Quantum Fourier Transform

Pablo Manrique

February 21, 2025

Abstract

This work presents a formalization of the Quantum Fourier Transform, a fundamental component of Shor's factoring algorithm, with proofs of its correctness and unitarity. The proof is carried out by induction, relying on the algorithm's recursive definition. This formalization builds upon the *Isabelle Marries Dirac* quantum computing library, developed by A. Bordg, H. Lachnitt, and Y. He.

Contents

theory *QFT*

imports
Isabelle-Marries-Dirac.Deutsch
begin

1 Some useful lemmas

lemma *gate-carrier-mat*[*simp*]:
assumes *gate n U*
shows $U \in$ *carrier-mat* $(2\hat{}n)$ $(2\hat{}n)$
⟨*proof*⟩

lemma *state-carrier-mat*[*simp*]:
assumes *state n* ψ
shows $\psi \in$ *carrier-mat* $(2\hat{}n)$ *1*
⟨*proof*⟩

lemma *state-basis-carrier-mat*[*simp*]:
|*state-basis n j*⟩ $\in$ *carrier-mat* $(2\hat{}n)$ *1*
⟨*proof*⟩

lemma *left-tensor-id*[*simp*]:
assumes $A \in$ *carrier-mat nr nc*
shows $(1_m\ 1) \bigotimes A = A$
⟨*proof*⟩

lemma *right-tensor-id*[*simp*]:
assumes $A \in$ *carrier-mat nr nc*
shows $A \bigotimes (1_m\ 1) = A$
⟨*proof*⟩

lemma *tensor-carrier-mat*[*simp*]:
assumes $A \in$ *carrier-mat ra ca*
and $B \in$ *carrier-mat rb cb*
shows $A \bigotimes B \in$ *carrier-mat* $(ra*rb)$ $(ca*cb)$
⟨*proof*⟩

lemma *smult-tensor*[*simp*]:
assumes *dim-col* $A > 0$ **and** *dim-col* $B > 0$
shows $(a \cdot_m A) \bigotimes (b \cdot_m B) = (a*b) \cdot_m (A \bigotimes B)$
⟨*proof*⟩

lemma *smult-tensor1*[*simp*]:
assumes *dim-col* $A > 0$ **and** *dim-col* $B > 0$
shows $a \cdot_m (A \bigotimes B) = (a \cdot_m A) \bigotimes B$
⟨*proof*⟩

lemma *set-list*:
set $[m..<n] = \{m..<n\}$

⟨*proof*⟩

lemma *sumof2*:
$(\sum k<(2::nat).\ f\ k) = f\ 0 + f\ 1$
⟨*proof*⟩

lemma *sumof4*:
$(\sum k<(4::nat).\ f\ k) = f\ 0 + f\ 1 + f\ 2 + f\ 3$
⟨*proof*⟩

2 The operator R_k

definition *R*:: *nat* ⇒ *complex Matrix.mat* **where**
R k = mat-of-cols-list 2 [[*1*, *0*],
[*0*, *exp*(*2*∗*pi*∗i/*2*^*k*)]]

3 The SWAP gate:

definition *SWAP*:: *complex Matrix.mat* **where**
SWAP ≡ *Matrix.mat 4 4* (λ(*i*,*j*). *if i=0* ∧ *j=0 then 1 else*
if i=1 ∧ *j=2 then 1 else*
if i=2 ∧ *j=1 then 1 else*
if i=3 ∧ *j=3 then 1 else 0*)

lemma *SWAP-index*:
SWAP $$ (*0*,*0*) = *1* ∧
SWAP $$ (*0*,*1*) = *0* ∧
SWAP $$ (*0*,*2*) = *0* ∧
SWAP $$ (*0*,*3*) = *0* ∧
SWAP $$ (*1*,*0*) = *0* ∧
SWAP $$ (*1*,*1*) = *0* ∧
SWAP $$ (*1*,*2*) = *1* ∧
SWAP $$ (*1*,*3*) = *0* ∧
SWAP $$ (*2*,*0*) = *0* ∧
SWAP $$ (*2*,*1*) = *1* ∧
SWAP $$ (*2*,*2*) = *0* ∧
SWAP $$ (*2*,*3*) = *0* ∧
SWAP $$ (*3*,*0*) = *0* ∧
SWAP $$ (*3*,*1*) = *0* ∧
SWAP $$ (*3*,*2*) = *0* ∧
SWAP $$ (*3*,*3*) = *1*
⟨*proof*⟩

lemma *SWAP-nrows*:
dim-row SWAP = *4*
⟨*proof*⟩

lemma *SWAP-ncols*:

dim-col SWAP = *4*
⟨*proof*⟩

lemma *SWAP-carrier-mat*[*simp*]:
SWAP ∈ *carrier-mat 4 4*
⟨*proof*⟩

The SWAP gate indeed swaps the states of two qubits (it is not necessary to assume unitarity)

lemma *SWAP-tensor*:
assumes *u* ∈ *carrier-mat 2 1*
and *v* ∈ *carrier-mat 2 1*
shows *SWAP* ∗ (*u* ⨂ *v*) = *v* ⨂ *u*
⟨*proof*⟩

3.1 Downwards SWAP cascade

fun *SWAP-down*:: *nat* ⇒ *complex Matrix.mat* **where**
SWAP-down 0 = 1_m *1*
| *SWAP-down* (*Suc 0*) = 1_m *2*
| *SWAP-down* (*Suc* (*Suc 0*)) = *SWAP*
| *SWAP-down* (*Suc* (*Suc n*)) = ((1_m (*2*^*n*)) ⨂ *SWAP*) ∗ ((*SWAP-down* (*Suc n*)) ⨂ (1_m *2*))

lemma *SWAP-down-carrier-mat*[*simp*]:
shows *SWAP-down n* ∈ *carrier-mat* (*2*^*n*) (*2*^*n*) (**is** *?P n*)
⟨*proof*⟩

3.2 Upwards SWAP cascade

fun *SWAP-up*:: *nat* ⇒ *complex Matrix.mat* **where**
SWAP-up 0 = 1_m *1*
| *SWAP-up* (*Suc 0*) = 1_m *2*
| *SWAP-up* (*Suc* (*Suc 0*)) = *SWAP*
| *SWAP-up* (*Suc* (*Suc n*)) = (*SWAP* ⨂ (1_m (*2*^*n*))) ∗ ((1_m *2*) ⨂ (*SWAP-up* (*Suc n*)))

lemma *SWAP-up-carrier-mat*[*simp*]:
shows *SWAP-up n* ∈ *carrier-mat* (*2*^*n*) (*2*^*n*) (**is** *?P n*)
⟨*proof*⟩

4 Reversing qubits

In order to reverse the order of n qubits, we iteratively swap opposite qubits (swap 0th and (n-1)th qubits, 1st and (n-2)th qubits, and so on).

fun *reverse-qubits*:: *nat* ⇒ *complex Matrix.mat* **where**
reverse-qubits 0 = 1_m *1*
| *reverse-qubits* (*Suc 0*) = (1_m *2*)

| *reverse-qubits* (*Suc* (*Suc 0*)) = *SWAP*
| *reverse-qubits* (*Suc n*) = ((*reverse-qubits n*) $\bigotimes$ (1_m *2*)) ∗ (*SWAP-down* (*Suc n*))

lemma *reverse-qubits-carrier-mat*[*simp*]:
(*reverse-qubits n*) ∈ *carrier-mat* (*2*^*n*) (*2*^*n*)
⟨*proof*⟩

5 Controlled operations

The two-qubit gate control2 performs a controlled U operation on the first qubit with the second qubit as control

definition *control2*:: *complex Matrix.mat* ⇒ *complex Matrix.mat* **where**
control2 U ≡ *mat-of-cols-list 4* [[*1*, *0*, *0*, *0*],
[*0*, *U*$$(*0*,*0*), *0*, *U*$$(*1*,*0*)],
[*0*, *0*, *1*, *0*],
[*0*, *U*$$(*0*,*1*), *0*, *U*$$(*1*,*1*)]]

lemma *control2-carrier-mat*[*simp*]:
shows *control2 U* ∈ *carrier-mat 4 4*
⟨*proof*⟩

lemma *control2-zero*:
assumes *dim-row v* = *2* **and** *dim-col v* = *1*
shows *control2 U* ∗ (*v* $\bigotimes$ |*zero*⟩) = *v* $\bigotimes$ |*zero*⟩
⟨*proof*⟩

lemma *vtensorone-index*[*simp*]:
assumes *dim-row v* = *2* **and** *dim-col v* = *1*
shows (*v* $\bigotimes$ |*one*⟩) $$ (*0*,*0*) = *0* ∧
(*v* $\bigotimes$ |*one*⟩) $$ (*1*,*0*) = *v* $$ (*0*,*0*) ∧
(*v* $\bigotimes$ |*one*⟩) $$ (*2*,*0*) = *0* ∧
(*v* $\bigotimes$ |*one*⟩) $$ (*3*,*0*) = *v* $$ (*1*,*0*)
⟨*proof*⟩

lemma *control2-one*:
assumes *dim-row v* = *2* **and** *dim-col v* = *1* **and** *dim-row U* = *2* **and** *dim-col U* = *2*
shows *control2 U* ∗ (*v* $\bigotimes$ |*one*⟩) = (*U*∗*v*) $\bigotimes$ |*one*⟩
⟨*proof*⟩

Given a single qubit gate U, control n U creates a quantum n-qubit gate that performs a controlled-U operation on the first qubit using the last qubit as control.

fun *control*:: *nat* ⇒ *complex Matrix.mat* ⇒ *complex Matrix.mat* **where**

control 0 U = 1_m *1*
| *control* (*Suc 0*) *U* = 1_m *2*
| *control* (*Suc* (*Suc 0*)) *U* = *control2 U*
| *control* (*Suc* (*Suc n*)) *U* =
((*1*$_m$ *2*) $\bigotimes$ *SWAP-down* (*Suc n*)) ∗ (*control2 U* $\bigotimes$ (*1*$_m$ (*2*^*n*))) ∗ ((*1*$_m$ *2*) $\bigotimes$ *SWAP-up* (*Suc n*))

lemma *control-carrier-mat*[*simp*]:
shows *control n U* ∈ *carrier-mat* (*2*^*n*) (*2*^*n*)
⟨*proof*⟩

6 Quantum Fourier Transform Circuit

6.1 QFT definition

The function kron is the generalization of the Kronecker product to a finite number of qubits

fun *kron*:: (*nat* ⇒ *complex Matrix.mat*) ⇒ *nat list* ⇒ *complex Matrix.mat* **where**
kron f [] = *1*$_m$ *1*
| *kron f* (*x*#*xs*) = (*f x*) $\bigotimes$ (*kron f xs*)

lemma *kron-carrier-mat*[*simp*]:
assumes ∀ *m*. *dim-row* (*f m*) = *2* ∧ *dim-col* (*f m*) = *1*
shows *kron f xs* ∈ *carrier-mat* (*2*^(*length xs*)) *1*
⟨*proof*⟩

lemma *kron-cons-right*:
shows *kron f* (*xs*@[*x*]) = *kron f xs* $\bigotimes$ *f x*
⟨*proof*⟩

We define the QFT product representation

definition *QFT-product-representation*:: *nat* ⇒ *nat* ⇒ *complex Matrix.mat* **where**
QFT-product-representation j n ≡ *1*/(*sqrt* (*2*^*n*)) ·$_m$
(*kron* (λ(*l*::*nat*). |*zero*⟩ + *exp* (*2*∗i∗*pi*∗*j*/(*2*^*l*)) ·$_m$ |*one*⟩)
(*map nat* [*1*..*n*]))

We also define the reverse version of the QFT product representation, which is the output state of the QFT circuit alone

definition *reverse-QFT-product-representation*:: *nat* ⇒ *nat* ⇒ *complex Matrix.mat* **where**
reverse-QFT-product-representation j n ≡ *1*/(*sqrt* (*2*^*n*)) ·$_m$
(*kron* (λ(*l*::*nat*). |*zero*⟩ + *exp* (*2*∗i∗*pi*∗*j*/(*2*^*l*)) ·$_m$ |*one*⟩)
(*map nat* (*rev* [*1*..*n*])))

6.2 QFT circuit

The recursive function controlled_rotations computes the controlled-R_k gates subcircuit of the QFT circuit at each stage (i.e. for each qubit).

fun *controlled-rotations*:: *nat* ⇒ *complex Matrix.mat* **where**
  controlled-rotations 0 = 1_m *1*
| *controlled-rotations* (*Suc 0*) = 1_m *2*
| *controlled-rotations* (*Suc n*) = (*control* (*Suc n*) (*R* (*Suc n*))) ∗
  ((*controlled-rotations n*) ⊗ (1_m *2*))

lemma *controlled-rotations-carrier-mat*[*simp*]:
  controlled-rotations n ∈ *carrier-mat* (*2^n*) (*2^n*)
⟨*proof*⟩

The recursive function QFT computes the Quantum Fourier Transform circuit.

fun *QFT*:: *nat* ⇒ *complex Matrix.mat* **where**
  QFT 0 = 1_m *1*
| *QFT* (*Suc 0*) = *H*
| *QFT* (*Suc n*) = ((1_m *2*) ⊗ (*QFT n*)) ∗ (*controlled-rotations* (*Suc n*)) ∗ (*H* ⊗ ((1_m (*2^n*))))

lemma *QFT-carrier-mat*[*simp*]:
  QFT n ∈ *carrier-mat* (*2^n*) (*2^n*)
⟨*proof*⟩

ordered_QFT reverses the order of the qubits at the end of the QFT circuit

definition *ordered-QFT*:: *nat* ⇒ *complex Matrix.mat* **where**
  ordered-QFT n ≡ (*reverse-qubits n*) ∗ (*QFT n*)

7 QFT circuit correctness

Some useful lemmas:

lemma *state-basis-dec*:
  assumes *j* < *2* ^ *Suc n*
  shows |*state-basis 1* (*j div 2^n*)⟩ ⊗ |*state-basis n* (*j mod 2^n*)⟩ = |*state-basis* (*Suc n*) *j*⟩
⟨*proof*⟩

lemma *state-basis-dec′*:
  ∀ *j*. *j* < *2* ^ *Suc n* ⟶
    |*state-basis n* (*j div 2*)⟩ ⊗ |*state-basis 1* (*j mod 2*)⟩ = |*state-basis* (*Suc n*) *j*⟩
⟨*proof*⟩

Action of the H gate in the circuit

lemma *H-on-first-qubit*:
assumes $j < 2 \hat{} Suc\ n$
shows $((H \bigotimes ((1_m (2\hat{}n))))) * |state\text{-}basis\ (Suc\ n)\ j\rangle =$
$1/sqrt\ 2 \cdot_m (|zero\rangle + exp(2*i*pi*(complex\text{-}of\text{-}nat\ (j\ div\ 2\hat{}n))/2) \cdot_m |one\rangle)$
$\bigotimes$
$|state\text{-}basis\ n\ (j\ mod\ 2\hat{}n)\rangle$
$\langle proof \rangle$

Action of the R gate in the circuit

lemma *R-action*:
assumes $j < 2 \hat{} Suc\ n$ **and** $j\ mod\ 2 = 1$
shows $(R\ (Suc\ n)) * (|zero\rangle + exp\ (2*i*pi*complex\text{-}of\text{-}nat\ (j\ div\ 2) / 2\hat{}n) \cdot_m |one\rangle) =$
$|zero\rangle + exp\ (2*i*pi*complex\text{-}of\text{-}nat\ j / 2\hat{}(Suc\ n)) \cdot_m |one\rangle$
$\langle proof \rangle$

Action of the SWAP cascades in the circuit

lemma *SWAP-up-action*:
$\forall j.\ j < 2 \hat{}(Suc\ (Suc\ n)) \longrightarrow$
$SWAP\text{-}up\ (Suc\ (Suc\ n)) * (|state\text{-}basis\ (Suc\ n)\ (j\ div\ 2)\rangle \bigotimes |state\text{-}basis\ 1\ (j\ mod\ 2)\rangle) =$
$|state\text{-}basis\ 1\ (j\ mod\ 2)\rangle \bigotimes |state\text{-}basis\ (Suc\ n)\ (j\ div\ 2)\rangle$
$\langle proof \rangle$

lemma *SWAP-down-action*:
$\forall j.\ j < 2 \hat{}Suc\ (Suc\ n) \longrightarrow$
$SWAP\text{-}down\ (Suc\ (Suc\ n)) * (|state\text{-}basis\ 1\ (j\ mod\ 2)\rangle \bigotimes |state\text{-}basis\ (Suc\ n)\ (j\ div\ 2)\rangle) =$
$|state\text{-}basis\ (Suc\ n)\ (j\ div\ 2)\rangle \bigotimes |state\text{-}basis\ 1\ (j\ mod\ 2)\rangle$
$\langle proof \rangle$

Action of the controlled-R gates in the circuit

lemma *controlR-action*:
assumes $j < 2 \hat{} Suc\ (Suc\ n)$
shows $(control\ (Suc\ (Suc\ n))\ (R\ (Suc\ (Suc\ n)))) *$
$((|zero\rangle + exp\ (2*i*pi*complex\text{-}of\text{-}nat\ (j\ div\ 2) / 2\hat{}(Suc\ n)) \cdot_m |one\rangle) \bigotimes$
$|state\text{-}basis\ n\ ((j\ mod\ 2\hat{}(Suc\ n))\ div\ 2)\rangle \bigotimes |state\text{-}basis\ 1\ (j\ mod\ 2)\rangle) =$
$(|zero\rangle + exp\ (2*i*pi*complex\text{-}of\text{-}nat\ j / 2\hat{}(Suc\ (Suc\ n))) \cdot_m |one\rangle) \bigotimes$
$|state\text{-}basis\ n\ ((j\ mod\ 2\hat{}(Suc\ n))\ div\ 2)\rangle \bigotimes |state\text{-}basis\ 1\ (j\ mod\ 2)\rangle$
$\langle proof \rangle$

Action of the controlled rotations subcircuit

lemma *controlled-rotations-ind*:
$\forall j.\ j < 2 \hat{} Suc\ n \longrightarrow$
$controlled\text{-}rotations\ (Suc\ n) *$
$((|zero\rangle + exp(2*i*pi*(complex\text{-}of\text{-}nat\ (j\ div\ 2\hat{}n))/2) \cdot_m |one\rangle) \bigotimes |state\text{-}basis\ n\ (j\ mod\ 2\hat{}n)\rangle) =$

($|zero\rangle$ + $exp(2{*}i{*}pi{*}j/(2\hat{\ }(Suc\ n)))\ \cdot_m\ |one\rangle$) $\bigotimes$ $|state\text{-}basis\ n\ (j\ mod\ 2\hat{\ }n)\rangle$
$\langle proof \rangle$

lemma *controlled-rotations-on-first-qubit*:
assumes $j < 2\ \hat{}\ Suc\ n$
shows $controlled\text{-}rotations\ (Suc\ n) *$
$(1/sqrt\ 2 \cdot_m (\ |zero\rangle + exp(2{*}i{*}pi{*}(complex\text{-}of\text{-}nat\ (j\ div\ 2\hat{\ }n))/2) \cdot_m |one\rangle)$
$\bigotimes$
$|state\text{-}basis\ n\ (j\ mod\ 2\hat{\ }n)\rangle) =$
$(1/sqrt\ 2 \cdot_m ((\ |zero\rangle + exp(2{*}i{*}pi{*}j/(2\hat{\ }(Suc\ n))) \cdot_m |one\rangle)) \bigotimes |state\text{-}basis\ n\ (j\ mod\ 2\hat{\ }n)\rangle)$
$\langle proof \rangle$

More useful lemmas:

lemma *exp-j*:
assumes $l < Suc\ n$
shows $exp\ (2{*}i{*}pi{*}j/(2\hat{\ }l)) = exp\ (2{*}i{*}pi{*}(j\ mod\ 2\hat{\ }n)/(2\hat{\ }l))$
$\langle proof \rangle$

lemma *kron-list-fun*[*simp*]:
$\forall x.\ List.member\ xs\ x \longrightarrow f\ x = g\ x \Longrightarrow kron\ f\ xs = kron\ g\ xs$
$\langle proof \rangle$

lemma *member-rev*:
shows $List.member\ (rev\ xs)\ x = List.member\ xs\ x$
$\langle proof \rangle$

lemma *kron-j*:
shows $kron\ (\lambda(l{::}nat).\ |zero\rangle + exp\ (2{*}i{*}pi{*}j/(2\hat{\ }l)) \cdot_m |one\rangle)\ (map\ nat\ (rev\ [1..n])) =$
$kron\ (\lambda(l{::}nat).\ |zero\rangle + exp\ (2{*}i{*}pi{*}(complex\text{-}of\text{-}nat\ (j\ mod\ 2\hat{\ }n))/(2\hat{\ }l)) \cdot_m |one\rangle)$
$(map\ nat\ (rev\ [1..n]))$
$\langle proof \rangle$

We proof that the QFT circuit is correct:

theorem *QFT-is-correct*:
shows $\forall j.\ j < 2\hat{\ }n \longrightarrow (QFT\ n) * |state\text{-}basis\ n\ j\rangle = reverse\text{-}QFT\text{-}product\text{-}representation\ j\ n$
$\langle proof \rangle$

7.1 QFT with qubits reordering correctness

lemma *SWAP-down-kron*:

assumes $\forall m.$ *dim-row* $(f\ m) = 2 \wedge$ *dim-col* $(f\ m) = 1$
shows *SWAP-down* $($*length* $(x\#xs)) *$ *kron f* $(x\#xs) =$ *kron f xs* $\bigotimes$ *f x*
⟨*proof*⟩

lemma *SWAP-down-kron-map-rev*:
assumes $\forall m.$ *dim-row* $(f\ m) = 2 \wedge$ *dim-col* $(f\ m) = 1$
shows $($*SWAP-down* $($*Suc k*$)) *$
kron f $($*map nat* $($*rev* $[1..$*int* $($*Suc k*$)])) =$
$($*kron f* $($*map nat* $($*rev* $[1..$*int k*$]))$ $\bigotimes$ $(f$ $($*Suc k*$)))$
⟨*proof*⟩

lemma *reverse-qubits-kron*:
assumes $\forall m.$ *dim-row* $(f\ m) = 2 \wedge$ *dim-col* $(f\ m) = 1$
shows $($*reverse-qubits n*$) * ($*kron f* $($*map nat* $($*rev* $[1..n]))) =$ *kron f* $($*map nat* $[1..n])$
⟨*proof*⟩

lemma *prod-rep-fun*:
assumes $f = (\lambda(l::nat).\ |zero\rangle +$ *exp* $(2*\mathrm{i}*pi*j/(2\hat{}l)) \cdot_m |one\rangle)$
shows $\forall m.$ *dim-row* $(f\ m) = 2 \wedge$ *dim-col* $(f\ m) = 1$
⟨*proof*⟩

lemma *rev-upto*:
assumes $n1 \leq n2$
shows *rev* $[n1..n2] = n2\ \#$ *rev* $[n1..(n2-1)]$
⟨*proof*⟩

lemma *dim-row-kron*:
shows *dim-row* $($*kron f xs*$) = (\prod x\leftarrow xs.$ *dim-row* $(f\ x))$
⟨*proof*⟩

lemma *dim-col-kron*:
shows *dim-col* $($*kron f xs*$) = (\prod x\leftarrow xs.$ *dim-col* $(f\ x))$
⟨*proof*⟩

lemma *prod-2-n*:
$(\prod x\leftarrow$ *map nat* $($*rev* $[1..$*int n*$]).\ 2) = 2\ \hat{}\ n$
⟨*proof*⟩

lemma *prod-2-n-b*:
$(\prod x\leftarrow$ *map nat* $[1..$*int n*$].\ 2) = 2\ \hat{}\ n$
⟨*proof*⟩

lemma *prod-1-n*:
$(\prod x\leftarrow$ *map nat* $($*rev* $[1..$*int n*$]).\ 1) = 1$
⟨*proof*⟩

lemma *prod-1-n-b*:
$(\prod x \leftarrow map\ nat\ [1..int\ n].\ Suc\ 0) = Suc\ 0$
⟨*proof*⟩

lemma *reverse-qubits-product-representation*:
reverse-qubits n ∗ *reverse-QFT-product-representation j n* = *QFT-product-representation j n*
⟨*proof*⟩

Finally, we proof the correctness of the algorithm

theorem *ordered-QFT-is-correct*:
assumes $j < 2\hat{}n$
shows (*ordered-QFT n*) ∗ |*state-basis n j*⟩ = *QFT-product-representation j n*
⟨*proof*⟩

8 Unitarity

Although unitarity is not required to proof QFT's correctness, in this section we will prove it, i.e., QFT and ordered_QFT functions create quantum gates and QFT product representation is a quantum state.

lemma *state-basis-is-state*:
assumes $j < n$
shows *state n* |*state-basis n j*⟩
⟨*proof*⟩

lemma *R-dagger-mat*:
shows $(R\ k)^\dagger = Matrix.mat\ 2\ 2\ (\lambda(i,j).\ if\ i{\neq}j\ then\ 0\ else\ (if\ i{=}0\ then\ 1\ else\ exp(-2{*}pi{*}\mathrm{i}/2\hat{}k)))$
⟨*proof*⟩

lemma *R-is-gate*:
shows *gate 1* (*R n*)
⟨*proof*⟩

lemma *SWAP-dagger-mat*:
shows $SWAP^\dagger = SWAP$
⟨*proof*⟩

lemma *SWAP-inv*:
shows $SWAP * (SWAP^\dagger) = 1_m\ 4$
⟨*proof*⟩

lemma *SWAP-inv'*:
shows $(SWAP^\dagger) * SWAP = 1_m\ 4$
⟨*proof*⟩

lemma *SWAP-is-gate*:

shows *gate 2 SWAP*
⟨*proof*⟩

lemma *control2-inv*:
assumes *gate 1 U*
shows $(control2\ U) * ((control2\ U)^{\dagger}) = 1_m\ 4$
⟨*proof*⟩

lemma *control2-inv′*:
assumes *gate 1 U*
shows $(control2\ U)^{\dagger} * (control2\ U) = 1_m\ 4$
⟨*proof*⟩

lemma *control2-is-gate*:
assumes *gate 1 U*
shows *gate 2* (*control2 U*)
⟨*proof*⟩

lemma *SWAP-down-is-gate*:
shows *gate n* (*SWAP-down n*)
⟨*proof*⟩

lemma *SWAP-up-is-gate*:
shows *gate n* (*SWAP-up n*)
⟨*proof*⟩

lemma *control-is-gate*:
assumes *gate 1 U*
shows *gate n* (*control n U*)
⟨*proof*⟩

lemma *controlled-rotations-is-gate*:
shows *gate n* (*controlled-rotations n*)
⟨*proof*⟩

theorem *QFT-is-gate*:
shows *gate n* (*QFT n*)
⟨*proof*⟩

corollary *QFT-is-unitary*:
shows *unitary* (*QFT n*)
⟨*proof*⟩

corollary *reverse-product-rep-is-state*:
assumes $j < 2\hat{}n$
shows *state n* (*reverse-QFT-product-representation j n*)
⟨*proof*⟩

lemma *reverse-qubits-is-gate*:
  shows *gate n* (*reverse-qubits n*)
⟨*proof*⟩

theorem *ordered-QFT-is-gate*:
  shows *gate n* (*ordered-QFT n*)
  ⟨*proof*⟩

corollary *ordered-QFT-is-unitary*:
  shows *unitary* (*ordered-QFT n*)
  ⟨*proof*⟩

corollary *product-rep-is-state*:
  assumes $j < 2\hat{}n$
  shows *state n* (*QFT-product-representation j n*)
  ⟨*proof*⟩

end

9 Acknowledgements

This work was conducted as part of my MSc Thesis [2] under the supervision of Prof. Francisco Jesús Martín Mateos, without whose advise and assistance its completion would not have been possible.

References

[1] A. Bordg, H. Lachnitt, and Y. He. Isabelle Marries Dirac: a Library for Quantum Computation and Quantum Information. *Archive of Formal Proofs*, November 2020. https://isa-afp.org/entries/Isabelle_Marries_Dirac.html, Formal proof development.

[2] P. Manrique. Computación Cuántica: formalización y demostración en Isabelle. Master's thesis, Universidad de Sevilla, 2024.

[3] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, 2010.

[4] Y. Peng, K. Hietala, R. Tao, L. Li, R. Rand, M. Hicks, and X. Wu. A Formally Certified End-to-End Implementation of Shor's Factorization Algorithm, 2022.