

Isabelle Marries Dirac: a Library for Quantum Computation and Quantum Information

Anthony Bordg, Hanna Lachnitt and Yijun He

February 6, 2026

Contents

1	Basic Results	2
1.1	Basic Set-Theoretic Results	3
1.2	Basic Arithmetic Results	3
1.3	Basic Results on Matrices	4
1.4	Basic Results on Sums	5
1.5	Basic Results Involving the Exponential Function.	5
1.6	Basic Results with Trigonometric Functions.	6
1.6.1	Basic Inequalities	6
1.6.2	Basic Equalities	6
2	Binary Representation of Natural Numbers	6
3	Qubits and Quantum Gates	8
3.1	Qubits	8
3.2	The Hermitian Conjugation	9
3.3	Unitary Matrices and Quantum Gates	10
3.4	Relations Between Complex Conjugation, Hermitian Conjugation, Transposition and Unitarity	11
3.5	The Inner Product	13
3.6	Unitary Matrices and Length-Preservation	15
3.6.1	Unitary Matrices are Length-Preserving	15
3.6.2	Length-Preserving Matrices are Unitary	17
3.7	A Few Well-known Quantum Gates	19
3.8	The Bell States	22
3.9	The Bitwise Inner Product	23
4	Complex Vectors	23
4.1	The Vector Space of Complex Vectors of Dimension n	23

5	Tensor Products	25
5.1	The Kronecker Product of Complex Vectors	25
5.2	The Tensor Product of Complex Matrices	26
6	Further Results on Tensor Products	29
7	Measurement	32
7.1	Measurements with Bell States	36
8	Quantum Entanglement	38
8.1	The Product States and Entangled States of a 2-qubits System	39
9	Quantum Teleportation	40
10	The Deutsch Algorithm	45
11	The Deutsch-Jozsa Algorithm	51
12	The No-Cloning Theorem	63
12.1	The Cauchy-Schwarz Inequality	64
12.2	The No-Cloning Theorem	64
13	Quantum Prisoner's Dilemma	65
13.1	The Set-Up	65
13.2	The Separable Case	71
13.3	The Maximally Entangled Case	72
13.4	The Unfair Strategy Case	74
14	Acknowledgements	75

Abstract

This work is an effort to formalise some quantum algorithms and results in quantum information theory. Formal methods being critical for the safety and security of algorithms and protocols, we foresee their widespread use for quantum computing in the future. We have developed a large library for quantum computing in Isabelle based on a matrix representation for quantum circuits, successfully formalising the no-cloning theorem, quantum teleportation, Deutsch's algorithm, the Deutsch-Jozsa algorithm and the quantum Prisoner's Dilemma.

1 Basic Results

theory *Basics*

imports

HOL.Set-Interval

HOL.Semiring-Normalization

HOL.Real-Vector-Spaces

HOL.Power
HOL.Complex
Jordan-Normal-Form.Jordan-Normal-Form
begin

1.1 Basic Set-Theoretic Results

lemma *set-2-atLeast0* [simp]: $\{0..<2::nat\} = \{0,1\}$ *<proof>*

lemma *set-2*: $\{..<2::nat\} = \{0,1\}$ *<proof>*

lemma *set-4-atLeast0* [simp]: $\{0..<4::nat\} = \{0,1,2,3\}$ *<proof>*

lemma *set-4*: $\{..<4::nat\} = \{0,1,2,3\}$ *<proof>*

lemma *set-4-disj* [simp]:
fixes $i::nat$
assumes $i < 4$
shows $i = 0 \vee i = 1 \vee i = 2 \vee i = 3$
<proof>

lemma *set-8-atLeast0* [simp]: $\{0..<8::nat\} = \{0,1,2,3,4,5,6,7\}$ *<proof>*

lemma *index-is-2* [simp]: $\forall i::nat. i \neq \text{Suc } 0 \longrightarrow i \neq 3 \longrightarrow 0 < i \longrightarrow i < 4 \longrightarrow i = 2$ *<proof>*

lemma *index-sl-four* [simp]: $\forall i::nat. i < 4 \longrightarrow i = 0 \vee i = 1 \vee i = 2 \vee i = 3$ *<proof>*

1.2 Basic Arithmetic Results

lemma *index-div-eq* [simp]:
fixes $i::nat$
shows $i \in \{a*b..<(a+1)*b\} \implies i \text{ div } b = a$
<proof>

lemma *index-mod-eq* [simp]:
fixes $i::nat$
shows $i \in \{a*b..<(a+1)*b\} \implies i \text{ mod } b = i - a*b$
<proof>

lemma *sqr-of-cmod-of-prod*:
shows $(\text{cmod } (z1 * z2))^2 = (\text{cmod } z1)^2 * (\text{cmod } z2)^2$
<proof>

lemma *less-power-add-imp-div-less* [simp]:
fixes $i m n::nat$
assumes $i < 2^{\wedge}(m+n)$
shows $i \text{ div } 2^{\wedge}n < 2^{\wedge}m$
<proof>

lemma *div-mult-mod-eq-minus*:

fixes $i\ j::\text{nat}$

shows $(i \text{ div } 2^{\hat{n}}) * 2^{\hat{n}} + i \text{ mod } 2^{\hat{n}} - (j \text{ div } 2^{\hat{n}}) * 2^{\hat{n}} - j \text{ mod } 2^{\hat{n}} = i - j$
<proof>

lemma *neq-imp-neq-div-or-mod*:

fixes $i\ j::\text{nat}$

assumes $i \neq j$

shows $i \text{ div } 2^{\hat{n}} \neq j \text{ div } 2^{\hat{n}} \vee i \text{ mod } 2^{\hat{n}} \neq j \text{ mod } 2^{\hat{n}}$
<proof>

lemma *index-one-mat-div-mod*:

assumes $i < 2^{\hat{m}+\hat{n}}$ **and** $j < 2^{\hat{m}+\hat{n}}$

shows $((1_m(2^{\hat{m}}) \text{ $$ } (i \text{ div } 2^{\hat{n}}, j \text{ div } 2^{\hat{n}})::\text{complex}) * 1_m(2^{\hat{n}}) \text{ $$ } (i \text{ mod } 2^{\hat{n}}, j \text{ mod } 2^{\hat{n}})) = 1_m(2^{\hat{m}+\hat{n}}) \text{ $$ } (i, j)$
<proof>

lemma *sqr-of-sqrt-2 [simp]*:

fixes $z::\text{complex}$

shows $z * 2 / (\text{complex-of-real } (\text{sqrt } 2) * \text{complex-of-real } (\text{sqrt } 2)) = z$
<proof>

lemma *two-div-sqrt-two [simp]*:

shows $2 * \text{complex-of-real } (\text{sqrt } (1/2)) = \text{complex-of-real } (\text{sqrt } 2)$
<proof>

lemma *two-div-sqr-of-cmd-sqrt-two [simp]*:

shows $2 * (\text{cmod } (1 / \text{complex-of-real } (\text{sqrt } 2)))^2 = 1$
<proof>

lemma *two-div-two [simp]*:

shows $2 \text{ div Suc } (\text{Suc } 0) = 1$ *<proof>*

lemma *two-mod-two [simp]*:

shows $2 \text{ mod Suc } (\text{Suc } 0) = 0$ *<proof>*

lemma *three-div-two [simp]*:

shows $3 \text{ div Suc } (\text{Suc } 0) = 1$ *<proof>*

lemma *three-mod-two [simp]*:

shows $3 \text{ mod Suc } (\text{Suc } 0) = 1$ *<proof>*

1.3 Basic Results on Matrices

lemma *index-matrix-prod [simp]*:

assumes $i < \text{dim-row } A$ **and** $j < \text{dim-col } B$ **and** $\text{dim-col } A = \text{dim-row } B$

shows $(A * B) \text{ $$ } (i, j) = (\sum_{k < \text{dim-row } B} (A \text{ $$ } (i, k)) * (B \text{ $$ } (k, j)))$
<proof>

1.4 Basic Results on Sums

lemma *sum-insert* [simp]:
 assumes $x \notin F$ **and** *finite F*
 shows $(\sum y \in \text{insert } x \ F. P \ y) = (\sum y \in F. P \ y) + P \ x$
 $\langle \text{proof} \rangle$

lemma *sum-of-index-diff* [simp]:
 fixes $f :: \text{nat} \Rightarrow 'a :: \text{comm-monoid-add}$
 shows $(\sum i \in \{a..<a+b\}. f(i-a)) = (\sum i \in \{..<b\}. f(i))$
 $\langle \text{proof} \rangle$

1.5 Basic Results Involving the Exponential Function.

lemma *exp-of-real-cnj*:
 fixes $x :: \text{real}$
 shows $\text{cnj} (\exp (i * x)) = \exp (-i * x)$
 $\langle \text{proof} \rangle$

lemma *exp-of-real-cnj2*:
 fixes $x :: \text{real}$
 shows $\text{cnj} (\exp (-i * x)) = \exp (i * x)$
 $\langle \text{proof} \rangle$

lemma *exp-of-half-pi*:
 fixes $x :: \text{real}$
 assumes $x = \pi/2$
 shows $\exp (i * \text{complex-of-real } x) = i$
 $\langle \text{proof} \rangle$

lemma *exp-of-minus-half-pi*:
 fixes $x :: \text{real}$
 assumes $x = \pi/2$
 shows $\exp (-i * \text{complex-of-real } x) = -i$
 $\langle \text{proof} \rangle$

lemma *exp-of-real*:
 fixes $x :: \text{real}$
 shows $\exp (i * x) = \cos x + i * (\sin x)$
 $\langle \text{proof} \rangle$

lemma *exp-of-real-inv*:
 fixes $x :: \text{real}$
 shows $\exp (-i * x) = \cos x - i * (\sin x)$
 $\langle \text{proof} \rangle$

1.6 Basic Results with Trigonometric Functions.

1.6.1 Basic Inequalities

lemma *sin-squared-le-one*:

fixes $x:: \text{real}$

shows $(\sin x)^2 \leq 1$

<proof>

lemma *cos-squared-le-one*:

fixes $x:: \text{real}$

shows $(\cos x)^2 \leq 1$

<proof>

1.6.2 Basic Equalities

lemma *sin-of-quarter-pi*:

fixes $x:: \text{real}$

assumes $x = \pi/2$

shows $\sin (x/2) = (\text{sqrt } 2)/2$

<proof>

lemma *cos-of-quarter-pi*:

fixes $x:: \text{real}$

assumes $x = \pi/2$

shows $\cos (x/2) = (\text{sqrt } 2)/2$

<proof>

end

2 Binary Representation of Natural Numbers

theory *Binary-Nat*

imports

HOL.Nat

HOL.List

Basics

begin

primrec *bin-rep-aux*:: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat list}$ **where**

bin-rep-aux 0 $m = [m]$

| *bin-rep-aux* (Suc n) $m = m \text{ div } 2^n \# \text{bin-rep-aux } n (m \text{ mod } 2^n)$

lemma *length-of-bin-rep-aux*:

fixes $n m:: \text{nat}$

assumes $m < 2^n$

shows $\text{length} (\text{bin-rep-aux } n m) = n+1$

<proof>

lemma *bin-rep-aux-neq-nil*:
fixes $n m :: \text{nat}$
shows $\text{bin-rep-aux } n \ m \neq []$
 $\langle \text{proof} \rangle$

lemma *last-of-bin-rep-aux*:
fixes $n m :: \text{nat}$
assumes $m < 2^n$ **and** $m \geq 0$
shows $\text{last } (\text{bin-rep-aux } n \ m) = 0$
 $\langle \text{proof} \rangle$

lemma *mod-mod-power-cancel*:
fixes $m \ n \ p :: \text{nat}$
assumes $m \leq n$
shows $p \bmod 2^n \bmod 2^m = p \bmod 2^m$
 $\langle \text{proof} \rangle$

lemma *bin-rep-aux-index*:
fixes $n \ m \ i :: \text{nat}$
assumes $n \geq 1$ **and** $m < 2^n$ **and** $m \geq 0$ **and** $i \leq n$
shows $\text{bin-rep-aux } n \ m ! i = (m \bmod 2^{(n-i)}) \text{ div } 2^{(n-1-i)}$
 $\langle \text{proof} \rangle$

lemma *bin-rep-aux-coeff*:
fixes $n \ m \ i :: \text{nat}$
assumes $m < 2^n$ **and** $i \leq n$ **and** $m \geq 0$
shows $\text{bin-rep-aux } n \ m ! i = 0 \vee \text{bin-rep-aux } n \ m ! i = 1$
 $\langle \text{proof} \rangle$

definition *bin-rep*: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat list}$ **where**
 $\text{bin-rep } n \ m = \text{butlast } (\text{bin-rep-aux } n \ m)$

lemma *length-of-bin-rep*:
fixes $n \ m :: \text{nat}$
assumes $m < 2^n$
shows $\text{length } (\text{bin-rep } n \ m) = n$
 $\langle \text{proof} \rangle$

lemma *bin-rep-coeff*:
fixes $n \ m \ i :: \text{nat}$
assumes $m < 2^n$ **and** $i < n$ **and** $m \geq 0$
shows $\text{bin-rep } n \ m ! i = 0 \vee \text{bin-rep } n \ m ! i = 1$
 $\langle \text{proof} \rangle$

lemma *bin-rep-index*:
fixes $n \ m \ i :: \text{nat}$
assumes $n \geq 1$ **and** $m < 2^n$ **and** $i < n$ **and** $m \geq 0$
shows $\text{bin-rep } n \ m ! i = (m \bmod 2^{(n-i)}) \text{ div } 2^{(n-1-i)}$
 $\langle \text{proof} \rangle$

lemma *bin-rep-eq*:
fixes $n m :: \text{nat}$
assumes $n \geq 1$ **and** $m \geq 0$ **and** $m < 2^n$ **and** $m \geq 0$
shows $m = (\sum_{i < n}. \text{bin-rep } n \ m \ ! \ i * 2^{(n-1-i)})$
 $\langle \text{proof} \rangle$

lemma *bin-rep-index-0*:
fixes $n m :: \text{nat}$
assumes $m < 2^n$ **and** $k > n$
shows $(\text{bin-rep } k \ m) \ ! \ 0 = 0$
 $\langle \text{proof} \rangle$

lemma *bin-rep-index-0-geq*:
fixes $n m :: \text{nat}$
assumes $m \geq 2^n$ **and** $m < 2^{(n+1)}$
shows $\text{bin-rep } (n+1) \ m \ ! \ 0 = 1$
 $\langle \text{proof} \rangle$

end

3 Qubits and Quantum Gates

theory *Quantum*
imports
Jordan-Normal-Form.Matrix
HOL-Library.Nonpos-Ints
Basics
Binary-Nat
begin

3.1 Qubits

In this theory *cpx* stands for *complex*.

definition *cpx-vec-length* :: *complex vec* \Rightarrow *real* ($\langle \| \cdot \| \rangle$) **where**
cpx-vec-length $v \equiv \text{sqrt}(\sum_{i < \text{dim-vec } v}. (\text{cmod } (v \ \$ \ i))^2)$

lemma *cpx-length-of-vec-of-list* [*simp*]:
 $\| \text{vec-of-list } l \| = \text{sqrt}(\sum_{i < \text{length } l}. (\text{cmod } (l \ ! \ i))^2)$
 $\langle \text{proof} \rangle$

lemma *norm-vec-index-unit-vec-is-0* [*simp*]:
assumes $j < n$ **and** $j \neq i$
shows $\text{cmod } ((\text{unit-vec } n \ i) \ \$ \ j) = 0$
 $\langle \text{proof} \rangle$

lemma *norm-vec-index-unit-vec-is-1* [*simp*]:
assumes $j < n$ **and** $j = i$

shows $\text{cmod } ((\text{unit-vec } n \ i) \ \$ \ j) = 1$
 $\langle \text{proof} \rangle$

lemma *unit-cpx-vec-length* [simp]:
assumes $i < n$
shows $\| \text{unit-vec } n \ i \| = 1$
 $\langle \text{proof} \rangle$

lemma *smult-vec-length* [simp]:
assumes $x \geq 0$
shows $\| \text{complex-of-real}(x) \cdot_v \ v \| = x * \|v\|$
 $\langle \text{proof} \rangle$

locale *state* =
fixes $n:: \text{nat}$ **and** $v:: \text{complex mat}$
assumes *is-column* [simp]: $\text{dim-col } v = 1$
and *dim-row* [simp]: $\text{dim-row } v = 2^{\widehat{n}}$
and *is-normal* [simp]: $\| \text{col } v \ 0 \| = 1$

Below the natural number n codes for the dimension of the complex vector space whose elements of norm 1 we call states.

lemma *unit-vec-of-right-length-is-state* [simp]:
assumes $i < 2^{\widehat{n}}$
shows $\text{unit-vec } (2^{\widehat{n}}) \ i \in \{v \mid v:: \text{complex vec. dim-vec } v = 2^{\widehat{n}} \wedge \|v\| = 1\}$
 $\langle \text{proof} \rangle$

definition *state-qbit* :: $\text{nat} \Rightarrow \text{complex vec set}$ **where**
 $\text{state-qbit } n \equiv \{v \mid v:: \text{complex vec. dim-vec } v = 2^{\widehat{n}} \wedge \|v\| = 1\}$

lemma (*in state*) *state-to-state-qbit* [simp]:
shows $\text{col } v \ 0 \in \text{state-qbit } n$
 $\langle \text{proof} \rangle$

3.2 The Hermitian Conjugation

The Hermitian conjugate of a complex matrix is the complex conjugate of its transpose.

definition *dagger* :: $\text{complex mat} \Rightarrow \text{complex mat}$ ($\langle \cdot^\dagger \rangle$) **where**
 $M^\dagger \equiv \text{mat } (\text{dim-col } M) (\text{dim-row } M) (\lambda(i,j). \text{cnj}(M \ \$\$ (j,i)))$

We introduce the type of complex square matrices.

typedef *cpx-sqr-mat* = $\{M \mid M:: \text{complex mat. square-mat } M\}$
 $\langle \text{proof} \rangle$

definition *cpx-sqr-mat-to-cpx-mat* :: $\text{cpx-sqr-mat} \Rightarrow \text{complex mat}$ **where**
 $\text{cpx-sqr-mat-to-cpx-mat } M \equiv \text{Rep-cpx-sqr-mat } M$

We introduce a coercion from the type of complex square matrices to the type of complex matrices.

declare $[[\text{coercion } \text{cpx-sqr-mat-to-cpx-mat}]]$

lemma *dim-row-of-dagger* [simp]:
 $\text{dim-row } (M^\dagger) = \text{dim-col } M$
 ⟨proof⟩

lemma *dim-col-of-dagger* [simp]:
 $\text{dim-col } (M^\dagger) = \text{dim-row } M$
 ⟨proof⟩

lemma *col-of-dagger* [simp]:
assumes $j < \text{dim-row } M$
shows $\text{col } (M^\dagger) j = \text{vec } (\text{dim-col } M) (\lambda i. \text{cnj } (M \$\$ (j, i)))$
 ⟨proof⟩

lemma *row-of-dagger* [simp]:
assumes $i < \text{dim-col } M$
shows $\text{row } (M^\dagger) i = \text{vec } (\text{dim-row } M) (\lambda j. \text{cnj } (M \$\$ (j, i)))$
 ⟨proof⟩

lemma *dagger-of-dagger-is-id*:
fixes $M :: \text{complex Matrix.mat}$
shows $(M^\dagger)^\dagger = M$
 ⟨proof⟩

lemma *dagger-of-sqr-is-sqr* [simp]:
 $\text{square-mat } ((M :: \text{cpx-sqr-mat})^\dagger)$
 ⟨proof⟩

lemma *dagger-of-id-is-id* [simp]:
 $(1_m \ n)^\dagger = 1_m \ n$
 ⟨proof⟩

3.3 Unitary Matrices and Quantum Gates

definition *unitary* $:: \text{complex mat} \Rightarrow \text{bool}$ **where**
 $\text{unitary } M \equiv (M^\dagger) * M = 1_m (\text{dim-col } M) \wedge M * (M^\dagger) = 1_m (\text{dim-row } M)$

lemma *id-is-unitary* [simp]:
 $\text{unitary } (1_m \ n)$
 ⟨proof⟩

locale *gate* =
fixes $n :: \text{nat}$ **and** $A :: \text{complex mat}$
assumes *dim-row* [simp]: $\text{dim-row } A = 2 \hat{n}$
and *square-mat* [simp]: $\text{square-mat } A$
and *unitary* [simp]: $\text{unitary } A$

We prove that a quantum gate is invertible and its inverse is given by its Hermitian conjugate.

lemma *mat-unitary-mat* [*intro*]:
assumes *unitary* M
shows *inverts-mat* M (M^\dagger)
 \langle *proof* \rangle

lemma *unitary-mat-mat* [*intro*]:
assumes *unitary* M
shows *inverts-mat* (M^\dagger) M
 \langle *proof* \rangle

lemma (*in gate*) *gate-is-inv*:
invertible-mat A
 \langle *proof* \rangle

3.4 Relations Between Complex Conjugation, Hermitian Conjugation, Transposition and Unitarity

notation *transpose-mat* $\langle(-^t)\rangle$

lemma *col-tranpose* [*simp*]:
assumes *dim-row* $M = n$ **and** $i < n$
shows *col* (M^t) $i =$ *row* M i
 \langle *proof* \rangle

lemma *row-transpose* [*simp*]:
assumes *dim-col* $M = n$ **and** $i < n$
shows *row* (M^t) $i =$ *col* M i
 \langle *proof* \rangle

definition *cpx-mat-cnj* :: *complex mat* \Rightarrow *complex mat* $\langle(-^*)\rangle$ **where**
cpx-mat-cnj $M \equiv$ *mat* (*dim-row* M) (*dim-col* M) $(\lambda(i,j). \text{cnj } (M \text{ $$$ } (i,j)))$

lemma *cpx-mat-cnj-id* [*simp*]:
 $(1_m \ n)^* = 1_m \ n$
 \langle *proof* \rangle

lemma *cpx-mat-cnj-cnj* [*simp*]:
 $(M^*)^* = M$
 \langle *proof* \rangle

lemma *dim-row-of-cjn-prod* [*simp*]:
dim-row $((M^*) * (N^*)) =$ *dim-row* M
 \langle *proof* \rangle

lemma *dim-col-of-cjn-prod* [*simp*]:
dim-col $((M^*) * (N^*)) =$ *dim-col* N
 \langle *proof* \rangle

lemma *cpx-mat-cnj-prod*:

assumes $\dim\text{-col } M = \dim\text{-row } N$
shows $(M * N)^* = (M^*) * (N^*)$
 $\langle\text{proof}\rangle$

lemma *transpose-of-prod*:
fixes $M N::\text{complex Matrix.mat}$
assumes $\dim\text{-col } M = \dim\text{-row } N$
shows $(M * N)^t = N^t * (M^t)$
 $\langle\text{proof}\rangle$

lemma *transpose-cnj-is-dagger* [simp]:
 $(M^t)^* = (M^\dagger)$
 $\langle\text{proof}\rangle$

lemma *cnj-transpose-is-dagger* [simp]:
 $(M^*)^t = (M^\dagger)$
 $\langle\text{proof}\rangle$

lemma *dagger-of-transpose-is-cnj* [simp]:
 $(M^t)^\dagger = (M^*)$
 $\langle\text{proof}\rangle$

lemma *dagger-of-prod*:
fixes $M N::\text{complex Matrix.mat}$
assumes $\dim\text{-col } M = \dim\text{-row } N$
shows $(M * N)^\dagger = N^\dagger * (M^\dagger)$
 $\langle\text{proof}\rangle$

The product of two quantum gates is a quantum gate.

lemma *prod-of-gate-is-gate*:
assumes *gate* n $G1$ **and** *gate* n $G2$
shows *gate* n $(G1 * G2)$
 $\langle\text{proof}\rangle$

lemma *left-inv-of-unitary-transpose* [simp]:
assumes *unitary* U
shows $(U^t)^\dagger * (U^t) = 1_m(\dim\text{-row } U)$
 $\langle\text{proof}\rangle$

lemma *right-inv-of-unitary-transpose* [simp]:
assumes *unitary* U
shows $U^t * ((U^t)^\dagger) = 1_m(\dim\text{-col } U)$
 $\langle\text{proof}\rangle$

lemma *transpose-of-unitary-is-unitary* [simp]:
assumes *unitary* U
shows *unitary* (U^t)
 $\langle\text{proof}\rangle$

3.5 The Inner Product

We introduce a coercion between complex vectors and (column) complex matrices.

definition *ket-vec* :: *complex vec* \Rightarrow *complex mat* ($\langle |-\rangle$) **where**
 $|v\rangle \equiv \text{mat } 1 \ (\text{dim-vec } v) \ (\lambda(i,j). v \$ i)$

lemma *ket-vec-index* [*simp*]:
assumes $i < \text{dim-vec } v$
shows $|v\rangle \$\$ (i,0) = v \$ i$
 $\langle \text{proof} \rangle$

lemma *ket-vec-col* [*simp*]:
 $\text{col } |v\rangle \ 0 = v$
 $\langle \text{proof} \rangle$

lemma *smult-ket-vec* [*simp*]:
 $|x \cdot_v v\rangle = x \cdot_m |v\rangle$
 $\langle \text{proof} \rangle$

lemma *smult-vec-length-bis* [*simp*]:
assumes $x \geq 0$
shows $\|\text{col } (\text{complex-of-real}(x) \cdot_m |v\rangle) \ 0\| = x * \|v\|$
 $\langle \text{proof} \rangle$

declare $[[\text{coercion } \text{ket-vec}]]$

definition *row-vec* :: *complex vec* \Rightarrow *complex mat* **where**
 $\text{row-vec } v \equiv \text{mat } 1 \ (\text{dim-vec } v) \ (\lambda(i,j). v \$ j)$

definition *bra-vec* :: *complex vec* \Rightarrow *complex mat* **where**
 $\text{bra-vec } v \equiv (\text{row-vec } v)^*$

lemma *row-bra-vec* [*simp*]:
 $\text{row } (\text{bra-vec } v) \ 0 = \text{vec } (\text{dim-vec } v) \ (\lambda i. \text{conj}(v \$ i))$
 $\langle \text{proof} \rangle$

We introduce a definition called *bra* to see a vector as a column matrix.

definition *bra* :: *complex mat* \Rightarrow *complex mat* ($\langle \langle |-\rangle$) **where**
 $\langle v| \equiv \text{mat } 1 \ (\text{dim-row } v) \ (\lambda(i,j). \text{conj}(v \$\$ (j,i)))$

The relation between *bra*, *bra-vec* and *ket-vec* is given as follows.

lemma *bra-bra-vec* [*simp*]:
 $\text{bra } (\text{ket-vec } v) = \text{bra-vec } v$
 $\langle \text{proof} \rangle$

lemma *row-bra* [*simp*]:
fixes $v :: \text{complex vec}$
shows $\text{row } \langle v| \ 0 = \text{vec } (\text{dim-vec } v) \ (\lambda i. \text{conj } (v \$ i))$ $\langle \text{proof} \rangle$

We introduce the inner product of two complex vectors in \mathbf{C}^n .

definition *inner-prod* :: *complex vec* \Rightarrow *complex vec* \Rightarrow *complex* ($\langle \langle - | - \rangle \rangle$) **where**
inner-prod $u\ v \equiv \sum_{i \in \{0..< \dim\text{-vec } v\}} \text{cnj}(u\ \$\ i) * (v\ \$\ i)$

lemma *inner-prod-with-row-bra-vec* [*simp*]:

assumes $\dim\text{-vec } u = \dim\text{-vec } v$
shows $\langle u | v \rangle = \text{row } (\text{bra-vec } u)\ 0 \cdot v$
 $\langle \text{proof} \rangle$

lemma *inner-prod-with-row-bra-vec-col-ket-vec* [*simp*]:

assumes $\dim\text{-vec } u = \dim\text{-vec } v$
shows $\langle u | v \rangle = (\text{row } \langle u | 0 \rangle) \cdot (\text{col } |v\rangle\ 0)$
 $\langle \text{proof} \rangle$

lemma *inner-prod-with-times-mat* [*simp*]:

assumes $\dim\text{-vec } u = \dim\text{-vec } v$
shows $\langle u | v \rangle = (\langle u | * |v \rangle)\ \$\$ (0,0)$
 $\langle \text{proof} \rangle$

lemma *orthogonal-unit-vec* [*simp*]:

assumes $i < n$ **and** $j < n$ **and** $i \neq j$
shows $\langle \text{unit-vec } n\ i | \text{unit-vec } n\ j \rangle = 0$
 $\langle \text{proof} \rangle$

We prove that our inner product is linear in its second argument.

lemma *vec-index-is-linear* [*simp*]:

assumes $\dim\text{-vec } u = \dim\text{-vec } v$ **and** $j < \dim\text{-vec } u$
shows $(k \cdot_v u + l \cdot_v v)\ \$\ j = k * (u\ \$\ j) + l * (v\ \$\ j)$
 $\langle \text{proof} \rangle$

lemma *inner-prod-is-linear* [*simp*]:

fixes $u::\text{complex vec}$ **and** $v::\text{nat} \Rightarrow \text{complex vec}$ **and** $l::\text{nat} \Rightarrow \text{complex}$
assumes $\forall i \in \{0, 1\}. \dim\text{-vec } u = \dim\text{-vec } (v\ i)$
shows $\langle u | l\ 0 \cdot_v v\ 0 + l\ 1 \cdot_v v\ 1 \rangle = (\sum_{i \leq 1}. l\ i * \langle u | v\ i \rangle)$
 $\langle \text{proof} \rangle$

lemma *inner-prod-cnj*:

assumes $\dim\text{-vec } u = \dim\text{-vec } v$
shows $\langle v | u \rangle = \text{cnj } (\langle u | v \rangle)$
 $\langle \text{proof} \rangle$

lemma *inner-prod-with-itself-Im* [*simp*]:

$\text{Im } (\langle u | u \rangle) = 0$
 $\langle \text{proof} \rangle$

lemma *inner-prod-with-itself-real* [*simp*]:

$\langle u | u \rangle \in \mathbf{R}$
 $\langle \text{proof} \rangle$

lemma *inner-prod-with-itself-eq0* [simp]:
assumes $u = 0_v$ (*dim-vec* u)
shows $\langle u|u \rangle = 0$
<proof>

lemma *inner-prod-with-itself-Re*:
Re $(\langle u|u \rangle) \geq 0$
<proof>

lemma *inner-prod-with-itself-nonneg-reals*:
fixes $u::\text{complex vec}$
shows $\langle u|u \rangle \in \text{nonneg-Reals}$
<proof>

lemma *inner-prod-with-itself-Re-non0*:
assumes $u \neq 0_v$ (*dim-vec* u)
shows $\text{Re} (\langle u|u \rangle) > 0$
<proof>

lemma *inner-prod-with-itself-nonneg-reals-non0*:
assumes $u \neq 0_v$ (*dim-vec* u)
shows $\langle u|u \rangle \neq 0$
<proof>

lemma *cpx-vec-length-inner-prod* [simp]:
 $\|v\|^2 = \langle v|v \rangle$
<proof>

lemma *inner-prod-csqrt* [simp]:
 $\text{csqrt} \langle v|v \rangle = \|v\|$
<proof>

3.6 Unitary Matrices and Length-Preservation

3.6.1 Unitary Matrices are Length-Preserving

The bra-vector $\langle A * v|$ is given by $\langle v| * A^\dagger$

lemma *dagger-of-ket-is-bra*:
fixes $v::\text{complex vec}$
shows $(|v \rangle)^\dagger = \langle v|$
<proof>

lemma *bra-mat-on-vec*:
fixes $v::\text{complex vec}$ **and** $A::\text{complex mat}$
assumes $\text{dim-col } A = \text{dim-vec } v$
shows $\langle A * v| = \langle v| * (A^\dagger)$
<proof>

lemma *mat-on-ket*:

fixes $v::\text{complex vec}$ **and** $A::\text{complex mat}$
assumes $\text{dim-col } A = \text{dim-vec } v$
shows $A * |v\rangle = |\text{col } (A * v) \ 0\rangle$
 $\langle\text{proof}\rangle$

lemma *dagger-of-mat-on-ket*:
fixes $v::\text{complex vec}$ **and** $A::\text{complex mat}$
assumes $\text{dim-col } A = \text{dim-vec } v$
shows $(A * |v\rangle)^\dagger = \langle v | * (A^\dagger)$
 $\langle\text{proof}\rangle$

definition *col-fst* :: 'a mat \Rightarrow 'a vec **where**
 $\text{col-fst } A = \text{vec } (\text{dim-row } A) (\lambda i. A \ \$\$ (i,0))$

lemma *col-fst-is-col* [*simp*]:
 $\text{col-fst } M = \text{col } M \ 0$
 $\langle\text{proof}\rangle$

We need to declare *col-fst* as a coercion from matrices to vectors in order to see a column matrix as a vector.

declare
 $[[\text{coercion-delete ket-vec}]]$
 $[[\text{coercion col-fst}]]$

lemma *unit-vec-to-col*:
assumes $\text{dim-col } A = n$ **and** $i < n$
shows $\text{col } A \ i = A * |\text{unit-vec } n \ i\rangle$
 $\langle\text{proof}\rangle$

lemma *mult-ket-vec-is-ket-vec-of-mult*:
fixes $A::\text{complex mat}$ **and** $v::\text{complex vec}$
assumes $\text{dim-col } A = \text{dim-vec } v$
shows $|A * |v\rangle\rangle = A * |v\rangle$
 $\langle\text{proof}\rangle$

lemma *unitary-is-sq-length-preserving* [*simp*]:
assumes *unitary* U **and** $\text{dim-vec } v = \text{dim-col } U$
shows $\|U * |v\rangle\|^2 = \|v\|^2$
 $\langle\text{proof}\rangle$

lemma *col-ket-vec* [*simp*]:
assumes $\text{dim-col } M = 1$
shows $|\text{col } M \ 0\rangle = M$
 $\langle\text{proof}\rangle$

lemma *state-col-ket-vec*:
assumes *state 1* v
shows *state 1* $|\text{col } v \ 0\rangle$
 $\langle\text{proof}\rangle$

lemma *col-ket-vec-index* [simp]:
assumes $i < \text{dim-row } v$
shows $|\text{col } v \ 0\rangle \ \$\$ (i,0) = v \ \$\$ (i,0)$
 $\langle \text{proof} \rangle$

lemma *col-index-of-mat-col* [simp]:
assumes $\text{dim-col } v = 1$ **and** $i < \text{dim-row } v$
shows $\text{col } v \ 0 \ \$ i = v \ \$\$ (i,0)$
 $\langle \text{proof} \rangle$

lemma *unitary-is-sq-length-preserving-bis* [simp]:
assumes *unitary* U **and** $\text{dim-row } v = \text{dim-col } U$ **and** $\text{dim-col } v = 1$
shows $\|\text{col } (U * v) \ 0\|^2 = \|\text{col } v \ 0\|^2$
 $\langle \text{proof} \rangle$

A unitary matrix is length-preserving, i.e. it acts on a vector to produce another vector of the same length.

lemma *unitary-is-length-preserving-bis* [simp]:
fixes $U::\text{complex mat}$ **and** $v::\text{complex mat}$
assumes *unitary* U **and** $\text{dim-row } v = \text{dim-col } U$ **and** $\text{dim-col } v = 1$
shows $\|\text{col } (U * v) \ 0\| = \|\text{col } v \ 0\|$
 $\langle \text{proof} \rangle$

lemma *unitary-is-length-preserving* [simp]:
fixes $U::\text{complex mat}$ **and** $v::\text{complex vec}$
assumes *unitary* U **and** $\text{dim-vec } v = \text{dim-col } U$
shows $\|U * |v\rangle\| = \|v\|$
 $\langle \text{proof} \rangle$

3.6.2 Length-Preserving Matrices are Unitary

lemma *inverts-mat-sym*:
fixes $A \ B::\text{complex mat}$
assumes *inverts-mat* $A \ B$ **and** $\text{dim-row } B = \text{dim-col } A$ **and** *square-mat* B
shows *inverts-mat* $B \ A$
 $\langle \text{proof} \rangle$

lemma *sum-of-unit-vec-length*:
fixes $i \ j \ n::\text{nat}$ **and** $c::\text{complex}$
assumes $i < n$ **and** $j < n$ **and** $i \neq j$
shows $\|\text{unit-vec } n \ i + c \cdot_v \text{unit-vec } n \ j\|^2 = 1 + \text{cnj}(c) * c$
 $\langle \text{proof} \rangle$

lemma *sum-of-unit-vec-to-col*:
assumes $\text{dim-col } A = n$ **and** $i < n$ **and** $j < n$
shows $\text{col } A \ i + c \cdot_v \text{col } A \ j = A * |\text{unit-vec } n \ i + c \cdot_v \text{unit-vec } n \ j\rangle$
 $\langle \text{proof} \rangle$

lemma *inner-prod-is-sesquilinear*:

fixes $u1\ u2\ v1\ v2::\text{complex vec}$ **and** $c1\ c2\ c3\ c4::\text{complex}$ **and** $n::\text{nat}$
assumes $\text{dim-vec } u1 = n$ **and** $\text{dim-vec } u2 = n$ **and** $\text{dim-vec } v1 = n$ **and** $\text{dim-vec } v2 = n$
shows $\langle c1 \cdot_v u1 + c2 \cdot_v u2 | c3 \cdot_v v1 + c4 \cdot_v v2 \rangle = \text{cnj } (c1) * c3 * \langle u1 | v1 \rangle +$
 $\text{cnj } (c2) * c3 * \langle u2 | v1 \rangle +$
 $\text{cnj } (c1) * c4 * \langle u1 | v2 \rangle + \text{cnj } (c2) * c4 * \langle u2 | v2 \rangle$
 $\langle \text{proof} \rangle$

A length-preserving matrix is unitary. So, unitary matrices are exactly the length-preserving matrices.

lemma *length-preserving-is-unitary*:

fixes $U::\text{complex mat}$
assumes $\text{square-mat } U$ **and** $\forall v::\text{complex vec. dim-vec } v = \text{dim-col } U \longrightarrow \|U * v\| = \|v\|$
shows $\text{unitary } U$
 $\langle \text{proof} \rangle$

lemma *inner-prod-with-unitary-mat* [simp]:

assumes $\text{unitary } U$ **and** $\text{dim-vec } u = \text{dim-col } U$ **and** $\text{dim-vec } v = \text{dim-col } U$
shows $\langle U * |u\rangle | U * |v\rangle \rangle = \langle u | v \rangle$
 $\langle \text{proof} \rangle$

As a consequence we prove that columns and rows of a unitary matrix are orthonormal vectors.

lemma *unitary-unit-col* [simp]:

assumes $\text{unitary } U$ **and** $\text{dim-col } U = n$ **and** $i < n$
shows $\|\text{col } U\ i\| = 1$
 $\langle \text{proof} \rangle$

lemma *unitary-unit-row* [simp]:

assumes $\text{unitary } U$ **and** $\text{dim-row } U = n$ **and** $i < n$
shows $\|\text{row } U\ i\| = 1$
 $\langle \text{proof} \rangle$

lemma *orthogonal-col-of-unitary* [simp]:

assumes $\text{unitary } U$ **and** $\text{dim-col } U = n$ **and** $i < n$ **and** $j < n$ **and** $i \neq j$
shows $\langle \text{col } U\ i | \text{col } U\ j \rangle = 0$
 $\langle \text{proof} \rangle$

lemma *orthogonal-row-of-unitary* [simp]:

fixes $U::\text{complex mat}$
assumes $\text{unitary } U$ **and** $\text{dim-row } U = n$ **and** $i < n$ **and** $j < n$ **and** $i \neq j$
shows $\langle \text{row } U\ i | \text{row } U\ j \rangle = 0$
 $\langle \text{proof} \rangle$

As a consequence, we prove that a quantum gate acting on a state of a system of n qubits give another state of that same system.

lemma *gate-on-state-is-state* [*intro, simp*]:
assumes *a1:gate n A and a2:state n v*
shows *state n (A * v)*
 ⟨*proof*⟩

3.7 A Few Well-known Quantum Gates

Any unitary operation on n qubits can be implemented exactly by composing single qubits and CNOT-gates (controlled-NOT gates). However, no straightforward method is known to implement these gates in a fashion which is resistant to errors. But, the Hadamard gate, the phase gate, the CNOT-gate and the $\pi/8$ gate are also universal for quantum computations, i.e. any quantum circuit on n qubits can be approximated to an arbitrary accuracy by using only these gates, and these gates can be implemented in a fault-tolerant way.

We introduce a coercion from real matrices to complex matrices.

definition *real-to-cpx-mat*:: *real mat* \Rightarrow *complex mat* **where**
real-to-cpx-mat $A \equiv \text{mat } (\text{dim-row } A) (\text{dim-col } A) (\lambda(i,j). A \$\$ (i,j))$

Our first quantum gate: the identity matrix! Arguably, not a very interesting one though!

definition *Id* :: *nat* \Rightarrow *complex mat* **where**
Id $n \equiv 1_m (2^{\wedge}n)$

lemma *id-is-gate* [*simp*]:
gate n (Id n)
 ⟨*proof*⟩

More interesting: the Pauli matrices.

definition *X* :: *complex mat* **where**
X $\equiv \text{mat } 2 \ 2 (\lambda(i,j). \text{if } i=j \text{ then } 0 \text{ else } 1)$

Be aware that *gate n A* means that the matrix A has dimension $2^{\wedge}n * 2^{\wedge}n$. For instance, with this convention a 2×2 matrix A which is unitary satisfies *gate 1 A* but not *gate 2 A* as one might have been expected.

lemma *dagger-of-X* [*simp*]:
 $X^{\dagger} = X$
 ⟨*proof*⟩

lemma *X-inv* [*simp*]:
 $X * X = 1_m \ 2$
 ⟨*proof*⟩

lemma *X-is-gate* [*simp*]:
gate 1 X
 ⟨*proof*⟩

definition Y :: complex mat where

$Y \equiv \text{mat } 2 \ 2 \ (\lambda(i,j). \text{ if } i=j \text{ then } 0 \text{ else (if } i=0 \text{ then } -i \text{ else } i))$

lemma dagger-of- Y [simp]:

$$Y^\dagger = Y$$

<proof>

lemma Y -inv [simp]:

$$Y * Y = 1_m \ 2$$

<proof>

lemma Y -is-gate [simp]:

$$\text{gate } 1 \ Y$$

<proof>

definition Z :: complex mat where

$Z \equiv \text{mat } 2 \ 2 \ (\lambda(i,j). \text{ if } i \neq j \text{ then } 0 \text{ else (if } i=0 \text{ then } 1 \text{ else } -1))$

lemma dagger-of- Z [simp]:

$$Z^\dagger = Z$$

<proof>

lemma Z -inv [simp]:

$$Z * Z = 1_m \ 2$$

<proof>

lemma Z -is-gate [simp]:

$$\text{gate } 1 \ Z$$

<proof>

The Hadamard gate

definition H :: complex mat where

$H \equiv 1/\text{sqrt}(2) \cdot_m (\text{mat } 2 \ 2 \ (\lambda(i,j). \text{ if } i \neq j \text{ then } 1 \text{ else (if } i=0 \text{ then } 1 \text{ else } -1)))$

lemma H -without-scalar-prod:

$$H = \text{mat } 2 \ 2 \ (\lambda(i,j). \text{ if } i \neq j \text{ then } 1/\text{sqrt}(2) \text{ else (if } i=0 \text{ then } 1/\text{sqrt}(2) \text{ else } -(1/\text{sqrt}(2))))$$

<proof>

lemma dagger-of- H [simp]:

$$H^\dagger = H$$

<proof>

lemma H -inv [simp]:

$$H * H = 1_m \ 2$$

<proof>

lemma H -is-gate [simp]:

gate 1 H
 ⟨proof⟩

lemma H -values:
 fixes $i\ j::\ \text{nat}$
 assumes $i < \text{dim-row } H$ and $j < \text{dim-col } H$ and $i \neq 1 \vee j \neq 1$
 shows $H \ \$\$ (i,j) = 1/\text{sqrt } 2$
 ⟨proof⟩

lemma H -values-right-bottom:
 fixes $i\ j::\ \text{nat}$
 assumes $i = 1 \wedge j = 1$
 shows $H \ \$\$ (i,j) = -\ 1/\text{sqrt } 2$
 ⟨proof⟩

The controlled-NOT gate

definition $CNOT$::complex mat **where**
 $CNOT \equiv \text{mat } 4\ 4$
 $(\lambda(i,j). \text{if } i=0 \wedge j=0 \text{ then } 1 \text{ else}$
 $(\text{if } i=1 \wedge j=1 \text{ then } 1 \text{ else}$
 $(\text{if } i=2 \wedge j=3 \text{ then } 1 \text{ else}$
 $(\text{if } i=3 \wedge j=2 \text{ then } 1 \text{ else } 0))))$

lemma dagger-of- $CNOT$ [simp]:
 $CNOT^\dagger = CNOT$
 ⟨proof⟩

lemma $CNOT$ -inv [simp]:
 $CNOT * CNOT = 1_m\ 4$
 ⟨proof⟩

lemma $CNOT$ -is-gate [simp]:
 gate 2 $CNOT$
 ⟨proof⟩

The phase gate, also known as the S-gate

definition S ::complex mat **where**
 $S \equiv \text{mat } 2\ 2 (\lambda(i,j). \text{if } i=0 \wedge j=0 \text{ then } 1 \text{ else } (\text{if } i=1 \wedge j=1 \text{ then } i \text{ else } 0))$

The $\pi/8$ gate, also known as the T-gate

definition T ::complex mat **where**
 $T \equiv \text{mat } 2\ 2 (\lambda(i,j). \text{if } i=0 \wedge j=0 \text{ then } 1 \text{ else } (\text{if } i=1 \wedge j=1 \text{ then } \exp(i*(\pi/4))$
 else 0))

A few relations between the Hadamard gate and the Pauli matrices

lemma HXH -is- Z [simp]:
 $H * X * H = Z$
 ⟨proof⟩

lemma *HYH-is-minusY* [simp]:

$$H * Y * H = - Y$$

<proof>

lemma *HZH-is-X* [simp]:

$$\text{shows } H * Z * H = X$$

<proof>

3.8 The Bell States

We introduce below the so-called Bell states, also known as EPR pairs (EPR stands for Einstein, Podolsky and Rosen).

definition *bell00* :: *complex mat* ($\langle |\beta_{00}\rangle \rangle$) **where**

$$\text{bell00} \equiv 1/\text{sqrt}(2) \cdot_m |\text{vec } 4 \text{ } (\lambda i. \text{if } i=0 \vee i=3 \text{ then } 1 \text{ else } 0)\rangle$$

definition *bell01* :: *complex mat* ($\langle |\beta_{01}\rangle \rangle$) **where**

$$\text{bell01} \equiv 1/\text{sqrt}(2) \cdot_m |\text{vec } 4 \text{ } (\lambda i. \text{if } i=1 \vee i=2 \text{ then } 1 \text{ else } 0)\rangle$$

definition *bell10* :: *complex mat* ($\langle |\beta_{10}\rangle \rangle$) **where**

$$\text{bell10} \equiv 1/\text{sqrt}(2) \cdot_m |\text{vec } 4 \text{ } (\lambda i. \text{if } i=0 \text{ then } 1 \text{ else if } i=3 \text{ then } -1 \text{ else } 0)\rangle$$

definition *bell11* :: *complex mat* ($\langle |\beta_{11}\rangle \rangle$) **where**

$$\text{bell11} \equiv 1/\text{sqrt}(2) \cdot_m |\text{vec } 4 \text{ } (\lambda i. \text{if } i=1 \text{ then } 1 \text{ else if } i=2 \text{ then } -1 \text{ else } 0)\rangle$$

lemma

shows *bell00-is-state* [simp]: *state 2* $|\beta_{00}\rangle$ **and** *bell01-is-state* [simp]: *state 2* $|\beta_{01}\rangle$

and

bell10-is-state [simp]: *state 2* $|\beta_{10}\rangle$ **and** *bell11-is-state* [simp]: *state 2* $|\beta_{11}\rangle$

<proof>

lemma *bell00-index* [simp]:

shows $|\beta_{00}\rangle \text{ } \$\$ (0,0) = 1/\text{sqrt } 2$ **and** $|\beta_{00}\rangle \text{ } \$\$ (1,0) = 0$ **and** $|\beta_{00}\rangle \text{ } \$\$ (2,0) = 0$ **and**

$$|\beta_{00}\rangle \text{ } \$\$ (3,0) = 1/\text{sqrt } 2$$

<proof>

lemma *bell01-index* [simp]:

shows $|\beta_{01}\rangle \text{ } \$\$ (0,0) = 0$ **and** $|\beta_{01}\rangle \text{ } \$\$ (1,0) = 1/\text{sqrt } 2$ **and** $|\beta_{01}\rangle \text{ } \$\$ (2,0) = 1/\text{sqrt } 2$ **and**

$$|\beta_{01}\rangle \text{ } \$\$ (3,0) = 0$$

<proof>

lemma *bell10-index* [simp]:

shows $|\beta_{10}\rangle \text{ } \$\$ (0,0) = 1/\text{sqrt } 2$ **and** $|\beta_{10}\rangle \text{ } \$\$ (1,0) = 0$ **and** $|\beta_{10}\rangle \text{ } \$\$ (2,0) = 0$ **and**

$$|\beta_{10}\rangle \text{ } \$\$ (3,0) = - 1/\text{sqrt } 2$$

<proof>

lemma *bell-11-index* [*simp*]:
shows $|\beta_{11}\rangle \text{ \}\ (0,0) = 0$ **and** $|\beta_{11}\rangle \text{ \}\ (1,0) = 1/\text{sqrt } 2$ **and** $|\beta_{11}\rangle \text{ \}\ (2,0) = -1/\text{sqrt } 2$ **and**
 $|\beta_{11}\rangle \text{ \}\ (3,0) = 0$
 $\langle \text{proof} \rangle$

3.9 The Bitwise Inner Product

definition *bitwise-inner-prod*:: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat}$ **where**
bitwise-inner-prod $n\ i\ j = (\sum_{k \in \{0..<n\}} (\text{bin-rep } n\ i) ! k * (\text{bin-rep } n\ j) ! k)$

abbreviation *bip*:: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat}$ ($\langle \cdot \cdot \cdot \rangle$) **where**
bip $i\ n\ j \equiv \text{bitwise-inner-prod } n\ i\ j$

lemma *bitwise-inner-prod-fst-el-0*:
assumes $i < 2^n \vee j < 2^n$
shows $(i \cdot_{\text{Suc } n} j) = (i \bmod 2^n) \cdot_n (j \bmod 2^n)$
 $\langle \text{proof} \rangle$

lemma *bitwise-inner-prod-fst-el-is-1*:
fixes $n\ i\ j$:: nat
assumes $i \geq 2^n \wedge j \geq 2^n$ **and** $i < 2^{n+1} \wedge j < 2^{n+1}$
shows $(i \cdot_{n+1} j) = 1 + ((i \bmod 2^n) \cdot_n (j \bmod 2^n))$
 $\langle \text{proof} \rangle$

lemma *bitwise-inner-prod-with-zero*:
assumes $m < 2^n$
shows $(0 \cdot_n m) = 0$
 $\langle \text{proof} \rangle$

end

4 Complex Vectors

theory *Complex-Vectors*
imports
Quantum
VectorSpace.VectorSpace
begin

4.1 The Vector Space of Complex Vectors of Dimension n

definition *module-cpx-vec*:: $\text{nat} \Rightarrow (\text{complex}, \text{complex } \text{vec})$ **module** **where**
module-cpx-vec $n \equiv \text{module-vec } \text{TYPE}(\text{complex})\ n$

definition *cpx-rng*:: *complex ring* **where**
cpx-rng $\equiv (\text{carrier} = \text{UNIV}, \text{mult} = (*), \text{one} = 1, \text{zero} = 0, \text{add} = (+))$

lemma *cpx-cring-is-field* [simp]:

field cpx-rng

<proof>

lemma *cpx-abelian-monoid* [simp]:

abelian-monoid cpx-rng

<proof>

lemma *vecspace-cpx-vec* [simp]:

vectorspace cpx-rng (module-cpx-vec n)

<proof>

lemma *module-cpx-vec* [simp]:

Module.module cpx-rng (module-cpx-vec n)

<proof>

definition *state-basis*:: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{complex vec}$ **where**

state-basis n i \equiv *unit-vec* ($2^{\wedge}n$) *i*

definition *unit-vectors*:: $\text{nat} \Rightarrow (\text{complex vec}) \text{ set}$ **where**

unit-vectors n \equiv $\{\text{unit-vec } n \ i \mid i::\text{nat. } 0 \leq i \wedge i < n\}$

lemma *unit-vectors-carrier-vec* [simp]:

unit-vectors n \subseteq *carrier-vec n*

<proof>

lemma (in *Module.module*) *finsum-over-singleton* [simp]:

assumes $f \ x \in \text{carrier } M$

shows $\text{finsum } M \ f \ \{x\} = f \ x$

<proof>

lemma *lincomb-over-singleton* [simp]:

assumes $x \in \text{carrier-vec } n$ **and** $f \in \{x\} \rightarrow \text{UNIV}$

shows $\text{module.lincomb } (\text{module-cpx-vec } n) \ f \ \{x\} = f \ x \cdot_v x$

<proof>

lemma *dim-vec-lincomb* [simp]:

assumes *finite* F **and** $f: F \rightarrow \text{UNIV}$ **and** $F \subseteq \text{carrier-vec } n$

shows $\text{dim-vec } (\text{module.lincomb } (\text{module-cpx-vec } n) \ f \ F) = n$

<proof>

lemma *lincomb-vec-index* [simp]:

assumes *finite* F **and** $a2:i < n$ **and** $F \subseteq \text{carrier-vec } n$ **and** $f: F \rightarrow \text{UNIV}$

shows $\text{module.lincomb } (\text{module-cpx-vec } n) \ f \ F \ \$ \ i = (\sum v \in F. f \ v * (v \ \$ \ i))$

<proof>

lemma *unit-vectors-is-lin-indpt* [simp]:

module.lin-indpt cpx-rng (module-cpx-vec n) (unit-vectors n)

<proof>

```

lemma unit-vectors-is-genset [simp]:
  module.gen-set cpx-rng (module-cpx-vec n) (unit-vectors n)
  ⟨proof⟩

lemma unit-vectors-is-basis [simp]:
  vectorspace.basis cpx-rng (module-cpx-vec n) (unit-vectors n)
  ⟨proof⟩

lemma state-qbit-is-lincomb [simp]:
  state-qbit n =
  { module.lincomb (module-cpx-vec (2^n)) a A | a A.
    finite A ∧ A ⊆ (unit-vectors (2^n)) ∧ a ∈ A → UNIV ∧ ||module.lincomb
    (module-cpx-vec (2^n)) a A|| = 1 }
  ⟨proof⟩

end

```

5 Tensor Products

```

theory Tensor
imports
  Complex-Vectors
  Matrix-Tensor.Matrix-Tensor
  Jordan-Normal-Form.Matrix
begin

```

There is already a formalization of tensor products in the Archive of Formal Proofs, namely `Matrix_Tensor.thy` in `Tensor Product of Matrices`[1] by T.V.H. Prathamesh, but it does not build on top of the formalization of vectors and matrices given in `Matrices`, `Jordan Normal Forms`, and `Spectral Radius Theory`[2] by René Thiemann and Akihisa Yamada. In the present theory our purpose consists in giving such a formalization. Of course, we will reuse Prathamesh’s code as much as possible, and in order to achieve that we formalize some lemmas that translate back and forth between vectors (resp. matrices) seen as lists (resp. lists of lists) and vectors (resp. matrices) as formalized in [2].

5.1 The Kronecker Product of Complex Vectors

```

definition tensor-vec:: complex Matrix.vec ⇒ complex Matrix.vec ⇒ complex Ma-
trix.vec (infixl ⟨ $\otimes$ ⟩ 63)
where tensor-vec u v ≡ vec-of-list (mult.vec-vec-Tensor (*) (list-of-vec u) (list-of-vec
v))

```

5.2 The Tensor Product of Complex Matrices

To see a matrix in the sense of [2] as a matrix in the sense of [1], we convert it into its list of column vectors.

definition *mat-to-cols-list*:: *complex Matrix.mat* \Rightarrow *complex list list* **where**
 $\text{mat-to-cols-list } A = [[A \ \$\$ (i,j) . i <- [0..< \text{dim-row } A]] . j <- [0..< \text{dim-col } A]]$

lemma *length-mat-to-cols-list* [*simp*]:
 $\text{length } (\text{mat-to-cols-list } A) = \text{dim-col } A$
<proof>

lemma *length-cols-mat-to-cols-list* [*simp*]:
assumes $j < \text{dim-col } A$
shows $\text{length } [A \ \$\$ (i,j) . i <- [0..< \text{dim-row } A]] = \text{dim-row } A$
<proof>

lemma *length-row-mat-to-cols-list* [*simp*]:
assumes $i < \text{dim-row } A$
shows $\text{length } (\text{row } (\text{mat-to-cols-list } A) \ i) = \text{dim-col } A$
<proof>

lemma *length-col-mat-to-cols-list* [*simp*]:
assumes $j < \text{dim-col } A$
shows $\text{length } (\text{col } (\text{mat-to-cols-list } A) \ j) = \text{dim-row } A$
<proof>

lemma *mat-to-cols-list-is-not-Nil* [*simp*]:
assumes $\text{dim-col } A > 0$
shows $\text{mat-to-cols-list } A \neq []$
<proof>

Link between *Matrix_Tensor.row_length* and *Matrix.dim_row*

lemma *row-length-mat-to-cols-list* [*simp*]:
assumes $\text{dim-col } A > 0$
shows $\text{mult.row-length } (\text{mat-to-cols-list } A) = \text{dim-row } A$
<proof>

mat-to-cols-list is a matrix in the sense of *Matrix.Matrix-Legacy*.

lemma *mat-to-cols-list-is-mat* [*simp*]:
assumes $\text{dim-col } A > 0$
shows $\text{mat } (\text{mult.row-length } (\text{mat-to-cols-list } A)) \ (\text{length } (\text{mat-to-cols-list } A))$
 $(\text{mat-to-cols-list } A)$
<proof>

definition *mat-of-cols-list*:: *nat* \Rightarrow *complex list list* \Rightarrow *complex Matrix.mat* **where**
 $\text{mat-of-cols-list } nr \ cs = \text{Matrix.mat } nr \ (\text{length } cs) \ (\lambda (i,j). \ cs \ ! \ j \ ! \ i)$

lemma *index-mat-of-cols-list* [*simp*]:

assumes $i < nr$ **and** $j < \text{length } cs$
shows $\text{mat-of-cols-list } nr \ cs \ \$\$ (i,j) = cs ! j ! i$
 $\langle \text{proof} \rangle$

lemma $\text{mat-to-cols-list-to-mat}$ [simp]:
 $\text{mat-of-cols-list } (\text{dim-row } A) (\text{mat-to-cols-list } A) = A$
 $\langle \text{proof} \rangle$

lemma plus-mult-cpx [simp]:
 $\text{plus-mult } 1 \ (*) \ 0 \ (+) \ (a\text{-inv } \text{cpx-rng})$
 $\langle \text{proof} \rangle$

lemma $\text{list-to-mat-to-cols-list}$ [simp]:
fixes $l::\text{complex list list}$
assumes $\text{mat } nr \ nc \ l$
shows $\text{mat-to-cols-list } (\text{mat-of-cols-list } nr \ l) = l$
 $\langle \text{proof} \rangle$

lemma $\text{col-mat-of-cols-list}$ [simp]:
assumes $j < \text{length } l$
shows $\text{Matrix.col } (\text{mat-of-cols-list } (\text{length } (l ! j)) \ l) \ j = \text{vec-of-list } (l ! j)$
 $\langle \text{proof} \rangle$

definition $\text{tensor-mat}:: [\text{complex Matrix.mat}, \text{complex Matrix.mat}] \Rightarrow \text{complex Matrix.mat}$ (**infixl** $\langle \otimes \rangle$ 63) **where**
 $\text{tensor-mat } A \ B \equiv$
 $\text{mat-of-cols-list } (\text{dim-row } A * \text{dim-row } B) (\text{mult.Tensor } (*) (\text{mat-to-cols-list } A)$
 $(\text{mat-to-cols-list } B))$

lemma $\text{dim-row-tensor-mat}$ [simp]:
 $\text{dim-row } (A \ \otimes \ B) = \text{dim-row } A * \text{dim-row } B$
 $\langle \text{proof} \rangle$

lemma $\text{dim-col-tensor-mat}$ [simp]:
 $\text{dim-col } (A \ \otimes \ B) = \text{dim-col } A * \text{dim-col } B$
 $\langle \text{proof} \rangle$

lemma $\text{mat-to-cols-list-nth-nth-eq-index-mat}$:
 $\langle \text{mat-to-cols-list } A ! j ! i = A \ \$\$ (i, j) \rangle$ **if** $\langle i < \text{dim-row } A \rangle \ \langle j < \text{dim-col } A \rangle$
 $\langle \text{proof} \rangle$

lemma index-tensor-mat [simp]:
assumes $a1:\text{dim-row } A = rA$ **and** $a2:\text{dim-col } A = cA$ **and** $a3:\text{dim-row } B = rB$
and $a4:\text{dim-col } B = cB$
and $a5:i < rA * rB$ **and** $a6:j < cA * cB$ **and** $a7:cA > 0$ **and** $a8:cB > 0$
shows $(A \ \otimes \ B) \ \$\$ (i,j) = A \ \$\$ (i \ \text{div } rB, \ j \ \text{div } cB) * B \ \$\$ (i \ \text{mod } rB, \ j \ \text{mod } cB)$
 $\langle \text{proof} \rangle$

To go from *Matrix.row* to *Matrix-Legacy.row*

lemma *Matrix-row-is-Legacy-row*:
assumes $i < \text{dim-row } A$
shows $\text{Matrix.row } A \ i = \text{vec-of-list } (\text{row } (\text{mat-to-cols-list } A) \ i)$
 $\langle \text{proof} \rangle$

To go from *Matrix-Legacy.row* to *Matrix.row*

lemma *Legacy-row-is-Matrix-row*:
assumes $i < \text{mult.row-length } A$
shows $\text{row } A \ i = \text{list-of-vec } (\text{Matrix.row } (\text{mat-of-cols-list } (\text{mult.row-length } A) \ A) \ i)$
 $\langle \text{proof} \rangle$

To go from *Matrix.col* to *Matrix-Legacy.col*

lemma *Matrix-col-is-Legacy-col*:
assumes $j < \text{dim-col } A$
shows $\text{Matrix.col } A \ j = \text{vec-of-list } (\text{col } (\text{mat-to-cols-list } A) \ j)$
 $\langle \text{proof} \rangle$

To go from *Matrix-Legacy.col* to *Matrix.col*

lemma *Legacy-col-is-Matrix-col*:
assumes $a1:j < \text{length } A$ **and** $a2:\text{length } (A \ ! \ j) = \text{mult.row-length } A$
shows $\text{col } A \ j = \text{list-of-vec } (\text{Matrix.col } (\text{mat-of-cols-list } (\text{mult.row-length } A) \ A) \ j)$
 $\langle \text{proof} \rangle$

Link between *plus-mult.scalar-product* and (\cdot)

lemma *scalar-prod-is-Matrix-scalar-prod* [*simp*]:
fixes $u::\text{complex list}$ **and** $v::\text{complex list}$
assumes $\text{length } u = \text{length } v$
shows $\text{plus-mult.scalar-product } (*) \ 0 \ (+) \ u \ v = (\text{vec-of-list } u) \cdot (\text{vec-of-list } v)$
 $\langle \text{proof} \rangle$

Link between $(*)$ and $\lambda f \text{ zer } g \ M1. \ \text{mat-multI } \text{zer } g \ f \ (\text{mult.row-length } M1) \ M1$

lemma *matrix-mult-to-times-mat*:
assumes $\text{dim-col } A > 0$ **and** $\text{dim-col } B > 0$ **and** $\text{dim-col } (A::\text{complex Matrix.mat}) = \text{dim-row } B$
shows $A * B = \text{mat-of-cols-list } (\text{dim-row } A) \ (\text{plus-mult.matrix-mult } (*) \ 0 \ (+) \ (\text{mat-to-cols-list } A) \ (\text{mat-to-cols-list } B))$
 $\langle \text{proof} \rangle$

lemma *mat-to-cols-list-times-mat* [*simp*]:
assumes $\text{dim-col } A = \text{dim-row } B$ **and** $\text{dim-col } A > 0$
shows $\text{mat-to-cols-list } (A * B) = \text{plus-mult.matrix-mult } (*) \ 0 \ (+) \ (\text{mat-to-cols-list } A) \ (\text{mat-to-cols-list } B)$
 $\langle \text{proof} \rangle$

Finally, we prove that the tensor product of complex matrices is distributive over the multiplication of complex matrices.

```

lemma mult-distr-tensor:
  assumes a1:dim-col A = dim-row B and a2:dim-col C = dim-row D and
a3:dim-col A > 0 and
a4:dim-col B > 0 and a5:dim-col C > 0 and a6:dim-col D > 0
  shows  $(A * B) \otimes (C * D) = (A \otimes C) * (B \otimes D)$ 
  <proof>

lemma tensor-mat-is-assoc:
  fixes A B C:: complex Matrix.mat
  shows  $A \otimes (B \otimes C) = (A \otimes B) \otimes C$ 
  <proof>

end

```

6 Further Results on Tensor Products

```

theory More-Tensor
imports
  Quantum
  Tensor
  Jordan-Normal-Form.Matrix
  Basics
begin

lemma tensor-prod-2 [simp]:
  mult.vec-vec-Tensor (*) [x1::complex,x2] [x3, x4] = [x1 * x3, x1 * x4, x2 * x3,
x2 * x4]
  <proof>

lemma list-vec [simp]:
  assumes v ∈ state-qbit 1
  shows list-of-vec v = [v $ 0, v $ 1]
  <proof>

lemma vec-tensor-prod-2 [simp]:
  assumes v ∈ state-qbit 1 and w ∈ state-qbit 1
  shows  $v \otimes w = \text{vec-of-list } [v \$ 0 * w \$ 0, v \$ 0 * w \$ 1, v \$ 1 * w \$ 0, v \$ 1$ 
 $* w \$ 1]$ 
  <proof>

lemma vec-dim-of-vec-of-list [simp]:
  assumes length l = n
  shows dim-vec (vec-of-list l) = n
  <proof>

lemma vec-tensor-prod-2-bis [simp]:
  assumes v ∈ state-qbit 1 and w ∈ state-qbit 1
  shows  $v \otimes w = \text{Matrix.vec } 4 \ (\lambda i. \text{if } i = 0 \text{ then } v \$ 0 * w \$ 0 \text{ else}$ 

```

if $i = 3$ then $v \$ 1 * w \$ 1$ else
 if $i = 1$ then $v \$ 0 * w \$ 1$ else $v \$ 1 * w \$ 0$)

<proof>

lemma *index-col-mat-of-cols-list* [simp]:

assumes $i < n$ **and** $j < \text{length } l$

shows $\text{Matrix.col } (\text{mat-of-cols-list } n \ l) \ j \ \$ \ i = l \ ! \ j \ ! \ i$

<proof>

lemma *multTensor2* [simp]:

assumes $a1:A = \text{Matrix.mat } 2 \ 1 \ (\lambda(i,j). \text{ if } i = 0 \text{ then } a0 \text{ else } a1)$ **and**

$a2:B = \text{Matrix.mat } 2 \ 1 \ (\lambda(i,j). \text{ if } i = 0 \text{ then } b0 \text{ else } b1)$

shows $\text{mult.Tensor } (*) \ (\text{mat-to-cols-list } A) \ (\text{mat-to-cols-list } B) = [[a0*b0, a0*b1, a1*b0, a1*b1]]$

<proof>

lemma *multTensor2-bis* [simp]:

assumes $a1:\text{dim-row } A = 2$ **and** $a2:\text{dim-col } A = 1$ **and** $a3:\text{dim-row } B = 2$ **and**
 $a4:\text{dim-col } B = 1$

shows $\text{mult.Tensor } (*) \ (\text{mat-to-cols-list } A) \ (\text{mat-to-cols-list } B) =$

$[[A \ \$\$ \ (0,0) * B \ \$\$ \ (0,0), A \ \$\$ \ (0,0) * B \ \$\$ \ (1,0), A \ \$\$ \ (1,0) * B \ \$\$ \ (0,0), A \ \$\$ \ (1,0) * B \ \$\$ \ (1,0)]]$

<proof>

lemma *mat-tensor-prod-2-prelim* [simp]:

assumes *state 1 v* **and** *state 1 w*

shows $v \otimes w = \text{mat-of-cols-list } 4$

$[[v \ \$\$ \ (0,0) * w \ \$\$ \ (0,0), v \ \$\$ \ (0,0) * w \ \$\$ \ (1,0), v \ \$\$ \ (1,0) * w \ \$\$ \ (0,0), v \ \$\$ \ (1,0) * w \ \$\$ \ (1,0)]]$

<proof>

lemma *mat-tensor-prod-2-col* [simp]:

assumes *state 1 v* **and** *state 1 w*

shows $\text{Matrix.col } (v \otimes w) \ 0 = \text{Matrix.col } v \ 0 \otimes \text{Matrix.col } w \ 0$

<proof>

lemma *mat-tensor-prod-2* [simp]:

assumes *state 1 v* **and** *state 1 w*

shows $v \otimes w = \text{Matrix.mat } 4 \ 1 \ (\lambda(i,j). \text{ if } i = 0 \text{ then } v \ \$\$ \ (0,0) * w \ \$\$ \ (0,0)$
else

if $i = 3$ then $v \ \$\$ \ (1,0) * w \ \$\$ \ (1,0)$ *else*
 if $i = 1$ then $v \ \$\$ \ (0,0) * w \ \$\$ \ (1,0)$ *else*
 $v \ \$\$ \ (1,0) * w \ \$\$ \ (0,0)$)

<proof>

lemma *mat-tensor-prod-2-bis*:

assumes *state 1 v* **and** *state 1 w*

shows $v \otimes w = |\text{Matrix.vec } 4 \ (\lambda i. \text{ if } i = 0 \text{ then } v \ \$\$ \ (0,0) * w \ \$\$ \ (0,0) \text{ else$

if $i = 3$ then $v \text{ \textit{\$} } (1,0) * w \text{ \textit{\$} } (1,0)$ else
 if $i = 1$ then $v \text{ \textit{\$} } (0,0) * w \text{ \textit{\$} } (1,0)$ else
 $v \text{ \textit{\$} } (1,0) * w \text{ \textit{\$} } (0,0)$

$\langle \textit{proof} \rangle$

lemma *eq-ket-vec*:

fixes $u v :: \textit{complex Matrix.vec}$

assumes $u = v$

shows $|u\rangle = |v\rangle$

$\langle \textit{proof} \rangle$

lemma *mat-tensor-ket-vec*:

assumes *state 1 v and state 1 w*

shows $v \otimes w = |(Matrix.col v 0) \otimes (Matrix.col w 0)|$

$\langle \textit{proof} \rangle$

The property of being a state (resp. a gate) is preserved by tensor product.

lemma *tensor-state2 [simp]*:

assumes *state 1 u and state 1 v*

shows *state 2 (u \otimes v)*

$\langle \textit{proof} \rangle$

lemma *sum-prod*:

fixes $f :: \textit{nat} \Rightarrow \textit{complex}$ **and** $g :: \textit{nat} \Rightarrow \textit{complex}$

shows $(\sum i < a * b. f(i \textit{ div } b) * g(i \textit{ mod } b)) = (\sum i < a. f(i)) * (\sum j < b. g(j))$

$\langle \textit{proof} \rangle$

lemma *tensor-state [simp]*:

assumes *state m u and state n v*

shows *state (m + n) (u \otimes v)*

$\langle \textit{proof} \rangle$

lemma *dim-row-of-tensor-gate*:

assumes *gate m G1 and gate n G2*

shows $\textit{dim-row} (G1 \otimes G2) = 2^{m+n}$

$\langle \textit{proof} \rangle$

lemma *tensor-gate-sqr-mat*:

assumes *gate m G1 and gate n G2*

shows *square-mat (G1 \otimes G2)*

$\langle \textit{proof} \rangle$

lemma *dim-row-of-one-mat-less-pow*:

assumes *gate m G1 and gate n G2 and $i < \textit{dim-row} (1_m(\textit{dim-col } G1 * \textit{dim-col } G2))$*

shows $i < 2^{m+n}$

$\langle \textit{proof} \rangle$

lemma *dim-col-of-one-mat-less-pow*:

assumes gate m $G1$ **and** gate n $G2$ **and** $j < \text{dim-col } (1_m(\text{dim-col } G1 * \text{dim-col } G2))$
shows $j < 2^{m+n}$
 $\langle \text{proof} \rangle$

lemma *index-tensor-gate-unitary1*:

assumes gate m $G1$ **and** gate n $G2$ **and** $i < \text{dim-row } (1_m(\text{dim-col } G1 * \text{dim-col } G2))$ **and**
 $j < \text{dim-col } (1_m(\text{dim-col } G1 * \text{dim-col } G2))$
shows $((G1 \otimes G2)^\dagger * (G1 \otimes G2)) \text{ $$ } (i, j) = 1_m(\text{dim-col } G1 * \text{dim-col } G2)$
 $\text{ $$ } (i, j)$
 $\langle \text{proof} \rangle$

lemma *tensor-gate-unitary1* [*simp*]:

assumes gate m $G1$ **and** gate n $G2$
shows $(G1 \otimes G2)^\dagger * (G1 \otimes G2) = 1_m(\text{dim-col } G1 * \text{dim-col } G2)$
 $\langle \text{proof} \rangle$

lemma *index-tensor-gate-unitary2* [*simp*]:

assumes gate m $G1$ **and** gate n $G2$ **and** $i < \text{dim-row } (1_m(\text{dim-col } G1 * \text{dim-col } G2))$ **and**
 $j < \text{dim-col } (1_m(\text{dim-col } G1 * \text{dim-col } G2))$
shows $((G1 \otimes G2) * ((G1 \otimes G2)^\dagger)) \text{ $$ } (i, j) = 1_m(\text{dim-col } G1 * \text{dim-col } G2)$
 $\text{ $$ } (i, j)$
 $\langle \text{proof} \rangle$

lemma *tensor-gate-unitary2* [*simp*]:

assumes gate m $G1$ **and** gate n $G2$
shows $(G1 \otimes G2) * ((G1 \otimes G2)^\dagger) = 1_m(\text{dim-col } G1 * \text{dim-col } G2)$
 $\langle \text{proof} \rangle$

lemma *tensor-gate* [*simp*]:

assumes gate m $G1$ **and** gate n $G2$
shows gate $(m + n)$ $(G1 \otimes G2)$
 $\langle \text{proof} \rangle$

end

7 Measurement

theory *Measurement*

imports

Quantum

begin

Given an element v such that *state* n v , its components $v \text{ \$ } i$ (when v is seen as a vector, v being a matrix column) for $0 \leq i < n$ have to be understood as the coefficients of the representation of v in the basis given by the unit vectors of dimension 2^n , unless stated otherwise. Such a vector v is a state

for a quantum system of n qubits. In the literature on quantum computing, for $n = 1$, i.e. for a quantum system of 1 qubit, the elements of the so-called computational basis are denoted $|0\rangle, |1\rangle$, and these last elements might be understood for instance as $(1, 0), (0, 1)$, i.e. as the zeroth and the first elements of a given basis ; for $n = 2$, i.e. for a quantum system of 2 qubits, the elements of the computational basis are denoted $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, and they might be understood for instance as $(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)$; and so on for higher values of n . The idea behind these standard notations is that the labels on the vectors of the computational basis are the binary expressions of the natural numbers indexing the elements in a given ordered basis interpreting the computational basis in a specific context, another point of view is that the order of the basis corresponds to the lexicographic order for the labels. Those labels also represent the possible outcomes of a measurement of the n qubits of the system, while the squared modules of the corresponding coefficients represent the probabilities for those outcomes. The fact that the vector v has to be normalized expresses precisely the fact that the squared modules of the coefficients represent some probabilities and hence their sum should be 1. Note that in the case of a system with multiple qubits, i.e. $n \geq 2$, one can model the simultaneous measurement of multiple qubits by sequential measurements of single qubits. Indeed, this last process leads to the same probabilities for the various possible outcomes. Given a system with n -qubits and i the index of one qubit among the n qubits of the system, where $0 \leq i \leq n - 1$ (i.e. we start the indexing from 0), we want to find the indices of the states of the computational basis whose labels have a 1 at the i th spot (counting from 0). For instance, if $n = 3$ and $i = 2$ then 1,3,5,7 are the indices of the elements of the computational basis with a 1 at the 2nd spot, namely $|001\rangle, |011\rangle, |101\rangle, |111\rangle$. To achieve that we define the predicate *select-index* below.

definition *select-index* :: $nat \Rightarrow nat \Rightarrow nat \Rightarrow bool$ **where**
select-index $n\ i\ j \equiv (i \leq n - 1) \wedge (j \leq 2^{\wedge} n - 1) \wedge (j \bmod 2^{\wedge} (n - i) \geq 2^{\wedge} (n - 1 - i))$

lemma *select-index-union*:
 $\{k \mid k :: nat. \text{select-index } n\ i\ k\} \cup \{k \mid k :: nat. (k < 2^{\wedge} n) \wedge \neg \text{select-index } n\ i\ k\} = \{0..<2^{\wedge} n :: nat\}$
<proof>

lemma *select-index-inter*:
 $\{k \mid k :: nat. \text{select-index } n\ i\ k\} \cap \{k \mid k :: nat. (k < 2^{\wedge} n) \wedge \neg \text{select-index } n\ i\ k\} = \{\}$
<proof>

lemma *outcomes-sum* [*simp*]:
fixes $f :: nat \Rightarrow real$
shows
 $(\sum_{j \in \{k \mid k :: nat. \text{select-index } n\ i\ k\}}. (f\ j)) +$

$$\frac{(\sum_{j \in \{k \mid k::\text{nat}. (k < 2^{\widehat{n}}) \wedge \neg \text{select-index } n \ i \ k\}. (f \ j)})}{(\sum_{j \in \{0..<2^{\widehat{n}}::\text{nat}\}. (f \ j)})} =$$

<proof>

Given a state v of a n -qbit system, we compute the probability that a measure of qubit i has the outcome 1.

definition *prob1* :: $\text{nat} \Rightarrow \text{complex mat} \Rightarrow \text{nat} \Rightarrow \text{real}$ **where**
prob1 $n \ v \ i \equiv \sum_{j \in \{k \mid k::\text{nat}. \text{select-index } n \ i \ k\}. (\text{cmod}(v \ \$\$ (j,0)))^2$

definition *prob0* :: $\text{nat} \Rightarrow \text{complex mat} \Rightarrow \text{nat} \Rightarrow \text{real}$ **where**
prob0 $n \ v \ i \equiv \sum_{j \in \{k \mid k::\text{nat}. (k < 2^{\widehat{n}}) \wedge \neg \text{select-index } n \ i \ k\}. (\text{cmod}(v \ \$\$ (j,0)))^2$

lemma

shows *prob1-geq-zero:prob1* $n \ v \ i \geq 0$ **and** *prob0-geq-zero:prob0* $n \ v \ i \geq 0$
<proof>

lemma *prob-sum-is-one* [*simp*]:

assumes *state* $n \ v$
shows *prob1* $n \ v \ i + \text{prob0}$ $n \ v \ i = 1$
<proof>

lemma

assumes *state* $n \ v$
shows *prob1-leq-one:prob1* $n \ v \ i \leq 1$ **and** *prob0-leq-one:prob0* $n \ v \ i \leq 1$
<proof>

lemma *prob0-is-prob*:

assumes *state* $n \ v$
shows *prob0* $n \ v \ i \geq 0 \wedge \text{prob0}$ $n \ v \ i \leq 1$
<proof>

lemma *prob1-is-prob*:

assumes *state* $n \ v$
shows *prob1* $n \ v \ i \geq 0 \wedge \text{prob1}$ $n \ v \ i \leq 1$
<proof>

Below we give the new state of a n -qubits system after a measurement of the i th qubit gave 0.

definition *post-meas0* :: $\text{nat} \Rightarrow \text{complex mat} \Rightarrow \text{nat} \Rightarrow \text{complex mat}$ **where**
post-meas0 $n \ v \ i \equiv$
of-real($1/\text{sqrt}(\text{prob0 } n \ v \ i)$) $\cdot_m \text{vec } (2^{\widehat{n}}) (\lambda j. \text{if } \neg \text{select-index } n \ i \ j \text{ then } v \ \$\$ (j,0) \text{ else } 0)$

Note that a division by 0 never occurs. Indeed, if $\text{sqrt}(\text{prob0 } n \ v \ i)$ would be 0 then *prob0* $n \ v \ i$ would be 0 and it would mean that the measurement of the i th qubit gave 1.

lemma *post-meas0-is-state* [*simp*]:

assumes *state* $n \ v$ **and** *prob0* $n \ v \ i \neq 0$

shows state n (*post-meas0* n v i)
 ⟨*proof*⟩

Below we give the new state of a n -qubits system after a measurement of the i th qubit gave 1.

definition *post-meas1* :: $\text{nat} \Rightarrow \text{complex mat} \Rightarrow \text{nat} \Rightarrow \text{complex mat}$ **where**
post-meas1 n v i \equiv
 of-real($1/\text{sqrt}(\text{prob1 } n$ v $i)$) \cdot_m |*vec* ($2^{\wedge}n$) (λj . if *select-index* n i j then v $\$ \$$ ($j,0$) else 0))

Note that a division by 0 never occurs. Indeed, if *sqrt*(*prob1* n v i) would be 0 then *prob1* n v i would be 0 and it would mean that the measurement of the i th qubit gave 0.

lemma *post-meas-1-is-state* [*simp*]:
assumes state n v **and** *prob1* n v i \neq 0
shows state n (*post-meas1* n v i)
 ⟨*proof*⟩

The measurement operator below takes a number of qubits n , a state v of a n -qubits system, a number i corresponding to the index (starting from 0) of one qubit among the n -qubits, and it computes a list whose first (resp. second) element is the pair made of the probability that the outcome of the measurement of the i th qubit is 0 (resp. 1) and the corresponding post-measurement state of the system. Of course, note that i should be strictly less than n and v should be a state of dimension n , i.e. state n v should hold".

definition *meas* :: $\text{nat} \Rightarrow \text{complex mat} \Rightarrow \text{nat} \Rightarrow \text{-list}$ **where**
meas n v i \equiv [(*prob0* n v i , *post-meas0* n v i), (*prob1* n v i , *post-meas1* n v i)]

We want to determine the probability that the first n qubits of an $n+1$ qubit system are 0. For this we need to find the indices of the states of the computational basis whose labels do not have a 1 at spot $i = 0, \dots, n$.

definition *prob0-fst-qubits*:: $\text{nat} \Rightarrow \text{complex Matrix.mat} \Rightarrow \text{real}$ **where**
prob0-fst-qubits n v \equiv
 $\sum_{j \in \{k \mid k::\text{nat}. (k < 2^{\wedge}(n+1)) \wedge (\forall i \in \{0..<n\}. \neg \text{select-index } (n+1) i k)\}}$. (*cm*od(v $\$ \$$ ($j,0$)))²

lemma *select-index-div-2*:
fixes n i j :: nat
assumes $i < 2^{\wedge}(n+1)$ **and** $j < n$
shows *select-index* n j ($i \text{ div } 2$) = *select-index* ($n+1$) j i
 ⟨*proof*⟩

lemma *select-index-suc-even*:
fixes n k i :: nat
assumes $k < 2^{\wedge}n$ **and** *select-index* n i k
shows *select-index* (*Suc* n) i ($2 * k$)

<proof>

lemma *select-index-suc-odd*:

fixes $n\ k\ i::\ \text{nat}$

assumes $k \leq 2^{\wedge}n - 1$ **and** *select-index* $n\ i\ k$

shows *select-index* $(\text{Suc } n)\ i\ (2*k+1)$

<proof>

lemma *aux-range*:

fixes $k::\ \text{nat}$

assumes $k < 2^{\wedge}(\text{Suc } n + 1)$ **and** $k \geq 2$

shows $k = 2 \vee k = 3 \vee (\exists l. l \geq 2 \wedge l \leq 2^{\wedge}(n+1) - 1 \wedge (k = 2*l \vee k = 2*l + 1))$

<proof>

lemma *select-index-with-1*:

fixes $n::\ \text{nat}$

assumes $n \geq 1$

shows $\forall k. k < 2^{\wedge}(n+1) \longrightarrow k \geq 2 \longrightarrow (\exists i < n. \text{select-index } (n+1)\ i\ k)$

<proof>

lemma *prob0-fst-qubits-index*:

fixes $n::\ \text{nat}$ **and** $v::\ \text{complex Matrix.mat}$

shows $\{k \mid k::\ \text{nat}. (k < 2^{\wedge}(n+1)) \wedge (\forall i \in \{0..<n\}. \neg \text{select-index } (n+1)\ i\ k)\} = \{0,1\}$

<proof>

lemma *prob0-fst-qubits-eq*:

fixes $n::\ \text{nat}$

shows *prob0-fst-qubits* $n\ v = (\text{cmod}(v\ \$\$ (0,0)))^2 + (\text{cmod}(v\ \$\$ (1,0)))^2$

<proof>

primrec *iter-post-meas0*:: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{complex Matrix.mat} \Rightarrow \text{complex Matrix.mat}$ **where**

iter-post-meas0 $n\ 0\ v = v$

| *iter-post-meas0* $n\ (\text{Suc } m)\ v = \text{post-meas0 } n\ (\text{iter-post-meas0 } n\ m\ v)\ m$

definition *iter-prob0*:: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{complex Matrix.mat} \Rightarrow \text{real}$ **where**

iter-prob0 $n\ m\ v = (\prod i < m. \text{prob0 } n\ (\text{iter-post-meas0 } n\ i\ v)\ i)$

7.1 Measurements with Bell States

A Bell state is a remarkable state. Indeed, if one makes one measure, either of the first or the second qubit, then one gets either 0 with probability 1/2 or 1 with probability 1/2. Moreover, in the case of two successive measurements of the first and second qubit, the outcomes are correlated. Indeed, in the case of $|\beta_{00}\rangle$ or $|\beta_{10}\rangle$ (resp. $|\beta_{01}\rangle$ or $|\beta_{11}\rangle$) if one measures the second qubit after

a measurement of the first qubit (or the other way around) then one gets the same outcomes (resp. opposite outcomes), i.e. for instance the probability of measuring 0 for the second qubit after a measure with outcome 0 for the first qubit is 1 (resp. 0).

lemma *prob0-bell-fst [simp]*:
assumes $v = |\beta_{00}\rangle \vee v = |\beta_{01}\rangle \vee v = |\beta_{10}\rangle \vee v = |\beta_{11}\rangle$
shows $\text{prob0 } 2 \ v \ 0 = 1/2$
<proof>

lemma *prob1-bell-fst [simp]*:
assumes $v = |\beta_{00}\rangle \vee v = |\beta_{01}\rangle \vee v = |\beta_{10}\rangle \vee v = |\beta_{11}\rangle$
shows $\text{prob1 } 2 \ v \ 0 = 1/2$
<proof>

lemma *prob0-bell-snd [simp]*:
assumes $v = |\beta_{00}\rangle \vee v = |\beta_{01}\rangle \vee v = |\beta_{10}\rangle \vee v = |\beta_{11}\rangle$
shows $\text{prob0 } 2 \ v \ 1 = 1/2$
<proof>

lemma *prob1-bell-snd [simp]*:
assumes $v = |\beta_{00}\rangle \vee v = |\beta_{01}\rangle \vee v = |\beta_{10}\rangle \vee v = |\beta_{11}\rangle$
shows $\text{prob1 } 2 \ v \ 1 = 1/2$
<proof>

lemma *post-meas0-bell00-fst [simp]*:
 $\text{post-meas0 } 2 \ |\beta_{00}\rangle \ 0 = |\text{unit-vec } 4 \ 0\rangle$
<proof>

lemma *post-meas0-bell00-snd [simp]*:
 $\text{post-meas0 } 2 \ |\beta_{00}\rangle \ 1 = |\text{unit-vec } 4 \ 0\rangle$
<proof>

lemma *post-meas0-bell01-fst [simp]*:
 $\text{post-meas0 } 2 \ |\beta_{01}\rangle \ 0 = |\text{unit-vec } 4 \ 1\rangle$
<proof>

lemma *post-meas0-bell01-snd [simp]*:
 $\text{post-meas0 } 2 \ |\beta_{01}\rangle \ 1 = |\text{unit-vec } 4 \ 2\rangle$
<proof>

lemma *post-meas0-bell10-fst [simp]*:
 $\text{post-meas0 } 2 \ |\beta_{10}\rangle \ 0 = |\text{unit-vec } 4 \ 0\rangle$
<proof>

lemma *post-meas0-bell10-snd [simp]*:
 $\text{post-meas0 } 2 \ |\beta_{10}\rangle \ 1 = |\text{unit-vec } 4 \ 0\rangle$
<proof>

lemma *post-meas0-bell11-fst [simp]*:

post-meas0 2 $|\beta_{11}\rangle$ 0 = $|\text{unit-vec } 4 \ 1\rangle$
<proof>

lemma *post-meas0-bell11-snd* [*simp*]:
post-meas0 2 $|\beta_{11}\rangle$ 1 = - $|\text{unit-vec } 4 \ 2\rangle$
<proof>

lemma *post-meas1-bell00-fst* [*simp*]:
post-meas1 2 $|\beta_{00}\rangle$ 0 = $|\text{unit-vec } 4 \ 3\rangle$
<proof>

lemma *post-meas1-bell00-snd* [*simp*]:
post-meas1 2 $|\beta_{00}\rangle$ 1 = $|\text{unit-vec } 4 \ 3\rangle$
<proof>

lemma *post-meas1-bell01-fst* [*simp*]:
post-meas1 2 $|\beta_{01}\rangle$ 0 = $|\text{unit-vec } 4 \ 2\rangle$
<proof>

lemma *post-meas1-bell01-snd* [*simp*]:
post-meas1 2 $|\beta_{01}\rangle$ 1 = $|\text{unit-vec } 4 \ 1\rangle$
<proof>

lemma *post-meas1-bell10-fst* [*simp*]:
post-meas1 2 $|\beta_{10}\rangle$ 0 = - $|\text{unit-vec } 4 \ 3\rangle$
<proof>

lemma *post-meas1-bell10-snd* [*simp*]:
post-meas1 2 $|\beta_{10}\rangle$ 1 = - $|\text{unit-vec } 4 \ 3\rangle$
<proof>

lemma *post-meas1-bell11-fst* [*simp*]:
post-meas1 2 $|\beta_{11}\rangle$ 0 = - $|\text{unit-vec } 4 \ 2\rangle$
<proof>

lemma *post-meas1-bell11-snd* [*simp*]:
post-meas1 2 $|\beta_{11}\rangle$ 1 = $|\text{unit-vec } 4 \ 1\rangle$
<proof>

end

8 Quantum Entanglement

theory *Entanglement*

imports

Quantum

More-Tensor

begin

8.1 The Product States and Entangled States of a 2-qubits System

Below we add the condition that v and w are two-dimensional states, otherwise u can always be represented by the tensor product of the 1-dimensional vector 1 and u itself.

definition *prod-state2*:: *complex Matrix.mat* \Rightarrow *bool* **where**
prod-state2 $u \equiv$ if state 2 u then $\exists v w$. state 1 $v \wedge$ state 1 $w \wedge u = v \otimes w$ else undefined

definition *entangled2*:: *complex Matrix.mat* \Rightarrow *bool* **where**
entangled2 $u \equiv \neg$ *prod-state2* u

The Bell states are entangled states.

lemma *bell00-is-entangled2* [*simp*]:
entangled2 $|\beta_{00}\rangle$
 <proof>

lemma *bell01-is-entangled2* [*simp*]:
entangled2 $|\beta_{01}\rangle$
 <proof>

lemma *bell10-is-entangled2* [*simp*]:
entangled2 $|\beta_{10}\rangle$
 <proof>

lemma *bell11-is-entangled2* [*simp*]:
entangled2 $|\beta_{11}\rangle$
 <proof>

An entangled state is a state that cannot be broken down as the tensor product of smaller states.

definition *prod-state*:: *nat* \Rightarrow *complex Matrix.mat* \Rightarrow *bool* **where**
prod-state $m u \equiv$ if state $m u$ then $\exists n p::nat.\exists v w$. state $n v \wedge$ state $p w \wedge n < m \wedge p < m \wedge u = v \otimes w$ else undefined

definition *entangled*:: *nat* \Rightarrow *complex Matrix.mat* \Rightarrow *bool* **where**
entangled $n v \equiv \neg$ (*prod-state* $n v$)

lemma *sanity-check*:
 \neg (*entangled* 2 (*mat-of-cols-list* 2 $[[1/\text{sqrt}(2), 1/\text{sqrt}(2)]] \otimes$ *mat-of-cols-list* 2 $[[1/\text{sqrt}(2), 1/\text{sqrt}(2)]]$))
 <proof>

end

9 Quantum Teleportation

theory *Quantum-Teleportation*

imports

More-Tensor

Basics

Measurement

begin

definition *alice*:: *complex Matrix.mat* \Rightarrow *complex Matrix.mat* **where**
alice $\varphi \equiv (H \otimes Id\ 2) * ((CNOT \otimes Id\ 1) * (\varphi \otimes |\beta_{00}\rangle))$

abbreviation *M1*:: *complex Matrix.mat* **where**

M1 \equiv *mat-of-cols-list* 8 [[1, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 1, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 0],
[0, 0, 0, 0, 0, 0, 0, 1],
[0, 0, 0, 0, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 0, 0]]

lemma *tensor-prod-of-cnot-id-1*:

shows $(CNOT \otimes Id\ 1) = M1$

<proof>

abbreviation *M2*:: *complex Matrix.mat* **where**

M2 \equiv *mat-of-cols-list* 8 [[1/sqrt(2), 0, 0, 0, 1/sqrt(2), 0, 0, 0],
[0, 1/sqrt(2), 0, 0, 0, 1/sqrt(2), 0, 0],
[0, 0, 1/sqrt(2), 0, 0, 0, 1/sqrt(2), 0],
[0, 0, 0, 1/sqrt(2), 0, 0, 0, 1/sqrt(2)],
[1/sqrt(2), 0, 0, 0, -1/sqrt(2), 0, 0, 0],
[0, 1/sqrt(2), 0, 0, 0, -1/sqrt(2), 0, 0],
[0, 0, 1/sqrt(2), 0, 0, 0, -1/sqrt(2), 0],
[0, 0, 0, 1/sqrt(2), 0, 0, 0, -1/sqrt(2)]]

lemma *tensor-prod-of-h-id-2*:

shows $(H \otimes Id\ 2) = M2$

<proof>

lemma *alice-step-1-state* [*simp*]:

assumes *state 1* φ

shows *state 3* $(\varphi \otimes |\beta_{00}\rangle)$

<proof>

lemma *alice-step-2-state*:

assumes *state 1* φ

shows *state 3* $((CNOT \otimes Id\ 1) * (\varphi \otimes |\beta_{00}\rangle))$

$\langle proof \rangle$

lemma *alice-state* [simp]:

assumes state 1 φ

shows state 3 (alice φ)

$\langle proof \rangle$

lemma *alice-step-1*:

assumes state 1 φ and $\alpha = \varphi$ $\$ \$ (0,0)$ and $\beta = \varphi$ $\$ \$ (1,0)$

shows $(\varphi \otimes |\beta_{00}\rangle) = \text{mat-of-cols-list } 8 \ [[\alpha/\text{sqrt}(2), 0, 0, \alpha/\text{sqrt}(2), \beta/\text{sqrt}(2), 0, 0, \beta/\text{sqrt}(2)]]$

$\langle proof \rangle$

lemma *alice-step-2*:

assumes state 1 φ and $\alpha = \varphi$ $\$ \$ (0,0)$ and $\beta = \varphi$ $\$ \$ (1,0)$

shows $(\text{CNOT} \otimes \text{Id } 1) * (\varphi \otimes |\beta_{00}\rangle) = \text{mat-of-cols-list } 8 \ [[\alpha/\text{sqrt}(2), 0, 0, \alpha/\text{sqrt}(2), 0, \beta/\text{sqrt}(2), \beta/\text{sqrt}(2),$

$\langle proof \rangle$

lemma *alice-result*:

assumes state 1 φ and $\alpha = \varphi$ $\$ \$ (0,0)$ and $\beta = \varphi$ $\$ \$ (1,0)$

shows alice $\varphi = \text{mat-of-cols-list } 8 \ [[\alpha/2, \beta/2, \beta/2, \alpha/2, \alpha/2, -\beta/2, -\beta/2,$

$\langle proof \rangle$

An application of function *alice* to a state φ of a 1-qubit system results in the following cases.

definition *alice-meas*:: complex Matrix.mat \Rightarrow -list **where**

alice-meas $\varphi = [$

$((\text{prob0 } 3 \text{ (alice } \varphi) 0) * (\text{prob0 } 3 \text{ (post-meas0 } 3 \text{ (alice } \varphi) 0) 1), \text{post-meas0 } 3$

$(\text{post-meas0 } 3 \text{ (alice } \varphi) 0) 1)$

$, ((\text{prob0 } 3 \text{ (alice } \varphi) 0) * (\text{prob1 } 3 \text{ (post-meas0 } 3 \text{ (alice } \varphi) 0) 1), \text{post-meas1 } 3$

$(\text{post-meas0 } 3 \text{ (alice } \varphi) 0) 1)$

$, ((\text{prob1 } 3 \text{ (alice } \varphi) 0) * (\text{prob0 } 3 \text{ (post-meas1 } 3 \text{ (alice } \varphi) 0) 1), \text{post-meas0 } 3$

$(\text{post-meas1 } 3 \text{ (alice } \varphi) 0) 1)$

$, ((\text{prob1 } 3 \text{ (alice } \varphi) 0) * (\text{prob1 } 3 \text{ (post-meas1 } 3 \text{ (alice } \varphi) 0) 1), \text{post-meas1 } 3$

$(\text{post-meas1 } 3 \text{ (alice } \varphi) 0) 1)$

$]$

definition *alice-pos*:: complex Matrix.mat \Rightarrow complex Matrix.mat \Rightarrow bool **where**

alice-pos φ $q \equiv q = \text{mat-of-cols-list } 8 \ [[\varphi \ \$ \$ (0,0), \varphi \ \$ \$ (1,0), 0, 0, 0, 0, 0, 0]]$

\vee

$q = \text{mat-of-cols-list } 8 \ [[0, 0, \varphi \ \$ \$ (1,0), \varphi \ \$ \$ (0,0), 0, 0, 0, 0]] \vee$

$q = \text{mat-of-cols-list } 8 \ [[0, 0, 0, 0, \varphi \ \$ \$ (0,0), -\varphi \ \$ \$ (1,0), 0, 0]]$

\vee

$q = \text{mat-of-cols-list } 8 \ [[0, 0, 0, 0, 0, 0, -\varphi \ \$ \$ (1,0), \varphi \ \$ \$ (0,0)]]$

lemma *phi-vec-length*:

assumes state 1 φ

shows $\text{cmod}(\varphi \ \$ \$ (0,0))^2 + \text{cmod}(\varphi \ \$ \$ (\text{Suc } 0,0))^2 = 1$

$\langle proof \rangle$

lemma *select-index-3-subsets* [simp]:

shows $\{j::\text{nat. } \text{select-index } 3 \ 0 \ j\} = \{4,5,6,7\} \wedge$
 $\{j::\text{nat. } j < 8 \wedge \neg \text{select-index } 3 \ 0 \ j\} = \{0,1,2,3\} \wedge$
 $\{j::\text{nat. } \text{select-index } 3 \ 1 \ j\} = \{2,3,6,7\} \wedge$
 $\{j::\text{nat. } j < 8 \wedge \neg \text{select-index } 3 \ 1 \ j\} = \{0,1,4,5\}$

<proof>

lemma *prob-index-0-alice*:

assumes *state 1* φ
shows $\text{prob0 } 3 \ (\text{alice } \varphi) \ 0 = 1/2 \wedge \text{prob1 } 3 \ (\text{alice } \varphi) \ 0 = 1/2$

<proof>

lemma *post-meas0-index-0-alice*:

assumes *state 1* φ **and** $\alpha = \varphi \ \$\$ \ (0,0)$ **and** $\beta = \varphi \ \$\$ \ (1,0)$

shows $\text{post-meas0 } 3 \ (\text{alice } \varphi) \ 0 =$

$\text{mat-of-cols-list } 8 \ [[\alpha/\text{sqrt}(2), \beta/\text{sqrt}(2), \beta/\text{sqrt}(2), \alpha/\text{sqrt}(2), 0, 0, 0, 0]]$

<proof>

lemma *post-meas1-index-0-alice*:

assumes *state 1* φ **and** $\alpha = \varphi \ \$\$ \ (0,0)$ **and** $\beta = \varphi \ \$\$ \ (1,0)$

shows $\text{post-meas1 } 3 \ (\text{alice } \varphi) \ 0 = \text{mat-of-cols-list } 8 \ [[0,0,0,0,\alpha/\text{sqrt}(2),-\beta/\text{sqrt}(2),-\beta/\text{sqrt}(2),\alpha/\text{sqrt}(2)]]$

<proof>

lemma *post-meas0-index-0-alice-state* [simp]:

assumes *state 1* φ

shows $\text{state } 3 \ (\text{post-meas0 } 3 \ (\text{alice } \varphi) \ 0)$

<proof>

lemma *post-meas1-index-0-alice-state* [simp]:

assumes *state 1* φ

shows $\text{state } 3 \ (\text{post-meas1 } 3 \ (\text{alice } \varphi) \ 0)$

<proof>

lemma *Alice-case* [simp]:

assumes *state 1* φ **and** *state 3* q **and** $\text{List.member } (\text{alice-meas } \varphi) \ (p, q)$

shows $\text{alice-pos } \varphi \ q$

<proof>

datatype *bit* = *zero* | *one*

definition *alice-out*:: $\text{complex Matrix.mat} \Rightarrow \text{complex Matrix.mat} \Rightarrow \text{bit} \times \text{bit}$

where

$\text{alice-out } \varphi \ q \equiv$

$\text{if } q = \text{mat-of-cols-list } 8 \ [[\varphi \ \$\$ \ (0,0), \varphi \ \$\$ \ (1,0), 0, 0, 0, 0, 0, 0]] \ \text{then } (\text{zero}, \text{zero}) \ \text{else}$

$\text{if } q = \text{mat-of-cols-list } 8 \ [[0, 0, \varphi \ \$\$ \ (1,0), \varphi \ \$\$ \ (0,0), 0, 0, 0, 0]] \ \text{then } (\text{zero}, \text{one}) \ \text{else}$

if $q = \text{mat-of-cols-list } 8 \ [[0, 0, 0, 0, \varphi \ \$\$ (0,0), -\varphi \ \$\$ (1,0), 0, 0]]$ then
 (one, zero) else
 if $q = \text{mat-of-cols-list } 8 \ [[0, 0, 0, 0, 0, 0, -\varphi \ \$\$ (1,0), \varphi \ \$\$ (0,0)]]$ then
 (one, one) else
 undefined

definition $\text{bob}:: \text{complex Matrix.mat} \Rightarrow \text{bit} \times \text{bit} \Rightarrow \text{complex Matrix.mat}$ **where**
 $\text{bob } q \ b \equiv$

if $(\text{fst } b, \text{snd } b) = (\text{zero}, \text{zero})$ then q else
 if $(\text{fst } b, \text{snd } b) = (\text{zero}, \text{one})$ then $(\text{Id } 2 \otimes X) * q$ else
 if $(\text{fst } b, \text{snd } b) = (\text{one}, \text{zero})$ then $(\text{Id } 2 \otimes Z) * q$ else
 if $(\text{fst } b, \text{snd } b) = (\text{one}, \text{one})$ then $(\text{Id } 2 \otimes Z * X) * q$ else
 undefined

lemma alice-out-unique [simp]:

assumes state 1 φ

shows $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[0, 0, \varphi \ \$\$ (\text{Suc } 0, 0), \varphi \ \$\$ (0, 0), 0, 0,$

$0, 0]]!j!i) \neq$
 $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[\varphi \ \$\$ (0, 0), \varphi \ \$\$ (\text{Suc } 0, 0), 0, 0, 0, 0,$

$0, 0]]!j!i) \wedge$
 $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[0, 0, 0, 0, \varphi \ \$\$ (0, 0), -\varphi \ \$\$ (\text{Suc } 0, 0),$

$0, 0]]!j!i) \neq$
 $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[\varphi \ \$\$ (0, 0), \varphi \ \$\$ (\text{Suc } 0, 0), 0, 0, 0, 0,$

$0, 0]]!j!i) \wedge$
 $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[0, 0, 0, 0, 0, 0, -\varphi \ \$\$ (\text{Suc } 0, 0), \varphi \ \$\$$

$(0, 0)]!j!i) \neq$
 $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[\varphi \ \$\$ (0, 0), \varphi \ \$\$ (\text{Suc } 0, 0), 0, 0, 0, 0,$

$0, 0]]!j!i) \wedge$
 $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[0, 0, 0, 0, \varphi \ \$\$ (0, 0), -\varphi \ \$\$ (\text{Suc } 0, 0),$

$0, 0]]!j!i) \neq$
 $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[0, 0, \varphi \ \$\$ (\text{Suc } 0, 0), \varphi \ \$\$ (0, 0), 0, 0,$

$0, 0]]!j!i) \wedge$
 $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[0, 0, 0, 0, 0, 0, -\varphi \ \$\$ (\text{Suc } 0, 0), \varphi \ \$\$$

$(0, 0)]!j!i) \neq$
 $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[0, 0, \varphi \ \$\$ (\text{Suc } 0, 0), \varphi \ \$\$ (0, 0), 0, 0,$

$0, 0]]!j!i) \wedge$
 $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[0, 0, 0, 0, 0, 0, -\varphi \ \$\$ (\text{Suc } 0, 0), \varphi \ \$\$$

$(0, 0)]!j!i) \neq$
 $\text{Matrix.mat } 8 \ (\text{Suc } 0) \ (\lambda(i,j). \ [[0, 0, 0, 0, \varphi \ \$\$ (0, 0), -\varphi \ \$\$ (\text{Suc } 0, 0),$

$0, 0]]!j!i)$
 <proof>

abbreviation $M3:: \text{complex Matrix.mat}$ **where**

$M3 \equiv \text{mat-of-cols-list } 8 \ [[0, 1, 0, 0, 0, 0, 0, 0],$

$[1, 0, 0, 0, 0, 0, 0, 0],$

$[0, 0, 0, 1, 0, 0, 0, 0],$

$[0, 0, 1, 0, 0, 0, 0, 0],$

$[0, 0, 0, 0, 0, 1, 0, 0],$

$[0, 0, 0, 0, 1, 0, 0, 0],$

$$\begin{aligned} & [0, 0, 0, 0, 0, 0, 0, 1], \\ & [0, 0, 0, 0, 0, 0, 1, 0] \end{aligned}$$

lemma *tensor-prod-of-id-2-x*:

shows $(Id\ 2 \otimes X) = M3$

<proof>

abbreviation $M4$:: *complex Matrix.mat* **where**

$M4 \equiv mat-of-cols-list\ 8\ [[0, -1, 0, 0, 0, 0, 0, 0],$

$$\begin{aligned} & [1, 0, 0, 0, 0, 0, 0, 0], \\ & [0, 0, 0, -1, 0, 0, 0, 0], \\ & [0, 0, 1, 0, 0, 0, 0, 0], \\ & [0, 0, 0, 0, 0, -1, 0, 0], \\ & [0, 0, 0, 0, 1, 0, 0, 0], \\ & [0, 0, 0, 0, 0, 0, 0, -1], \\ & [0, 0, 0, 0, 0, 0, 1, 0] \end{aligned}$$

abbreviation ZX :: *complex Matrix.mat* **where**

$ZX \equiv mat-of-cols-list\ 2\ [[0, -1], [1, 0]]$

lemma *l-inv-of-ZX*:

shows $ZX^\dagger * ZX = 1_m\ 2$

<proof>

lemma *r-inv-of-ZX*:

shows $ZX * (ZX^\dagger) = 1_m\ 2$

<proof>

lemma *ZX-is-gate* [*simp*]:

shows *gate 1 ZX*

<proof>

lemma *prod-of-ZX*:

shows $Z * X = ZX$

<proof>

lemma *tensor-prod-of-id-2-y*:

shows $(Id\ 2 \otimes Z * X) = M4$

<proof>

abbreviation $M5$:: *complex Matrix.mat* **where**

$M5 \equiv mat-of-cols-list\ 8\ [[1, 0, 0, 0, 0, 0, 0, 0],$

$$\begin{aligned} & [0, -1, 0, 0, 0, 0, 0, 0], \\ & [0, 0, 1, 0, 0, 0, 0, 0], \\ & [0, 0, 0, -1, 0, 0, 0, 0], \\ & [0, 0, 0, 0, 1, 0, 0, 0], \\ & [0, 0, 0, 0, 0, -1, 0, 0], \\ & [0, 0, 0, 0, 0, 0, 1, 0], \\ & [0, 0, 0, 0, 0, 0, 0, -1] \end{aligned}$$

lemma *tensor-prod-of-id-2-z*:

shows $(Id\ 2 \otimes Z) = M5$
<proof>

lemma *teleportation*:

assumes *state 1* φ **and** *state 3* q **and** *List.member (alice-meas φ) (p, q)*
shows $\exists r. \text{state } 2\ r \wedge \text{bob } q\ (\text{alice-out } \varphi\ q) = r \otimes \varphi$
<proof>

end

10 The Deutsch Algorithm

theory *Deutsch*

imports

More-Tensor

Measurement

begin

Given a function $f : 0, 1 \mapsto 0, 1$, Deutsch's algorithm decides if this function is constant or balanced with a single $f(x)$ circuit to evaluate the function for multiple values of x simultaneously. The algorithm makes use of quantum parallelism and quantum interference.

A constant function with values in $0, 1$ returns either always 0 or always 1. A balanced function is 0 for half of the inputs and 1 for the other half.

locale *deutsch* =

fixes $f :: \text{nat} \Rightarrow \text{nat}$

assumes $\text{dom}: f \in (\{0, 1\} \rightarrow_E \{0, 1\})$

context *deutsch*

begin

definition *is-swap* :: *bool* **where**

$\text{is-swap} = (\forall x \in \{0, 1\}. f\ x = 1 - x)$

lemma *is-swap-values*:

assumes *is-swap*

shows $f\ 0 = 1$ **and** $f\ 1 = 0$

<proof>

lemma *is-swap-sum-mod-2*:

assumes *is-swap*

shows $(f\ 0 + f\ 1) \bmod 2 = 1$

<proof>

definition *const*:: *nat* \Rightarrow *bool* **where**

const *n* = $(\forall x \in \{0,1\}.(f\ x = n))$

definition *is-const*:: *bool* **where**

is-const \equiv *const* 0 \vee *const* 1

definition *is-balanced*:: *bool* **where**

is-balanced \equiv $(\forall x \in \{0,1\}.(f\ x = x)) \vee$ *is-swap*

lemma *f-values*: $(f\ 0 = 0 \vee f\ 0 = 1) \wedge (f\ 1 = 0 \vee f\ 1 = 1)$

<proof>

lemma *f-cases*:

shows *is-const* \vee *is-balanced*

<proof>

lemma *const-0-sum-mod-2*:

assumes *const* 0

shows $(f\ 0 + f\ 1) \bmod 2 = 0$

<proof>

lemma *const-1-sum-mod-2*:

assumes *const* 1

shows $(f\ 0 + f\ 1) \bmod 2 = 0$

<proof>

lemma *is-const-sum-mod-2*:

assumes *is-const*

shows $(f\ 0 + f\ 1) \bmod 2 = 0$

<proof>

lemma *id-sum-mod-2*:

assumes *f* = *id*

shows $(f\ 0 + f\ 1) \bmod 2 = 1$

<proof>

lemma *is-balanced-sum-mod-2*:

assumes *is-balanced*

shows $(f\ 0 + f\ 1) \bmod 2 = 1$

<proof>

lemma *f-ge-0*: $\forall x. (f\ x \geq 0)$ *<proof>*

end

The Deutsch's Transform U_f .

definition (**in deutsch**) *deutsch-transform*:: *complex Matrix.mat* ($\langle U_f \rangle$) **where**

$U_f \equiv$ *mat-of-cols-list* 4 $[[1 - f(0), f(0), 0, 0],$

$$\begin{aligned} &[f(0), 1 - f(0), 0, 0], \\ &[0, 0, 1 - f(1), f(1)], \\ &[0, 0, f(1), 1 - f(1)] \end{aligned}$$

lemma (in *deutsch*) *deutsch-transform-dim* [simp]:
 shows $\dim\text{-row } U_f = 4$ and $\dim\text{-col } U_f = 4$
 ⟨proof⟩

lemma (in *deutsch*) *deutsch-transform-coeff-is-zero* [simp]:
 shows $U_f \$(0,2) = 0$ and $U_f \$(0,3) = 0$
 and $U_f \$(1,2) = 0$ and $U_f \$(1,3) = 0$
 and $U_f \$(2,0) = 0$ and $U_f \$(2,1) = 0$
 and $U_f \$(3,0) = 0$ and $U_f \$(3,1) = 0$
 ⟨proof⟩

lemma (in *deutsch*) *deutsch-transform-coeff* [simp]:
 shows $U_f \$(0,1) = f(0)$ and $U_f \$(1,0) = f(0)$
 and $U_f \$(2,3) = f(1)$ and $U_f \$(3,2) = f(1)$
 and $U_f \$(0,0) = 1 - f(0)$ and $U_f \$(1,1) = 1 - f(0)$
 and $U_f \$(2,2) = 1 - f(1)$ and $U_f \$(3,3) = 1 - f(1)$
 ⟨proof⟩

abbreviation (in *deutsch*) V_f :: *complex Matrix.mat* where
 $V_f \equiv \text{Matrix.mat } 4 \ 4 \ (\lambda(i,j).$
 if $i=0 \wedge j=0$ then $1 - f(0)$ else
 (if $i=0 \wedge j=1$ then $f(0)$ else
 (if $i=1 \wedge j=0$ then $f(0)$ else
 (if $i=1 \wedge j=1$ then $1 - f(0)$ else
 (if $i=2 \wedge j=2$ then $1 - f(1)$ else
 (if $i=2 \wedge j=3$ then $f(1)$ else
 (if $i=3 \wedge j=2$ then $f(1)$ else
 (if $i=3 \wedge j=3$ then $1 - f(1)$ else 0))))))

lemma (in *deutsch*) *deutsch-transform-alt-rep-coeff-is-zero* [simp]:
 shows $V_f \$(0,2) = 0$ and $V_f \$(0,3) = 0$
 and $V_f \$(1,2) = 0$ and $V_f \$(1,3) = 0$
 and $V_f \$(2,0) = 0$ and $V_f \$(2,1) = 0$
 and $V_f \$(3,0) = 0$ and $V_f \$(3,1) = 0$
 ⟨proof⟩

lemma (in *deutsch*) *deutsch-transform-alt-rep-coeff* [simp]:
 shows $V_f \$(0,1) = f(0)$ and $V_f \$(1,0) = f(0)$
 and $V_f \$(2,3) = f(1)$ and $V_f \$(3,2) = f(1)$
 and $V_f \$(0,0) = 1 - f(0)$ and $V_f \$(1,1) = 1 - f(0)$
 and $V_f \$(2,2) = 1 - f(1)$ and $V_f \$(3,3) = 1 - f(1)$
 ⟨proof⟩

lemma (in *deutsch*) *deutsch-transform-alt-rep*:
 shows $U_f = V_f$

<proof>

U_f is a gate.

lemma (*in deutsch*) *transpose-of-deutsch-transform:*

shows $(U_f)^t = U_f$

<proof>

lemma (*in deutsch*) *adjoint-of-deutsch-transform:*

shows $(U_f)^\dagger = U_f$

<proof>

lemma (*in deutsch*) *deutsch-transform-is-gate:*

shows *gate* 2 U_f

<proof>

Two qubits are prepared. The first one in the state $|0\rangle$, the second one in the state $|1\rangle$.

abbreviation *zero* **where** *zero* \equiv *unit-vec* 2 0

abbreviation *one* **where** *one* \equiv *unit-vec* 2 1

lemma *ket-zero-is-state:*

shows *state* 1 $|zero\rangle$

<proof>

lemma *ket-one-is-state:*

shows *state* 1 $|one\rangle$

<proof>

lemma *ket-zero-to-mat-of-cols-list [simp]:* $|zero\rangle = \text{mat-of-cols-list } 2 \ [[1, 0]]$

<proof>

lemma *ket-one-to-mat-of-cols-list [simp]:* $|one\rangle = \text{mat-of-cols-list } 2 \ [[0, 1]]$

<proof>

Applying the Hadamard gate to the state $|0\rangle$ results in the new state $\psi_{00} = \frac{(|0\rangle + |1\rangle)}{\sqrt{2}}$

abbreviation ψ_{00} **where** *complex Matrix.mat* **where**

$\psi_{00} \equiv \text{mat-of-cols-list } 2 \ [[1/\text{sqrt}(2), 1/\text{sqrt}(2)]]$

lemma *H-on-ket-zero:*

shows $(H * |zero\rangle) = \psi_{00}$

<proof>

lemma *H-on-ket-zero-is-state:*

shows *state* 1 $(H * |zero\rangle)$

<proof>

Applying the Hadamard gate to the state $|0\rangle$ results in the new state $\psi_{01} =$

$$\frac{(|0\rangle - |1\rangle)}{\sqrt{2}}.$$

abbreviation ψ_{01} :: *complex Matrix.mat* **where**
 $\psi_{01} \equiv \text{mat-of-cols-list } 2 \text{ } [[1/\text{sqrt}(2), -1/\text{sqrt}(2)]]$

lemma *H-on-ket-one*:
shows $(H * |one\rangle) = \psi_{01}$
 $\langle \text{proof} \rangle$

lemma *H-on-ket-one-is-state*:
shows *state 1* $(H * |one\rangle)$
 $\langle \text{proof} \rangle$

Then, the state $\psi_1 = \frac{(|00\rangle - |01\rangle + |10\rangle - |11\rangle)}{2}$ is obtained by taking the tensor product of the states $\psi_{00} = \frac{(|0\rangle + |1\rangle)}{\sqrt{2}}$ and $\psi_{01} = \frac{(|0\rangle - |1\rangle)}{\sqrt{2}}$.

abbreviation ψ_1 :: *complex Matrix.mat* **where**
 $\psi_1 \equiv \text{mat-of-cols-list } 4 \text{ } [[1/2, -1/2, 1/2, -1/2]]$

lemma ψ_0 -to- ψ_1 :
shows $(\psi_{00} \otimes \psi_{01}) = \psi_1$
 $\langle \text{proof} \rangle$

lemma ψ_1 -is-state:
shows *state 2* ψ_1
 $\langle \text{proof} \rangle$

Next, the gate U_f is applied to the state $\psi_1 = \frac{(|00\rangle - |01\rangle + |10\rangle - |11\rangle)}{2}$ and $\psi_2 = \frac{(|0f(0) \oplus 0\rangle - |0f(0) \oplus 1\rangle + |1f(1) \oplus 0\rangle - |1f(1) \oplus 1\rangle)}{2}$ is obtained.

This simplifies to $\psi_2 = \frac{(|0f(0)\rangle - |0\overline{f(0)}\rangle + |1f(1)\rangle - |1\overline{f(1)}\rangle)}{2}$

abbreviation (*in deutsch*) ψ_2 :: *complex Matrix.mat* **where**
 $\psi_2 \equiv \text{mat-of-cols-list } 4 \text{ } [[(1 - f(0))/2 - f(0)/2,$
 $f(0)/2 - (1 - f(0))/2,$
 $(1 - f(1))/2 - f(1)/2,$
 $f(1)/2 - (1 - f(1))/2]]$

lemma (*in deutsch*) ψ_1 -to- ψ_2 :
shows $U_f * \psi_1 = \psi_2$
 $\langle \text{proof} \rangle$

lemma (*in deutsch*) ψ_2 -is-state:
shows *state 2* ψ_2
 $\langle \text{proof} \rangle$

lemma *H-tensor-Id-1*:

defines $d:v \equiv \text{mat-of-cols-list } 4 \text{ } [[1/\text{sqrt}(2), 0, 1/\text{sqrt}(2), 0],$
 $[0, 1/\text{sqrt}(2), 0, 1/\text{sqrt}(2)],$
 $[1/\text{sqrt}(2), 0, -1/\text{sqrt}(2), 0],$
 $[0, 1/\text{sqrt}(2), 0, -1/\text{sqrt}(2)]]$

shows $(H \otimes Id\ 1) = v$
 $\langle \text{proof} \rangle$

lemma *H-tensor-Id-1-is-gate:*

shows $\text{gate } 2 \text{ } (H \otimes Id\ 1)$
 $\langle \text{proof} \rangle$

Applying the Hadamard gate to the first qubit of ψ_2 results in $\psi_3 = \pm|f(0) \oplus$
 $f(1)\rangle \left[\frac{(|0\rangle - |1\rangle)}{\sqrt{2}} \right]$

abbreviation (in *deutsch*) $\psi_3:: \text{complex Matrix.mat}$ **where**

$\psi_3 \equiv \text{mat-of-cols-list } 4$
 $[[(1-f(0))/(2*\text{sqrt}(2)) - f(0)/(2*\text{sqrt}(2)) + (1-f(1))/(2*\text{sqrt}(2)) - f(1)/(2*\text{sqrt}(2)),$
 $f(0)/(2*\text{sqrt}(2)) - (1-f(0))/(2*\text{sqrt}(2)) + (f(1)/(2*\text{sqrt}(2)) - (1-f(1))/(2*\text{sqrt}(2))),$
 $(1-f(0))/(2*\text{sqrt}(2)) - f(0)/(2*\text{sqrt}(2)) - (1-f(1))/(2*\text{sqrt}(2)) + f(1)/(2*\text{sqrt}(2)),$
 $f(0)/(2*\text{sqrt}(2)) - (1-f(0))/(2*\text{sqrt}(2)) - f(1)/(2*\text{sqrt}(2)) + (1-f(1))/(2*\text{sqrt}(2))]]$

lemma (in *deutsch*) $\psi_2\text{-to-}\psi_3:$

shows $(H \otimes Id\ 1) * \psi_2 = \psi_3$
 $\langle \text{proof} \rangle$

lemma (in *deutsch*) $\psi_3\text{-is-state:}$

shows $\text{state } 2 \text{ } \psi_3$
 $\langle \text{proof} \rangle$

Finally, all steps are put together. The result depends on the function f .
 If f is constant the first qubit of $\pm|f(0) \oplus f(1)\rangle \left[\frac{(|0\rangle - |1\rangle)}{\sqrt{2}} \right]$ is 0, if it is
 is_balanced it is 1. The algorithm only uses one evaluation of $f(x)$ and will
 always succeed.

definition (in *deutsch*) $\text{deutsch-algo}:: \text{complex Matrix.mat}$ **where**

$\text{deutsch-algo} \equiv (H \otimes Id\ 1) * (U_f * ((H * |zero\rangle) \otimes (H * |one\rangle)))$

lemma (in *deutsch*) $\text{deutsch-algo-result}$ [simp]:

shows $\text{deutsch-algo} = \psi_3$
 $\langle \text{proof} \rangle$

lemma (in *deutsch*) $\text{deutsch-algo-result-is-state:}$

shows $\text{state } 2 \text{ } \text{deutsch-algo}$
 $\langle \text{proof} \rangle$

If the function is constant then the measurement of the first qubit should
 result in the state $|0\rangle$ with probability 1.

lemma (in *deutsch*) $\text{prob0-deutsch-algo-const:}$

assumes *is-const*
shows *prob0 2 deutsch-algo 0 = 1*
 ⟨*proof*⟩

lemma (in *deutsch*) *prob1-deutsch-algo-const*:
assumes *is-const*
shows *prob1 2 deutsch-algo 0 = 0*
 ⟨*proof*⟩

If the function is balanced the measurement of the first qubit should result in the state $|1\rangle$ with probability 1.

lemma (in *deutsch*) *prob0-deutsch-algo-balanced*:
assumes *is-balanced*
shows *prob0 2 deutsch-algo 0 = 0*
 ⟨*proof*⟩

lemma (in *deutsch*) *prob1-deutsch-algo-balanced*:
assumes *is-balanced*
shows *prob1 2 deutsch-algo 0 = 1*
 ⟨*proof*⟩

Eventually, the measurement of the first qubit results in $f(0) \oplus f(1)$.

definition (in *deutsch*) *deutsch-algo-eval*:: *real* **where**
deutsch-algo-eval \equiv *prob1 2 deutsch-algo 0*

lemma (in *deutsch*) *sum-mod-2-cases*:
shows $(f\ 0 + f\ 1) \bmod 2 = 0 \longrightarrow$ *is-const*
and $(f\ 0 + f\ 1) \bmod 2 = 1 \longrightarrow$ *is-balanced*
 ⟨*proof*⟩

lemma (in *deutsch*) *deutsch-algo-eval-is-sum-mod-2*:
shows *deutsch-algo-eval* $= (f\ 0 + f\ 1) \bmod 2$
 ⟨*proof*⟩

If the algorithm returns 0 then one concludes that the input function is constant and if it returns 1 then the function is balanced.

theorem (in *deutsch*) *deutsch-algo-is-correct*:
shows *deutsch-algo-eval* $= 0 \longrightarrow$ *is-const* **and** *deutsch-algo-eval* $= 1 \longrightarrow$ *is-balanced*
 ⟨*proof*⟩

end

11 The Deutsch-Jozsa Algorithm

theory *Deutsch-Jozsa*
imports
Deutsch
More-Tensor

Binary-Nat

begin

Given a function $f : 0, 1^n \mapsto 0, 1$, the Deutsch-Jozsa algorithm decides if this function is constant or balanced with a single $f(x)$ circuit to evaluate the function for multiple values of x simultaneously. The algorithm makes use of quantum parallelism and quantum interference.

A constant function with values in $0, 1$ returns either always 0 or always 1. A balanced function is 0 for half of the inputs and 1 for the other half.

locale *bob-fun* =

fixes $f :: \text{nat} \Rightarrow \text{nat}$ **and** $n :: \text{nat}$

assumes $\text{dom}: f \in (\{i :: \text{nat}. i < 2^n\} \rightarrow_E \{0, 1\})$

assumes $\text{dim}: n \geq 1$

context *bob-fun*

begin

definition *const* :: $\text{nat} \Rightarrow \text{bool}$ **where**

$\text{const } c = (\forall x \in \{i :: \text{nat}. i < 2^n\}. f x = c)$

definition *is-const* :: bool **where**

$\text{is-const} \equiv \text{const } 0 \vee \text{const } 1$

definition *is-balanced* :: bool **where**

$\text{is-balanced} \equiv \exists A B :: \text{nat set}. A \subseteq \{i :: \text{nat}. i < 2^n\} \wedge B \subseteq \{i :: \text{nat}. i < 2^n\}$
 $\wedge \text{card } A = 2^{n-1} \wedge \text{card } B = 2^{n-1}$
 $\wedge (\forall x \in A. f x = 0) \wedge (\forall x \in B. f x = 1)$

lemma *is-balanced-inter*:

fixes $A B :: \text{nat set}$

assumes $\forall x \in A. f x = 0$ **and** $\forall x \in B. f x = 1$

shows $A \cap B = \{\}$

<proof>

lemma *is-balanced-union*:

fixes $A B :: \text{nat set}$

assumes $A \subseteq \{i :: \text{nat}. i < 2^n\}$ **and** $B \subseteq \{i :: \text{nat}. i < 2^n\}$

and $\text{card } A = 2^{n-1}$ **and** $\text{card } B = 2^{n-1}$

and $A \cap B = \{\}$

shows $A \cup B = \{i :: \text{nat}. i < 2^n\}$

<proof>

lemma *f-ge-0*: $\forall x. f x \geq 0$ *<proof>*

lemma *f-dom-not-zero*:

shows $f \in (\{i :: \text{nat}. n \geq 1 \wedge i < 2^n\} \rightarrow_E \{0, 1\})$

<proof>

lemma *f-values*: $\forall x \in \{(i::\text{nat}). i < 2^n\}. f x = 0 \vee f x = 1$
 ⟨proof⟩

end

The input function has to be constant or balanced.

locale *jozsa* = *bob-fun* +
assumes *const-or-balanced*: *is-const* \vee *is-balanced*

Introduce two customised rules: disjunctions with four disjuncts and induction starting from one instead of zero.

lemma *disj-four-cases*:
assumes $A \vee B \vee C \vee D$ **and** $A \implies P$ **and** $B \implies P$ **and** $C \implies P$ **and** $D \implies P$
shows P
 ⟨proof⟩

The unitary transform U_f .

definition (in *jozsa*) *jozsa-transform*:: *complex Matrix.mat* ($\langle U_f \rangle$) **where**
 $U_f \equiv \text{Matrix.mat } (2^{n+1}) (2^{n+1}) (\lambda(i,j).$
 if $i = j$ *then* $(1 - f(i \text{ div } 2))$ *else*
 if $i = j + 1 \wedge \text{odd } i$ *then* $f(i \text{ div } 2)$ *else*
 if $i = j - 1 \wedge \text{even } i \wedge j \geq 1$ *then* $f(i \text{ div } 2)$ *else* 0)

lemma (in *jozsa*) *jozsa-transform-dim* [*simp*]:
shows *dim-row* $U_f = 2^{n+1}$ **and** *dim-col* $U_f = 2^{n+1}$
 ⟨proof⟩

lemma (in *jozsa*) *jozsa-transform-coeff-is-zero* [*simp*]:
assumes $i < \text{dim-row } U_f \wedge j < \text{dim-col } U_f$
shows $(i \neq j \wedge \neg(i=j+1 \wedge \text{odd } i) \wedge \neg(i=j-1 \wedge \text{even } i \wedge j \geq 1)) \longrightarrow U_f \text{ \$(\$ (i,j) = 0$
 ⟨proof⟩

lemma (in *jozsa*) *jozsa-transform-coeff* [*simp*]:
assumes $i < \text{dim-row } U_f \wedge j < \text{dim-col } U_f$
shows $i = j \longrightarrow U_f \text{ \$(\$ (i,j) = } 1 - f(i \text{ div } 2)$
and $i = j + 1 \wedge \text{odd } i \longrightarrow U_f \text{ \$(\$ (i,j) = } f(i \text{ div } 2)$
and $j \geq 1 \wedge i = j - 1 \wedge \text{even } i \longrightarrow U_f \text{ \$(\$ (i,j) = } f(i \text{ div } 2)$
 ⟨proof⟩

lemma (in *jozsa*) *U_f-mult-without-empty-summands-sum-even*:
fixes $i j A$
assumes $i < \text{dim-row } U_f$ **and** $j < \text{dim-col } A$ **and** *even* i **and** $\text{dim-col } U_f = \text{dim-row } A$
shows $(\sum k \in \{0.. < \text{dim-row } A\}. U_f \text{ \$(\$ (i,k) * } A \text{ \$(\$ (k,j) = } (\sum k \in \{i, i+1\}. U_f \text{ \$(\$ (i,k) * } A \text{ \$(\$ (k,j))$
 ⟨proof⟩

lemma (in *jozsa*) *U_f-mult-without-empty-summands-even*:

fixes $i\ j\ A$

assumes $i < \dim\text{-row } U_f$ **and** $j < \dim\text{-col } A$ **and** *even* i **and** $\dim\text{-col } U_f = \dim\text{-row } A$

shows $(U_f * A) \text{ \$(\$ (i,j)) } = (\sum k \in \{i, i+1\}. U_f \text{ \$(\$ (i,k)) } * A \text{ \$(\$ (k,j)) }$

<proof>

lemma (in *jozsa*) *U_f-mult-without-empty-summands-sum-odd*:

fixes $i\ j\ A$

assumes $i < \dim\text{-row } U_f$ **and** $j < \dim\text{-col } A$ **and** *odd* i **and** $\dim\text{-col } U_f = \dim\text{-row } A$

shows $(\sum k \in \{0.. < \dim\text{-row } A\}. U_f \text{ \$(\$ (i,k)) } * A \text{ \$(\$ (k,j)) }) = (\sum k \in \{i-1, i\}. U_f \text{ \$(\$ (i,k)) } * A \text{ \$(\$ (k,j)) }$

<proof>

lemma (in *jozsa*) *U_f-mult-without-empty-summands-odd*:

fixes $i\ j\ A$

assumes $i < \dim\text{-row } U_f$ **and** $j < \dim\text{-col } A$ **and** *odd* i **and** $\dim\text{-col } U_f = \dim\text{-row } A$

shows $(U_f * A) \text{ \$(\$ (i,j)) } = (\sum k \in \{i-1, i\}. U_f \text{ \$(\$ (i,k)) } * A \text{ \$(\$ (k,j)) }$

<proof>

U_f is a gate.

lemma (in *jozsa*) *transpose-of-jozsa-transform*:

shows $(U_f)^t = U_f$

<proof>

lemma (in *jozsa*) *adjoint-of-jozsa-transform*:

shows $(U_f)^\dagger = U_f$

<proof>

lemma (in *jozsa*) *jozsa-transform-is-unitary-index-even*:

fixes $i\ j:: \text{nat}$

assumes $i < \dim\text{-row } U_f$ **and** $j < \dim\text{-col } U_f$ **and** *even* i

shows $(U_f * U_f) \text{ \$(\$ (i,j)) } = 1_m (\dim\text{-col } U_f) \text{ \$(\$ (i,j)) }$

<proof>

lemma (in *jozsa*) *jozsa-transform-is-unitary-index-odd*:

fixes $i\ j:: \text{nat}$

assumes $i < \dim\text{-row } U_f$ **and** $j < \dim\text{-col } U_f$ **and** *odd* i

shows $(U_f * U_f) \text{ \$(\$ (i,j)) } = 1_m (\dim\text{-col } U_f) \text{ \$(\$ (i,j)) }$

<proof>

lemma (in *jozsa*) *jozsa-transform-is-gate*:

shows *gate* $(n+1) U_f$

<proof>

N-fold application of the tensor product

fun *iter-tensor*:: *complex Matrix.mat* \Rightarrow *nat* \Rightarrow *complex Matrix.mat* ($\langle \cdot \otimes \cdot \rangle$ 75)

where

$$A \otimes^{(Suc\ 0)} = A$$
$$| A \otimes^{(Suc\ k)} = A \otimes (A \otimes^k)$$

lemma *one-tensor-is-id* [simp]:

fixes A

shows $A \otimes^1 = A$

<proof>

lemma *iter-tensor-suc*:

fixes n

assumes $n \geq 1$

shows $A \otimes^{(Suc\ n)} = A \otimes (A \otimes^n)$

<proof>

lemma *dim-row-of-iter-tensor* [simp]:

fixes $A\ n$

assumes $n \geq 1$

shows $\dim\text{-row}(A \otimes^n) = (\dim\text{-row}\ A) \wedge^n$

<proof>

lemma *dim-col-of-iter-tensor* [simp]:

fixes $A\ n$

assumes $n \geq 1$

shows $\dim\text{-col}(A \otimes^n) = (\dim\text{-col}\ A) \wedge^n$

<proof>

lemma *iter-tensor-values*:

fixes $A\ n\ i\ j$

assumes $n \geq 1$ **and** $i < \dim\text{-row}\ (A \otimes (A \otimes^n))$ **and** $j < \dim\text{-col}\ (A \otimes (A \otimes^n))$

shows $(A \otimes^{(Suc\ n)}) \ \$\$ (i,j) = (A \otimes (A \otimes^n)) \ \$\$ (i,j)$

<proof>

lemma *iter-tensor-mult-distr*:

assumes $n \geq 1$ **and** $\dim\text{-col}\ A = \dim\text{-row}\ B$ **and** $\dim\text{-col}\ A > 0$ **and** $\dim\text{-col}\ B > 0$

shows $(A \otimes^{(Suc\ n)}) * (B \otimes^{(Suc\ n)}) = (A * B) \otimes ((A \otimes^n) * (B \otimes^n))$

<proof>

lemma *index-tensor-mat-with-vec2-row-cond*:

fixes $A\ B:: \text{complex Matrix.mat}$ **and** $i:: \text{nat}$

assumes $i < 2 * (\dim\text{-row}\ B)$ **and** $i \geq \dim\text{-row}\ B$ **and** $\dim\text{-col}\ B > 0$ **and** $\dim\text{-row}\ A = 2$ **and** $\dim\text{-col}\ A = 1$

shows $(A \otimes B) \ \$\$ (i,0) = (A \ \$\$ (1,0)) * (B \ \$\$ (i - \dim\text{-row}\ B, 0))$

<proof>

lemma *iter-tensor-of-gate-is-gate*:

fixes $A:: \text{complex Matrix.mat}$ **and** $n\ m:: \text{nat}$

assumes *gate* m A **and** $n \geq 1$
shows *gate* $(m*n)$ $(A \otimes^n)$
 \langle *proof* \rangle

lemma *iter-tensor-of-state-is-state*:
fixes $A::$ *complex Matrix.mat* **and** n $m::$ *nat*
assumes *state* m A **and** $n \geq 1$
shows *state* $(m*n)$ $(A \otimes^n)$
 \langle *proof* \rangle

We prepare $n+1$ qubits. The first n qubits in the state $|0\rangle$, the last one in the state $|1\rangle$.

abbreviation $\psi_{10}::$ *nat* \Rightarrow *complex Matrix.mat* **where**
 ψ_{10} $n \equiv$ *Matrix.mat* $(2^{\wedge}n)$ 1 $(\lambda(i,j). 1/(\text{sqrt } 2)^{\wedge}n)$

lemma *ψ_{10} -values*:
fixes i j n
assumes $i < \text{dim-row } (\psi_{10} n)$ **and** $j < \text{dim-col } (\psi_{10} n)$
shows $(\psi_{10} n) \$\$ (i,j) = 1/(\text{sqrt } 2)^{\wedge}n$
 \langle *proof* \rangle

$H^{\otimes n}$ is applied to $|0\rangle^{\otimes n}$.

lemma *H-on-ket-zero*:
shows $(H * |zero\rangle) = \psi_{10} 1$
 \langle *proof* \rangle

lemma *ψ_{10} -tensor*:
assumes $n \geq 1$
shows $(\psi_{10} 1) \otimes (\psi_{10} n) = (\psi_{10} (\text{Suc } n))$
 \langle *proof* \rangle

lemma *ψ_{10} -tensor-is-state*:
assumes $n \geq 1$
shows *state* n $(|zero\rangle \otimes^n)$
 \langle *proof* \rangle

lemma *iter-tensor-of-H-is-gate*:
assumes $n \geq 1$
shows *gate* n $(H \otimes^n)$
 \langle *proof* \rangle

lemma *iter-tensor-of-H-on-zero-tensor*:
assumes $n \geq 1$
shows $(H \otimes^n) * (|zero\rangle \otimes^n) = \psi_{10} n$
 \langle *proof* \rangle

lemma *ψ_{10} -is-state*:
assumes $n \geq 1$
shows *state* n $(\psi_{10} n)$

<proof>

abbreviation ψ_{11} :: *complex Matrix.mat where*

$\psi_{11} \equiv \text{Matrix.mat } 2 \ 1 \ (\lambda(i,j). \text{ if } i=0 \text{ then } 1/\text{sqrt}(2) \text{ else } -1/\text{sqrt}(2))$

lemma *H-on-ket-one-is- ψ_{11} :*

shows $(H * |one\rangle) = \psi_{11}$

<proof>

abbreviation ψ_1 :: *nat \Rightarrow complex Matrix.mat where*

$\psi_1 \ n \equiv \text{Matrix.mat } (2^{n+1}) \ 1 \ (\lambda(i,j). \text{ if even } i \text{ then } 1/(\text{sqrt } 2)^{n+1} \text{ else } -1/(\text{sqrt } 2)^{n+1})$

lemma ψ_1 -values-even[simp]:

fixes $i \ j \ n$

assumes $i < \text{dim-row } (\psi_1 \ n)$ **and** $j < \text{dim-col } (\psi_1 \ n)$ **and even** i

shows $(\psi_1 \ n) \ \$\$ (i,j) = 1/(\text{sqrt } 2)^{n+1}$

<proof>

lemma ψ_1 -values-odd [simp]:

fixes $i \ j \ n$

assumes $i < \text{dim-row } (\psi_1 \ n)$ **and** $j < \text{dim-col } (\psi_1 \ n)$ **and odd** i

shows $(\psi_1 \ n) \ \$\$ (i,j) = -1/(\text{sqrt } 2)^{n+1}$

<proof>

lemma ψ_{10} -tensor- ψ_{11} -is- ψ_1 :

assumes $n \geq 1$

shows $(\psi_{10} \ n) \ \otimes \ \psi_{11} = \psi_1 \ n$

<proof>

lemma ψ_1 -is-state:

assumes $n \geq 1$

shows $\text{state } (n+1) (\psi_1 \ n)$

<proof>

abbreviation (in *jozsa*) ψ_2 :: *complex Matrix.mat where*

$\psi_2 \equiv \text{Matrix.mat } (2^{n+1}) \ 1 \ (\lambda(i,j). \text{ if even } i \text{ then } (-1)^{f(i \text{ div } 2)}/(\text{sqrt } 2)^{n+1}$

else $(-1)^{f(i \text{ div } 2)+1}/(\text{sqrt } 2)^{n+1})$

lemma (in *jozsa*) ψ_2 -values-even [simp]:

fixes $i \ j$

assumes $i < \text{dim-row } \psi_2$ **and** $j < \text{dim-col } \psi_2$ **and even** i

shows $\psi_2 \ \$\$ (i,j) = (-1)^{f(i \text{ div } 2)}/(\text{sqrt } 2)^{n+1}$

<proof>

lemma (in *jozsa*) ψ_2 -values-odd [simp]:

fixes $i \ j$

assumes $i < \text{dim-row } \psi_2$ **and** $j < \text{dim-col } \psi_2$ **and odd** i

shows $\psi_2 \text{ \#\# } (i,j) = (-1)^{\frown(f(i \text{ div } 2)+1)}/(\text{sqrt } 2)^{\frown(n+1)}$
 ⟨proof⟩

lemma (in *jozsa*) ψ_2 -values-odd-hidden [simp]:
assumes $2*k+1 < \text{dim-row } \psi_2$ **and** $j < \text{dim-col } \psi_2$
shows $\psi_2 \text{ \#\# } (2*k+1,j) = ((-1)^{\frown(f((2*k+1) \text{ div } 2)+1))}/(\text{sqrt } 2)^{\frown(n+1)}$
 ⟨proof⟩

lemma (in *jozsa*) *snd-rep-of- ψ_2* :
assumes $i < \text{dim-row } \psi_2$
shows $((1-f(i \text{ div } 2)) + -f(i \text{ div } 2)) * 1/(\text{sqrt } 2)^{\frown(n+1)} = (-1)^{\frown(f(i \text{ div } 2))}/(\text{sqrt } 2)^{\frown(n+1)}$
and $(-(1-f(i \text{ div } 2))+f(i \text{ div } 2)) * 1/(\text{sqrt } 2)^{\frown(n+1)} = (-1)^{\frown(f(i \text{ div } 2)+1)}/(\text{sqrt } 2)^{\frown(n+1)}$
 ⟨proof⟩

lemma (in *jozsa*) *jozsa-transform-times- ψ_1 -is- ψ_2* :
shows $U_f * (\psi_1 \text{ } n) = \psi_2$
 ⟨proof⟩

lemma (in *jozsa*) ψ_2 -is-state:
shows *state* $(n+1) \psi_2$
 ⟨proof⟩

$H^{\frown}_{\otimes} n$ is the result of taking the n th tensor product of H

abbreviation *iter-tensor-of-H-rep*:: $\text{nat} \Rightarrow \text{complex Matrix.mat } (\langle H^{\frown}_{\otimes} \text{ } \rangle)$ **where**
iter-tensor-of-H-rep $n \equiv \text{Matrix.mat } (2^{\frown}n) (2^{\frown}n) (\lambda(i,j).(-1)^{\frown(i \cdot_n j)}/(\text{sqrt } 2)^{\frown}n)$

lemma *tensor-of-H-values* [simp]:
fixes $n \ i \ j$:: nat
assumes $i < \text{dim-row } (H^{\frown}_{\otimes} n)$ **and** $j < \text{dim-col } (H^{\frown}_{\otimes} n)$
shows $(H^{\frown}_{\otimes} n) \text{ \#\# } (i,j) = (-1)^{\frown(i \cdot_n j)}/(\text{sqrt } 2)^{\frown}n$
 ⟨proof⟩

lemma *dim-row-of-iter-tensor-of-H* [simp]:
assumes $n \geq 1$
shows $1 < \text{dim-row } (H^{\frown}_{\otimes} n)$
 ⟨proof⟩

lemma *iter-tensor-of-H-fst-pos*:
fixes $n \ i \ j$:: nat
assumes $i < 2^{\frown}n \vee j < 2^{\frown}n$ **and** $i < 2^{\frown}(n+1) \wedge j < 2^{\frown}(n+1)$
shows $(H^{\frown}_{\otimes} (\text{Suc } n)) \text{ \#\# } (i,j) = 1/\text{sqrt}(2) * ((H^{\frown}_{\otimes} n) \text{ \#\# } (i \text{ mod } 2^{\frown}n, j \text{ mod } 2^{\frown}n))$
 ⟨proof⟩

lemma *iter-tensor-of-H-fst-neg*:
fixes $n \ i \ j$:: nat
assumes $i \geq 2^{\frown}n \wedge j \geq 2^{\frown}n$ **and** $i < 2^{\frown}(n+1) \wedge j < 2^{\frown}(n+1)$

shows $(H^{\widehat{\otimes}} (Suc\ n))\ \$\$ (i,j) = -1/\text{sqrt}(2) * (H^{\widehat{\otimes}}\ n)\ \$\$ (i \bmod 2^{\widehat{n}}, j \bmod 2^{\widehat{n}})$
 $\langle\text{proof}\rangle$

lemma *H-tensor-iter-tensor-of-H*:

fixes $n::\ \text{nat}$

shows $(H \otimes H^{\widehat{\otimes}}\ n) = H^{\widehat{\otimes}} (Suc\ n)$

$\langle\text{proof}\rangle$

We prove that *Matrix.mat* $2^n\ 2^n$ ($\lambda x.$ *complex-of-real* (case x of $(i, j) \Rightarrow (-1)^{i \cdot n} j / (\text{sqrt}\ 2)^n$)) is indeed the matrix representation of $H \otimes^n$, the iterated tensor product of the Hadamard gate H .

lemma *one-tensor-of-H-is-H*:

shows $(H^{\widehat{\otimes}}\ 1) = H$

$\langle\text{proof}\rangle$

lemma *iter-tensor-of-H-rep-is-correct*:

fixes $n::\ \text{nat}$

assumes $n \geq 1$

shows $(H \otimes^n) = H^{\widehat{\otimes}}\ n$

$\langle\text{proof}\rangle$

$HId^{\widehat{\otimes}}\ 1$ is the result of taking the tensor product of the n th tensor of H and $Id\ 1$

abbreviation *tensor-of-H-tensor-Id*:: $\text{nat} \Rightarrow \text{complex Matrix.mat } (\lambda x.HId^{\widehat{\otimes}}\ x)$ **where**
tensor-of-H-tensor-Id $n \equiv \text{Matrix.mat } (2^{\widehat{(n+1)}})\ (2^{\widehat{(n+1)}})\ (\lambda(i,j).$

$\text{if } (i \bmod 2 = j \bmod 2) \text{ then } (-1)^{\widehat{(i \text{ div } 2)} \cdot n} (j \text{ div } 2) / (\text{sqrt } 2)^{\widehat{n}} \text{ else } 0)$

lemma *mod-2-is-both-even-or-odd*:

$((\text{even } i \wedge \text{even } j) \vee (\text{odd } i \wedge \text{odd } j)) \longleftrightarrow (i \bmod 2 = j \bmod 2)$

$\langle\text{proof}\rangle$

lemma *HId-values [simp]*:

assumes $n \geq 1$ **and** $i < \text{dim-row } (HId^{\widehat{\otimes}}\ n)$ **and** $j < \text{dim-col } (HId^{\widehat{\otimes}}\ n)$

shows $\text{even } i \wedge \text{even } j \longrightarrow (HId^{\widehat{\otimes}}\ n)\ \$\$ (i,j) = (-1)^{\widehat{(i \text{ div } 2)} \cdot n} (j \text{ div } 2) / (\text{sqrt } 2)^{\widehat{n}}$

and $\text{odd } i \wedge \text{odd } j \longrightarrow (HId^{\widehat{\otimes}}\ n)\ \$\$ (i,j) = (-1)^{\widehat{(i \text{ div } 2)} \cdot n} (j \text{ div } 2) / (\text{sqrt } 2)^{\widehat{n}}$

and $(i \bmod 2 = j \bmod 2) \longrightarrow (HId^{\widehat{\otimes}}\ n)\ \$\$ (i,j) = (-1)^{\widehat{(i \text{ div } 2)} \cdot n} (j \text{ div } 2) / (\text{sqrt } 2)^{\widehat{n}}$

and $\neg(i \bmod 2 = j \bmod 2) \longrightarrow (HId^{\widehat{\otimes}}\ n)\ \$\$ (i,j) = 0$

$\langle\text{proof}\rangle$

lemma *iter-tensor-of-H-tensor-Id-is-HId*:

shows $(H^{\widehat{\otimes}}\ n) \otimes Id\ 1 = HId^{\widehat{\otimes}}\ n$

$\langle\text{proof}\rangle$

lemma *HId-is-gate*:

assumes $n \geq 1$

shows $gate\ (n+1)\ (H\hat{I}d_{\otimes}\ n)$
 $\langle proof \rangle$

State ψ_3 is obtained by the multiplication of $Matrix.mat\ 2^{n+1}\ 2^{n+1}$
 $(\lambda x. complex-of-real\ (case\ x\ of\ (i, j) \Rightarrow if\ i\ mod\ 2 = j\ mod\ 2\ then\ (-1)^{i\ div\ 2} \cdot_n\ j\ div\ 2 / (sqrt\ 2)^n\ else\ 0))$ and ψ_2

abbreviation (in *jozsa*) $\psi_3:: complex\ Matrix.mat\ where$

$\psi_3 \equiv Matrix.mat\ (2^{n+1})\ 1\ (\lambda(i,j).$

if even i

then $(\sum_{k < 2^n} (-1)^{f(k) + ((i\ div\ 2) \cdot_n\ k)}) / ((sqrt\ 2)^n * (sqrt\ 2)^{n+1}))$

else $(\sum_{k < 2^n} (-1)^{f(k) + 1 + ((i\ div\ 2) \cdot_n\ k)}) / ((sqrt\ 2)^n * (sqrt\ 2)^{n+1}))$

lemma (in *jozsa*) ψ_3 -values:

assumes $i < dim-row\ \psi_3$

shows $odd\ i \longrightarrow \psi_3\ \$\$ (i,0) = (\sum_{k < 2^n} (-1)^{f(k) + 1 + ((i\ div\ 2) \cdot_n\ k)}) / ((sqrt\ 2)^n * (sqrt\ 2)^{n+1}))$

$\langle proof \rangle$

lemma (in *jozsa*) ψ_3 -dim [simp]:

shows $1 < dim-row\ \psi_3$

$\langle proof \rangle$

lemma *sum-every-odd-summand-is-zero*:

fixes $n:: nat$

assumes $n \geq 1$

shows $\forall f::(nat \Rightarrow complex). (\forall i. i < 2^{n+1} \wedge odd\ i \longrightarrow f\ i = 0) \longrightarrow$
 $(\sum_{k \in \{0.. < 2^{n+1}\}}. f\ k) = (\sum_{k \in \{0.. < 2^n\}}. f\ (2*k))$

$\langle proof \rangle$

lemma *sum-every-even-summand-is-zero*:

fixes $n:: nat$

assumes $n \geq 1$

shows $\forall f::(nat \Rightarrow complex). (\forall i. i < 2^{n+1} \wedge even\ i \longrightarrow f\ i = 0) \longrightarrow$
 $(\sum_{k \in \{0.. < 2^{n+1}\}}. f\ k) = (\sum_{k \in \{0.. < 2^n\}}. f\ (2*k+1))$

$\langle proof \rangle$

lemma (in *jozsa*) *iter-tensor-of-H-times-psi2-is-psi3*:

shows $((H\hat{I}d_{\otimes}\ n) \otimes Id\ 1) * \psi_2 = \psi_3$

$\langle proof \rangle$

lemma (in *jozsa*) ψ_3 -is-state:

shows $state\ (n+1)\ \psi_3$

$\langle proof \rangle$

Finally, all steps are put together. The result depends on the function f . If f is constant the first n qubits are 0, if f is balanced there is at least one qubit in state 1 among the first n qubits. The algorithm only uses one evaluation of $f(x)$ and will always succeed.

definition (in *jozsa*) *jozsa-algo*: complex Matrix.mat **where**
 $jozsa_algo \equiv ((H \otimes^n) \otimes Id\ 1) * (U_f * (((H \otimes^n) * (|zero\rangle \otimes^n)) \otimes (H * |one\rangle)))$

lemma (in *jozsa*) *jozsa-algo-result* [simp]:
shows $jozsa_algo = \psi_3$
 ⟨proof⟩

lemma (in *jozsa*) *jozsa-algo-result-is-state*:
shows $state\ (n+1)\ jozsa_algo$
 ⟨proof⟩

lemma (in *jozsa*) *prob0-fst-qubits-of-jozsa-algo*:
shows $(prob0_fst_qubits\ n\ jozsa_algo) = (\sum_{j \in \{0,1\}} (cmod(jozsa_algo\ \$\$ (j,0)))^2)$
 ⟨proof⟩

General lemmata required to compute probabilities.

lemma *aux-comp-with-sqrt2*:
shows $(\sqrt{2})^n * (\sqrt{2})^n = 2^n$
 ⟨proof⟩

lemma *aux-comp-with-sqrt2-bis* [simp]:
shows $2^n / (\sqrt{2})^n * \sqrt{2}^{(n+1)} = 1/\sqrt{2}$
 ⟨proof⟩

lemma *aux-ineq-with-card*:
fixes $g:: nat \Rightarrow nat$ **and** $A:: nat\ set$
assumes *finite A*
shows $(\sum_{k \in A} (-1)^{(g\ k)}) \leq card\ A$ **and** $(\sum_{k \in A} (-1)^{(g\ k)}) \geq -card\ A$
 ⟨proof⟩

lemma *aux-comp-with-cmod*:
fixes $g:: nat \Rightarrow nat$
assumes $(\forall x < 2^n. g\ x = 0) \vee (\forall x < 2^n. g\ x = 1)$
shows $(cmod\ (\sum_{k < 2^n} (-1)^{(g\ k)})^2) = 2^{(2*n)}$
 ⟨proof⟩

lemma *cmod-less*:
fixes $a\ n:: int$
assumes $a < n$ **and** $a > -n$
shows $cmod\ a < n$
 ⟨proof⟩

lemma *square-less*:
fixes $a\ n:: real$
assumes $a < n$ **and** $a > -n$
shows $a^2 < n^2$
 ⟨proof⟩

lemma *cmod-square-real* [simp]:

fixes $n:: \text{real}$
shows $(\text{cmod } n)^2 = n^2$
 $\langle \text{proof} \rangle$

lemma *aux-comp-sum-divide-cmod*:

fixes $n:: \text{nat}$ **and** $g:: \text{nat} \Rightarrow \text{int}$ **and** $a:: \text{real}$
shows $(\text{cmod}(\text{complex-of-real}(\sum_{k < n} g \ k / a)))^2 = (\text{cmod}(\sum_{k < n} g \ k) / a)^2$
 $\langle \text{proof} \rangle$

The function is constant if and only if the first n qubits are 0. So, if the function is constant, then the probability of measuring 0 for the first n qubits is 1.

lemma (in *jozsa*) *prob0-jozsa-algo-of-const-0*:

assumes $\text{const } 0$
shows $\text{prob0-fst-qubits } n \ \text{jozsa-algo} = 1$
 $\langle \text{proof} \rangle$

lemma (in *jozsa*) *prob0-jozsa-algo-of-const-1*:

assumes $\text{const } 1$
shows $\text{prob0-fst-qubits } n \ \text{jozsa-algo} = 1$
 $\langle \text{proof} \rangle$

If the probability of measuring 0 for the first n qubits is 1, then the function is constant.

lemma (in *jozsa*) *max-value-of-not-const-less*:

assumes $\neg \text{const } 0$ **and** $\neg \text{const } 1$
shows $(\text{cmod}(\sum_{k:: \text{nat} < 2^n} (-1)^{f \ k}))^2 < (2:: \text{nat})^{2*n}$
 $\langle \text{proof} \rangle$

lemma (in *jozsa*) *max-value-of-not-const-less-bis*:

assumes $\neg \text{const } 0$ **and** $\neg \text{const } 1$
shows $(\text{cmod}(\sum_{k:: \text{nat} < 2^n} (-1)^{f \ k + 1}))^2 < (2:: \text{nat})^{2*n}$
 $\langle \text{proof} \rangle$

lemma (in *jozsa*) *f-const-has-max-value*:

assumes $\text{const } 0 \vee \text{const } 1$
shows $(\text{cmod}(\sum_{k < (2:: \text{nat})^n} (-1)^{f \ k}))^2 = (2:: \text{nat})^{2*n}$
and $(\text{cmod}(\sum_{k < (2:: \text{nat})^n} (-1)^{f \ k + 1}))^2 = (2:: \text{nat})^{2*n}$
 $\langle \text{proof} \rangle$

lemma (in *jozsa*) *prob0-fst-qubits-leq*:

shows $(\text{cmod}(\sum_{k < (2:: \text{nat})^n} (-1)^{f \ k}))^2 \leq (2:: \text{nat})^{2*n}$
and $(\text{cmod}(\sum_{k < (2:: \text{nat})^n} (-1)^{f \ k + 1}))^2 \leq (2:: \text{nat})^{2*n}$
 $\langle \text{proof} \rangle$

lemma (in *jozsa*) *prob0-jozsa-algo-1-is-const*:

assumes $\text{prob0-fst-qubits } n \ \text{jozsa-algo} = 1$
shows $\text{const } 0 \vee \text{const } 1$
 $\langle \text{proof} \rangle$

The function is balanced if and only if at least one qubit among the first n qubits is not zero. So, if the function is balanced then the probability of measuring 0 for the first n qubits is 0.

lemma *sum-union-disjoint-finite-set:*

fixes $C::\text{nat set}$ **and** $g::\text{nat} \Rightarrow \text{int}$
assumes *finite C*
shows $\forall A B. A \cap B = \{\} \wedge A \cup B = C \longrightarrow (\sum_{k \in C}. g\ k) = (\sum_{k \in A}. g\ k) + (\sum_{k \in B}. g\ k)$
 $\langle \text{proof} \rangle$

lemma (*in jozsa*) *balanced-pos-and-neg-terms-cancel-out1:*

assumes *is-balanced*
shows $(\sum_{k < (2::\text{nat})^{\wedge} n}. -(1::\text{nat})^{\wedge} (f\ k)) = 0$
 $\langle \text{proof} \rangle$

lemma (*in jozsa*) *balanced-pos-and-neg-terms-cancel-out2:*

assumes *is-balanced*
shows $(\sum_{k < (2::\text{nat})^{\wedge} n}. -(1::\text{nat})^{\wedge} (f\ k + 1)) = 0$
 $\langle \text{proof} \rangle$

lemma (*in jozsa*) *prob0-jozsa-algo-of-balanced:*

assumes *is-balanced*
shows *prob0-fst-qubits n jozsa-algo = 0*
 $\langle \text{proof} \rangle$

If the probability that the first n qubits are 0 is 0, then the function is balanced.

lemma (*in jozsa*) *balanced-prob0-jozsa-algo:*

assumes *prob0-fst-qubits n jozsa-algo = 0*
shows *is-balanced*
 $\langle \text{proof} \rangle$

We prove the correctness of the algorithm.

definition (*in jozsa*) *jozsa-algo-eval:: real* **where**
jozsa-algo-eval \equiv *prob0-fst-qubits n jozsa-algo*

theorem (*in jozsa*) *jozsa-algo-is-correct:*

shows *jozsa-algo-eval = 1* \longleftrightarrow *is-const*
and *jozsa-algo-eval = 0* \longleftrightarrow *is-balanced*
 $\langle \text{proof} \rangle$

end

12 The No-Cloning Theorem

theory *No-Cloning*

imports

Quantum

Tensor
begin

12.1 The Cauchy-Schwarz Inequality

lemma *inner-prod-expand*:

assumes $\text{dim-vec } a = \text{dim-vec } b$ **and** $\text{dim-vec } a = \text{dim-vec } c$ **and** $\text{dim-vec } a = \text{dim-vec } d$

shows $\langle a + b | c + d \rangle = \langle a | c \rangle + \langle a | d \rangle + \langle b | c \rangle + \langle b | d \rangle$
<proof>

lemma *inner-prod-distrib-left*:

assumes $\text{dim-vec } a = \text{dim-vec } b$

shows $\langle c \cdot_v a | b \rangle = \text{cnj}(c) * \langle a | b \rangle$
<proof>

lemma *inner-prod-distrib-right*:

assumes $\text{dim-vec } a = \text{dim-vec } b$

shows $\langle a | c \cdot_v b \rangle = c * \langle a | b \rangle$
<proof>

lemma *cauchy-schwarz-ineq*:

assumes $\text{dim-vec } v = \text{dim-vec } w$

shows $(\text{cmod}(\langle v | w \rangle))^2 \leq \text{Re} (\langle v | v \rangle * \langle w | w \rangle)$
<proof>

lemma *cauchy-schwarz-eq [simp]*:

assumes $v = (l \cdot_v w)$

shows $(\text{cmod}(\langle v | w \rangle))^2 = \text{Re} (\langle v | v \rangle * \langle w | w \rangle)$
<proof>

lemma *cauchy-schwarz-col [simp]*:

assumes $\text{dim-vec } v = \text{dim-vec } w$ **and** $(\text{cmod}(\langle v | w \rangle))^2 = \text{Re} (\langle v | v \rangle * \langle w | w \rangle)$

shows $\exists l. v = (l \cdot_v w) \vee w = (l \cdot_v v)$
<proof>

12.2 The No-Cloning Theorem

lemma *eq-from-inner-prod [simp]*:

assumes $\text{dim-vec } v = \text{dim-vec } w$ **and** $\langle v | w \rangle = 1$ **and** $\langle v | v \rangle = 1$ **and** $\langle w | w \rangle = 1$

shows $v = w$
<proof>

lemma *hermite-cnj-of-tensor*:

shows $(A \otimes B)^\dagger = (A^\dagger) \otimes (B^\dagger)$

<proof>

locale *quantum-machine* =

fixes $n:: \text{nat}$ **and** $s:: \text{complex Matrix.vec}$ **and** $U:: \text{complex Matrix.mat}$

assumes $\text{dim-vec } [simp]: \text{dim-vec } s = 2^n$

and *dim-col* [*simp*]: *dim-col* $U = 2^{\wedge}n * 2^{\wedge}n$
and *square* [*simp*]: *square-mat* U **and** *unitary* [*simp*]: *unitary* U

lemma *inner-prod-of-unit-vec*:

fixes $n i :: \text{nat}$
assumes $i < n$
shows $\langle \text{unit-vec } n \ i \mid \text{unit-vec } n \ i \rangle = 1$
 $\langle \text{proof} \rangle$

theorem (**in** *quantum-machine*) *no-cloning*:

assumes [*simp*]: *dim-vec* $v = 2^{\wedge}n$ **and** [*simp*]: *dim-vec* $w = 2^{\wedge}n$ **and**
cloning1: $\bigwedge s. U * (|v\rangle \otimes |s\rangle) = |v\rangle \otimes |v\rangle$ **and**
cloning2: $\bigwedge s. U * (|w\rangle \otimes |s\rangle) = |w\rangle \otimes |w\rangle$ **and**
 $\langle v|v \rangle = 1$ **and** $\langle w|w \rangle = 1$
shows $v = w \vee \langle v|w \rangle = 0$
 $\langle \text{proof} \rangle$

end

13 Quantum Prisoner's Dilemma

theory *Quantum-Prisoners-Dilemma*

imports

More-Tensor

Measurement

Basics

begin

In the 2-parameter strategic space of Eisert, Wilkens and Lewenstein [EWL], Prisoner's Dilemma ceases to pose a dilemma if quantum strategies are allowed for. Indeed, Alice and Bob both choosing to defect is no longer a Nash equilibrium. However, a new Nash equilibrium appears which is at the same time Pareto optimal. Moreover, there exists a quantum strategy which always gives reward if played against any classical strategy. Below the parameter γ can be seen as a measure of the game's entanglement. The game behaves classically if $\gamma = 0$, and for the maximally entangled case ($\gamma = 2*\pi$) the dilemma disappears as pointed out above.

13.1 The Set-Up

locale *prisoner* =

fixes $\gamma :: \text{real}$

assumes $\gamma \leq \text{pi}/2$ **and** $\gamma \geq 0$

abbreviation (**in** *prisoner*) $J :: \text{complex Matrix.mat}$ **where**

$J \equiv \text{mat-of-cols-list } 4 \ [[\cos(\gamma/2), 0, 0, i*\sin(\gamma/2)],$
 $[0, \cos(\gamma/2), -i*\sin(\gamma/2), 0],$
 $[0, -i*\sin(\gamma/2), \cos(\gamma/2), 0],$

$$[i*\sin(\gamma/2), 0, 0, \cos(\gamma/2)]$$

abbreviation (in prisoner) ψ_1 :: complex Matrix.mat where
 $\psi_1 \equiv \text{mat-of-cols-list } 4 \text{ } [[\cos(\gamma/2), 0, 0, i*\sin(\gamma/2)]]$

lemma (in prisoner) psi-one:
 shows $J * |\text{unit-vec } 4 \ 0\rangle = \psi_1$
 <proof>

locale strategic-space-2p = prisoner +
fixes ϑ_A :: real
and φ_A :: real
and ϑ_B :: real
and φ_B :: real
assumes $0 \leq \vartheta_A \wedge \vartheta_A \leq \pi$
and $0 \leq \varphi_A \wedge \varphi_A \leq \pi/2$
and $0 \leq \vartheta_B \wedge \vartheta_B \leq \pi$
and $0 \leq \varphi_B \wedge \varphi_B \leq \pi/2$

abbreviation (in strategic-space-2p) U_A :: complex Matrix.mat where
 $U_A \equiv \text{mat-of-cols-list } 2 \text{ } [[\exp(i*\varphi_A)*\cos(\vartheta_A/2), -\sin(\vartheta_A/2)],$
 $[\sin(\vartheta_A/2), \exp(-i*\varphi_A)*\cos(\vartheta_A/2)]]$

abbreviation (in strategic-space-2p) U_B :: complex Matrix.mat where
 $U_B \equiv \text{mat-of-cols-list } 2 \text{ } [[\exp(i*\varphi_B)*\cos(\vartheta_B/2), -\sin(\vartheta_B/2)],$
 $[\sin(\vartheta_B/2), \exp(-i*\varphi_B)*\cos(\vartheta_B/2)]]$

abbreviation (in strategic-space-2p) ψ_2 :: complex Matrix.mat where
 $\psi_2 \equiv$
 $\text{mat-of-cols-list } 4 \text{ } [[\exp(i*\varphi_A)*\cos(\vartheta_A/2) * \exp(i*\varphi_B)*\cos(\vartheta_B/2) * \cos(\gamma/2) + \sin(\vartheta_A/2)$
 $* \sin(\vartheta_B/2) * i*\sin(\gamma/2),$
 $\exp(i*\varphi_A)*\cos(\vartheta_A/2) * -\sin(\vartheta_B/2) * \cos(\gamma/2) + \sin(\vartheta_A/2) *$
 $\exp(-i*\varphi_B)*\cos(\vartheta_B/2) * i*\sin(\gamma/2),$
 $-\sin(\vartheta_A/2) * \exp(i*\varphi_B)*\cos(\vartheta_B/2) * \cos(\gamma/2) + \exp(-i*\varphi_A)*\cos(\vartheta_A/2)$
 $* \sin(\vartheta_B/2) * i*\sin(\gamma/2),$
 $\sin(\vartheta_A/2) * \sin(\vartheta_B/2) * \cos(\gamma/2) + \exp(-i*\varphi_A)*\cos(\vartheta_A/2) *$
 $\exp(-i*\varphi_B)*\cos(\vartheta_B/2) * i*\sin(\gamma/2)]]$

abbreviation (in strategic-space-2p) U_{AB} :: complex Matrix.mat where
 $U_{AB} \equiv$
 $\text{mat-of-cols-list } 4 \text{ } [[\exp(i*\varphi_A)*\cos(\vartheta_A/2) * \exp(i*\varphi_B)*\cos(\vartheta_B/2), \exp(i*\varphi_A)*\cos(\vartheta_A/2)$
 $* -\sin(\vartheta_B/2),$
 $-\sin(\vartheta_A/2) * \exp(i*\varphi_B)*\cos(\vartheta_B/2), -\sin(\vartheta_A/2) *$
 $-\sin(\vartheta_B/2)],$
 $[\exp(i*\varphi_A)*\cos(\vartheta_A/2) * \sin(\vartheta_B/2), \exp(i*\varphi_A)*\cos(\vartheta_A/2) *$
 $\exp(-i*\varphi_B)*\cos(\vartheta_B/2),$
 $-\sin(\vartheta_A/2) * \sin(\vartheta_B/2), -\sin(\vartheta_A/2) * \exp(-i*\varphi_B)*\cos(\vartheta_B/2)],$
 $[\sin(\vartheta_A/2) * \exp(i*\varphi_B)*\cos(\vartheta_B/2), -\sin(\vartheta_A/2) * \sin(\vartheta_B/2),$
 $\exp(-i*\varphi_A)*\cos(\vartheta_A/2) * \exp(i*\varphi_B)*\cos(\vartheta_B/2),$

$$\begin{aligned} & \exp(-i\varphi_A) * \cos(\vartheta_A/2) * -\sin(\vartheta_B/2)], \\ & \quad [\sin(\vartheta_A/2) * \sin(\vartheta_B/2), \sin(\vartheta_A/2) * \exp(-i\varphi_B) * \cos(\vartheta_B/2), \\ & \quad \quad \exp(-i\varphi_A) * \cos(\vartheta_A/2) * \sin(\vartheta_B/2), \exp(-i\varphi_A) * \cos(\vartheta_A/2) \\ & * \exp(-i\varphi_B) * \cos(\vartheta_B/2)] \end{aligned}$$

lemma (in *strategic-space-2p*) U_A -tensor- U_B :

$$\text{shows } (U_A \otimes U_B) = U_{AB}$$

<proof>

lemma (in *strategic-space-2p*) *psi-two*:

$$\text{shows } (U_A \otimes U_B) * \psi_1 = \psi_2$$

<proof>

abbreviation (in *prisoner*) J -cnj :: complex Matrix.mat where

$$\begin{aligned} J\text{-cnj} \equiv \text{mat-of-cols-list } 4 \quad & [[\cos(\gamma/2), 0, 0, -i\sin(\gamma/2)], \\ & [0, \cos(\gamma/2), i\sin(\gamma/2), 0], \\ & [0, i\sin(\gamma/2), \cos(\gamma/2), 0], \\ & [-i\sin(\gamma/2), 0, 0, \cos(\gamma/2)]] \end{aligned}$$

lemma (in *prisoner*) *hermite-cnj-of-J* [simp]:

$$\text{shows } J^\dagger = J\text{-cnj}$$

<proof>

abbreviation (in *strategic-space-2p*) ψ_f :: complex Matrix.mat where

$$\begin{aligned} \psi_f \equiv \text{mat-of-cols-list } 4 \quad & [[\\ & \cos(\gamma/2) * (\exp(i\varphi_A) * \cos(\vartheta_A/2) * \exp(i\varphi_B) * \cos(\vartheta_B/2) * \cos(\gamma/2) + \sin(\vartheta_A/2) \\ & * \sin(\vartheta_B/2) * i\sin(\gamma/2)) \\ & + (-i\sin(\gamma/2)) * (\sin(\vartheta_A/2) * \sin(\vartheta_B/2) * \cos(\gamma/2) + \exp(-i\varphi_A) * \cos(\vartheta_A/2) \\ & * \exp(-i\varphi_B) * \cos(\vartheta_B/2) * i\sin(\gamma/2)), \end{aligned}$$

$$\begin{aligned} & \cos(\gamma/2) * (\exp(i\varphi_A) * \cos(\vartheta_A/2) * -\sin(\vartheta_B/2) * \cos(\gamma/2) + \sin(\vartheta_A/2) * \exp(-i\varphi_B) * \cos(\vartheta_B/2) \\ & * i\sin(\gamma/2)) \\ & + (i\sin(\gamma/2)) * (-\sin(\vartheta_A/2) * \exp(i\varphi_B) * \cos(\vartheta_B/2) * \cos(\gamma/2) + \exp(-i\varphi_A) * \cos(\vartheta_A/2) \\ & * \sin(\vartheta_B/2) * i\sin(\gamma/2)), \end{aligned}$$

$$\begin{aligned} & (i\sin(\gamma/2)) * (\exp(i\varphi_A) * \cos(\vartheta_A/2) * -\sin(\vartheta_B/2) * \cos(\gamma/2) + \sin(\vartheta_A/2) * \\ & \exp(-i\varphi_B) * \cos(\vartheta_B/2) * i\sin(\gamma/2)) \\ & + \cos(\gamma/2) * (-\sin(\vartheta_A/2) * \exp(i\varphi_B) * \cos(\vartheta_B/2) * \cos(\gamma/2) + \exp(-i\varphi_A) * \cos(\vartheta_A/2) \\ & * \sin(\vartheta_B/2) * i\sin(\gamma/2)), \end{aligned}$$

$$\begin{aligned} & (-i\sin(\gamma/2)) * (\exp(i\varphi_A) * \cos(\vartheta_A/2) * \exp(i\varphi_B) * \cos(\vartheta_B/2) * \cos(\gamma/2) + \sin(\vartheta_A/2) \\ & * \sin(\vartheta_B/2) * i\sin(\gamma/2)) \\ & + \cos(\gamma/2) * (\sin(\vartheta_A/2) * \sin(\vartheta_B/2) * \cos(\gamma/2) + \exp(-i\varphi_A) * \cos(\vartheta_A/2) * \\ & \exp(-i\varphi_B) * \cos(\vartheta_B/2) * i\sin(\gamma/2)) \\ &]] \end{aligned}$$

lemma (in *strategic-space-2p*) *psi-f*:

$$\text{shows } (J^\dagger) * \psi_2 = \psi_f$$

<proof>

lemma (in *prisoner*) *unit-vec-4-0-ket-is-state*:

shows $state\ 2\ |\text{unit-vec}\ 4\ 0\rangle$

<proof>

lemma *cos-sin-squared-add-cpx*:

$complex\text{-of-real}\ (\cos\ (\gamma/2))\ *\ complex\text{-of-real}\ (\cos\ (\gamma/2))\ -$

$i*complex\text{-of-real}\ (\sin\ (\gamma/2))\ *\ (i*complex\text{-of-real}\ (\sin\ (\gamma/2)))\ =\ 1$

<proof>

lemma *sin-cos-squared-add-cpx*:

$i*complex\text{-of-real}\ (\sin\ (\gamma/2))\ *\ (i*complex\text{-of-real}\ (\sin\ (\gamma/2)))\ -$

$complex\text{-of-real}\ (\cos\ (\gamma/2))\ *\ complex\text{-of-real}\ (\cos\ (\gamma/2))\ =\ -1$

<proof>

lemma (in *prisoner*) *J-cnj-times-J*:

shows $J^\dagger * J = 1_m\ 4$

<proof>

lemma (in *prisoner*) *J-times-J-cnj*:

shows $J * (J^\dagger) = 1_m\ 4$

<proof>

lemma (in *prisoner*) *J-is-gate*:

shows $gate\ 2\ J$

<proof>

lemma (in *strategic-space-2p*) *psi-one-is-state*:

shows $state\ 2\ \psi_1$

<proof>

abbreviation (in *strategic-space-2p*) $U_A\text{-cnj} :: complex\ Matrix.mat$ **where**

$U_A\text{-cnj} \equiv mat\text{-of-cols-list}\ 2\ [[(exp(-i*\varphi_A))*cos(\vartheta_A/2),\ sin(\vartheta_A/2)],$
 $[-sin(\vartheta_A/2),\ (exp(i*\varphi_A))*cos(\vartheta_A/2)]]$

abbreviation (in *strategic-space-2p*) $U_B\text{-cnj} :: complex\ Matrix.mat$ **where**

$U_B\text{-cnj} \equiv mat\text{-of-cols-list}\ 2\ [[(exp(-i*\varphi_B))*cos(\vartheta_B/2),\ sin(\vartheta_B/2)],$
 $[-sin(\vartheta_B/2),\ (exp(i*\varphi_B))*cos(\vartheta_B/2)]]$

lemma (in *strategic-space-2p*) *hermite-cnj-of-U_A*:

shows $U_A^\dagger = U_A\text{-cnj}$

<proof>

lemma (in *strategic-space-2p*) *hermite-cnj-of-U_B*:

shows $U_B^\dagger = U_B\text{-cnj}$

<proof>

lemma *exp-sin-cos-squared-add*:

fixes $x\ y :: real$

shows $\exp(-i * x) * \cos(y) * (\exp(i * x) * \cos(y)) + \sin(y) * \sin(y) = 1$
<proof>

lemma (in *strategic-space-2p*) *U_A-conj-times-U_A*:
shows $U_A^\dagger * U_A = 1_m$ 2
<proof>

lemma (in *strategic-space-2p*) *U_A-times-U_A-conj*:
shows $U_A * (U_A^\dagger) = 1_m$ 2
<proof>

lemma (in *strategic-space-2p*) *U_B-conj-times-U_B*:
shows $U_B^\dagger * U_B = 1_m$ 2
<proof>

lemma (in *strategic-space-2p*) *U_B-times-U_B-conj*:
shows $U_B * (U_B^\dagger) = 1_m$ 2
<proof>

lemma (in *strategic-space-2p*) *U_A-is-gate*:
shows *gate 1 U_A*
<proof>

lemma (in *strategic-space-2p*) *U_B-is-gate*:
shows *gate 1 U_B*
<proof>

lemma (in *strategic-space-2p*) *U_{AB}-is-gate*:
shows *gate 2 (U_A ⊗ U_B)*
<proof>

lemma (in *strategic-space-2p*) *psi-two-is-state*:
shows *state 2 ψ₂*
<proof>

lemma (in *strategic-space-2p*) *J-conj-is-gate*:
shows *gate 2 (J[†])*
<proof>

lemma (in *strategic-space-2p*) *psi-f-is-state*:
shows *state 2 ψ_f*
<proof>

lemma (in *strategic-space-2p*) *equation-one*:
shows $(J^\dagger) * ((U_A \otimes U_B) * (J * |unit-vec 4 0\rangle)) = \psi_f$
<proof>

abbreviation (in *strategic-space-2p*) *prob00* :: *complex Matrix.mat* ⇒ *real where*

$prob00\ v \equiv \text{if state } 2\ v \text{ then } (cmod\ (v\ \$\$ (0,0)))^2 \text{ else undefined}$

abbreviation (in *strategic-space-2p*) $prob01 :: \text{complex Matrix.mat} \Rightarrow \text{real}$ **where**
 $prob01\ v \equiv \text{if state } 2\ v \text{ then } (cmod\ (v\ \$\$ (1,0)))^2 \text{ else undefined}$

abbreviation (in *strategic-space-2p*) $prob10 :: \text{complex Matrix.mat} \Rightarrow \text{real}$ **where**
 $prob10\ v \equiv \text{if state } 2\ v \text{ then } (cmod\ (v\ \$\$ (2,0)))^2 \text{ else undefined}$

abbreviation (in *strategic-space-2p*) $prob11 :: \text{complex Matrix.mat} \Rightarrow \text{real}$ **where**
 $prob11\ v \equiv \text{if state } 2\ v \text{ then } (cmod\ (v\ \$\$ (3,0)))^2 \text{ else undefined}$

definition (in *strategic-space-2p*) $alice\text{-payoff} :: \text{real}$ **where**
 $alice\text{-payoff} \equiv 3 * (prob00\ \psi_f) + 1 * (prob11\ \psi_f) + 0 * (prob01\ \psi_f) + 5 * (prob10\ \psi_f)$

definition (in *strategic-space-2p*) $bob\text{-payoff} :: \text{real}$ **where**
 $bob\text{-payoff} \equiv 3 * (prob00\ \psi_f) + 1 * (prob11\ \psi_f) + 5 * (prob01\ \psi_f) + 0 * (prob10\ \psi_f)$

definition (in *strategic-space-2p*) $is\text{-nash}\text{-eq} :: \text{bool}$ **where**
 $is\text{-nash}\text{-eq} \equiv$
 $(\forall\ tA\ pA.\ \text{strategic-space-2p}\ \gamma\ tA\ pA\ \vartheta_B\ \varphi_B \longrightarrow$
 $\quad alice\text{-payoff} \geq \text{strategic-space-2p.alice-payoff}\ \gamma\ tA\ pA\ \vartheta_B\ \varphi_B)$
 \wedge
 $(\forall\ tB\ pB.\ \text{strategic-space-2p}\ \gamma\ \vartheta_A\ \varphi_A\ tB\ pB \longrightarrow$
 $\quad bob\text{-payoff} \geq \text{strategic-space-2p.bob-payoff}\ \gamma\ \vartheta_A\ \varphi_A\ tB\ pB)$

definition (in *strategic-space-2p*) $is\text{-pareto}\text{-opt} :: \text{bool}$ **where**
 $is\text{-pareto}\text{-opt} \equiv \forall\ tA\ pA\ tB\ pB.\ \text{strategic-space-2p}\ \gamma\ tA\ pA\ tB\ pB \longrightarrow$
 $((\text{strategic-space-2p.alice-payoff}\ \gamma\ tA\ pA\ tB\ pB > alice\text{-payoff} \longrightarrow$
 $\quad \text{strategic-space-2p.bob-payoff}\ \gamma\ tA\ pA\ tB\ pB < bob\text{-payoff}) \wedge$
 $(\text{strategic-space-2p.bob-payoff}\ \gamma\ tA\ pA\ tB\ pB > bob\text{-payoff} \longrightarrow$
 $\quad \text{strategic-space-2p.alice-payoff}\ \gamma\ tA\ pA\ tB\ pB < alice\text{-payoff}))$

lemma (in *strategic-space-2p*) $sum\text{-of}\text{-prob}$:
fixes $v :: \text{complex Matrix.mat}$
assumes $state\ 2\ v$
shows $(prob00\ v) + (prob11\ v) + (prob01\ v) + (prob10\ v) = 1$
(proof)

lemma (in *strategic-space-2p*) $sum\text{-payoff}\text{-le-6}$:
fixes $tA\ pA\ tB\ pB :: \text{real}$
shows $alice\text{-payoff} + bob\text{-payoff} \leq 6$
(proof)

lemma (in *strategic-space-2p*) $coop\text{-is}\text{-pareto}\text{-opt}$:
assumes $alice\text{-payoff} = 3 \wedge bob\text{-payoff} = 3$
shows $is\text{-pareto}\text{-opt}$
(proof)

13.2 The Separable Case

lemma (in *strategic-space-2p*) *separable-case-CC*:

assumes $\gamma = 0$

shows $\varphi_A = 0 \wedge \vartheta_A = 0 \wedge \varphi_B = 0 \wedge \vartheta_B = 0 \longrightarrow \text{alice-payoff} = 3 \wedge \text{bob-payoff} = 3$

<proof>

lemma (in *strategic-space-2p*) *separable-case-DD*:

assumes $\gamma = 0$

shows $\varphi_A = 0 \wedge \vartheta_A = pi \wedge \varphi_B = 0 \wedge \vartheta_B = pi \longrightarrow \text{alice-payoff} = 1 \wedge \text{bob-payoff} = 1$

<proof>

lemma (in *strategic-space-2p*) *separable-case-DC*:

assumes $\gamma = 0$

shows $\varphi_A = 0 \wedge \vartheta_A = pi \wedge \varphi_B = 0 \wedge \vartheta_B = 0 \longrightarrow \text{alice-payoff} = 5 \wedge \text{bob-payoff} = 0$

<proof>

lemma (in *strategic-space-2p*) *separable-alice-payoff-DB*:

assumes $\gamma = 0$ **and** $\varphi_B = 0 \wedge \vartheta_B = pi$

shows $\text{alice-payoff} \leq 1$

<proof>

lemma (in *strategic-space-2p*) *separable-bob-payoff-DA*:

assumes $\gamma = 0$ **and** $\varphi_A = 0 \wedge \vartheta_A = pi$

shows $\text{bob-payoff} \leq 1$

<proof>

lemma (in *strategic-space-2p*) *separable-case-DD-alice-opt*:

assumes $\gamma = 0$ **and** $\varphi_A = 0 \wedge \vartheta_A = pi \wedge \varphi_B = 0 \wedge \vartheta_B = pi$

shows $\bigwedge tA pA. \text{strategic-space-2p } \gamma tA pA \vartheta_B \varphi_B \longrightarrow \text{strategic-space-2p.alice-payoff } \gamma tA pA \vartheta_B \varphi_B \leq \text{alice-payoff}$

<proof>

lemma (in *strategic-space-2p*) *separable-case-DD-bob-opt*:

assumes $\gamma = 0$ **and** $\varphi_A = 0 \wedge \vartheta_A = pi \wedge \varphi_B = 0 \wedge \vartheta_B = pi$

shows $\bigwedge tB pB. \text{strategic-space-2p } \gamma \vartheta_A \varphi_A tB pB \longrightarrow \text{strategic-space-2p.bob-payoff } \gamma \vartheta_A \varphi_A tB pB \leq \text{bob-payoff}$

<proof>

lemma (in *strategic-space-2p*) *separable-case-DD-is-nash-eq*:

assumes $\gamma = 0$

shows $\varphi_A = 0 \wedge \vartheta_A = pi \wedge \varphi_B = 0 \wedge \vartheta_B = pi \longrightarrow \text{is-nash-eq}$

<proof>

lemma (in *strategic-space-2p*) *separable-case-CC-is-not-nash-eq*:

assumes $\gamma = 0$
shows $\varphi_A = 0 \wedge \vartheta_A = 0 \wedge \varphi_B = 0 \wedge \vartheta_B = 0 \longrightarrow \neg \text{is-nash-eq}$
 ⟨proof⟩

lemma (in *strategic-space-2p*) *separable-case-CC-is-pareto-optimal*:
assumes $\gamma = 0$
shows $\varphi_A = 0 \wedge \vartheta_A = 0 \wedge \varphi_B = 0 \wedge \vartheta_B = 0 \longrightarrow \text{is-pareto-opt}$
 ⟨proof⟩

13.3 The Maximally Entangled Case

lemma *exp-to-sin*:
fixes $x:: \text{real}$
shows $\exp(i * x) - \exp(-i * x) = 2 * i * (\sin x)$
 ⟨proof⟩

lemma *exp-to-cos*:
fixes $x:: \text{real}$
shows $\exp(i * x) + \exp(-i * x) = 2 * (\cos x)$
 ⟨proof⟩

lemma *cmod-real-prod-squared*:
fixes $x y:: \text{real}$
shows $(\text{cmod}(\text{complex-of-real } x * \text{complex-of-real } y))^2 = x^2 * y^2$
 ⟨proof⟩

lemma *quantum-payoff-simp*:
fixes $x y:: \text{real}$
shows $3 * (\text{cmod}(\text{complex-of-real } (\sin x) * \text{complex-of-real } (\cos y)))^2 +$
 $(\text{cmod}(\text{complex-of-real } (\cos x) * \text{complex-of-real } (\cos y)))^2 =$
 $2 * (\sin x)^2 * (\cos y)^2 + (\cos y)^2$
 ⟨proof⟩

lemma *quantum-payoff-le-3*:
fixes $x y:: \text{real}$
shows $2 * (\sin x)^2 * (\cos y)^2 + (\cos y)^2 \leq 3$
 ⟨proof⟩

lemma *sqrt-two-squared-cpx*: $\text{complex-of-real } (\text{sqrt } 2) * \text{complex-of-real } (\text{sqrt } 2) =$
 2
 ⟨proof⟩

lemma *hidden-sqrt-two-squared-cpx*: $\text{complex-of-real } (\text{sqrt } 2) * (\text{complex-of-real } (\text{sqrt } 2) * x) / 4 = x/2$
 ⟨proof⟩

lemma (in *strategic-space-2p*) *max-entangled-DD*:
assumes $\gamma = \text{pi}/2$

shows $\varphi_A = 0 \wedge \vartheta_A = \pi \wedge \varphi_B = 0 \wedge \vartheta_B = \pi \longrightarrow \text{alice-payoff} = 1 \wedge \text{bob-payoff} = 1$
 <proof>

lemma (in *strategic-space-2p*) *max-entangled-QQ*:

assumes $\gamma = \pi/2$
shows $\varphi_A = \pi/2 \wedge \vartheta_A = 0 \wedge \varphi_B = \pi/2 \wedge \vartheta_B = 0 \longrightarrow \text{alice-payoff} = 3 \wedge \text{bob-payoff} = 3$
 <proof>

lemma (in *strategic-space-2p*) *max-entangled-QD*:

assumes $\gamma = \pi/2$
shows $\varphi_A = \pi/2 \wedge \vartheta_A = 0 \wedge \varphi_B = 0 \wedge \vartheta_B = \pi \longrightarrow \text{alice-payoff} = 5 \wedge \text{bob-payoff} = 0$
 <proof>

lemma (in *strategic-space-2p*) *max-entangled-alice-payoff-QB*:

assumes $\gamma = \pi/2$
shows $\varphi_B = \pi/2 \wedge \vartheta_B = 0 \longrightarrow \text{alice-payoff} \leq 3$
 <proof>

lemma (in *strategic-space-2p*) *max-entangled-bob-payoff-QA*:

assumes $\gamma = \pi/2$
shows $\varphi_A = \pi/2 \wedge \vartheta_A = 0 \longrightarrow \text{bob-payoff} \leq 3$
 <proof>

lemma (in *strategic-space-2p*) *max-entangled-DD-is-not-nash-eq*:

assumes $\gamma = \pi/2$
shows $\varphi_A = 0 \wedge \vartheta_A = \pi \wedge \varphi_B = 0 \wedge \vartheta_B = \pi \longrightarrow \neg \text{is-nash-eq}$
 <proof>

lemma (in *strategic-space-2p*) *max-entangled-alice-opt*:

assumes $\gamma = \pi/2$ **and** $\varphi_A = \pi/2 \wedge \vartheta_A = 0 \wedge \varphi_B = \pi/2 \wedge \vartheta_B = 0$
shows $\bigwedge tA \ pA. \text{strategic-space-2p } \gamma \ tA \ pA \ \vartheta_B \ \varphi_B \longrightarrow \text{strategic-space-2p.alice-payoff } \gamma \ tA \ pA \ \vartheta_B \ \varphi_B \leq \text{alice-payoff}$
 <proof>

lemma (in *strategic-space-2p*) *max-entangled-bob-opt*:

assumes $\gamma = \pi/2$ **and** $\varphi_A = \pi/2 \wedge \vartheta_A = 0 \wedge \varphi_B = \pi/2 \wedge \vartheta_B = 0$
shows $\bigwedge tB \ pB. \text{strategic-space-2p } \gamma \ \vartheta_A \ \varphi_A \ tB \ pB \longrightarrow \text{strategic-space-2p.bob-payoff } \gamma \ \vartheta_A \ \varphi_A \ tB \ pB \leq \text{bob-payoff}$
 <proof>

lemma (in *strategic-space-2p*) *max-entangled-QQ-is-nash-eq*:

assumes $\gamma = \pi/2$

shows $\varphi_A = \pi/2 \wedge \vartheta_A = 0 \wedge \varphi_B = \pi/2 \wedge \vartheta_B = 0 \longrightarrow is\text{-nash}\text{-eq}$
 ⟨proof⟩

lemma (in *strategic-space-2p*) *max-entangled-QQ-is-pareto-optimal*:
assumes $\gamma = \pi/2$
shows $\varphi_A = \pi/2 \wedge \vartheta_A = 0 \wedge \varphi_B = \pi/2 \wedge \vartheta_B = 0 \longrightarrow is\text{-pareto}\text{-opt}$
 ⟨proof⟩

13.4 The Unfair Strategy Case

lemma *half-sqrt-two-squared*: $2 * (\text{sqrt } 2 / 2)^2 = 1$
 ⟨proof⟩

lemma (in *strategic-space-2p*) *max-entangled-MD*:

assumes $\gamma = \pi/2$
shows $\varphi_A = \pi/2 \wedge \vartheta_A = \pi/2 \wedge \varphi_B = 0 \wedge \vartheta_B = \pi \longrightarrow \text{alice}\text{-payoff} = 3 \wedge$
 $\text{bob}\text{-payoff} = 1/2$
 ⟨proof⟩

lemma (in *strategic-space-2p*) *max-entangled-MC*:

assumes $\gamma = \pi/2$
shows $\varphi_A = \pi/2 \wedge \vartheta_A = \pi/2 \wedge \varphi_B = 0 \wedge \vartheta_B = 0 \longrightarrow \text{alice}\text{-payoff} = 3 \wedge$
 $\text{bob}\text{-payoff} = 1/2$
 ⟨proof⟩

lemma (in *strategic-space-2p*) *max-entangled-MH*:

assumes $\gamma = \pi/2$
shows $\varphi_A = \pi/2 \wedge \vartheta_A = \pi/2 \wedge \varphi_B = 0 \wedge \vartheta_B = \pi/2 \longrightarrow \text{alice}\text{-payoff} = 1$
 $\wedge \text{bob}\text{-payoff} = 1$
 ⟨proof⟩

abbreviation $M :: \text{complex Matrix.mat where}$
 $M \equiv \text{mat-of-cols-list } 2 \text{ [[i * sqrt(2)/2, -1 * sqrt(2)/2],}$
 $\text{[1 * sqrt(2)/2, -i * sqrt(2)/2]]$

lemma (in *strategic-space-2p*) *M-correct*:

assumes $\varphi_A = \pi/2 \wedge \vartheta_A = \pi/2$
shows $U_A = M$
 ⟨proof⟩

lemma *hidden-sqrt-two-squared-cpx2*:

fixes $x y :: \text{complex}$
shows $(\text{sqrt } 2) * ((\text{sqrt } 2) * (x * y)) / 2 = x * y$
 ⟨proof⟩

lemma (in *strategic-space-2p*) *unfair-strategy-no-benefit*:

assumes $\gamma = \pi/2$

shows $\varphi_A = \pi/2 \wedge \varphi_B = 0 \wedge \vartheta_A = \vartheta_B \longrightarrow \text{alice-payoff} = 1 \wedge \text{bob-payoff} = 1$
(*proof*)

end

14 Acknowledgements

The work has been jointly supported by the Cambridge Mathematics Placements (CMP) Programme and the ERC Advanced Grant ALEXANDRIA (Project GA 742178).

References

- [1] J. Boender, F. Kammüller, and R. Nagarajan. Formalization of quantum protocols using coq. In *QPL*, 2015.
- [2] J. Eisert, M. Wilkens, and M. Lewenstein. Quantum Games and Quantum Strategies. *Physical Review Letters*, 83:3077–3080, Oct. 1999.
- [3] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.