A Touch of Topological Quantum Computation in Haskell

Philip Zucker

Overview

Where are we going today:

- Quantum Computation
- Haskell
- Vectors
- Anyons
- Category Theory?

Quantum Mechanics in 5 Minutes

Probability	Quantum
$[0, 1]$	$\mathbb{C}$
p_i	ψ_i
$\sum_i p_i = 1$	$\sum_i \lvert a_i \rvert^2 = 1$
T_{ij}	U
$\sum_j T_{ij} = 1$	$U^\dagger U = I$
sampling	*measurement*
d^n	d^n
blase	magic

Quantum Computation

- Applications in Cryptography, Optimization, Physical Simulations
- Hard to keep quantum quantum. Error Correction
- topological quantum computation
- anyonic vector spaces

Haskell

- Programming Language 1990
- Functional
- Statically Typed
- Pure / Immutable
- Lazily evaluated
- Polymorphic
- Type classes

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

Mathematical Vectors

- little arrows
- direction and magnitude
- Abstract
- basis elements mean thing
- little, medium, big
- geometrical, 2d, 3d, 4d
- color 3 dimensional BGR RGB
- discretized pde spaces
- probability, and quantum mechanics

Computer Implemented Vectors

- arrays
- sparse vs dense
- structures
- indexed on integer because computers like integers
- Color Vectors = Arrays?
- np.array()
- [Double]
- Data.Vector
- HMatrix
- Repa
- Massiv
- accelerate

Free Vectors

- good for sparse
- make the domain part of the vector historical row column formats
- b -> r
- Map b r
- [(b,r)]

Mathematical Linear Maps

Concept $A(\alpha\vec{x} + \beta\vec{y}) = \alpha A\vec{x} + \beta A\vec{y}$

Implementing Linear Maps

- Matrix a
- (Vec a -> Vec a)

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

The Linear Monad

```
bs >>= f = [   |   (b, r) <- bs]
pure
```

Physics of Fibonacci Anyons

- Particles I and τ
- Possible production rules

GADTs

Implementation of Fibonacci Anyons

- Possible production rules
- Particles I and τ

```
data Tau
data Id
data FibTree root leaves where
   TTT :: FibTree Tau l -> FibTree Tau r -> FibTree Tau (l,
   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
```

References

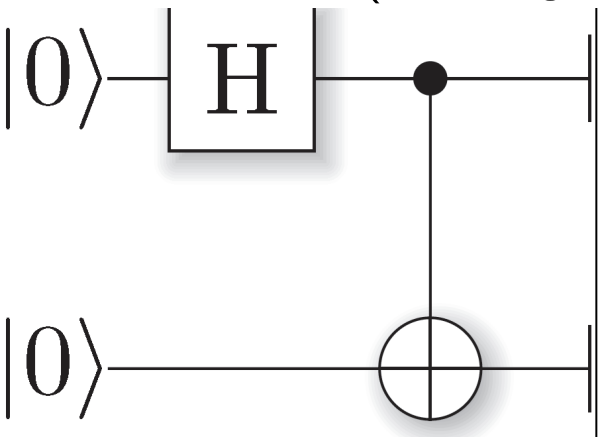
https://arxiv.org/pdf/1705.04103.pdf - topological QM review Nielson and Chuang Quantum compute review Moggi and Wadler Monad originals? Probability monad original and Piponi articles Wikipedia Kitaev possibly Baez and Stay Preskill notes http://blog.sigfpe.com/2007/03/monads-vector-spaces-and-quantum.html

my blog posts

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/

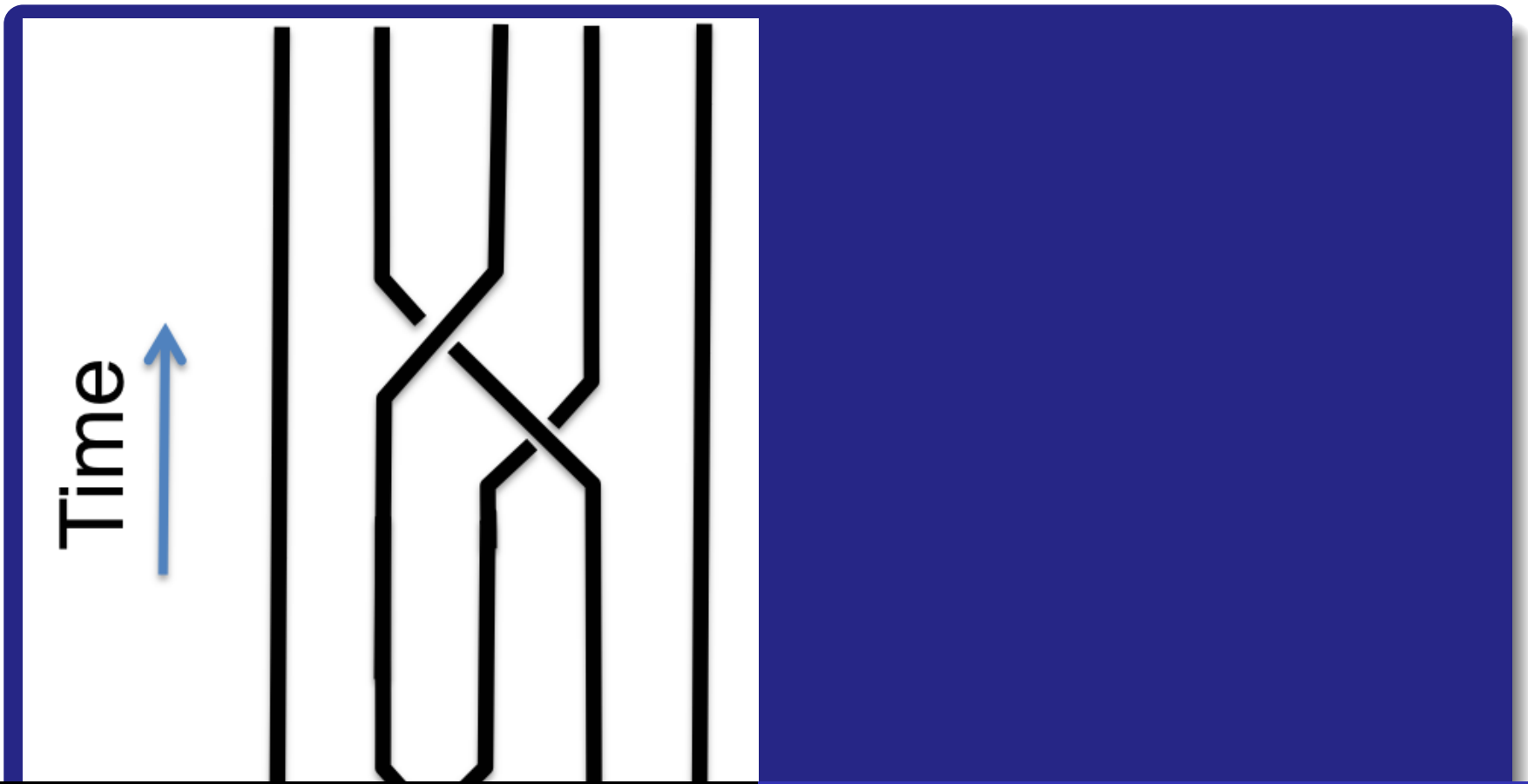
Quantum Gates

- Nand complete for classical circuits
- Complete Set, CNot and 1-spin qubits
- Build big ole matrices by decomposing them into matrices acting of small kronecker pieces
- Measurement (reading some current or voltage or something)



Topological Quantum Gates

- Anyons - Quasiparticles.
- If it smells and quacks like a duck
- Baseballs, atoms, nuclei, electrons quarks?
- Braiding of quasiparticles
- Annihilating quasiparticles - Measurement



Representable Functors

Any functor that is isomorphic to `(->) a` Gibbons Naperian Functors Connects the Functor Vector to the Free Vector style One slide? For everything?

Functor Vectors

(„ „ „) ntuples Kmett's linear package Compose = Kron

Product = Direct Sum Natural Transforomations Indexing into with `fmap`

Hey ho buddi

here we is

```
myprog :: IO ()
myprog = println "foo"
```

- item one
- item two

pandoc -t beamer -V theme:Warsaw topo.md -o topo.pdf

Vectors

Automating mechanical aspects

Category Theory

Category: Objects Morphisms Monoidal Category Braided

Point free programming exposes categorical underpinnings Vect Kliesli Arrow of Linear Monad Physical processes

Linear Types $\sim$ Categories don't have to be cartesian (dup and proj).

2 Vect anyons

Since I'm not done with this, I am unlikely to talk about it