

# An Algebraic Foundation for Quantum Programming Languages

Andrew Petersen &  
Mark Oskin  
Department of Computer Science  
The University of Washington

## The Quantum Divide

### Advances in Quantum Hardware

- 7-bit computers created
- Silicon devices proposed
- Solid-state bits entangled
- Photons teleported

### Stagnation in Quantum Software

- Few new algorithms discovered
- Little discussion of higher level languages

## What is the Problem?

- No more quantum algorithms exist...?
- We're just not smart enough...?
- No representation developed for computing
  - Traditional notations describe physical systems
    - Dirac notation: describes system state
    - Matrix notation: represents system evolution
  - Enabling computation requires more
    - Assist in guiding systems to “interesting” states
    - Support reasoning about system evolution

## Objective

Develop an alternative notation for quantum computing

- Representation: dealing with groups of bits is hard
  - Ensure operations are insensitive to state space size
  - Introduce shorthand for common entangled states
  - Facilitate computation on large, highly entangled states
- Reasoning: interesting states are difficult to identify
  - Identify quantum properties explicitly
  - Define operations by the quantum properties they induce
  - Favor local transformations over global ones
- Not a language ... yet

## Qubits: Quantum Bits

Bits and qubits both have two states: 0 and 1

- Superposition:  
A qubit may be in both states simultaneously
- Phase:  
A qubit may have a negative quantity of a state
- Entanglement:  
Multiple qubits may share a single state

## Dirac Notation: a Qubit

Probability Amplitude

State

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle$$

Superposition

## Matrix Notation: a Qubit

Probability Amplitude  
for State 0

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Probability Amplitude  
for State 1

## Representing Qubit Systems

$$|p,q\rangle = \alpha|00\rangle + \beta|01\rangle + \chi|10\rangle + \delta|11\rangle$$

$$\begin{bmatrix} \alpha \\ \beta \\ \chi \\ \delta \end{bmatrix} \begin{array}{l} \longleftarrow \text{Amplitude for } 00 \\ \longleftarrow \text{Amplitude for } 01 \\ \longleftarrow \text{Amplitude for } 10 \\ \longleftarrow \text{Amplitude for } 11 \end{array}$$

## The New Algebra: a Qubit

$$q = \alpha q^0 + \chi^k \beta q^1$$

Name  $\uparrow$   $q$       Superposition  $\downarrow$   $+$       State  $\downarrow$   $q^1$   
 Unit of Phase  $\nearrow$   $\chi^k$       Weight  $\uparrow$   $\beta$

## The New Algebra: Operators

- Superposition: +
  - Identity exists:  $x^y + \mathbf{0} = x^y$
  - Inverses exist:  $x^y + (-1) x^y = \mathbf{0}$
- Association: \*
  - Identity exists:  $1 x^y = x^y x^y = x^y$
  - $x^y x^z = \mathbf{0}$  for  $y \neq z$
- Other axioms hold for both operators
  - Associativity and commutativity of + and \*
  - Distributivity of \* over +

$$q^0 + q^1$$

$$p^0 q^1$$

## Association and Entanglement

Distributivity reveals the lack of entanglement.

00	→	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$
01	→	$1 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$1 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
10	→	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$	$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
11	→	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
		Unentangled (a)	Entangled (b)

$$p^0 q^0 + p^1 q^0 = q^0 (p^0 + p^1)$$

Unentangled  
(a)

$$p^0 q^0 + p^1 q^1 = p | q$$

Entangled  
(b)

## Weight and Phase in the Algebra

- Concepts of weight and phase separated
  - Weights are positive real values
  - Phases are complex values
- State probabilities are easy to compute

$$q = 1 q^0 + \chi^{2^n} 3 q^1$$

## Phase and Interference

- Fundamental unit of phase  $\chi$  introduced
  - Phase is manipulated in discrete increments
  - $\chi^k \equiv e^{k * i\pi / 2^n}$
  - $\chi^{2^n} = -1, \chi^0 = \chi^{2 * 2^n} = 1$
- Addition simulates phase interactions

$$q = (q^0 + q^1) + (q^0 + (-1)q^1) = 2q^0$$

## Matrix Notation: Procedures

Procedures are represented as matrices

- Larger state spaces require larger matrices
- The effect of the matrix may not be apparent

$$H = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \quad H_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

## The New Algebra: Procedures

A procedure has four parts:

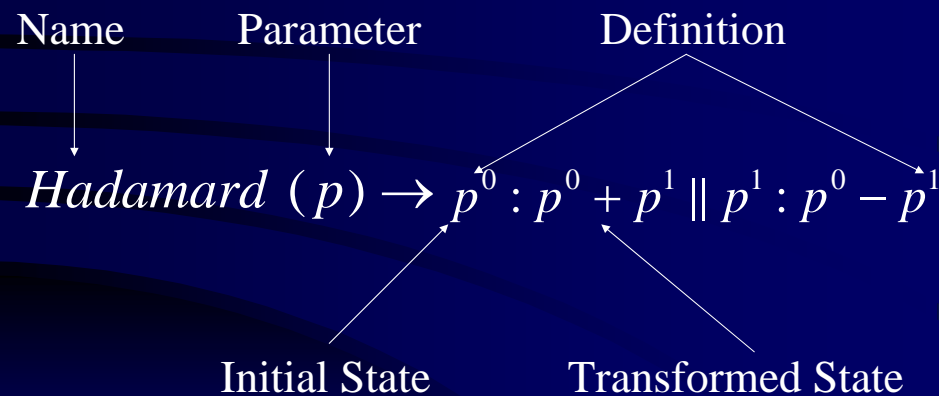
*name (parameters) → definition*

on *oldstate*  $\Rightarrow$  *newstate*

Computation is performed via pattern matching:

*definition*  $\equiv$  *oldpattern*<sub>1</sub> : *newpattern*<sub>1</sub> ||  
*oldpattern*<sub>2</sub> : *newpattern*<sub>2</sub> || ...

## Example: The Hadamard Gate



## Computation in the New Algebra

$$H(p) \rightarrow p^0 : p^0 + p^1 \parallel p^1 : p^0 - p^1$$

on  $p^0 q^0 + p^1 q^1$

$$\Rightarrow (p^0 + p^1)q^0 + (p^0 - p^1)q^1$$

### 1. Consolidate

Associate the states of all arguments

### 2. Match

Find affected patterns and replace them

### 3. Simplify

## Example: Controlled Not

$$CNot(p, q) \rightarrow p^0 q^x : p^0 q^x \parallel p^1 q^x : p^1 Not(q^x)$$

$$Not(p) \rightarrow p^0 : p^1 \parallel p^1 : p^0$$

- Expressions can contain wild-cards
- Patterns can call other procedures

## An Example: EPR Pairs

In matrix notation:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

In the algebra:

$$\text{qubit } p \Rightarrow p^0$$

$$\text{qubit } q \Rightarrow q^0$$

$$H(p) \text{ on } p^0 \Rightarrow p^0 + p^1$$

$$CNot(p, q) \text{ on } (p^0 + p^1)q^0$$

$$\Rightarrow p^0 q^0 + p^1 q^1 = p \mid q$$

## Universality and Illegal States

$$H(q) \rightarrow q^0 : q^0 + q^1 \parallel q^1 : q^0 - q^1$$

$$CNot(p, q) \rightarrow p^0 q^x : p^0 q^x \parallel p^1 q^x : p^1 Not(q^x)$$

$$T(q) \rightarrow q^0 : q^0 \parallel q^1 : \chi^{2^{n-2}} q^1$$

The algebra is complete and expressive

- All legal operations can be defined (Boykin et al.)
- All legal states can be expressed
- Illegal states cannot be reached using legal gates

## Negation

$$\bar{x} = U^x - x$$

Represents all states *not* present

$$x = p^0$$

$$\bar{x} = p^1$$

$$y = p^0 q^0 + p^0 q^1 + p^1 q^1$$

$$\bar{y} = p^1 q^0$$

$$z = p^0 q^0 r^0 t^0 + p^0 q^0 r^0 t^1 + p^0 q^0 r^1 t^0 + p^0 q^0 r^1 t^1 +$$

$$p^0 q^1 r^0 t^0 + p^0 q^1 r^0 t^1 + p^0 q^1 r^1 t^0 + p^0 q^1 r^1 t^1 +$$

$$p^1 q^0 r^0 t^0 + p^1 q^0 r^0 t^1 + p^1 q^0 r^1 t^0 + p^1 q^0 r^1 t^1 +$$

$$p^1 q^1 r^0 t^0 + p^1 q^1 r^0 t^1 + p^1 q^1 r^1 t^0 + p^1 q^1 r^1 t^1$$

$$\bar{z} = p^1 q^0 r^0 t^1$$

## Computing on Negations

Computation may be performed directly

1. Add all the cases of the gate being applied
2. Apply the gate to the negated state
3. Subtract the result from 2) from 1)

$H(p)$  on  $\overline{p^1 q^1}$

$$\Rightarrow ((p^0 + p^1) + (p^0 - p^1))(q^0 + q^1) - (p^0 - p^1)q^1$$

$$= 2p^0 q^0 + 2p^0 q^1 - p^0 q^1 + p^1 q^1$$

$$= p^0 q^0 + \overline{p^1 q^0}$$

## Example: Grover's Algorithm

- Fast search algorithm
- The desired solution is designated with a hat
- An Oracle is required
  - Adds a negative phase to the desired solutions
- A PhaseFlip operation is needed
  - Adds a negative phase to all non-zero states

## Example: Grover Iteration

$GroverIteration(p[n] \text{ as } 1\hat{p} + r\hat{p}) \{$

$$Oracle(p) \Rightarrow \hat{p} - r\hat{p} = U^P - (r+1)\hat{p}$$

forall  $p_i \{$

$$H(p_i)$$

$$\} \Rightarrow 2^n (p_n^0 \dots p_1^0) - (r+1)U^{\hat{P}}$$

$$= (2^n - (r+1))(p_n^0 \dots p_1^0) - (r+1)\overline{(p_n^0 \dots p_1^0)}$$

## Example: Grover Iteration

$$\begin{aligned} \text{PhaseFlip}(p) &\Rightarrow (2^n - (r+1))(p_n^0 \dots p_1^0) + (r+1)\overline{(p_n^0 \dots p_1^0)} \\ &= (2^n - 2(r+1))(p_n^0 \dots p_1^0) + (r+1)U^{\hat{p}} \end{aligned}$$

forall  $p_i$  {

$H(p_i)$

}  $\Rightarrow (2^n - 2(r+1))U^P + 2^n(r+1)\hat{p}$

}  $\rightarrow (1 - \frac{r+1}{2^{n-1}})\bar{\hat{p}} + (r+2 - \frac{r+1}{2^{n-1}})\hat{p}$

## Summary

- Explicit indicators of quantum properties
  - Superposition operator +
  - Basic unit of phase  $\chi$
  - Entanglement via distributivity
- Support for computing on large systems
  - Size-independent procedures
  - Focus on local transformations
- Methods for reasoning about entangled states
  - Symmetric entanglement operator |
  - Computation on negations

## Future Work

- Reasoning about phase is still difficult
  - A compact representation for complex phases is needed
  - Weight and phase interactions may be further formalized
- Types could be used to enforce constraints
  - Separate quantum and classical types would properly restrict interactions
  - Linear types would prevent copying of state
- User studies will indicate problem areas
- Our final goal is a language implementation

Questions?

## Previous Work

- Quantum circuits (Deutsch, Yao)  
Circuit representation with state annotations
- QCL (Ömer)  
Imperative language based on defining matrices
- qGCL (Sanders and Zuliani)  
Probabilistic language featuring a refinement calculus
- Quantum C++ (Bertelli)  
C++ with quantum operations defined as data structures
- Block-QPL (Selinger)  
A functional, graph-based computational model

## Grover's Algorithm

```
Grover(val n) {  
  qubit p[n] ⇒ pn0...p10  
  forall pi {  
    H(pi) → pi0 : pi0 + pi1 || pi1 : pi0 - pi1  
  } ⇒ (pn0 + pn1)...(p00 + p01) = Up =  $\bar{p}$  +  $\hat{p}$   
  for i = 1 to (√n + 1) {  
    GroverIteration(p) → (1 -  $\frac{r+1}{2^{n-1}}$ ) $\bar{p}$  + (r + 2 -  $\frac{r+1}{2^{n-1}}$ ) $\hat{p}$   
  } ⇒  $\bar{p}$  + r $\hat{p}$ , r > (2n - 1)  
  Measure(p) ⇒  $\bar{p}$  : r $\hat{p}$ , r > (2n - 1)  
} →  $\bar{p}$  : r $\hat{p}$ , r > (2n - 1)
```