- The QCL data type `qucond` allows for complex quantum predicates (the example below shows a 4-qubit primality test).



Quantum If-Statements

- Quantum if-statements (QISs) allow conditional branching on arbitrary quantum conditions.
- A QIS which contains classical assignments is called *dirty*.
- QIS may contain if- and else-branches and can be nested.
- Scratch registers for the evaluation of complex quantum conditions are transparently allocated and uncomputed again.

```
qcl> qureg a[1]; qureg b[1]; qureg q[3];
qcl> H(a & b);                  // prepare test state
0.5 |0,0,0> + 0.5 |1,0,0> + 0.5 |0,1,0> + 0.5 |1,1,0>

qcl> if a { inc(q); }
0.5 |0,0,0> + 0.5 |0,1,0> + 0.5 |1,0,1> + 0.5 |1,1,1>

qcl> if a and b { inc(q); }
0.5 |0,0,0> + 0.5 |0,1,0> + 0.5 |1,0,1> + 0.5 |1,1,2>

qcl> if a or b { inc(q); }
0.5 |0,0,0> + 0.5 |0,1,1> + 0.5 |1,0,2> + 0.5 |1,1,3>

qcl> if b { Phase(pi); } else { inc(q); }
0.5 |0,0,1> - 0.5 |0,1,1> + 0.5 |1,0,3> - 0.5 |1,1,3>

qcl> if not a { if b { inc(q); } else { !inc(q); } }
0.5 |0,0,0> - 0.5 |0,1,2> + 0.5 |1,0,3> - 0.5 |1,1,3>
```

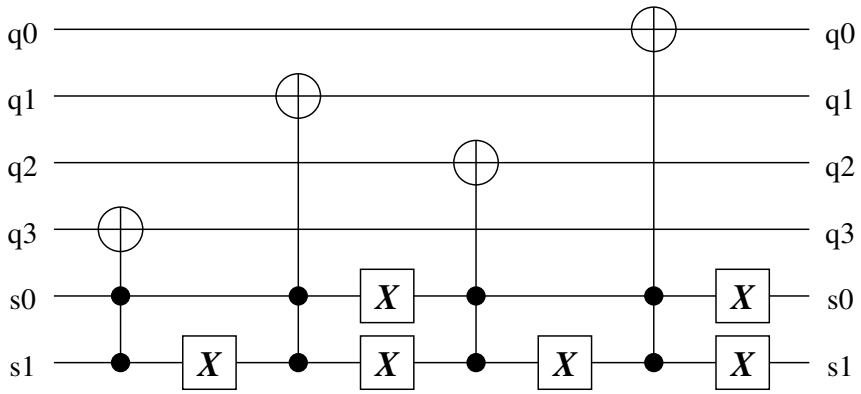
Quantum Forking

If the body of a quantum if-statement contains statements which change the program state (e.g. assignments to local variables), then subsequent operator calls may differ, depending on whether the if- or the else-branch has been executed *(dirty QIS)*.

In that case, QCL follows all possible classical paths throughout the operator definition *(threaded execution)*, accumulates the conditions of all visited quantum if-statements and serializes the generated sequence of operators by conditional composition.

Forking if-statements may only appear within an operator definition to assure that the different execution threads can be joined again.

```
cond qufunct mux(quconst s,qureg o) {   // Multiplexer
  int i;
  int n = 0;
  for i=0 to #s-1 {          // accumulate content of
    if s[i] { n=n+2^i; }     //   selection register in a
  }                          //   classical variable
  Not(o[n]);                 // flip selected output qubit
}
```




```
qcl> qureg s[2]; qureg q[4]; H(s);
0.5 |0,0> + 0.5 |1,0> + 0.5 |2,0> + 0.5 |3,0>
qcl> mux(s,q);
0.5 |0,1> + 0.5 |1,2> + 0.5 |2,4> + 0.5 |3,8>
```