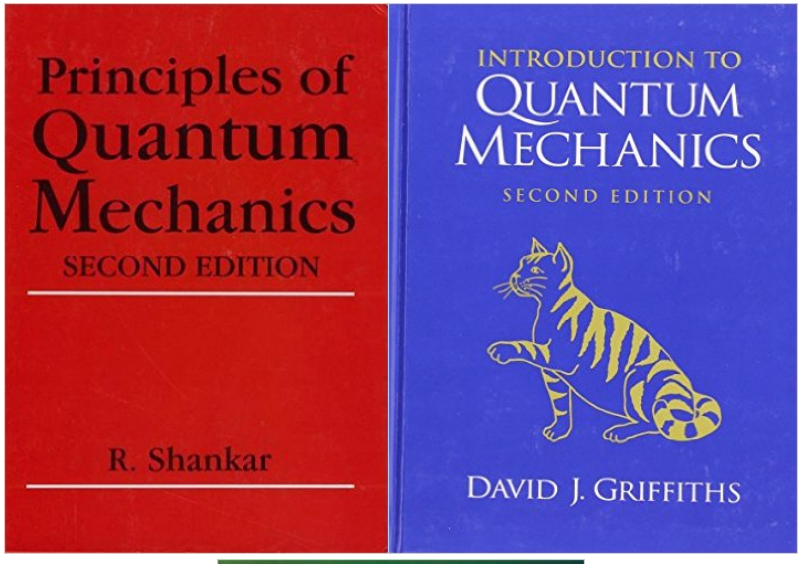
Learn Quantum Mechanics with Haskell

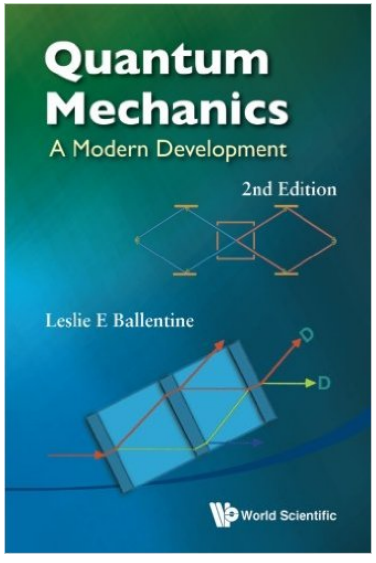
Scott N. Walck
Department of Physics
Lebanon Valley College
Annville, Pennsylvania, USA

TFPIE 2016
College Park, Maryland, USA
June 7, 2016

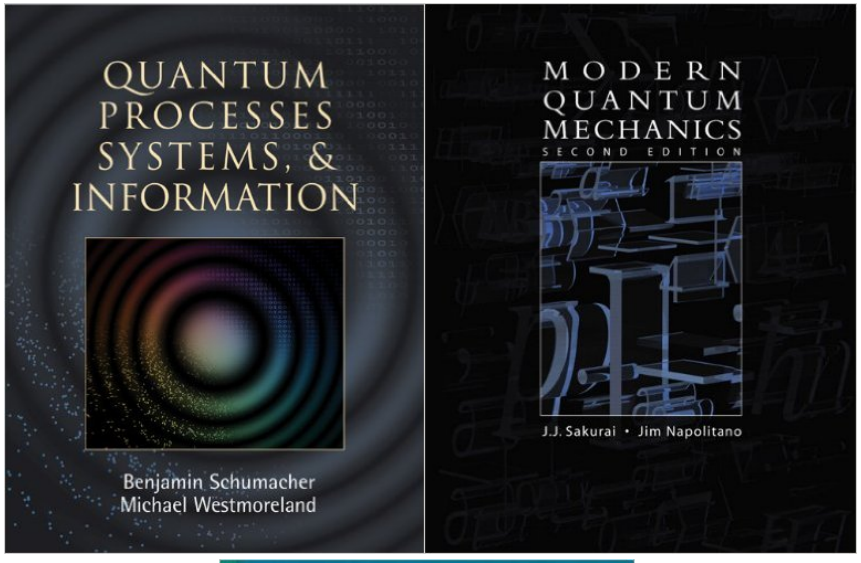
First choice in teaching quantum mechanics

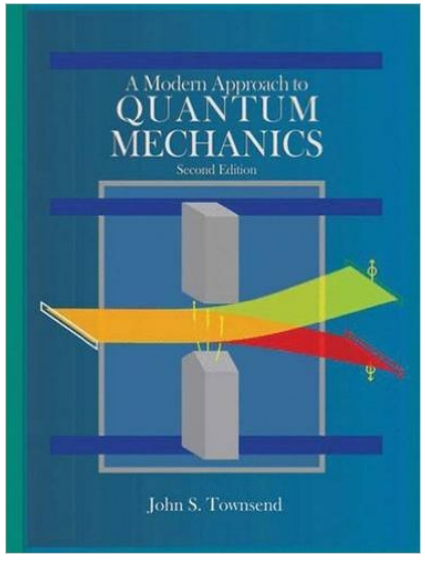
Waves first





Qubits first





Attitude toward computers in learning physics

1. Give students computational/lingustic building blocks.
   - could be raw computer language constructs
   - could be specially designed functions and structures
2. Ask them to build something.
   - could be solving a homework problem
   - could be expressing a theory
   - creative
   - computer gives feedback
   - fun

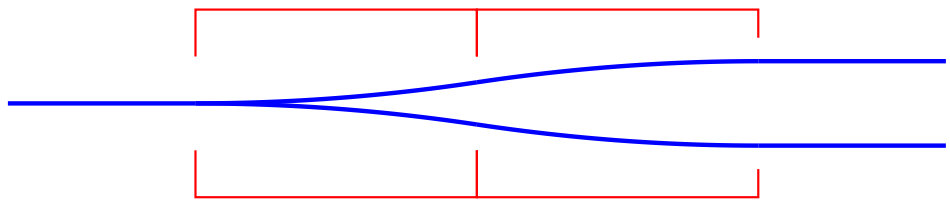
Want the creative process to engage the important ideas we're teaching.

Building Blocks for Quantum Mechanics

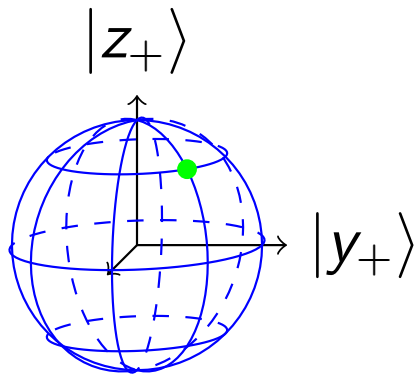
Laboratory experiments use a different language from theory.

Talk Outline:

1. `BeamStack`: a small language for describing Stern-Gerlach experiments

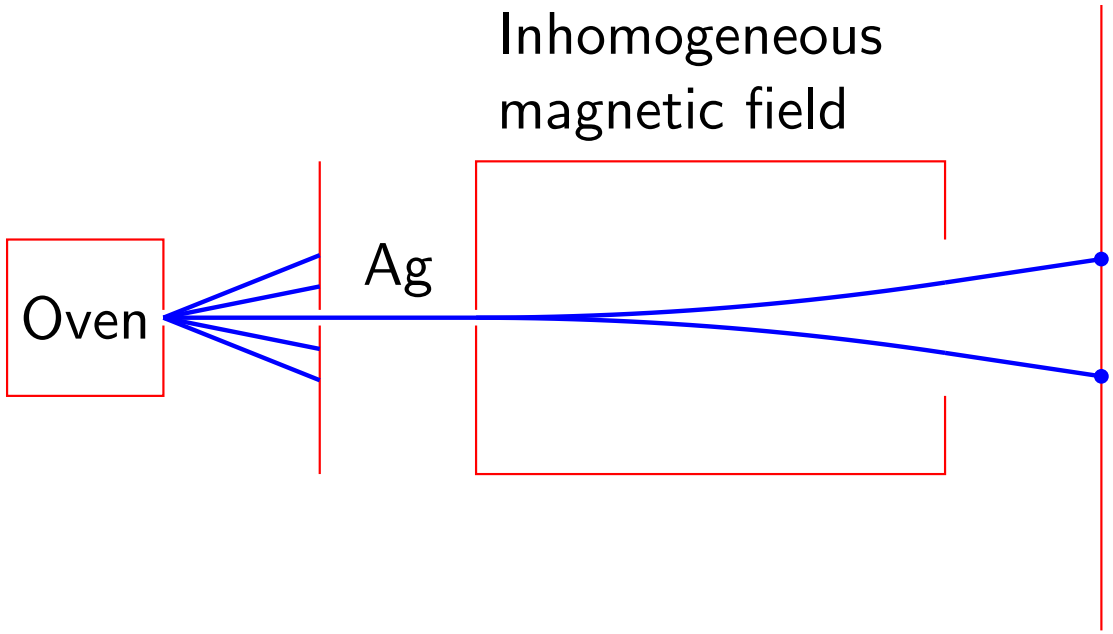


2. `Ket`: a calculational language that supports modern quantum notation



3. Using `Ket` to implement a simplified beam language

Stern-Gerlach experiment (1922)



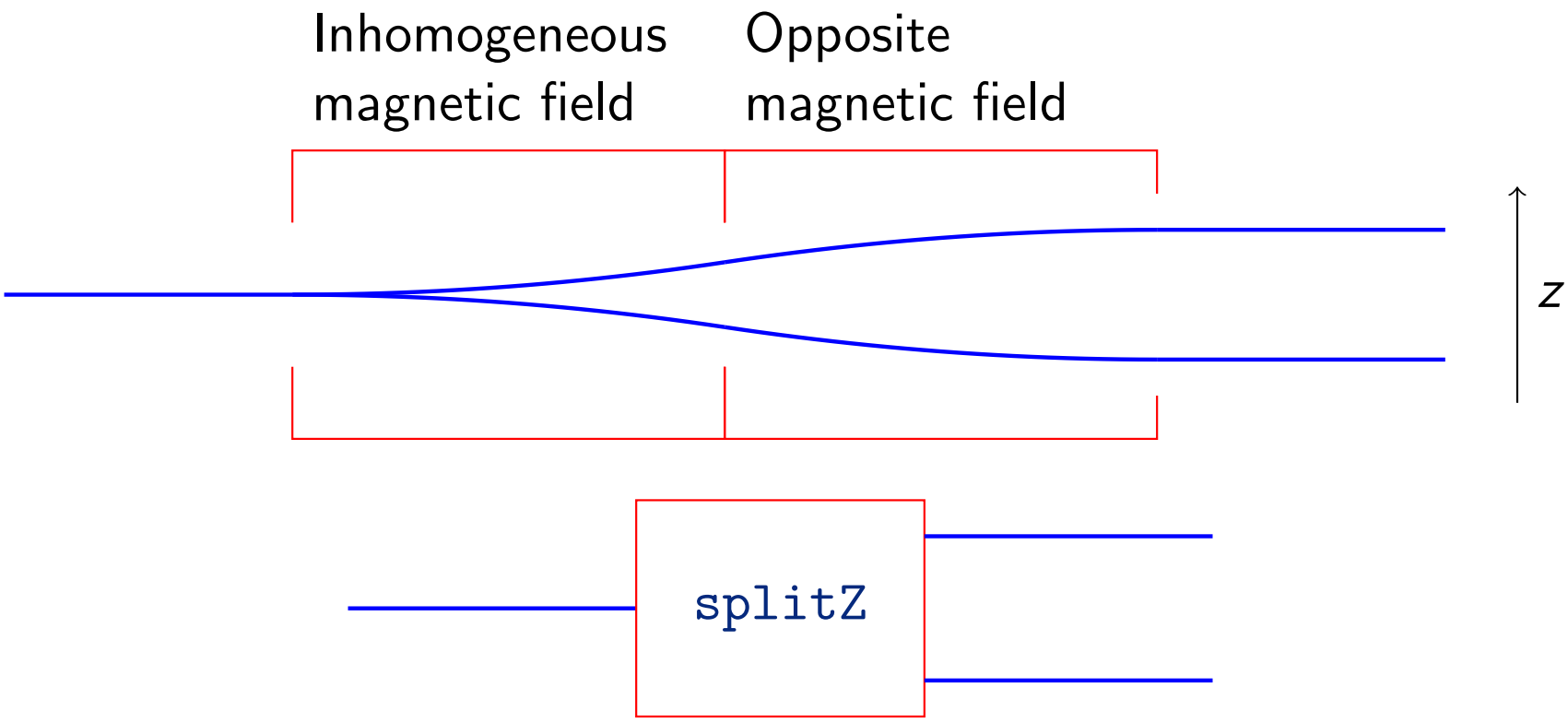
- classical mechanics predicts a range in amount deflection
- get exactly *two* types of deflection
- puts the *quantum* in quantum mechanics
- Ag acts like a spin-1/2 particle

A type for a collection of beams

```
data BeamStack
randomBeam :: BeamStack
dropBeam   :: BeamStack -> BeamStack
flipBeams  :: BeamStack -> BeamStack
```

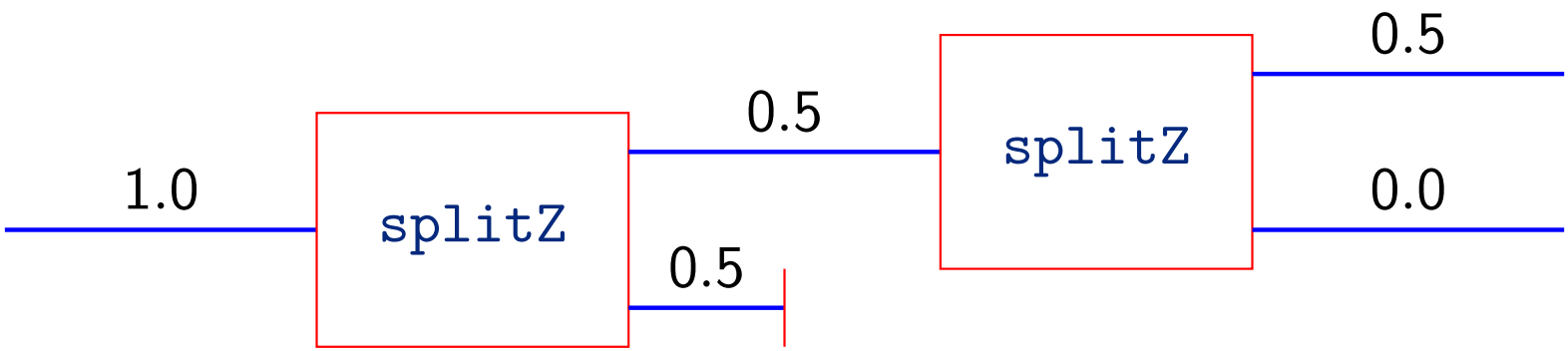
provided by `Physics.Learn.BeamStack` of *learn-physics* package

Stern-Gerlach beam splitter




```
splitX :: BeamStack -> BeamStack
splitY :: BeamStack -> BeamStack
splitZ :: BeamStack -> BeamStack
split  :: Double -> Double -> BeamStack -> BeamStack
```

Townsend's Experiment 1: Reproducibility

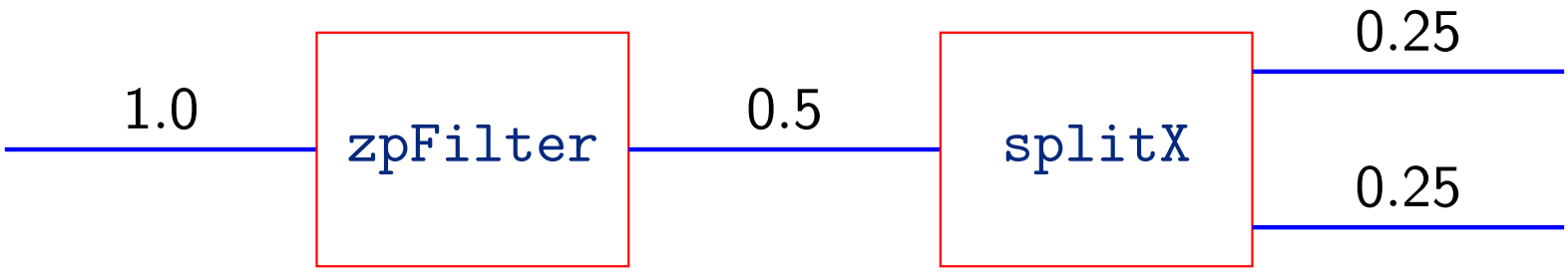



```
GHCi, version 7.10.2: http://www.haskell.org/ghc/  :? for help
Prelude> :m Physics.Learn.BeamStack
Prelude Physics.Learn.BeamStack> randomBeam
Beam of intensity 1.0
Prelude Physics.Learn.BeamStack> splitZ it
Beam of intensity 0.5
Beam of intensity 0.5
Prelude Physics.Learn.BeamStack> dropBeam it
Beam of intensity 0.5
Prelude Physics.Learn.BeamStack> splitZ it
Beam of intensity 0.5
Beam of intensity 0.0
```

To filter is to split and then drop

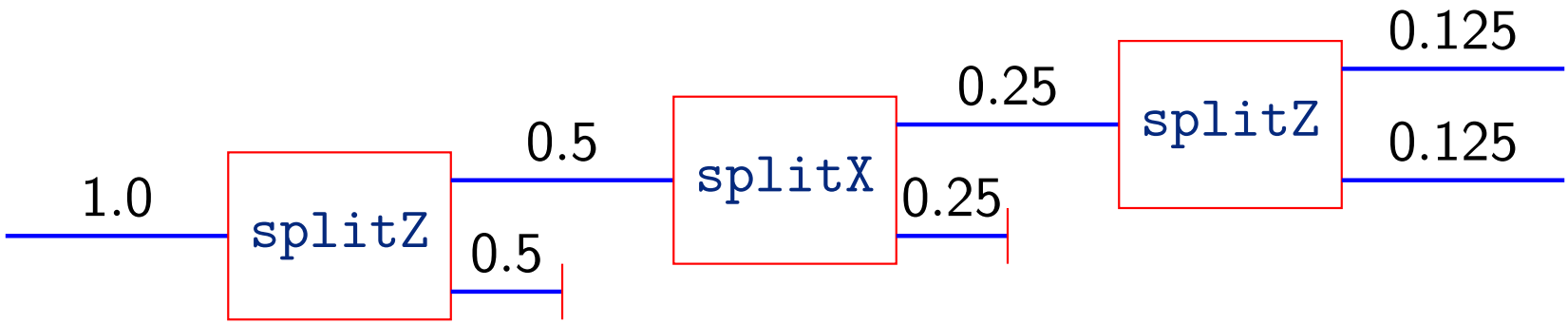
```
xpFilter :: BeamStack -> BeamStack
xpFilter = dropBeam . splitX

xmFilter :: BeamStack -> BeamStack
xmFilter = dropBeam . flipBeams . splitX

zpFilter :: BeamStack -> BeamStack
zpFilter = dropBeam . splitZ

zmFilter :: BeamStack -> BeamStack
zmFilter = dropBeam . flipBeams . splitZ
```

Townsend's Experiment 2: Z then X



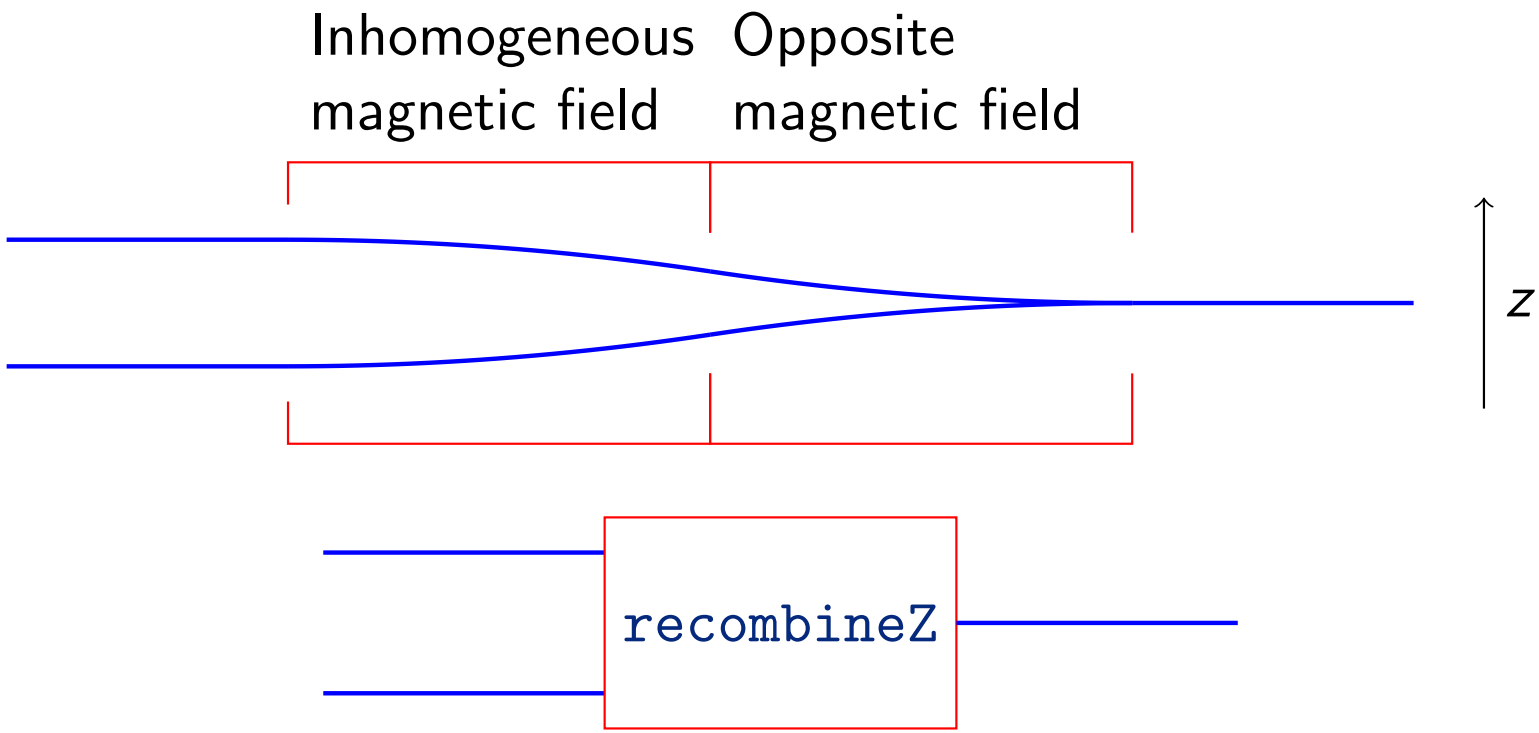

```
GHCi, version 7.10.2: http://www.haskell.org/ghc/  :? for help
Prelude> :m Physics.Learn.BeamStack
Prelude Physics.Learn.BeamStack> zpFilter randomBeam
Beam of intensity 0.5
Prelude Physics.Learn.BeamStack> splitX it
Beam of intensity 0.2500000000000006
Beam of intensity 0.2499999999999994
```

Townsend's Experiment 3: Z then X then Z



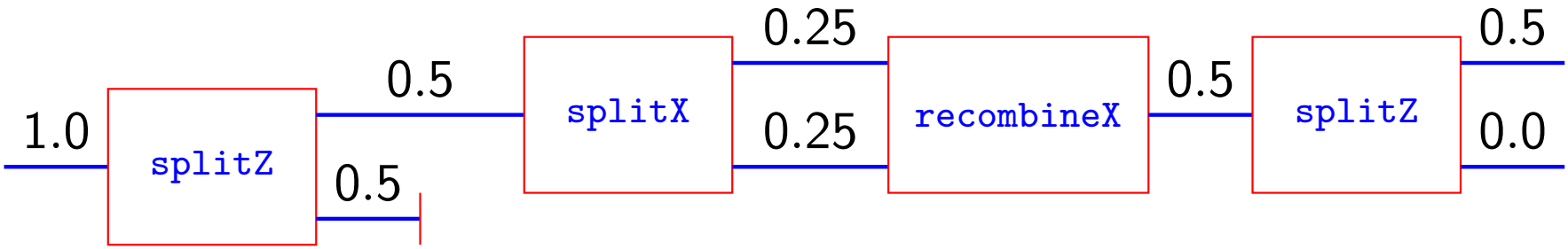

```
Prelude Physics.Learn.BeamStack> randomBeam
Beam of intensity 1.0
Prelude Physics.Learn.BeamStack> splitZ it
Beam of intensity 0.5
Beam of intensity 0.5
Prelude Physics.Learn.BeamStack> dropBeam it
Beam of intensity 0.5
Prelude Physics.Learn.BeamStack> splitX it
Beam of intensity 0.25000000000000006
Beam of intensity 0.24999999999999994
Prelude Physics.Learn.BeamStack> dropBeam it
Beam of intensity 0.25000000000000006
Prelude Physics.Learn.BeamStack> splitZ it
Beam of intensity 0.12500000000000006
Beam of intensity 0.125
```

Stern-Gerlach beam recombiner




```
recombineX :: BeamStack -> BeamStack
recombineY :: BeamStack -> BeamStack
recombineZ :: BeamStack -> BeamStack
recombine  :: Double -> Double -> BeamStack -> BeamStack
```

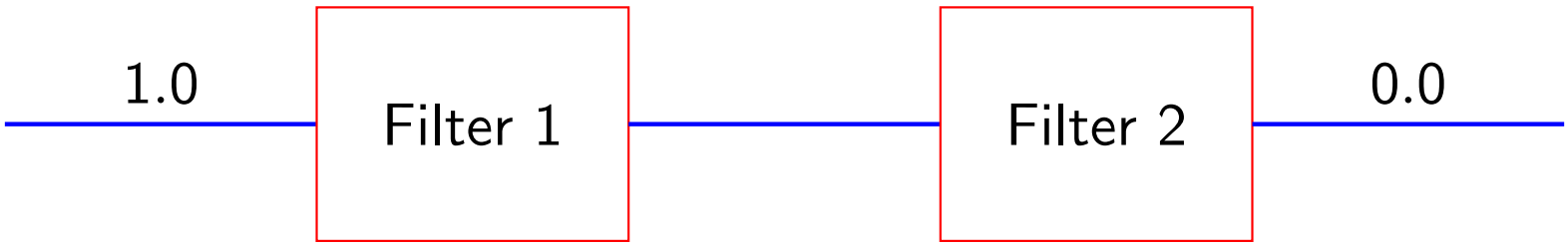
Townsend's Experiment 4: Recombination




```
Prelude Physics.Learn.BeamStack> randomBeam
Beam of intensity 1.0
Prelude Physics.Learn.BeamStack> splitZ it
Beam of intensity 0.5
Beam of intensity 0.5
Prelude Physics.Learn.BeamStack> dropBeam it
Beam of intensity 0.5
Prelude Physics.Learn.BeamStack> splitX it
Beam of intensity 0.2500000000000006
Beam of intensity 0.24999999999999994
Prelude Physics.Learn.BeamStack> recombineX it
Beam of intensity 0.5
Prelude Physics.Learn.BeamStack> splitZ it
Beam of intensity 0.5
Beam of intensity 0.0
```

A puzzle for students

Find a sequence of two filters such that no particles exit the second filter.



Is it possible to find a third filter to place between the first two, such that particles now flow from the last filter?

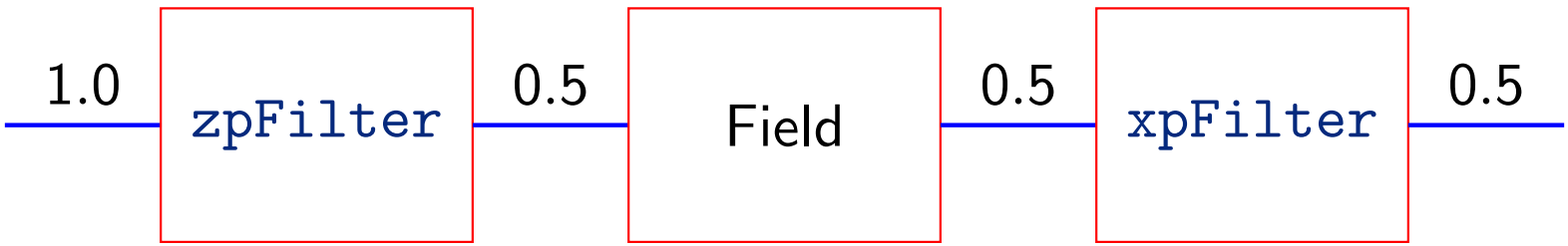
Applying a magnetic field to a beam

```
applyBFieldX :: Double -> BeamStack -> BeamStack
applyBFieldY :: Double -> BeamStack -> BeamStack
applyBFieldZ :: Double -> BeamStack -> BeamStack
applyBField  :: Double -> Double -> Double
             -> BeamStack -> BeamStack
```

- B is the symbol for magnetic field
- Field is applied to top beam of the stack
- First argument is intensity/duration combination

Another puzzle for students

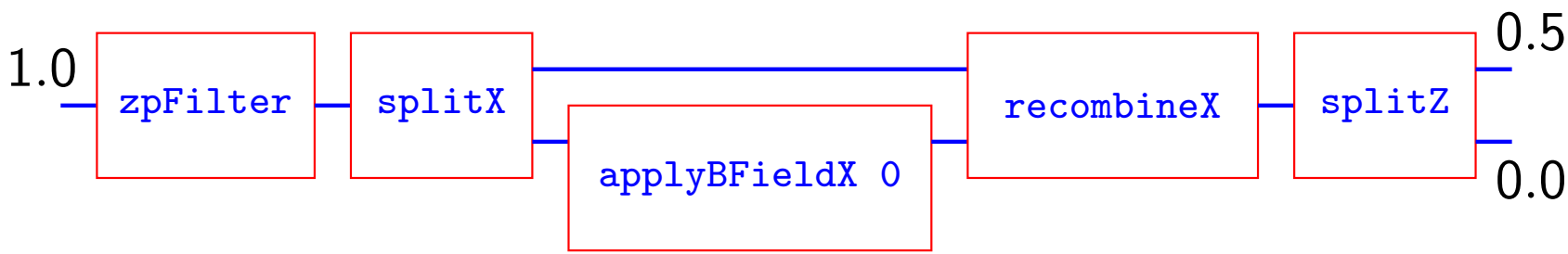
Can you find a direction and duration for a uniform magnetic field to act on a beam exiting a `zpFilter` so that the entire beam intensity will make it through an `xpFilter`?



Does this suggest a way to think about what a uniform magnetic field does?

Last puzzle for now

In Townsend's Experiment 4, suppose we apply a uniform magnetic field in the x direction to the lower beam between the x-splitter and x-recombiner. If the duration of application of the magnetic field is zero, the results will match that of Experiment 4.

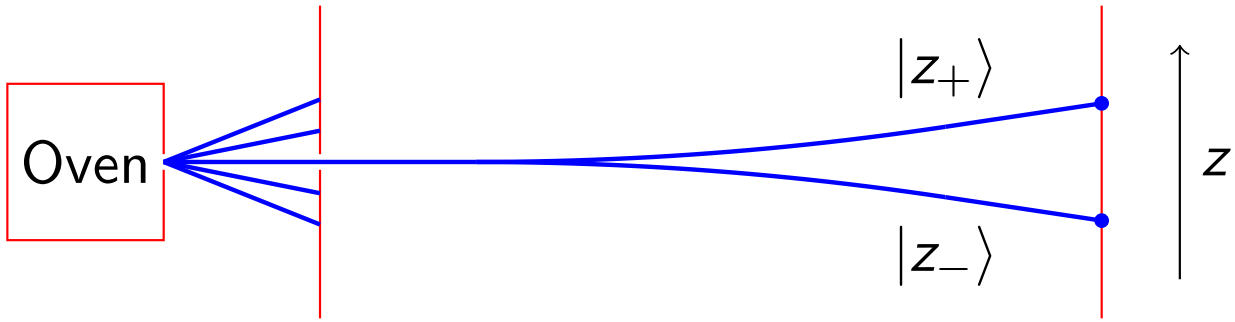


What is the next shortest duration when the results match again? Is the answer surprising?

BeamStack language summary

```
data BeamStack
randomBeam :: BeamStack
dropBeam   :: BeamStack -> BeamStack
flipBeams  :: BeamStack -> BeamStack

splitZ :: BeamStack -> BeamStack

recombineZ :: BeamStack -> BeamStack

applyBFieldZ :: Double -> BeamStack -> BeamStack
```

Quantum Theory uses a vector space



Every state of this spin-1/2 particle is a superposition of the basis states $|z_+\rangle$ and $|z_-\rangle$.

$$|\psi\rangle = \alpha_+ |z_+\rangle + \alpha_- |z_-\rangle$$

$$|x_+\rangle = \tfrac{1}{\sqrt{2}} |z_+\rangle + \tfrac{1}{\sqrt{2}} |z_-\rangle \qquad |y_+\rangle = \tfrac{1}{\sqrt{2}} |z_+\rangle + \tfrac{i}{\sqrt{2}} |z_-\rangle$$

$$|x_-\rangle = \tfrac{1}{\sqrt{2}} |z_+\rangle - \tfrac{1}{\sqrt{2}} |z_-\rangle \qquad |y_-\rangle = \tfrac{1}{\sqrt{2}} |z_+\rangle - \tfrac{i}{\sqrt{2}} |z_-\rangle$$

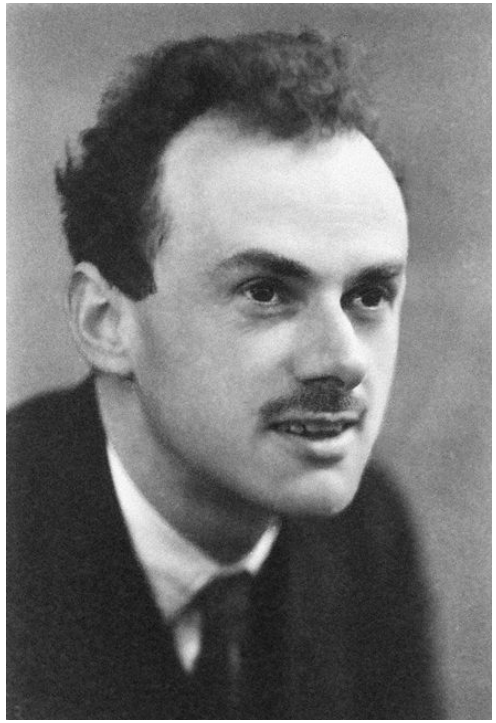
Dirac Notation is modern QM notation

$$\langle \phi | \psi \rangle$$

bracket

$\langle \phi |$ bra

$| \psi \rangle$ ket



Paul Dirac
1902–1984

Kets for spin-1/2 particles

A *ket* is a vector that describes the *state* of a particle.

```
data Ket
xp :: Ket
xm :: Ket
yp :: Ket
ym :: Ket
zp :: Ket
zm :: Ket
np :: Double -> Double
   -> Ket
nm :: Double -> Double
   -> Ket
```

$$|x_+\rangle = \frac{1}{\sqrt{2}}|z_+\rangle + \frac{1}{\sqrt{2}}|z_-\rangle$$
$$|x_-\rangle = \frac{1}{\sqrt{2}}|z_+\rangle - \frac{1}{\sqrt{2}}|z_-\rangle$$
$$|y_+\rangle = \frac{1}{\sqrt{2}}|z_+\rangle + \frac{i}{\sqrt{2}}|z_-\rangle$$
$$|y_-\rangle = \frac{1}{\sqrt{2}}|z_+\rangle - \frac{i}{\sqrt{2}}|z_-\rangle$$
$$|z_+\rangle$$
$$|z_-\rangle$$
$$|n_+(\theta,\phi)\rangle = \cos\frac{\theta}{2}|z_+\rangle + e^{i\phi}\sin\frac{\theta}{2}|z_-\rangle$$
$$|n_-(\theta,\phi)\rangle = \sin\frac{\theta}{2}|z_+\rangle - e^{i\phi}\cos\frac{\theta}{2}|z_-\rangle$$

provided by `Physics.Learn.Ket` of *learn-physics* package

Calculating probabilities

- A measurement result is associated with an *outcome ket* $|\phi\rangle$.

$$P = |\langle\phi|\psi\rangle|^2$$

```
magnitude (dagger phi <> psi) ** 2
```

Homework problem: probability calculation

Consider a spin-1/2 particle in the state

$$|\psi\rangle = \left(\frac{2}{5} + \frac{2}{5}i\right)|z_+\rangle + \left(\frac{1}{5} + \frac{4}{5}i\right)|z_-\rangle\,.$$

If a measurement of spin in the y-direction is made, what is the probability of obtaining spin down?

```
GHCi, version 7.10.2: http://www.haskell.org/ghc/  :? for help
Prelude> :m Physics.Learn.Ket
Prelude Physics.Learn.Ket> let psi = (2/5+2/5*i) <> zp + (1/5+4/5*i) <> zm
Prelude Physics.Learn.Ket> magnitude (dagger ym <> psi) ** 2
0.26
```

Operators for spin-1/2 particles

- Operators are used to describe *observables* like position, momentum, angular momentum, energy

```
data Operator
sx :: Operator
sy :: Operator
sz :: Operator
sn :: Double -> Double
   -> Operator
```

$$\sigma_x = |x_+\rangle\langle x_+| - |x_-\rangle\langle x_-|$$

$$\sigma_y = |y_+\rangle\langle y_+| - |y_-\rangle\langle y_-|$$

$$\sigma_z = |z_+\rangle\langle z_+| - |z_-\rangle\langle z_-|$$

$$|n_+\rangle\langle n_+| - |n_-\rangle\langle n_-|$$

$\frac{\hbar}{2}\sigma_z$ z component of angular momentum

Schrödinger equation describes time evolution

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H |\psi(t)\rangle$$

`Physics.Learn.Ket` provides

```
timeEv :: Double -> Operator -> Ket -> Ket

evolutionBlochSphereK :: Ket
                      -> (Double -> Operator)
                      -> IO ()
```

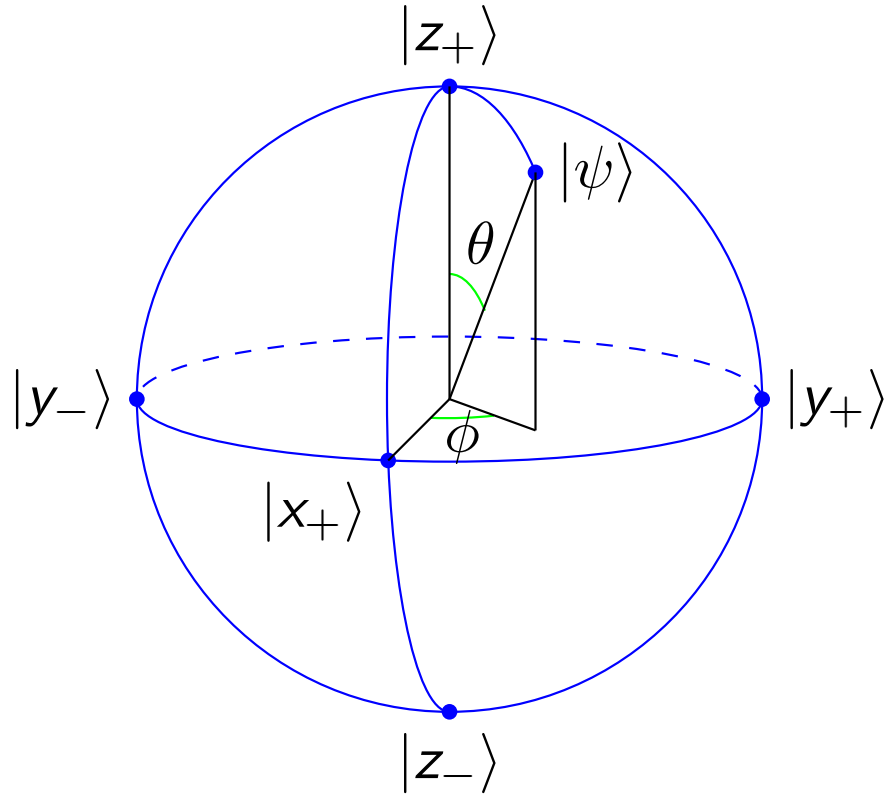
Bloch Sphere: Visualization for state of spin-1/2 particle

$$|\psi\rangle = \alpha_+ |z_+\rangle + \alpha_- |z_-\rangle$$

- global magnitude is irrelevant
- global phase is irrelevant

$$\tan\frac{\theta}{2} = \left|\frac{\alpha_-}{\alpha_+}\right| \qquad \phi = \arg\frac{\alpha_-}{\alpha_+}$$

- relative magnitude controls latitude
- relative phase controls longitude



Student Activity: Nuclear Magnetic Resonance

Activity: Show, on the Bloch Sphere, how the state of a spin-1/2 particle evolves under conditions of nuclear magnetic resonance.

```
ham :: Double ->  Double ->  Double
    ->  Double -> Operator
ham omega0 omegaR omega t
    = (omega0 / 2 :+ 0) <> sz
      + (omegaR / 2 * cos (omega * t) :+ 0) <> sx

main :: IO ()
main = evolutionBlochSphereK zm (ham 5 1 5)
```

`Ket` language summary

- data types `Ket`, `Bra`, `Operator`
- Dirac product to support modern Dirac notation
- calculation of probabilities
- Schrödinger equation solver
- visualization tools

Simplified Laboratory Language

```
data Beam
xpBeam :: Beam
xmBeam :: Beam
zpBeam :: Beam
zmBeam :: Beam
intensity :: Beam -> Double
splitX :: Beam -> (Beam,Beam)
splitZ :: Beam -> (Beam,Beam)
split  :: Double -> Double -> Beam -> (Beam,Beam)
zpFilter :: Beam -> Beam
zmFilter :: Beam -> Beam
recombineX :: (Beam,Beam) -> Beam
recombineZ :: (Beam,Beam) -> Beam
recombine  :: Double -> Double -> (Beam,Beam) -> Beam
applyBFieldX :: Double -> Beam -> Beam
applyBFieldZ :: Double -> Beam -> Beam
```

Thanks for Listening!