Schrodinger's cat alive!

http://mx-clojure.blogspot.com/

Oscar Riveros

October 9, 2013

In this chapter of the series "Quantum Programming with Quipper" try to discover an quantum algorithm that allows the Schrodinger's cat alive all 9 lives, whenever, it's enter the sinister black box of life and death.

1 Introduction

A state is a complete description of a physical system, In quantum Mdechanics, a state $|\psi\rangle$ of a system is a vector in the Hilbert space $\mathcal{H}$.

Consider a canonical base $\{|0\rangle, |1\rangle, |2\rangle, \ldots, |k\rangle \ldots |n-1\rangle\} \in \mathcal{H}_n$; The state $|\psi\rangle \in \mathcal{H}_n$ can be written as linear combination of basis states:

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

The coeficients

$$a_k = \langle k \mid \psi \rangle$$

are complex numbers and represent the ***probability amplitudes***; the probability of observing the state $|k\rangle$ is $p_k = \mid a_k \mid^2$.

By convetion, state vectors are assumed to be normal(ized), i.e, $\langle \psi \mid \psi \rangle = 1$. Therefore:

$$\sum_{k=0}^{n-1} \mid a_k \mid^2 = 1$$

2 Philosophical Programming (Rigorously Funny)

A *Qubit* is essentially a Schrodinger's cat, then as this has 9 lives, we can represent the super cat, as $|9\rangle :: QDInt$, then as $9_{10} \equiv 1001_2$:

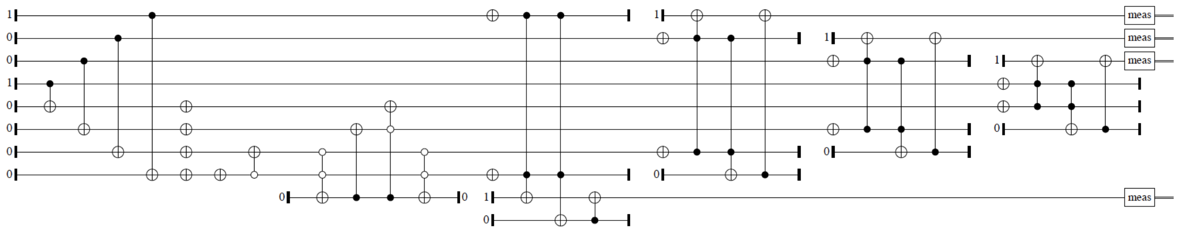
$$Cat = |1001\rangle = |1\rangle \otimes |0\rangle \otimes |0\rangle \otimes |1\rangle$$

Problem 1. *(Use Quipper)* There is a quantum algorithm Λ able to save our Schrodinger's cat of death? ie, such that $\Lambda |9\rangle = [1, 1, 1, 1](Bits)$ for all possible states of the cat?

```haskell
import Quipper
import QuipperLib.Simulation
import QuipperLib.Arith
import Control.Monad (zipWithM)

eru :: Qubit -> Qubit -> Circ Qubit
eru qlive qdeath                = do
        live                    <- qinit True
        death                   <- qinit False
        label                   live "There was Eru, the One."
        qdeath                  <- qnot qdeath
        qlive                   <- qnot qlive
        label                   qdeath "God's dice"
        label                   qlive "God's dice"
        qnot_at                 live `controlled` [qlive, qdeath]
        qnot_at                 death `controlled` [qdeath, qlive]
        livedeath                <- qnot live `controlled` death
        qdiscard                [death,qlive,qdeath]
        return                  livedeath

limbus :: QDInt -> QDInt -> Circ [Qubit]
limbus live death               = do
        let qlive               = qulist_of_qdint_lh live
        label qlive             "Live"
        let qdeath              = qulist_of_qdint_lh death
        label qlive             "Death"
        livedeath               <- zipWithM eru qlive qdeath
        return                  (map (\(nothing) -> nothing) livedeath)

schrodingers_cat :: [Qubit] -> Circ [Qubit]
schrodingers_cat xs             = do
        let cat                 = qdint_of_qulist_lh xs
        label                   cat "Cat IN Limbus"
        (live,death)            <- q_negate cat
        qcat                    <- limbus live death
        label                   qcat "Cat OUT Limbus"
        return                  qcat

black_box :: String -> Circ [Bit]
black_box scat                  = do
        qcat                    <- qinit_of_string scat
        label                   qcat "Cat IN Box"
        cats                    <- schrodingers_cat qcat
        label                   cats "Cat OUT Box"
        cats'                   <- measure cats
        return                  cats'

main :: IO ()
main                            = do
        print_generic           Preview black_box "1001"
        -- Cat State Nº 1
        out                     <- run_generic_io db black_box cat1
        putStrLn                ("Cat In State Nº 1 Is Alive? = " ++ show out)
        -- Cat State Nº 2
        out                     <- run_generic_io db black_box cat2
        putStrLn                ("Cat In State Nº 2 Is Alive? = " ++ show out)
        -- Cat State Nº 3
        out                     <- run_generic_io db black_box cat3
        putStrLn                ("Cat In State Nº 3 Is Alive? = " ++ show out)
        -- Cat State Nº 4
        out                     <- run_generic_io db black_box cat4
        putStrLn                ("Cat In State Nº 4 Is Alive? = " ++ show out)
        -- Cat State Nº 5
        out                     <- run_generic_io db black_box cat5
        putStrLn                 ("Cat In State Nº 5 Is Alive? = " ++ show out)
        -- Cat State Nº 6
        out                     <- run_generic_io db black_box cat6
        putStrLn                ("Cat In State Nº 6 Is Alive? = " ++ show out)
        where
                db :: Double
                db = undefined
                cat1 :: String
                cat1 = "1100"
                cat2 :: String
                cat2 = "1010"
                cat3 :: String
                cat3 = "1001"
                cat4 :: String
                cat4 = "0110"
                cat5 :: String
                cat5 = "0101"
                cat6 :: String
                cat6 = "0011"
```

Now running our quantum algorithm on the super cat.

Cat In State Nº 1 Is Alive? = [**True**, **True**, **True**, **True**]
Cat In State Nº 2 Is Alive? = [**True**, **True**, **True**, **True**]
Cat In State Nº 3 Is Alive? = [**True**, **True**, **True**, **True**]
Cat In State Nº 4 Is Alive? = [**True**, **True**, **True**, **True**]
Cat In State Nº 5 Is Alive? = [**True**, **True**, **True**, **True**]
Cat In State Nº 6 Is Alive? = [**True**, **True**, **True**, **True**]



[1]

[1] oscar.riveros@gmail.com