A Touch of Topological Quantum Computation in Haskell

Philip Zucker

Overview

- Quantum Computation
  - Quantum Mechanics
  - Topological Quantum Computation
  - Anyons
- Haskell
  - Dense Vectors
  - Sparse Vectors
  - Linear Monad
  - Implementing Fibonacci Anyons

Quantum Computation

- Use Quantum Mechanics to Solve Problems Fast
- Applications
  - Cryptography
  - Optimization
  - Physical Simulations

Quantum Mechanics in 5 Minutes

Probability	Quantum
familiar	magic
$S \to [0,1]$	$S \to \mathbb{C}$
p_i	ψ_i
$T_{ij} = P(i \mid j)$	$U_{ij} = e^{\frac{iHt}{\hbar}}$
d^n scaling	d^n scaling
sampling	measurement

Topological Quantum Computation Anyons

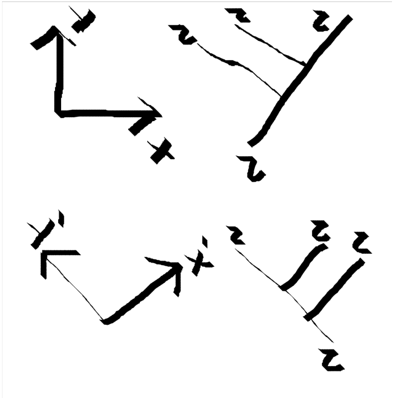
- Core Abstraction for Topological Quantum Computation
- Anyons are peculiar particles
- Production Rules N_{ab}^{c}
  - Similar to chemistry or nuclear.
- Quantum Vector Space
  - Number and Types of particles.

Fibonacci Anyons

- Fibonacci Anyons
  - Simple
  - Universal Quantum Computation
- Two particle types:
  - τ
  - I
- one nontrivial production:
  - $\tau \rightarrow \tau\tau$

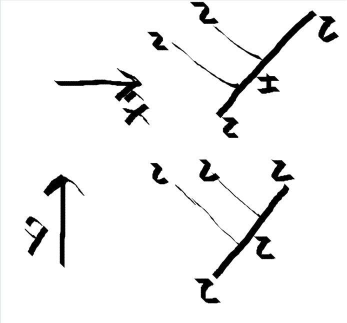
Basis

- Choice of basis = Production Tree Shape



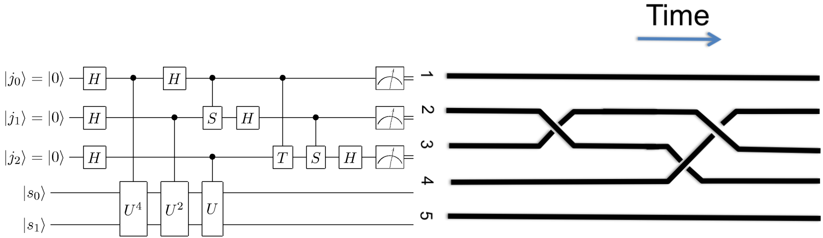
Basis Vector

- Basis Vector = Interior Tree Labelling



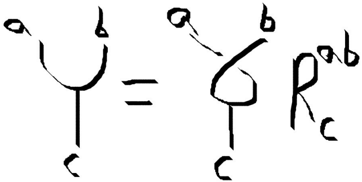
Computation Using Anyons

- Use subspace of anyon space as qubits
- Gates built using braiding

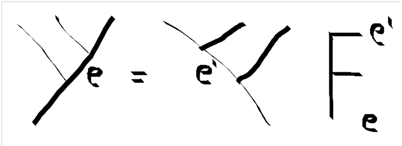


Computation Using Anyons (Cont.)

- Linear maps
  - Braiding = Physical Process.



  - Reassociating = Change of Basis



How do I implement this monster on a computer I have?

Haskell

- Typed Functional Programming Language
- Since 1990
- Optimizing Compiler
- Pure
- Polymorphic

```haskell
factorial :: (Eq a, Num a) => a -> a
factorial 0 = 1
factorial n = n * (factorial (n-1))
```

Dense Vectors

- Arrays
  - `[a]`
  - `Vector a`
- Types as Intent and Design
  - `data V4 a = V4 a a a a`
  - `Vec 7 a`

Free Vectors

- You can make a vector space from any set.
- Explicit type for index/basis
  - Plays elegantly with Haskell
  - `b -> r`
  - `Map b r`
  - `[(b,r)]`
- Similar to Sparse Vectors

Generalized Abstract Data Types (GADTs) for Anyon Trees

```haskell
data FibTree root leaves where
      TTT :: FibTree Tau l -> FibTree Tau r
                           -> FibTree Tau (l,r)
      ITT :: FibTree Tau l -> FibTree Tau r
                           -> FibTree Id (l,r)
      TIT :: FibTree Id  l -> FibTree Tau r
                           -> FibTree Tau (l,r)
      TTI :: FibTree Tau l -> FibTree Id r
                           -> FibTree Tau (l,r)
      III :: FibTree Id  l -> FibTree Id r
                           -> FibTree Id (l,r)
      TLeaf :: FibTree Tau Tau
      ILeaf :: FibTree Id Id

exampleTree :: FibTree Tau (Tau,(Tau,Tau))
exampleTree = TTI (TLeaf) (ITT TLeaf TLeaf)
```

Implementing Linear Maps

- Matrices
  - `[[r]]`
  - `V4 (V4 r)`
  - `[((b,b),r)]`
- What about `(Vec a -> Vec a)`?
  - Excellent fit with Haskell
  - Possibly very fast and flexible.

```
vfun :: Vec Double -> Vec Double
vfun = fmap square
```

The Linear Monad

- Monad pattern abstract away repetitive pipework
  - Null checking
  - Error handling
- Vectors as a monad over the index type.
- $A(\alpha\hat{x} + \beta\hat{y}) = \alpha(A\hat{x}) + \beta(A\hat{y})$

```haskell
newtype Q b = Q [(b, Complex Double)]
flip (>>=) :: (b -> Q a) -> (Q b -> Q a)
return :: b -> Q b
```

Braiding & ReAssociation

```haskell
braid :: FibTree a (l,r) -> Q (FibTree a (r,l))
braid (ITT l r) = W [(ITT r l,  cis $ 4 * pi / 5)]
braid (TTT l r) = W [(TTT r l,  (cis $ - 3 * pi / 5))]
braid (TTI l r) = pure $ TIT r l
braid (TIT l r) = pure $ TTI r l
braid (III l r) = pure $ III r l
```

- Reassocation is similar

```haskell
fmove :: FibTree a (c,(d,e)) -> Q (FibTree a ((c,d),e))
```

Final Thoughts

- Greedy pursuit of performance is mind closing
- Types are good
  - Good For Design
  - Good For Safety
- Haskell is Good
  - Useful in scientific contexts.
- Further Work
  - Category Theory for Anyons
  - 2Vect
  - Actually emulating an algorithm

Thank You

References

- http://www.philipzucker.com/a-touch-of-topological-quantum-computation-in-haskell-pt-i/
- http://www.philipzucker.com/a-touch-of-topological-quantum-computation-in-haskell-pt-ii-automating-drudgery/
- http://www.philipzucker.com/a-touch-of-topological-computation-3-categorical-interlude/
- http://www.theory.caltech.edu/people/preskill/ph229/
- http://blog.sigfpe.com/2007/03/monads-vector-spaces-and-quantum.html