ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΜΣ ΕΠΙΣΤΗΜΗ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑ Η/Υ

ΟΔΗΓΟΣ ΕΓΚΑΤΑΣΤΑΣΗΣ
Γλώσσας Quipper

ΚΟΥΒΕΛΑΣ ΘΕΟΔΩΡΟΣ
ΚΥΤΕΑ ΚΩΝΣΤΑΝΤΙΝΑ
ΧΡΗΣΤΙΔΗ ΑΓΓΕΛΙΚΗ-ΜΑΡΙΑ

Πάτρα, Ιούλιος 2017

Περιεχόμενα

Βήματα εγκατάστασης Quipper σε WINDOWS

ΒΗΜΑ 1

Λήψη της Haskell από το link https://www.haskell.org/platform/windows.html

Θα κατεβάσετε την έκδοση 8.0.2 core, για x86 ή x64, αναλόγως της έκδοσης των Windows που θα χρησιμοποιήσετε. Κάνετε εγκατάσταση εκτελώντας το αρχείο που κατεβάσατε.

Σημείωση:

Μετά την εγκατάσταση, για σύστημα Windows 10 με έκδοση 1703 ή μεγαλύτερη (*ελέγξτε εκτελώντας την εντολή winver*), θα πρέπει να αντικαταστήσετε με μια τροποποιημένη έκδοση, το αρχείο gcc.exe που βρίσκεται στο φάκελο:

Πχ C:\Program Files\Haskell Platform\8.0.2\mingw\bin

με το αρχείο που θα κατεβάσετε για την έκδοση των Windows που έχετε (x86 ή x64) από την ακόλουθη σελίδα, στο φάκελο binaries:

https://github.com/Mistuke/ghc-compat/tree/master/binaries/ghc-8.0.1

ΒΗΜΑ 2

Κατεβάζετε το βοηθητικό πρόγραμμα MSYS από τον παρακάτω σύνδεσμο και το εκτελείτε, ακολουθώντας τις οδηγίες που εμφανίζει.

http://downloads.sourceforge.net/mingw/MSYS-1.0.11.exe

ΒΗΜΑ 3

Μεταβείτε στις ιδιότητες του συστήματος : επιλέξτε για προχωρημένους -> μεταβλητές περιβάλλοντος (στο κάτω μέρος) -> επεξεργαστείτε τη μεταβλητή χρήστη PATH, προσθέτοντας στο τέλος :

;C:\msys\1.0\bin

Στη συνέχεια μεταβείτε στην γραμμή εντολών (cmd) και εκτελέστε την ακόλουθη εντολή για την επιβεβαίωση της σωστής εγκατάστασης του MSYS:

sh --login –i

ΒΗΜΑ 4

Εκτελέστε την αρχικοποίηση για το cabal-install με την εντολή

```
cabal user-config init
```

Τροποποιήστε το αρχείο ρυθμίσεων cabal για να περιέχει τις ακόλουθες γραμμές :

```
extra-prog-path: C:\Program Files\Haskell Platform\8.0.2\msys\usr\bin
extra-lib-dirs: C:\Program Files\Haskell Platform\8.0.2\mingw\lib
extra-include-dirs: C:\Program Files\Haskell Platform\8.0.2\mingw\include
```

ΒΗΜΑ 5

Μεταβείτε στην γραμμή εντολών (cmd) και εκτελέστε τις ακόλουθες εντολές με τη σειρά που δίνονται :

- cabal update
- cabal install random
- cabal install mtl
- cabal install primes
- cabal install Lattices
- cabal install zlib
- cabal install easyrender
- cabal install fixedprec
- cabal install newsynth
- cabal install containers
- cabal install set-monad
- cabal install QuickCheck

ΒΗΜΑ 6

Κάντε λήψη και ύστερα αποσυμπίεση της Quipper στο C:\quipper

(από www.mathstat.dal.ca/~selinger/quipper/)

ΒΗΜΑ 7

Προσθέστε " {-# OVERLAPPABLE #-} " στα instance στο Libraries\ShowAll.hs

Προσθέστε " {-# OVERLAPPABLE #-} " στα instance στο quipper\Quipper\Labels.hs

Προσθέστε " {-# OVERLAPPABLE #-} " στα instance στο quipper\Quipper\QData.hs

ΒΗΜΑ 8

Μεταβείτε στην γραμμή εντολών (cmd) και εκτελέστε **make** στον κεντρικό φάκελο της quipper (χρόνος μεταγλώττισης ~1h με 1h 30m σε Dual Core Pentium 2GHz).

Αυτό θα μεταγλωττίσει όλες τις απαραίτητες βιβλιοθήκες για κβαντικούς υπολογισμούς, αλλά και την κεντρική βιβλιοθήκη για τη γλώσσα Quipper και τα παραδείγματα που περιέχονται στη γλώσσα.

ΒΗΜΑ 9

Πρόσθεσε το "C:\quipper\quipper\scripts" στο PATH του χρήστη. Έτσι θα μπορεί να εκτελείται το βοηθητικό πρόγραμμα “quipper” για την μεταγλώττιση των προγραμμάτων.

ΒΗΜΑ 10

Για να μεταγλωττίσετε και να εκτελέσετε κάποιο πρόγραμμα, πρέπει να μεταβείτε σε γραμμή εντολών (cmd) και στον φάκελο που έχετε το πρόγραμμα προς μεταγλώττιση και να εκτελέσετε π.χ. quipper.bat And_gate.hs

ΕΚΤΕΛΕΣΗ

- Μετάβαση στην γραμμή εντολών (cmd)
- Εκτέλεση των εξής εντολών:
    - cd /
    - cd file (όπου file ο φάκελος στον οποίο βρίσκονται τα αρχεία προς μεταγλώττιση)
    - quipper name.hs (όπου name το όνομα του αρχείου που θέλουμε να μεταγλωττίσουμε)
    - name.exe (εκτέλεση του εκτελέσιμου αρχείου που δημιουργήθηκε από την παραπάνω εντολή)

Βήματα εγκατάστασης Quipper σε linux

Ακολουθούν τα βήματα για εγκατάσταση της Quipper σε σύστημα linux. Για την εγκατάσταση χρησιμοποιήθηκε σύστημα Linux Ubuntu 16.04.02 LTS 64bit. Γενικά τα βήματα που θα χρειαστούν είναι τα ακόλουθα:

1. Εγκατάσταση Haskell-platform 8.0.2
2. Εγκατάσταση γλώσσας Quipper.

Στη συνέχεια παρουσιάζονται αναλυτικά.

Εγκατάσταση Haskell-platform 8.0.2

Για τα επόμενα βήματα θεωρούμε ότι το κατέβασμα των αρχείων γίνεται στον προσωπικό φάκελο του ενεργού χρήστη. Έτσι χρησιμοποιούμε λογαριασμό χρήστη με όνομα **user**, με τον προσωπικό του φάκελο να βρίσκεται στην θέση: **/home/user**

Σε περίπτωση διαφορετικού ονόματος χρήστη, όπου παρακάτω αναφέρονται τα στοιχεία του user, τροποποιείτε ανάλογα.

Βήμα 1 - Κατέβασμα της Haskell

Κατεβάζουμε την τρέχουσα generic έκδοση 8.0.2 χειροκίνητα, επειδή στο repository του Ubuntu η υπάρχουσα έκδοση είναι η 7.10.3 η οποία έχει πρόβλημα με την Quipper.

Θα κάνουμε λήψη της Haskell από την διεύθυνση *https://www.haskell.org/platform/linux.html*, επιλέγοντας την έκδοση ***generic***. Στη συνέχεια θα κατεβάσουμε την έκδοση 8.0.2 core, για x86 ή x64. Η έκδοση που θα κατεβάσουμε περιέχει το cabal-install.

```
wget https://haskell.org/platform/download/8.0.2/haskell-platform-
8.0.2-unknown-posix--minimal-x86_64.tar.gz
```

Βήμα 2 - Αποσυμπίεση και εγκατάσταση Haskell 8.0.2

Αποσυμπιέζουμε το αρχείο του προηγούμενου βήματος και κάνουμε την εγκατάσταση με τις ακόλουθες εντολές. Η εγκατάσταση απαιτεί δικαιώματα διαχειριστή.

```
$ tar xf haskell-platform-8.0.2-unknown-posix--minimal-x86_64.tar.gz
$ sudo ./install-haskell-platform.sh
```

```
user@snf-761898:~$ sudo ./install-haskell-platform.sh
Unpacking ./hp-usr-local.tar.gz to /...
Running /usr/local/haskell/ghc-8.0.2-x86_64/bin/activate-hs ...

Haskell set to:
    GHC         /usr/local/haskell/ghc-8.0.2-x86_64
    Haddocks    file:///usr/local/haskell/ghc-8.0.2-x86_64/doc/frames.html
    Other doc   file:///usr/local/haskell/ghc-8.0.2-x86_64/share/doc/ghc/html/index.html

Symlinks for command line tools (ghc, cabal, etc..) added to:
    /usr/local/bin

user@snf-761898:~$
```

Σημ.: Εάν η εγκατάσταση γίνει σε νεότερη έκδοση linux (π.χ. Ubuntu > 16.04) ή σε σύστημα που υποστηρίζει εξατομικευμένη εγκατάσταση εκτελέσιμων αρχείων, θα πρέπει να αλλαχθεί στο αρχείο

usr/local/haskell/ghc-___/lib/ghc-___/settings

η flag *"compiler supports -no-pie"* από *"NO"* σε *"YES"*.

Βήμα 3 – Ενημέρωση του cabal

Πριν την εγκατάσταση των απαιτούμενων για την μεταγλώττιση της Quipper βιβλιοθηκών, θα πρέπει να γίνει ενημέρωση του συστήματος cabal. Αυτό γίνεται εκτελώντας την εντολή

cabal update

```
user@snf-761898:~$ cabal update
Config file path source is default config file.
Config file /home/user/.cabal/config not found.
Writing default configuration to /home/user/.cabal/config
Downloading the latest package list from hackage.haskell.org
user@snf-761898:~$
```

Βήμα 4 - Εγκατάσταση απαραίτητων βιβλιοθηκών

Εγκαθιστούμε τις απαραίτητες για την μεταγλώττιση της Quipper βιβλιοθήκες με την σειρά που ακολουθεί :

cabal install random

cabal install mtl
(αν βγάζει σφάλμα ότι δεν βρίσκει το -lgmp, εγκαθιστούμε την βιβλιοθήκη libgmp10: sudo apt-get install libgmp10)

cabal install primes

cabal install Lattices

cabal install zlib
(αν βγάζει σφάλμα ότι δεν βρίσκει το zlib.h, εγκαθιστούμε τη βιβλιοθήκη zlib: sudo apt-get install zlib1g-dev)

cabal install easyrender

```
cabal install fixedprec

cabal install newsynth

cabal install containers

cabal install set-monad

cabal install QuickCheck
```

Εάν υπάρξει σφάλμα κατά την εγκατάσταση σε κάποια από τις προηγούμενες βιβλιοθήκες, η συνήθης λύση είναι παρόμοια με τις υποδείξεις στις παρενθέσεις. Δηλαδή, ψάχνουμε να εγκαταστήσουμε την βιβλιοθήκη που εμφανίζεται να λείπει από το σύστημα

Εγκατάσταση γλώσσας Quipper

Βήμα 5 - Κατέβασμα της Quipper

Κατεβάζουμε το αρχείο της Quipper με την εντολή:

```
wget http://www.mathstat.dal.ca/~selinger/quipper/downloads/quipper-0.8.tgz
```

Κατόπιν αποσυμπιέζουμε το αρχείο και μπαίνουμε στον φάκελο.

```
$ tar xf quipper-0.8.tgz
$ cd quipper-0.8
```

Βήμα 6 - Διόρθωση αρχείων της Quipper 0.8

Για να ολοκληρωθεί σωστά η μεταγλώττιση της Quipper την τρέχουσα έκδοση της Haskell, θα πρέπει να επέμβουμε στον κώδικα τριών (3) αρχείων.

Θα πρέπει στα αρχεία :

- Libraries/ShowAll.hs
- quipper/Quipper/Labels.hs
- quipper/Quipper/QData.hs

να προσθέσουμε μπροστά από κάθε instance το {-# OVERLAPPABLE #-}.

Για παράδειγμα στο αρχείο ShowAll.hs στη γραμμή 28, η εντολή :

```
instance Show (a -> b) where
```

θα πρέπει να γίνει :

```
instance {-# OVERLAPPABLE #-} Show (a -> b) where
```

Παρόμοια τροποποιούμε και τα υπόλοιπα instance στα προαναφερθέντα αρχεία.

Βήμα 7 - Μεταγλώττιση της Quipper

Κάνουμε μεταγλώττιση της γλώσσας Quipper εκτελώντας την εντολή

```
make
```

στον κεντρικό φάκελο.

Βήμα 8 - Ενημέρωση του PATH

Η εγκατάσταση της γλώσσας περιλαμβάνει και ένα εκτελέσιμο με όνομα “quipper” το οποίο θα μας βοηθήσει στο να μεταγλωττίσουμε τα δικά μας προγράμματα. Στην ουσία καλεί με τη σειρά του τον μεταγλωττιστή της Haskell (ghc) αλλά πρώτα κάνει μερικές απαραίτητες ρυθμίσεις για τις ανάγκες της Quipper.

Στο τρέχον σύστημα Ubuntu το περιβάλλον της κονσόλας λειτουργεί με το bash shell. Έτσι στην τελευταία γραμμή του αρχείου .bashrc προσθέτουμε την εντολή:

```
export PATH="$HOME/quipper-0.8/quipper/scripts:$PATH"
```

Αντίστοιχα πρέπει να γίνει η προσθήκη στο αρχείο εκκίνησης ανάλογα με το shell που χρησιμοποιείται, σε άλλη έκδοση του linux.

Παραδείγματα χρήσης

Ακολουθούν δυο παραδείγματα χρήσης της quipper για την μεταγλώττιση ενός προγράμματος σε περιβάλλοντα Windows και Linux. Το παράδειγμα Windows πραγματοποιείται σε υπολογιστή με Windows 7 32bit, ενώ το παράδειγμα με το Linux σε υπολογιστή με Linux Ubuntu 16.02 LTS 64bit χωρίς γραφικό περιβάλλον.

Παράδειγμα χρήσης σε Windows

Έστω το παρακάτω πρόγραμμα σε Quipper.

```
import Quipper

feynman :: (Qubit, Qubit) -> Circ (Qubit, Qubit)
feynman (q1, q2) = do
  qnot_at q2 `controlled` [q1]
  return (q1, q2)

main = print_simple Preview feynman
```

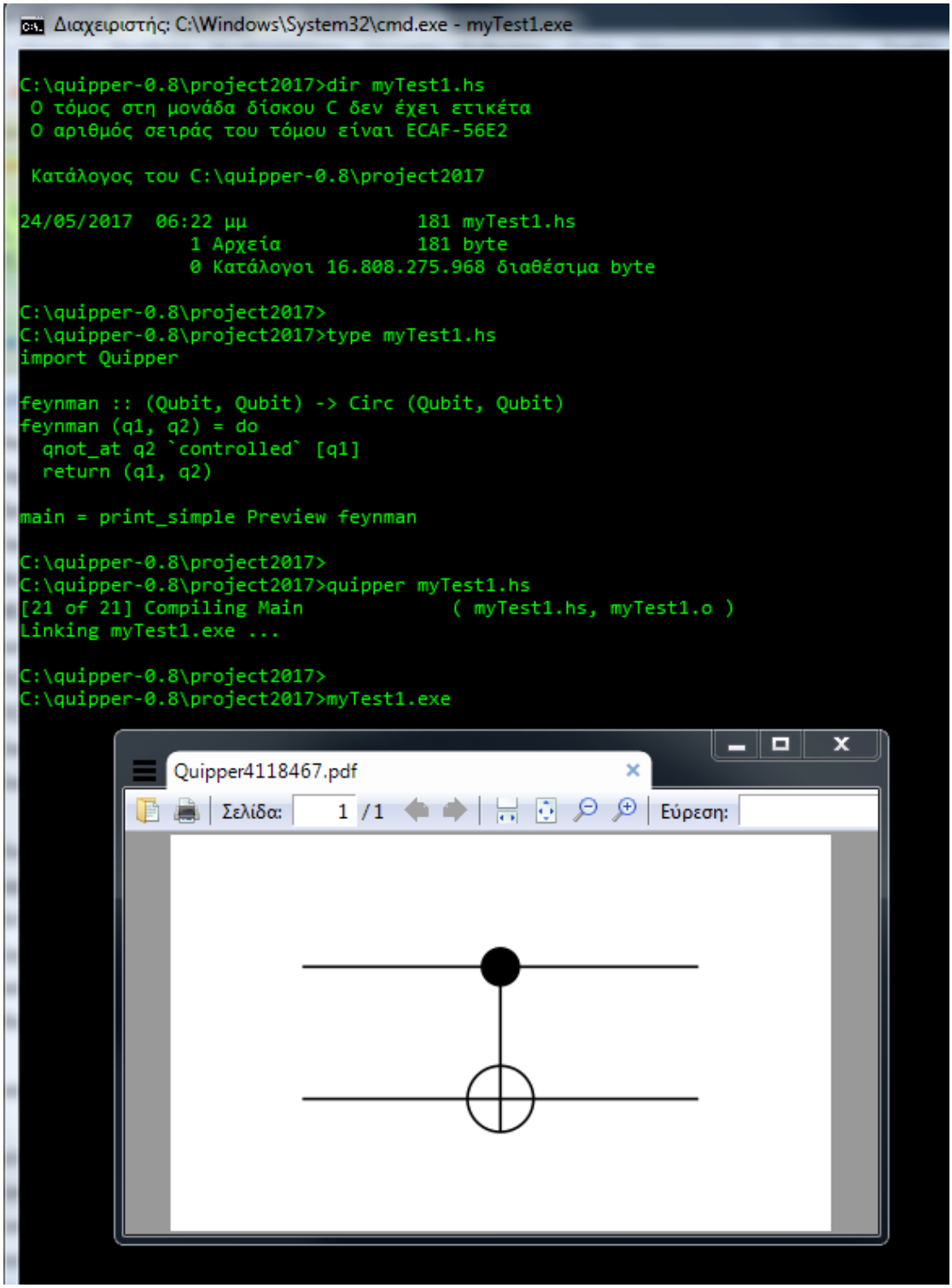
Οι εντολές του αρχείου συντάσσονται σε κάποιον κειμενογράφο, πχ Notepad.exe, και αποθηκεύουμε σε αρχείο κειμένου με κατάληξη .hs , έστω στο αρχείο: myTest1.hs

Κατόπιν περνάμε σε περιβάλλον γραμμής εντολών. Μεταβαίνουμε στο φάκελο με το αρχείο που δημιουργήσαμε και καλούμε την quipper για να το μεταγλωττίσουμε. Για παράδειγμα έστω το αρχείο myTest1.hs στο φάκελο C:\quipper-0.8\project2017. Θα το μεταγλωττίσουμε όπως παρακάτω:

C:\quipper-0.8\project2017>quipper myTest1.hs

Το αποτέλεσμα θα είναι η παραγωγή του εκτελέσιμου myTest1.exe στον ίδιο φάκελο. Επειδή η τελευταία γραμμή του παραπάνω κώδικα κάνει χρήση της 'Preview', τρέχοντας το myTest1.exe θα ανοίξει ο Acrobat Reader δείχνοντάς μας το προσωρινά δημιουργημένο αρχείο pdf του κυκλώματος.

Δείτε το παράδειγμα στο ακόλουθο στιγμιότυπο.



Εξομοίωση παραδείγματος

Για να ελέγξουμε τη λειτουργία του παραπάνω παραδείγματος θα πρέπει να κάνουμε μερικές προσθήκες στον κώδικα και να εκτελέσουμε μια εξομοίωση του κυκλώματος.

Θα χρειαστεί να προσθέσουμε στις βιβλιοθήκες στην κορυφή του αρχείου την QuipperLib, η οποία περιλαμβάνει την συνάρτηση εξομοίωσης run_generic_io. Στην βιβλιοθήκη αυτή περιέχονται και άλλες υλοποιήσεις της εξομοίωσης κβαντικού

κυκλώματος, αλλά εδώ θα δείξουμε την run_generic_io η οποία χρησιμοποιεί ως πηγή τυχαιότητας την αντίστοιχη του IO Monad της Haskell.

Έτσι τώρα ο κώδικας θα γίνει:

```
import Quipper
import QuipperLib.Simulation

feynman :: (Qubit, Qubit) -> Circ (Qubit, Qubit)
feynman (q1, q2) = do
  qnot_at q2 `controlled` [q1]
  return (q1, q2)

main = do
  test_run <- run_generic_io d feynman(True, True)
  putStrLn("Feynman gate test run = " ++ show test_run)
    where
      d :: Double
      d = undefined
```

Εδώ ορίζουμε μια μεταβλητή d ως Double και την περνάμε ως παράμετρο της run_generic_io, μαζί με την συνάρτηση που θα εξομοιώσουμε. Η μεταβλητή d χρειάζεται για να ορίσουμε απλά τον τύπο double.

Αφού τροποποιήσουμε το αρχείο και το αποθηκεύσουμε ως *myTest1b.hs* περνάμε σε περιβάλλον γραμμής εντολών. Μεταβαίνουμε στο φάκελο με το αρχείο και καλούμε την *quipper* για να το μεταγλωττίσουμε. Σύμφωνα με το προηγούμενο παράδειγμα, μεταβαίνουμε στο φάκελο *C:\quipper-0.8\project2017,* και κάνουμε την μεταγλώττιση όπως παρακάτω:

C:\quipper-0.8\project2017>quipper myTest1b.hs

Κατόπιν τρέχοντας το *myTest1b.exe* θα εκτελεστεί η εξομοίωση του κυκλώματος και θα μας εμφανίσει το αποτέλεσμα στην κονσόλα:

```
C:\quipper-0.8\project2017>myTest1b.exe
Feynman gate test run = (True,False)
```

Στο παραπάνω παράδειγμα βλέπουμε ότι το πρώτο qubit έχει τιμή TRUE άρα θα περιμένουμε να επηρεάσει την τιμή του δεύτερου, πράγμα που επιβεβαιώνουμε από το αποτέλεσμα της εκτέλεσης του προγράμματος.

Παράδειγμα χρήσης σε Linux

Έστω το παρακάτω πρόγραμμα σε Quipper.

```
import Quipper
feynman :: (Qubit, Qubit) -> Circ (Qubit, Qubit)
feynman (q1, q2) = do
  qnot_at q2 `controlled` [q1]
  return (q1, q2)
main = print_simple PDF feynman
```

Οι εντολές του αρχείου συντάσσονται σε κάποιον κειμενογράφο, πχ pico, και αποθηκεύουμε σε αρχείο κειμένου με κατάληξη .hs , έστω στο αρχείο: myTest1.hs

Κατόπιν περνάμε σε περιβάλλον γραμμής εντολών. Μεταβαίνουμε στο φάκελο με το αρχείο που δημιουργήσαμε και καλούμε την quipper για να το μεταγλωττίσουμε. Για παράδειγμα έστω το αρχείο myTest1.hs στο φάκελο ~/project2017/. Θα το μεταγλωττίσουμε όπως παρακάτω:

~$ cd project2017/
~/project2017$ quipper myTest1.hs

Αυτό θα δημιουργήσει το εκτελέσιμο αρχείο *myTest1* στον τρέχον φάκελο. Επειδή στο συγκεκριμένο παράδειγμα, στην τελευταία γραμμή ζητάμε παραγωγή ενός αρχείου pdf, κατά την εκτέλεση του προγράμματος θα πρέπει να κάνουμε ανακατεύθυνση της εξόδου του σε κάποιο αρχείο. Έτσι, για παράδειγμα, αν θέλουμε να αποθηκεύσουμε το κύκλωμα σε ένα αρχείο με όνομα *kikloma.pdf* τότε θα εκτελέσουμε το πρόγραμμά μας με την ακόλουθη εντολή:

~/project2017$ ./myTest1 >> kikloma.pdf

Έτσι θα έχει δημιουργηθεί στον τρέχον φάκελο το αρχείο *kikloma.pdf* με το αντίστοιχο κύκλωμα του κώδικά μας. Δείτε το παράδειγμα στο παρακάτω στιγμιότυπο:

```
user@snf-761898:~/project2017$ cat myTest1.hs
import Quipper

feynman :: (Qubit, Qubit) -> Circ (Qubit, Qubit)
feynman (q1, q2) = do
  qnot_at q2 `controlled` [q1]
  return (q1, q2)

main = print_simple PDF feynman
user@snf-761898:~/project2017$
user@snf-761898:~/project2017$ quipper myTest1.hs
[21 of 21] Compiling Main             ( myTest1.hs, myTest1.o )
Linking myTest1 ...
user@snf-761898:~/project2017$ ./myTest1 >> kikloma.pdf
user@snf-761898:~/project2017$
```

Εξομοίωση παραδείγματος

Για να ελέγξουμε τη λειτουργία του παραπάνω παραδείγματος θα πρέπει να κάνουμε μερικές προσθήκες στον κώδικα και να εκτελέσουμε μια εξομοίωση του κυκλώματος.

Θα χρειαστεί να προσθέσουμε στις βιβλιοθήκες στην κορυφή του αρχείου την QuipperLib, η οποία περιλαμβάνει την συνάρτηση εξομοίωσης run_generic_io. Στην βιβλιοθήκη αυτή περιέχονται και άλλες υλοποιήσεις της εξομοίωσης κβαντικού κυκλώματος, αλλά εδώ θα δείξουμε την run_generic_io η οποία χρησιμοποιεί ως πηγή τυχαιότητας την αντίστοιχη του IO Monad της Haskell.

Έτσι τώρα ο κώδικας θα γίνει:

```
import Quipper
import QuipperLib.Simulation

feynman :: (Qubit, Qubit) -> Circ (Qubit, Qubit)
feynman (q1, q2) = do
  qnot_at q2 `controlled` [q1]
  return (q1, q2)

main = do
  test_run <- run_generic_io d feynman(True, True)
  putStrLn("Feynman gate test run = " ++ show test_run)
    where
      d :: Double
      d = undefined
```

Εδώ ορίζουμε μια μεταβλητή d ως Double και την περνάμε ως παράμετρο της run_generic_io, μαζί με την συνάρτηση που θα εξομοιώσουμε. Η μεταβλητή d χρειάζεται για να ορίσουμε απλά τον τύπο double.

Αφού τροποποιήσουμε το αρχείο και το αποθηκεύσουμε ως *myTest1b.hs* περνάμε σε περιβάλλον γραμμής εντολών. Μεταβαίνουμε στο φάκελο με το αρχείο και καλούμε την *quipper* για να το μεταγλωττίσουμε. Σύμφωνα με το προηγούμενο παράδειγμα, μεταβαίνουμε στο φάκελο ~/project2017/, και κάνουμε την μεταγλώττιση όπως παρακάτω:

~/project2017$ quipper myTest1b.hs

Κατόπιν τρέχοντας το *myTest1b* θα εκτελεστεί η εξομοίωση του κυκλώματος και θα μας εμφανίσει το αποτέλεσμα στην κονσόλα:

```
user@snf-761898:~/project2017$ ./myTest1b
Feynman gate test run = (True,False)
```

Στο παραπάνω παράδειγμα βλέπουμε ότι το πρώτο qubit έχει τιμή TRUE άρα θα περιμένουμε να επηρεάσει την τιμή του δεύτερου, πράγμα που επιβεβαιώνουμε από το αποτέλεσμα της εκτέλεσης του προγράμματος.