

Quantum Hoare Logic

Junyi Liu, Bohua Zhan, Shuling Wang, Shenggang Ying,
Tao Liu, Yangjia Li, Mingsheng Ying, and Naijun Zhan

May 26, 2024

Abstract

We formalize quantum Hoare logic as given in [1]. In particular, we specify the syntax and denotational semantics of a simple model of quantum programs. Then, we write down the rules of quantum Hoare logic for partial correctness, and show the soundness and completeness of the resulting proof system. As an application, we verify the correctness of Grover's algorithm.

Contents

1	Complex matrices	2
1.1	Trace of a matrix	2
1.2	Conjugate of a vector	3
1.3	Inner product	3
1.4	Hermitian adjoint of a matrix	5
1.5	Algebraic manipulations on matrices	6
1.6	Hermitian matrices	7
1.7	Inverse matrices	8
1.8	Unitary matrices	8
1.9	Normalization of vectors	9
1.10	Spectral decomposition of normal complex matrices	10
1.11	Outer product	13
1.12	Semi-definite matrices	15
1.13	Löwner partial order	17
1.14	Density operators	20
2	Matrix limits	21
2.1	Definition of limit of matrices	21
2.2	Existence of least upper bound for the Löwner order	24
2.3	Finite sum of matrices	27
2.4	Measurement	29

3	Quantum programs	30
3.1	Syntax	30
3.2	Denotational semantics	31
4	Partial state	38
4.1	Encodings	40
4.2	Tensor product of vectors and matrices	42
4.3	Extension of matrices	46
4.4	Partial tensor product	47
4.5	Partial extensions	50
4.6	Commands on subset of variables	54
5	Standard gates	54
6	Partial and total correctness	56
6.1	Weakest liberal preconditions	57
6.2	Hoare triples for partial correctness	60
6.3	Consequences of completeness	61
7	Grover's algorithm	61
7.1	Basic definitions	61
7.2	Grover operator	65
7.3	State of Grover's algorithm	67
7.4	Grover's algorithm	70
7.5	Correctness	73

1 Complex matrices

```

theory Complex-Matrix
  imports
    Jordan-Normal-Form.Matrix
    Jordan-Normal-Form.Conjugate
    Jordan-Normal-Form.Jordan-Normal-Form-Existence
begin

```

1.1 Trace of a matrix

definition *trace* :: 'a::ring mat \Rightarrow 'a **where**
trace A = (\sum i \in {0 ..< dim-row A}. A \$\$ (i,i))

lemma *trace-zero* [simp]:
trace (0_m n n) = 0
 <proof>

lemma *trace-id* [simp]:
trace (1_m n) = n
 <proof>

lemma *trace-comm*:
fixes $A B :: 'a::comm-ring\ mat$
assumes $A: A \in carrier-mat\ n\ n$ **and** $B: B \in carrier-mat\ n\ n$
shows $trace\ (A * B) = trace\ (B * A)$
 $\langle proof \rangle$

lemma *trace-add-linear*:
fixes $A B :: 'a::comm-ring\ mat$
assumes $A: A \in carrier-mat\ n\ n$ **and** $B: B \in carrier-mat\ n\ n$
shows $trace\ (A + B) = trace\ A + trace\ B$ (**is** $?lhs = ?rhs$)
 $\langle proof \rangle$

lemma *trace-minus-linear*:
fixes $A B :: 'a::comm-ring\ mat$
assumes $A: A \in carrier-mat\ n\ n$ **and** $B: B \in carrier-mat\ n\ n$
shows $trace\ (A - B) = trace\ A - trace\ B$ (**is** $?lhs = ?rhs$)
 $\langle proof \rangle$

lemma *trace-smult*:
assumes $A \in carrier-mat\ n\ n$
shows $trace\ (c \cdot_m A) = c * trace\ A$
 $\langle proof \rangle$

1.2 Conjugate of a vector

lemma *conjugate-scalar-prod*:
fixes $v w :: 'a::conjugatable-ring\ vec$
assumes $dim-vec\ v = dim-vec\ w$
shows $conjugate\ (v \cdot w) = conjugate\ v \cdot conjugate\ w$
 $\langle proof \rangle$

1.3 Inner product

abbreviation *inner-prod* $:: 'a\ vec \Rightarrow 'a\ vec \Rightarrow 'a :: conjugatable-ring$
where $inner-prod\ v\ w \equiv w \cdot c\ v$

lemma *conjugate-scalar-prod-Im* [*simp*]:
 $Im\ (v \cdot c\ v) = 0$
 $\langle proof \rangle$

lemma *conjugate-scalar-prod-Re* [*simp*]:
 $Re\ (v \cdot c\ v) \geq 0$
 $\langle proof \rangle$

lemma *self-cscalar-prod-geq-0*:
fixes $v :: 'a::conjugatable-ordered-field\ vec$
shows $v \cdot c\ v \geq 0$
 $\langle proof \rangle$

lemma *inner-prod-distrib-left*:

fixes $u\ v\ w :: ('a::\text{conjugatable-field})\ \text{vec}$
assumes $\text{dim}u: u \in \text{carrier-vec}\ n$ **and** $\text{dim}v:v \in \text{carrier-vec}\ n$ **and** $\text{dim}w: w \in \text{carrier-vec}\ n$
shows $\text{inner-prod}\ (v + w)\ u = \text{inner-prod}\ v\ u + \text{inner-prod}\ w\ u$ (**is** $?lhs = ?rhs$)
<proof>

lemma *inner-prod-distrib-right*:

fixes $u\ v\ w :: ('a::\text{conjugatable-field})\ \text{vec}$
assumes $\text{dim}u: u \in \text{carrier-vec}\ n$ **and** $\text{dim}v:v \in \text{carrier-vec}\ n$ **and** $\text{dim}w: w \in \text{carrier-vec}\ n$
shows $\text{inner-prod}\ u\ (v + w) = \text{inner-prod}\ u\ v + \text{inner-prod}\ u\ w$ (**is** $?lhs = ?rhs$)
<proof>

lemma *inner-prod-minus-distrib-right*:

fixes $u\ v\ w :: ('a::\text{conjugatable-field})\ \text{vec}$
assumes $\text{dim}u: u \in \text{carrier-vec}\ n$ **and** $\text{dim}v:v \in \text{carrier-vec}\ n$ **and** $\text{dim}w: w \in \text{carrier-vec}\ n$
shows $\text{inner-prod}\ u\ (v - w) = \text{inner-prod}\ u\ v - \text{inner-prod}\ u\ w$ (**is** $?lhs = ?rhs$)
<proof>

lemma *inner-prod-smult-right*:

fixes $u\ v :: \text{complex}\ \text{vec}$
assumes $\text{dim}u: u \in \text{carrier-vec}\ n$ **and** $\text{dim}v:v \in \text{carrier-vec}\ n$
shows $\text{inner-prod}\ (a \cdot_v u)\ v = \text{conjugate}\ a * \text{inner-prod}\ u\ v$ (**is** $?lhs = ?rhs$)
<proof>

lemma *inner-prod-smult-left*:

fixes $u\ v :: \text{complex}\ \text{vec}$
assumes $\text{dim}u: u \in \text{carrier-vec}\ n$ **and** $\text{dim}v: v \in \text{carrier-vec}\ n$
shows $\text{inner-prod}\ u\ (a \cdot_v v) = a * \text{inner-prod}\ u\ v$ (**is** $?lhs = ?rhs$)
<proof>

lemma *inner-prod-smult-left-right*:

fixes $u\ v :: \text{complex}\ \text{vec}$
assumes $\text{dim}u: u \in \text{carrier-vec}\ n$ **and** $\text{dim}v: v \in \text{carrier-vec}\ n$
shows $\text{inner-prod}\ (a \cdot_v u)\ (b \cdot_v v) = \text{conjugate}\ a * b * \text{inner-prod}\ u\ v$ (**is** $?lhs = ?rhs$)
<proof>

lemma *inner-prod-swap*:

fixes $x\ y :: \text{complex}\ \text{vec}$
assumes $y \in \text{carrier-vec}\ n$ **and** $x \in \text{carrier-vec}\ n$
shows $\text{inner-prod}\ y\ x = \text{conjugate}\ (\text{inner-prod}\ x\ y)$
<proof>

Cauchy-Schwarz theorem for complex vectors. This is analogous to `aux_Cauchy` and `Cauchy_Schwarz_ineq` in `Generalizations2.thy` in `QR_De-composition`. Consider merging and moving to Isabelle library.

lemma *aux-Cauchy*:

fixes $x\ y :: \text{complex vec}$
assumes $x \in \text{carrier-vec } n$ **and** $y \in \text{carrier-vec } n$
shows $0 \leq \text{inner-prod } x\ x + a * (\text{inner-prod } x\ y) + (\text{cnj } a) * ((\text{cnj } (\text{inner-prod } x\ y)) + a * (\text{inner-prod } y\ y))$
<proof>

lemma *Cauchy-Schwarz-complex-vec*:

fixes $x\ y :: \text{complex vec}$
assumes $x \in \text{carrier-vec } n$ **and** $y \in \text{carrier-vec } n$
shows $\text{inner-prod } x\ y * \text{inner-prod } y\ x \leq \text{inner-prod } x\ x * \text{inner-prod } y\ y$
<proof>

1.4 Hermitian adjoint of a matrix

abbreviation *adjoint* **where** $\text{adjoint} \equiv \text{mat-adjoint}$

lemma *adjoint-dim-row [simp]*:

$\text{dim-row } (\text{adjoint } A) = \text{dim-col } A$ *<proof>*

lemma *adjoint-dim-col [simp]*:

$\text{dim-col } (\text{adjoint } A) = \text{dim-row } A$ *<proof>*

lemma *adjoint-dim*:

$A \in \text{carrier-mat } n\ n \implies \text{adjoint } A \in \text{carrier-mat } n\ n$
<proof>

lemma *adjoint-def*:

$\text{adjoint } A = \text{mat } (\text{dim-col } A) (\text{dim-row } A) (\lambda(i,j). \text{conjugate } (A\ \$\$ (j,i)))$
<proof>

lemma *adjoint-eval*:

assumes $i < \text{dim-col } A$ $j < \text{dim-row } A$
shows $(\text{adjoint } A)\ \$\$ (i,j) = \text{conjugate } (A\ \$\$ (j,i))$
<proof>

lemma *adjoint-row*:

assumes $i < \text{dim-col } A$
shows $\text{row } (\text{adjoint } A)\ i = \text{conjugate } (\text{col } A\ i)$
<proof>

lemma *adjoint-col*:

assumes $i < \text{dim-row } A$
shows $\text{col } (\text{adjoint } A)\ i = \text{conjugate } (\text{row } A\ i)$
<proof>

The identity $\langle v, A\ w \rangle = \langle A^* v, w \rangle$

lemma *adjoint-def-alter*:

fixes $v\ w :: 'a::\text{conjugatable-field vec}$
and $A :: 'a::\text{conjugatable-field mat}$

assumes $dims: v \in carrier-vec\ n\ w \in carrier-vec\ m\ A \in carrier-mat\ n\ m$
shows $inner-prod\ v\ (A *_{\nu}\ w) = inner-prod\ (adjoint\ A *_{\nu}\ v)\ w$ (**is** ?lhs = ?rhs)
 <proof>

lemma *adjoint-one*:
shows $adjoint\ (1_m\ n) = (1_m\ n::complex\ mat)$
 <proof>

lemma *adjoint-scale*:
fixes $A :: 'a::conjugatable-field\ mat$
shows $adjoint\ (a \cdot_m\ A) = (conjugate\ a) \cdot_m\ adjoint\ A$
 <proof>

lemma *adjoint-add*:
fixes $A\ B :: 'a::conjugatable-field\ mat$
assumes $A \in carrier-mat\ n\ m\ B \in carrier-mat\ n\ m$
shows $adjoint\ (A + B) = adjoint\ A + adjoint\ B$
 <proof>

lemma *adjoint-minus*:
fixes $A\ B :: 'a::conjugatable-field\ mat$
assumes $A \in carrier-mat\ n\ m\ B \in carrier-mat\ n\ m$
shows $adjoint\ (A - B) = adjoint\ A - adjoint\ B$
 <proof>

lemma *adjoint-mult*:
fixes $A\ B :: 'a::conjugatable-field\ mat$
assumes $A \in carrier-mat\ n\ m\ B \in carrier-mat\ m\ l$
shows $adjoint\ (A * B) = adjoint\ B * adjoint\ A$
 <proof>

lemma *adjoint-adjoint*:
fixes $A :: 'a::conjugatable-field\ mat$
shows $adjoint\ (adjoint\ A) = A$
 <proof>

lemma *trace-adjoint-positive*:
fixes $A :: complex\ mat$
shows $trace\ (A * adjoint\ A) \geq 0$
 <proof>

1.5 Algebraic manipulations on matrices

lemma *right-add-zero-mat[simp]*:
 $(A :: 'a :: monoid-add\ mat) \in carrier-mat\ nr\ nc \implies A + 0_m\ nr\ nc = A$
 <proof>

lemma *add-carrier-mat'*:
 $A \in carrier-mat\ nr\ nc \implies B \in carrier-mat\ nr\ nc \implies A + B \in carrier-mat\ nr$

nc
<proof>

lemma *minus-carrier-mat'*:

$A \in \text{carrier-mat } nr \ nc \implies B \in \text{carrier-mat } nr \ nc \implies A - B \in \text{carrier-mat } nr \ nc$
<proof>

lemma *swap-plus-mat*:

fixes $A \ B \ C :: 'a::\text{semiring-1} \ \text{mat}$
assumes $A \in \text{carrier-mat } n \ n \ B \in \text{carrier-mat } n \ n \ C \in \text{carrier-mat } n \ n$
shows $A + B + C = A + C + B$
<proof>

lemma *uminus-mat*:

fixes $A :: \text{complex mat}$
assumes $A \in \text{carrier-mat } n \ n$
shows $-A = (-1) \cdot_m A$
<proof>

<ML>

lemma *mat-assoc-test*:

fixes $A \ B \ C \ D :: \text{complex mat}$
assumes $A \in \text{carrier-mat } n \ n \ B \in \text{carrier-mat } n \ n \ C \in \text{carrier-mat } n \ n \ D \in \text{carrier-mat } n \ n$
shows
 $(A * B) * (C * D) = A * B * C * D$
 $\text{adjoint } (A * \text{adjoint } B) * C = B * (\text{adjoint } A * C)$
 $A * 1_m \ n * 1_m \ n * B * 1_m \ n = A * B$
 $(A - B) + (B - C) = A + (-B) + B + (-C)$
 $A + (B - C) = A + B - C$
 $A - (B + C + D) = A - B - C - D$
 $(A + B) * (B + C) = A * B + B * B + A * C + B * C$
 $A - B = A + (-1) \cdot_m B$
 $A * (B - C) * D = A * B * D - A * C * D$
 $\text{trace } (A * B * C) = \text{trace } (B * C * A)$
 $\text{trace } (A * B * C * D) = \text{trace } (C * D * A * B)$
 $\text{trace } (A + B * C) = \text{trace } A + \text{trace } (C * B)$
 $A + B = B + A$
 $A + B + C = C + B + A$
 $A + B + (C + D) = A + C + (B + D)$
<proof>

1.6 Hermitian matrices

A Hermitian matrix is a matrix that is equal to its Hermitian adjoint.

definition *hermitian* $:: 'a::\text{conjugatable-field} \ \text{mat} \Rightarrow \text{bool}$ **where**
 $\text{hermitian } A \longleftrightarrow (\text{adjoint } A = A)$

lemma *hermitian-one*:
shows *hermitian* $((1_m\ n)::('a::\text{conjugatable-field mat}))$
 $\langle\text{proof}\rangle$

1.7 Inverse matrices

lemma *inverts-mat-symm*:
fixes $A\ B :: 'a::\text{field mat}$
assumes $\text{dim}: A \in \text{carrier-mat } n\ n\ B \in \text{carrier-mat } n\ n$
and $AB: \text{inverts-mat } A\ B$
shows $\text{inverts-mat } B\ A$
 $\langle\text{proof}\rangle$

lemma *inverts-mat-unique*:
fixes $A\ B\ C :: 'a::\text{field mat}$
assumes $\text{dim}: A \in \text{carrier-mat } n\ n\ B \in \text{carrier-mat } n\ n\ C \in \text{carrier-mat } n\ n$
and $AB: \text{inverts-mat } A\ B$ **and** $AC: \text{inverts-mat } A\ C$
shows $B = C$
 $\langle\text{proof}\rangle$

1.8 Unitary matrices

A unitary matrix is a matrix whose Hermitian adjoint is also its inverse.

definition *unitary* $:: 'a::\text{conjugatable-field mat} \Rightarrow \text{bool}$ **where**
 $\text{unitary } A \iff A \in \text{carrier-mat } (\text{dim-row } A)\ (\text{dim-row } A) \wedge \text{inverts-mat } A\ (\text{adjoint } A)$

lemma *unitaryD2*:
assumes $A \in \text{carrier-mat } n\ n$
shows $\text{unitary } A \implies \text{inverts-mat } (\text{adjoint } A)\ A$
 $\langle\text{proof}\rangle$

lemma *unitary-simps [simp]*:
 $A \in \text{carrier-mat } n\ n \implies \text{unitary } A \implies \text{adjoint } A * A = 1_m\ n$
 $A \in \text{carrier-mat } n\ n \implies \text{unitary } A \implies A * \text{adjoint } A = 1_m\ n$
 $\langle\text{proof}\rangle$

lemma *unitary-adjoint [simp]*:
assumes $A \in \text{carrier-mat } n\ n\ \text{unitary } A$
shows $\text{unitary } (\text{adjoint } A)$
 $\langle\text{proof}\rangle$

lemma *unitary-one*:
shows *unitary* $((1_m\ n)::('a::\text{conjugatable-field mat}))$
 $\langle\text{proof}\rangle$

lemma *unitary-zero*:
fixes $A :: 'a::\text{conjugatable-field mat}$

assumes $A \in \text{carrier-mat } 0 \ 0$
shows *unitary* A
 $\langle \text{proof} \rangle$

lemma *unitary-elim*:

assumes $\text{dims}: A \in \text{carrier-mat } n \ n \ B \in \text{carrier-mat } n \ n \ P \in \text{carrier-mat } n \ n$
and $uP: \text{unitary } P$ **and** $\text{eq}: P * A * \text{adjoint } P = P * B * \text{adjoint } P$
shows $A = B$
 $\langle \text{proof} \rangle$

lemma *unitary-is-corthogonal*:

fixes $U :: 'a::\text{conjugatable-field mat}$
assumes $\text{dim}: U \in \text{carrier-mat } n \ n$
and $U: \text{unitary } U$
shows *corthogonal-mat* U
 $\langle \text{proof} \rangle$

lemma *unitary-times-unitary*:

fixes $P \ Q :: 'a::\text{conjugatable-field mat}$
assumes $\text{dim}: P \in \text{carrier-mat } n \ n \ Q \in \text{carrier-mat } n \ n$
and $uP: \text{unitary } P$ **and** $uQ: \text{unitary } Q$
shows *unitary* $(P * Q)$
 $\langle \text{proof} \rangle$

lemma *unitary-operator-keep-trace*:

fixes $U \ A :: \text{complex mat}$
assumes $dU: U \in \text{carrier-mat } n \ n$ **and** $dA: A \in \text{carrier-mat } n \ n$ **and** $u: \text{unitary } U$
shows $\text{trace } A = \text{trace } (\text{adjoint } U * A * U)$
 $\langle \text{proof} \rangle$

1.9 Normalization of vectors

definition *vec-norm* $:: \text{complex vec} \Rightarrow \text{complex}$ **where**
 $\text{vec-norm } v \equiv \text{csqrt } (v \cdot c \ v)$

lemma *vec-norm-geq-0*:

fixes $v :: \text{complex vec}$
shows $\text{vec-norm } v \geq 0$
 $\langle \text{proof} \rangle$

lemma *vec-norm-zero*:

fixes $v :: \text{complex vec}$
assumes $\text{dim}: v \in \text{carrier-vec } n$
shows $\text{vec-norm } v = 0 \longleftrightarrow v = 0_v \ n$
 $\langle \text{proof} \rangle$

lemma *vec-norm-ge-0*:

fixes $v :: \text{complex vec}$

assumes $\text{dim-}v: v \in \text{carrier-vec } n$ **and** $\text{neq0}: v \neq 0_v n$
shows $\text{vec-norm } v > 0$
 $\langle \text{proof} \rangle$

definition $\text{vec-normalize} :: \text{complex vec} \Rightarrow \text{complex vec}$ **where**
 $\text{vec-normalize } v = (\text{if } (v = 0_v (\text{dim-vec } v)) \text{ then } v \text{ else } 1 / (\text{vec-norm } v) \cdot_v v)$

lemma $\text{normalized-vec-dim}[\text{simp}]$:
assumes $(v :: \text{complex vec}) \in \text{carrier-vec } n$
shows $\text{vec-normalize } v \in \text{carrier-vec } n$
 $\langle \text{proof} \rangle$

lemma $\text{vec-eq-norm-smult-normalized}$:
shows $v = \text{vec-norm } v \cdot_v \text{vec-normalize } v$
 $\langle \text{proof} \rangle$

lemma $\text{normalized-cscalar-prod}$:
fixes $v w :: \text{complex vec}$
assumes $\text{dim-}v: v \in \text{carrier-vec } n$ **and** $\text{dim-}w: w \in \text{carrier-vec } n$
shows $v \cdot c w = (\text{vec-norm } v * \text{vec-norm } w) * (\text{vec-normalize } v \cdot c \text{vec-normalize } w)$
 $\langle \text{proof} \rangle$

lemma $\text{normalized-vec-norm}$:
fixes $v :: \text{complex vec}$
assumes $\text{dim-}v: v \in \text{carrier-vec } n$
and $\text{neq0}: v \neq 0_v n$
shows $\text{vec-normalize } v \cdot c \text{vec-normalize } v = 1$
 $\langle \text{proof} \rangle$

lemma normalize-zero :
assumes $v \in \text{carrier-vec } n$
shows $\text{vec-normalize } v = 0_v n \iff v = 0_v n$
 $\langle \text{proof} \rangle$

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

1.10 Spectral decomposition of normal complex matrices

lemma $\text{normalize-keep-corthogonal}$:
fixes $vs :: \text{complex vec list}$
assumes $\text{cor}: \text{corthogonal } vs$ **and** $\text{dims}: \text{set } vs \subseteq \text{carrier-vec } n$
shows $\text{corthogonal } (\text{map } \text{vec-normalize } vs)$
 $\langle \text{proof} \rangle$

lemma $\text{normalized-corthogonal-mat-is-unitary}$:
assumes $W: \text{set } ws \subseteq \text{carrier-vec } n$

and *orth*: *corthogonal ws*
and *len*: *length ws = n*
shows *unitary (mat-of-cols n (map vec-normalize ws)) (is unitary ?W)*
<proof>

lemma *normalize-keep-eigenvector*:
assumes *ev*: *eigenvector A v e*
and *dim*: *A ∈ carrier-mat n n v ∈ carrier-vec n*
shows *eigenvector A (vec-normalize v) e*
<proof>

lemma *four-block-mat-adjoint*:
fixes *A B C D :: 'a::conjugatable-field mat*
assumes *dim*: *A ∈ carrier-mat nr1 nc1 B ∈ carrier-mat nr1 nc2*
C ∈ carrier-mat nr2 nc1 D ∈ carrier-mat nr2 nc2
shows *adjoint (four-block-mat A B C D)*
= four-block-mat (adjoint A) (adjoint C) (adjoint B) (adjoint D)
<proof>

fun *unitary-schur-decomposition* :: *complex mat ⇒ complex list ⇒ complex mat ×*
complex mat × complex mat **where**
unitary-schur-decomposition A [] = (A, 1_m (dim-row A), 1_m (dim-row A))
| *unitary-schur-decomposition A (e # es) = (let*
n = dim-row A;
n1 = n - 1;
v' = find-eigenvector A e;
v = vec-normalize v';
ws0 = gram-schmidt n (basis-completion v);
ws = map vec-normalize ws0;
W = mat-of-cols n ws;
W' = corthogonal-inv W;
*A' = W' * A * W;*
(A1,A2,A0,A3) = split-block A' 1 1;
(B,P,Q) = unitary-schur-decomposition A3 es;
z-row = (0_m 1 n1);
z-col = (0_m n1 1);
one-1 = 1_m 1
*in (four-block-mat A1 (A2 * P) A0 B,*
*W * four-block-mat one-1 z-row z-col P,*
*four-block-mat one-1 z-row z-col Q * W'))*

theorem *unitary-schur-decomposition*:
assumes *A*: *(A::complex mat) ∈ carrier-mat n n*
and *c*: *char-poly A = (∏ (e :: complex) ← es. [:- e, 1:])*
and *B*: *unitary-schur-decomposition A es = (B,P,Q)*
shows *similar-mat-wit A B P Q ∧ upper-triangular B ∧ diag-mat B = es ∧*
unitary P ∧ (Q = adjoint P)
<proof>

lemma *complex-mat-char-poly-factorizable:*

fixes $A :: \text{complex mat}$
assumes $A \in \text{carrier-mat } n \ n$
shows $\exists as. \text{char-poly } A = (\prod a \leftarrow as. [:- a, 1:]) \wedge \text{length } as = n$
<proof>

lemma *complex-mat-has-unitary-schur-decomposition:*

fixes $A :: \text{complex mat}$
assumes $A \in \text{carrier-mat } n \ n$
shows $\exists B \ P \ es. \text{similar-mat-wit } A \ B \ P \ (\text{adjoint } P) \wedge \text{unitary } P$
 $\wedge \text{char-poly } A = (\prod (e :: \text{complex}) \leftarrow es. [:- e, 1:]) \wedge \text{diag-mat } B = es$
<proof>

lemma *normal-upper-triangular-matrix-is-diagonal:*

fixes $A :: 'a::\text{conjugatable-ordered-field mat}$
assumes $A \in \text{carrier-mat } n \ n$
and $tri: \text{upper-triangular } A$
and $norm: A * \text{adjoint } A = \text{adjoint } A * A$
shows $\text{diagonal-mat } A$
<proof>

lemma *normal-complex-mat-has-spectral-decomposition:*

assumes $A: (A::\text{complex mat}) \in \text{carrier-mat } n \ n$
and $normal: A * \text{adjoint } A = \text{adjoint } A * A$
and $c: \text{char-poly } A = (\prod (e :: \text{complex}) \leftarrow es. [:- e, 1:])$
and $B: \text{unitary-schur-decomposition } A \ es = (B, P, Q)$
shows $\text{similar-mat-wit } A \ B \ P \ (\text{adjoint } P) \wedge \text{diagonal-mat } B \wedge \text{diag-mat } B = es$
 $\wedge \text{unitary } P$
<proof>

lemma *complex-mat-has-jordan-nf:*

fixes $A :: \text{complex mat}$
assumes $A \in \text{carrier-mat } n \ n$
shows $\exists n\text{-as. jordan-nf } A \ n\text{-as}$
<proof>

lemma *hermitian-is-normal:*

assumes $\text{hermitian } A$
shows $A * \text{adjoint } A = \text{adjoint } A * A$
<proof>

lemma *hermitian-eigenvalue-real:*

assumes $dim: (A::\text{complex mat}) \in \text{carrier-mat } n \ n$
and $hA: \text{hermitian } A$
and $c: \text{char-poly } A = (\prod (e :: \text{complex}) \leftarrow es. [:- e, 1:])$
and $B: \text{unitary-schur-decomposition } A \ es = (B, P, Q)$
shows $\text{similar-mat-wit } A \ B \ P \ (\text{adjoint } P) \wedge \text{diagonal-mat } B \wedge \text{diag-mat } B = es$
 $\wedge \text{unitary } P \wedge (\forall i < n. B\$\$(i, i) \in \text{Reals})$
<proof>

lemma *hermitian-inner-prod-real*:
assumes $dimA: (A::complex\ mat) \in carrier\ mat\ n\ n$
and $dimv: v \in carrier\ vec\ n$
and $hA: hermitian\ A$
shows $inner\ prod\ v\ (A\ *_v\ v) \in Reals$
 $\langle proof \rangle$

lemma *unit-vec-bracket*:
fixes $A :: complex\ mat$
assumes $dimA: A \in carrier\ mat\ n\ n$ **and** $i: i < n$
shows $inner\ prod\ (unit\ vec\ n\ i)\ (A\ *_v\ (unit\ vec\ n\ i)) = A\ \(i, i)
 $\langle proof \rangle$

lemma *spectral-decomposition-extract-diag*:
fixes $P\ B :: complex\ mat$
assumes $dimP: P \in carrier\ mat\ n\ n$ **and** $dimB: B \in carrier\ mat\ n\ n$
and $uP: unitary\ P$ **and** $dB: diagonal\ mat\ B$ **and** $i: i < n$
shows $inner\ prod\ (col\ P\ i)\ (P\ *_v\ B\ *_v\ (adjoint\ P)\ *_v\ (col\ P\ i)) = B\ \(i, i)
 $\langle proof \rangle$

lemma *hermitian-inner-prod-zero*:
fixes $A :: complex\ mat$
assumes $dimA: A \in carrier\ mat\ n\ n$ **and** $hA: hermitian\ A$
and $zero: \forall v \in carrier\ vec\ n. inner\ prod\ v\ (A\ *_v\ v) = 0$
shows $A = 0_m\ n\ n$
 $\langle proof \rangle$

lemma *complex-mat-decomposition-to-hermitian*:
fixes $A :: complex\ mat$
assumes $dim: A \in carrier\ mat\ n\ n$
shows $\exists B\ C. hermitian\ B \wedge hermitian\ C \wedge A = B + i \cdot_m C \wedge B \in carrier\ mat\ n\ n \wedge C \in carrier\ mat\ n\ n$
 $\langle proof \rangle$

1.11 Outer product

definition *outer-prod* :: $'a::conjugatable\ field\ vec \Rightarrow 'a\ vec \Rightarrow 'a\ mat$ **where**
 $outer\ prod\ v\ w = mat\ (dim\ vec\ v)\ 1\ (\lambda(i, j). v\ \$\ i) \cdot mat\ 1\ (dim\ vec\ w)\ (\lambda(i, j). (conjugate\ w)\ \$\ j)$

lemma *outer-prod-dim[simp]*:
fixes $v\ w :: 'a::conjugatable\ field\ vec$
assumes $v: v \in carrier\ vec\ n$ **and** $w: w \in carrier\ vec\ m$
shows $outer\ prod\ v\ w \in carrier\ mat\ n\ m$
 $\langle proof \rangle$

lemma *mat-of-vec-mult-eq-scalar-prod*:
fixes $v\ w :: 'a::conjugatable\ field\ vec$

assumes $v \in \text{carrier-vec } n$ **and** $w \in \text{carrier-vec } n$
shows $\text{mat } 1 \ 1 \ (\lambda(i, j). (\text{conjugate } v) \ \$ \ j) * \text{mat } (\text{dim-vec } w) \ 1 \ (\lambda(i, j). w \ \$ \ i)$
 $= \text{mat } 1 \ 1 \ (\lambda k. \text{inner-prod } v \ w)$
 $\langle \text{proof} \rangle$

lemma *one-dim-mat-mult-is-scale*:
fixes $A \ B :: 'a::\text{conjugatable-field } \text{mat}$
assumes $B \in \text{carrier-mat } 1 \ n$
shows $(\text{mat } 1 \ 1 \ (\lambda k. a)) * B = a \cdot_m B$
 $\langle \text{proof} \rangle$

lemma *outer-prod-mult-outer-prod*:
fixes $a \ b \ c \ d :: 'a::\text{conjugatable-field } \text{vec}$
assumes $a: a \in \text{carrier-vec } d1$ **and** $b: b \in \text{carrier-vec } d2$
and $c: c \in \text{carrier-vec } d2$ **and** $d: d \in \text{carrier-vec } d3$
shows $\text{outer-prod } a \ b * \text{outer-prod } c \ d = \text{inner-prod } b \ c \cdot_m \text{outer-prod } a \ d$
 $\langle \text{proof} \rangle$

lemma *index-outer-prod*:
fixes $v \ w :: 'a::\text{conjugatable-field } \text{vec}$
assumes $v: v \in \text{carrier-vec } n$ **and** $w: w \in \text{carrier-vec } m$
and $ij: i < n \ j < m$
shows $(\text{outer-prod } v \ w) \ \$ \ \$ (i, j) = v \ \$ \ i * \text{conjugate } (w \ \$ \ j)$
 $\langle \text{proof} \rangle$

lemma *mat-of-vec-mult-vec*:
fixes $a \ b \ c :: 'a::\text{conjugatable-field } \text{vec}$
assumes $a: a \in \text{carrier-vec } d$ **and** $b: b \in \text{carrier-vec } d$
shows $\text{mat } 1 \ d \ (\lambda(i, j). (\text{conjugate } a) \ \$ \ j) *_v b = \text{vec } 1 \ (\lambda k. \text{inner-prod } a \ b)$
 $\langle \text{proof} \rangle$

lemma *mat-of-vec-mult-one-dim-vec*:
fixes $a \ b :: 'a::\text{conjugatable-field } \text{vec}$
assumes $a: a \in \text{carrier-vec } d$
shows $\text{mat } d \ 1 \ (\lambda(i, j). a \ \$ \ i) *_v \text{vec } 1 \ (\lambda k. c) = c \cdot_v a$
 $\langle \text{proof} \rangle$

lemma *outer-prod-mult-vec*:
fixes $a \ b \ c :: 'a::\text{conjugatable-field } \text{vec}$
assumes $a: a \in \text{carrier-vec } d1$ **and** $b: b \in \text{carrier-vec } d2$
and $c: c \in \text{carrier-vec } d2$
shows $\text{outer-prod } a \ b *_v c = \text{inner-prod } b \ c \cdot_v a$
 $\langle \text{proof} \rangle$

lemma *trace-outer-prod-right*:
fixes $A :: 'a::\text{conjugatable-field } \text{mat}$ **and** $v \ w :: 'a \ \text{vec}$
assumes $A: A \in \text{carrier-mat } n \ n$
and $v: v \in \text{carrier-vec } n$ **and** $w: w \in \text{carrier-vec } n$

shows $\text{trace } (A * \text{outer-prod } v \ w) = \text{inner-prod } w \ (A *_{\nu} v)$ (**is** $?lhs = ?rhs$)
 ⟨proof⟩

lemma *trace-outer-prod*:

fixes $v \ w :: 'a::\text{conjugatable-field } \text{vec}$
assumes $v: v \in \text{carrier-vec } n$ **and** $w: w \in \text{carrier-vec } n$
shows $\text{trace } (\text{outer-prod } v \ w) = \text{inner-prod } w \ v$ (**is** $?lhs = ?rhs$)
 ⟨proof⟩

lemma *inner-prod-outer-prod*:

fixes $a \ b \ c \ d :: 'a::\text{conjugatable-field } \text{vec}$
assumes $a: a \in \text{carrier-vec } n$ **and** $b: b \in \text{carrier-vec } n$
and $c: c \in \text{carrier-vec } m$ **and** $d: d \in \text{carrier-vec } m$
shows $\text{inner-prod } a \ (\text{outer-prod } b \ c *_{\nu} d) = \text{inner-prod } a \ b * \text{inner-prod } c \ d$ (**is** $?lhs = ?rhs$)
 ⟨proof⟩

1.12 Semi-definite matrices

definition *positive* :: $\text{complex mat} \Rightarrow \text{bool}$ **where**

$\text{positive } A \longleftrightarrow$
 $A \in \text{carrier-mat } (\text{dim-col } A) \ (\text{dim-col } A) \wedge$
 $(\forall v. \text{dim-vec } v = \text{dim-col } A \longrightarrow \text{inner-prod } v \ (A *_{\nu} v) \geq 0)$

lemma *positive-iff-normalized-vec*:

$\text{positive } A \longleftrightarrow$
 $A \in \text{carrier-mat } (\text{dim-col } A) \ (\text{dim-col } A) \wedge$
 $(\forall v. (\text{dim-vec } v = \text{dim-col } A \wedge \text{vec-norm } v = 1) \longrightarrow \text{inner-prod } v \ (A *_{\nu} v) \geq 0)$
 ⟨proof⟩

lemma *positive-is-hermitian*:

fixes $A :: \text{complex mat}$
assumes $pA: \text{positive } A$
shows $\text{hermitian } A$
 ⟨proof⟩

lemma *positive-eigenvalue-positive*:

assumes $\text{dim}A: (A::\text{complex mat}) \in \text{carrier-mat } n \ n$
and $pA: \text{positive } A$
and $c: \text{char-poly } A = (\prod (e :: \text{complex}) \leftarrow \text{es. } [:- e, 1:])$
and $B: \text{unitary-schur-decomposition } A \ \text{es} = (B, P, Q)$
shows $\bigwedge i. i < n \Longrightarrow B\$\$(i, i) \geq 0$
 ⟨proof⟩

lemma *diag-mat-mult-diag-mat*:

fixes $B \ D :: 'a::\text{semiring-0 } \text{mat}$
assumes $\text{dim}B: B \in \text{carrier-mat } n \ n$ **and** $\text{dim}D: D \in \text{carrier-mat } n \ n$
and $dB: \text{diagonal-mat } B$ **and** $dD: \text{diagonal-mat } D$

shows $B * D = \text{mat } n \ n \ (\lambda(i,j). (\text{if } i = j \text{ then } (B\$\$(i, i)) * (D\$\$(i, i)) \text{ else } 0))$
(proof)

lemma *positive-only-if-decomp*:

assumes $\text{dim}A: A \in \text{carrier-mat } n \ n$ **and** $pA: \text{positive } A$

shows $\exists M \in \text{carrier-mat } n \ n. M * \text{adjoint } M = A$

(proof)

lemma *positive-if-decomp*:

assumes $\text{dim}A: A \in \text{carrier-mat } n \ n$ **and** $\exists M. M * \text{adjoint } M = A$

shows $\text{positive } A$

(proof)

lemma *positive-iff-decomp*:

assumes $\text{dim}A: A \in \text{carrier-mat } n \ n$

shows $\text{positive } A \longleftrightarrow (\exists M \in \text{carrier-mat } n \ n. M * \text{adjoint } M = A)$

(proof)

lemma *positive-dim-eq*:

assumes $\text{positive } A$

shows $\text{dim-row } A = \text{dim-col } A$

(proof)

lemma *positive-zero*:

$\text{positive } (0_m \ n \ n)$

(proof)

lemma *positive-one*:

$\text{positive } (1_m \ n)$

(proof)

lemma *positive-antisym*:

assumes $pA: \text{positive } A$ **and** $pnA: \text{positive } (-A)$

shows $A = 0_m \ (\text{dim-col } A) \ (\text{dim-col } A)$

(proof)

lemma *positive-add*:

assumes $pA: \text{positive } A$ **and** $pB: \text{positive } B$

and $\text{dim}A: A \in \text{carrier-mat } n \ n$ **and** $\text{dim}B: B \in \text{carrier-mat } n \ n$

shows $\text{positive } (A + B)$

(proof)

lemma *positive-trace*:

assumes $A \in \text{carrier-mat } n \ n$ **and** $\text{positive } A$

shows $\text{trace } A \geq 0$

(proof)

lemma *positive-close-under-left-right-mult-adjoint*:

fixes $M \ A :: \text{complex mat}$

assumes $dM: M \in \text{carrier-mat } n \ n$ **and** $dA: A \in \text{carrier-mat } n \ n$
and $pA: \text{positive } A$
shows $\text{positive } (M * A * \text{adjoint } M)$
 $\langle \text{proof} \rangle$

lemma *positive-same-outer-prod*:
fixes $v \ w :: \text{complex vec}$
assumes $v: v \in \text{carrier-vec } n$
shows $\text{positive } (\text{outer-prod } v \ v)$
 $\langle \text{proof} \rangle$

lemma *smult-smult-mat*:
fixes $k :: \text{complex}$ **and** $l :: \text{complex}$
assumes $A \in \text{carrier-mat } nr \ n$
shows $k \cdot_m (l \cdot_m A) = (k * l) \cdot_m A$ $\langle \text{proof} \rangle$

lemma *positive-smult*:
assumes $A \in \text{carrier-mat } n \ n$
and $\text{positive } A$
and $c \geq 0$
shows $\text{positive } (c \cdot_m A)$
 $\langle \text{proof} \rangle$

Version of previous theorem for real numbers

lemma *positive-scale*:
fixes $c :: \text{real}$
assumes $A \in \text{carrier-mat } n \ n$
and $\text{positive } A$
and $c \geq 0$
shows $\text{positive } (c \cdot_m A)$
 $\langle \text{proof} \rangle$

1.13 Löwner partial order

definition *lowner-le* :: $\text{complex mat} \Rightarrow \text{complex mat} \Rightarrow \text{bool}$ (**infix** \leq_L 50) **where**
 $A \leq_L B \iff \text{dim-row } A = \text{dim-row } B \wedge \text{dim-col } A = \text{dim-col } B \wedge \text{positive } (B - A)$

lemma *lowner-le-refl*:
assumes $A \in \text{carrier-mat } n \ n$
shows $A \leq_L A$
 $\langle \text{proof} \rangle$

lemma *lowner-le-antisym*:
assumes $A: A \in \text{carrier-mat } n \ n$ **and** $B: B \in \text{carrier-mat } n \ n$
and $L1: A \leq_L B$ **and** $L2: B \leq_L A$
shows $A = B$
 $\langle \text{proof} \rangle$

lemma *lowner-le-inner-prod-le*:

fixes $A B :: \text{complex mat}$ **and** $v :: \text{complex vec}$
assumes $A: A \in \text{carrier-mat } n \ n$ **and** $B: B \in \text{carrier-mat } n \ n$
and $v: v \in \text{carrier-vec } n$
and $A \leq_L B$
shows $\text{inner-prod } v (A *_v v) \leq \text{inner-prod } v (B *_v v)$
 $\langle \text{proof} \rangle$

lemma *lower-le-trans*:
fixes $A B C :: \text{complex mat}$
assumes $A: A \in \text{carrier-mat } n \ n$ **and** $B: B \in \text{carrier-mat } n \ n$ **and** $C: C \in \text{carrier-mat } n \ n$
and $L1: A \leq_L B$ **and** $L2: B \leq_L C$
shows $A \leq_L C$
 $\langle \text{proof} \rangle$

lemma *lower-le-imp-trace-le*:
assumes $A \in \text{carrier-mat } n \ n$ **and** $B \in \text{carrier-mat } n \ n$
and $A \leq_L B$
shows $\text{trace } A \leq \text{trace } B$
 $\langle \text{proof} \rangle$

lemma *lower-le-add*:
assumes $A \in \text{carrier-mat } n \ n$ $B \in \text{carrier-mat } n \ n$ $C \in \text{carrier-mat } n \ n$ $D \in \text{carrier-mat } n \ n$
and $A \leq_L B$ $C \leq_L D$
shows $A + C \leq_L B + D$
 $\langle \text{proof} \rangle$

lemma *lower-le-swap*:
assumes $A \in \text{carrier-mat } n \ n$ $B \in \text{carrier-mat } n \ n$
and $A \leq_L B$
shows $-B \leq_L -A$
 $\langle \text{proof} \rangle$

lemma *lower-le-minus*:
assumes $A \in \text{carrier-mat } n \ n$ $B \in \text{carrier-mat } n \ n$ $C \in \text{carrier-mat } n \ n$ $D \in \text{carrier-mat } n \ n$
and $A \leq_L B$ $C \leq_L D$
shows $A - D \leq_L B - C$
 $\langle \text{proof} \rangle$

lemma *outer-prod-le-one*:
assumes $v \in \text{carrier-vec } n$
and $\text{inner-prod } v v \leq 1$
shows $\text{outer-prod } v v \leq_L 1_m \ n$
 $\langle \text{proof} \rangle$

lemma *zero-lower-le-positiveD*:
fixes $A :: \text{complex mat}$

assumes $dA: A \in \text{carrier-mat } n \ n$ **and** $le: 0_m \ n \ n \leq_L A$
shows *positive A*
 $\langle \text{proof} \rangle$

lemma *zero-lowner-le-positiveI:*
fixes $A :: \text{complex mat}$
assumes $dA: A \in \text{carrier-mat } n \ n$ **and** $le: \text{positive } A$
shows $0_m \ n \ n \leq_L A$
 $\langle \text{proof} \rangle$

lemma *lowner-le-trans-positiveI:*
fixes $A \ B :: \text{complex mat}$
assumes $dA: A \in \text{carrier-mat } n \ n$ **and** $pA: \text{positive } A$ **and** $le: A \leq_L B$
shows *positive B*
 $\langle \text{proof} \rangle$

lemma *lowner-le-keep-under-measurement:*
fixes $M \ A \ B :: \text{complex mat}$
assumes $dM: M \in \text{carrier-mat } n \ n$ **and** $dA: A \in \text{carrier-mat } n \ n$ **and** $dB: B \in \text{carrier-mat } n \ n$
and $le: A \leq_L B$
shows $\text{adjoint } M * A * M \leq_L \text{adjoint } M * B * M$
 $\langle \text{proof} \rangle$

lemma *smult-distrib-left-minus-mat:*
fixes $A \ B :: 'a::\text{comm-ring-1 mat}$
assumes $A \in \text{carrier-mat } n \ n$ $B \in \text{carrier-mat } n \ n$
shows $c \cdot_m (B - A) = c \cdot_m B - c \cdot_m A$
 $\langle \text{proof} \rangle$

lemma *lowner-le-smultc:*
fixes $c :: \text{complex}$
assumes $c \geq 0$ $A \leq_L B$ $A \in \text{carrier-mat } n \ n$ $B \in \text{carrier-mat } n \ n$
shows $c \cdot_m A \leq_L c \cdot_m B$
 $\langle \text{proof} \rangle$

lemma *lowner-le-smult:*
fixes $c :: \text{real}$
assumes $c \geq 0$ $A \leq_L B$ $A \in \text{carrier-mat } n \ n$ $B \in \text{carrier-mat } n \ n$
shows $c \cdot_m A \leq_L c \cdot_m B$
 $\langle \text{proof} \rangle$

lemma *minus-smult-vec-distrib:*
fixes $w :: 'a::\text{comm-ring-1 vec}$
shows $(a - b) \cdot_v w = a \cdot_v w - b \cdot_v w$
 $\langle \text{proof} \rangle$

lemma *smult-mat-mult-mat-vec-assoc:*
fixes $A :: 'a::\text{comm-ring-1 mat}$

assumes $A: A \in \text{carrier-mat } n \ m$ **and** $w: w \in \text{carrier-vec } m$
shows $a \cdot_m A *_v w = a \cdot_v (A *_v w)$
 $\langle \text{proof} \rangle$

lemma *mult-mat-vec-smult-vec-assoc*:
fixes $A :: 'a::\text{comm-ring-1 } \text{mat}$
assumes $A: A \in \text{carrier-mat } n \ m$ **and** $w: w \in \text{carrier-vec } m$
shows $A *_v (a \cdot_v w) = a \cdot_v (A *_v w)$
 $\langle \text{proof} \rangle$

lemma *outer-prod-left-right-mat*:
fixes $A \ B :: \text{complex mat}$
assumes $du: u \in \text{carrier-vec } d2$ **and** $dv: v \in \text{carrier-vec } d3$
and $dA: A \in \text{carrier-mat } d1 \ d2$ **and** $dB: B \in \text{carrier-mat } d3 \ d4$
shows $A * (\text{outer-prod } u \ v) * B = (\text{outer-prod } (A *_v u) \ (\text{adjoint } B *_v v))$
 $\langle \text{proof} \rangle$

1.14 Density operators

definition *density-operator* $:: \text{complex mat} \Rightarrow \text{bool}$ **where**
density-operator $A \iff \text{positive } A \wedge \text{trace } A = 1$

definition *partial-density-operator* $:: \text{complex mat} \Rightarrow \text{bool}$ **where**
partial-density-operator $A \iff \text{positive } A \wedge \text{trace } A \leq 1$

lemma *pure-state-self-outer-prod-is-partial-density-operator*:
fixes $v :: \text{complex vec}$
assumes $\text{dim}v: v \in \text{carrier-vec } n$ **and** $nv: \text{vec-norm } v = 1$
shows *partial-density-operator* $(\text{outer-prod } v \ v)$
 $\langle \text{proof} \rangle$

lemma *lower-le-trace*:
assumes $A: A \in \text{carrier-mat } n \ n$
and $B: B \in \text{carrier-mat } n \ n$
shows $A \leq_L B \iff (\forall \rho \in \text{carrier-mat } n \ n. \text{partial-density-operator } \rho \implies \text{trace } (A * \rho) \leq \text{trace } (B * \rho))$
 $\langle \text{proof} \rangle$

lemma *lower-le-traceI*:
assumes $A \in \text{carrier-mat } n \ n$
and $B \in \text{carrier-mat } n \ n$
and $\bigwedge \rho. \rho \in \text{carrier-mat } n \ n \implies \text{partial-density-operator } \rho \implies \text{trace } (A * \rho) \leq \text{trace } (B * \rho)$
shows $A \leq_L B$
 $\langle \text{proof} \rangle$

lemma *trace-pdo-eq-imp-eq*:
assumes $A: A \in \text{carrier-mat } n \ n$

and $B: B \in \text{carrier-mat } n \ n$
and $\text{teq}: \bigwedge \varrho. \varrho \in \text{carrier-mat } n \ n \implies \text{partial-density-operator } \varrho \implies \text{trace } (A * \varrho) = \text{trace } (B * \varrho)$
shows $A = B$
 $\langle \text{proof} \rangle$

lemma *lower-le-traceD*:

assumes $A \in \text{carrier-mat } n \ n \ B \in \text{carrier-mat } n \ n \ \varrho \in \text{carrier-mat } n \ n$
and $A \leq_L B$
and $\text{partial-density-operator } \varrho$
shows $\text{trace } (A * \varrho) \leq \text{trace } (B * \varrho)$
 $\langle \text{proof} \rangle$

lemma *sum-only-one-neq-0*:

assumes $\text{finite } A$ **and** $j \in A$ **and** $\bigwedge i. i \in A \implies i \neq j \implies g \ i = 0$
shows $\text{sum } g \ A = g \ j$
 $\langle \text{proof} \rangle$

end

2 Matrix limits

theory *Matrix-Limit*

imports *Complex-Matrix*

begin

2.1 Definition of limit of matrices

definition *limit-mat* :: $(\text{nat} \Rightarrow \text{complex mat}) \Rightarrow \text{complex mat} \Rightarrow \text{nat} \Rightarrow \text{bool}$ **where**
 $\text{limit-mat } X \ A \ m \longleftrightarrow (\forall n. X \ n \in \text{carrier-mat } m \ m \wedge A \in \text{carrier-mat } m \ m \wedge$
 $(\forall i < m. \forall j < m. (\lambda n. (X \ n) \ \text{\$} \ \$ (i, j)) \longrightarrow (A \ \text{\$} \ \$ (i, j))))$

lemma *limit-mat-unique*:

assumes $\text{lim}A: \text{limit-mat } X \ A \ m$ **and** $\text{lim}B: \text{limit-mat } X \ B \ m$
shows $A = B$
 $\langle \text{proof} \rangle$

lemma *limit-mat-const*:

fixes $A :: \text{complex mat}$
assumes $A \in \text{carrier-mat } m \ m$
shows $\text{limit-mat } (\lambda k. A) \ A \ m$
 $\langle \text{proof} \rangle$

lemma *limit-mat-scale*:

fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A :: \text{complex mat}$
assumes $\text{lim}X: \text{limit-mat } X \ A \ m$
shows $\text{limit-mat } (\lambda n. c \cdot_m X \ n) \ (c \cdot_m A) \ m$
 $\langle \text{proof} \rangle$

lemma *limit-mat-add*:

fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $Y :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A :: \text{complex mat}$

and $m :: \text{nat}$ **and** $B :: \text{complex mat}$

assumes $\text{lim}X: \text{limit-mat } X \ A \ m$ **and** $\text{lim}Y: \text{limit-mat } Y \ B \ m$

shows $\text{limit-mat } (\lambda k. X \ k + Y \ k) \ (A + B) \ m$

<proof>

lemma *limit-mat-minus*:

fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $Y :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A :: \text{complex mat}$

and $m :: \text{nat}$ **and** $B :: \text{complex mat}$

assumes $\text{lim}X: \text{limit-mat } X \ A \ m$ **and** $\text{lim}Y: \text{limit-mat } Y \ B \ m$

shows $\text{limit-mat } (\lambda k. X \ k - Y \ k) \ (A - B) \ m$

<proof>

lemma *limit-mat-mult*:

fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $Y :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A :: \text{complex mat}$

and $m :: \text{nat}$ **and** $B :: \text{complex mat}$

assumes $\text{lim}X: \text{limit-mat } X \ A \ m$ **and** $\text{lim}Y: \text{limit-mat } Y \ B \ m$

shows $\text{limit-mat } (\lambda k. X \ k * Y \ k) \ (A * B) \ m$

<proof>

Adding matrix A to the sequence X

definition *mat-add-seq* :: $\text{complex mat} \Rightarrow (\text{nat} \Rightarrow \text{complex mat}) \Rightarrow \text{nat} \Rightarrow \text{complex mat}$ **where**

$\text{mat-add-seq } A \ X = (\lambda n. A + X \ n)$

lemma *mat-add-limit*:

fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A :: \text{complex mat}$ **and** $m :: \text{nat}$ **and** $B :: \text{complex mat}$

assumes $\text{dim}B: B \in \text{carrier-mat } m \ m$ **and** $\text{lim}X: \text{limit-mat } X \ A \ m$

shows $\text{limit-mat } (\text{mat-add-seq } B \ X) \ (B + A) \ m$

<proof>

lemma *mat-minus-limit*:

fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A :: \text{complex mat}$ **and** $m :: \text{nat}$ **and** $B :: \text{complex mat}$

assumes $\text{dim}B: B \in \text{carrier-mat } m \ m$ **and** $\text{lim}X: \text{limit-mat } X \ A \ m$

shows $\text{limit-mat } (\lambda n. B - X \ n) \ (B - A) \ m$

<proof>

Multiply matrix A by the sequence X

definition *mat-mult-seq* :: $\text{complex mat} \Rightarrow (\text{nat} \Rightarrow \text{complex mat}) \Rightarrow \text{nat} \Rightarrow \text{complex mat}$ **where**

$\text{mat-mult-seq } A \ X = (\lambda n. A * X \ n)$

lemma *mat-mult-limit*:

fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A B :: \text{complex mat}$ **and** $m :: \text{nat}$
assumes $\text{dim}B: B \in \text{carrier-mat } m \ m$ **and** $\text{lim}X: \text{limit-mat } X \ A \ m$
shows $\text{limit-mat } (\text{mat-mult-seq } B \ X) \ (B * A) \ m$
 $\langle \text{proof} \rangle$

lemma *mult-mat-limit*:

fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A B :: \text{complex mat}$ **and** $m :: \text{nat}$
assumes $\text{dim}B: B \in \text{carrier-mat } m \ m$ **and** $\text{lim}X: \text{limit-mat } X \ A \ m$
shows $\text{limit-mat } (\lambda k. X \ k * B) \ (A * B) \ m$
 $\langle \text{proof} \rangle$

lemma *quadratic-form-mat*:

fixes $A :: \text{complex mat}$ **and** $v :: \text{complex vec}$ **and** $m :: \text{nat}$
assumes $\text{dim}v: \text{dim-vec } v = m$ **and** $\text{dim}A: A \in \text{carrier-mat } m \ m$
shows $\text{inner-prod } v \ (A *_v v) = (\sum i=0..<m. (\sum j=0..<m. \text{conjugate } (v\$i) * A\$$(i, j) * v\$j))$
 $\langle \text{proof} \rangle$

lemma *sum-subtractff*:

fixes $h \ g :: \text{nat} \Rightarrow \text{nat} \Rightarrow 'a::\text{ab-group-add}$
shows $(\sum x \in A. \sum y \in B. h \ x \ y - g \ x \ y) = (\sum x \in A. \sum y \in B. h \ x \ y) - (\sum x \in A. \sum y \in B. g \ x \ y)$
 $\langle \text{proof} \rangle$

lemma *sum-abs-complex*:

fixes $h :: \text{nat} \Rightarrow \text{nat} \Rightarrow \text{complex}$
shows $\text{cmod } (\sum x \in A. \sum y \in B. h \ x \ y) \leq (\sum x \in A. \sum y \in B. \text{cmod}(h \ x \ y))$
 $\langle \text{proof} \rangle$

lemma *hermitian-mat-lim-is-hermitian*:

fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A :: \text{complex mat}$ **and** $m :: \text{nat}$
assumes $\text{lim}X: \text{limit-mat } X \ A \ m$ **and** $\text{her}X: \forall n. \text{hermitian } (X \ n)$
shows $\text{hermitian } A$
 $\langle \text{proof} \rangle$

lemma *quantifier-change-order-once*:

fixes $P :: \text{nat} \Rightarrow \text{nat} \Rightarrow \text{bool}$ **and** $m :: \text{nat}$
shows $\forall j < m. \exists no. \forall n \geq no. P \ n \ j \implies \exists no. \forall j < m. \forall n \geq no. P \ n \ j$
 $\langle \text{proof} \rangle$

lemma *quantifier-change-order-twice*:

fixes $P :: \text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{bool}$ **and** $m \ n :: \text{nat}$
shows $\forall i < m. \forall j < n. \exists no. \forall n \geq no. P \ n \ i \ j \implies \exists no. \forall i < m. \forall j < n. \forall n \geq no. P \ n \ i \ j$
 $\langle \text{proof} \rangle$

lemma *pos-mat-lim-is-pos*:

fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A :: \text{complex mat}$ **and** $m :: \text{nat}$
assumes $\text{lim}X: \text{limit-mat } X \ A \ m$ **and** $\text{pos}X: \forall n. \text{positive } (X \ n)$

shows *positive A*
 ⟨*proof*⟩

lemma *limit-mat-ignore-initial-segment*:
 $limit\text{-}mat\ g\ A\ d \implies limit\text{-}mat\ (\lambda n. g\ (n + k))\ A\ d$
 ⟨*proof*⟩

lemma *mat-trace-limit*:
 $limit\text{-}mat\ g\ A\ d \implies (\lambda n. trace\ (g\ n)) \longrightarrow trace\ A$
 ⟨*proof*⟩

2.2 Existence of least upper bound for the Löwner order

definition *lower-is-lub* :: $(nat \Rightarrow complex\ mat) \Rightarrow complex\ mat \Rightarrow bool$ **where**
 $lower\text{-}is\text{-}lub\ f\ M \longleftrightarrow (\forall n. f\ n \leq_L M) \wedge (\forall M'. (\forall n. f\ n \leq_L M') \longrightarrow M \leq_L M')$

locale *matrix-seq* =
fixes $dim :: nat$
and $f :: nat \Rightarrow complex\ mat$
assumes
 $dim: \bigwedge n. f\ n \in carrier\text{-}mat\ dim\ dim$ **and**
 $pdo: \bigwedge n. partial\text{-}density\text{-}operator\ (f\ n)$ **and**
 $inc: \bigwedge n. lower\text{-}le\ (f\ n)\ (f\ (Suc\ n))$
begin

definition *lower-is-lub* :: $complex\ mat \Rightarrow bool$ **where**
 $lower\text{-}is\text{-}lub\ M \longleftrightarrow (\forall n. f\ n \leq_L M) \wedge (\forall M'. (\forall n. f\ n \leq_L M') \longrightarrow M \leq_L M')$

lemma *lower-is-lub-dim*:
assumes *lower-is-lub M*
shows $M \in carrier\text{-}mat\ dim\ dim$
 ⟨*proof*⟩

lemma *trace-adjoint-eq-u*:
fixes $A :: complex\ mat$
shows $trace\ (A * adjoint\ A) = (\sum\ i \in \{0 ..< dim\text{-}row\ A\}. \sum\ j \in \{0 ..< dim\text{-}col\ A\}. (norm(A\ \$\$ (i,j)))^2)$
 ⟨*proof*⟩

lemma *trace-adjoint-element-ineq*:
fixes $A :: complex\ mat$
assumes $rindex: i \in \{0 ..< dim\text{-}row\ A\}$
and $cindex: j \in \{0 ..< dim\text{-}col\ A\}$
shows $(norm(A\ \$\$ (i,j)))^2 \leq trace\ (A * adjoint\ A)$
 ⟨*proof*⟩

lemma *positive-is-normal*:
fixes $A :: complex\ mat$

assumes *pos*: positive *A*
shows $A * \text{adjoint } A = \text{adjoint } A * A$
⟨*proof*⟩

lemma *diag-mat-mul-diag-diag*:
fixes *A B* :: complex mat
assumes *dimA*: $A \in \text{carrier-mat } n \ n$ **and** *dimB*: $B \in \text{carrier-mat } n \ n$
and *dA*: diagonal-mat *A* **and** *dB*: diagonal-mat *B*
shows diagonal-mat ($A * B$)
⟨*proof*⟩

lemma *diag-mat-mul-diag-ele*:
fixes *A B* :: complex mat
assumes *dimA*: $A \in \text{carrier-mat } n \ n$ **and** *dimB*: $B \in \text{carrier-mat } n \ n$
and *dA*: diagonal-mat *A* **and** *dB*: diagonal-mat *B*
shows $\forall i < n. (A*B) \$\$ (i,i) = A \$\$ (i, i) * B \$\$ (i, i)$
⟨*proof*⟩

lemma *trace-square-less-square-trace*:
fixes *B* :: complex mat
assumes *dimB*: $B \in \text{carrier-mat } n \ n$
and *dB*: diagonal-mat *B* **and** *pB*: $\bigwedge i. i < n \implies B \$\$ (i, i) \geq 0$
shows $\text{trace } (B*B) \leq (\text{trace } B)^2$
⟨*proof*⟩

lemma *trace-positive-eq*:
fixes *A* :: complex mat
assumes *pos*: positive *A*
shows $\text{trace } (A * \text{adjoint } A) \leq (\text{trace } A)^2$
⟨*proof*⟩

lemma *lowner-le-transitive*:
fixes *m n* :: nat
assumes *re*: $n \geq m$
shows positive ($f \ n - f \ m$)
⟨*proof*⟩

The sequence of matrices converges pointwise.

lemma *inc-partial-density-operator-converge*:
assumes *i*: $i \in \{0 ..< \text{dim}\}$ **and** *j*: $j \in \{0 ..< \text{dim}\}$
shows convergent ($\lambda n. f \ n \ \$\$ (i, j)$)
⟨*proof*⟩

definition *mat-seq-minus* :: $(\text{nat} \Rightarrow \text{complex mat}) \Rightarrow \text{complex mat} \Rightarrow \text{nat} \Rightarrow \text{complex mat}$ **where**
 $\text{mat-seq-minus } X \ A = (\lambda n. X \ n - A)$

definition *minus-mat-seq* :: $\text{complex mat} \Rightarrow (\text{nat} \Rightarrow \text{complex mat}) \Rightarrow \text{nat} \Rightarrow \text{com-}$

plex mat **where**
minus-mat-seq $A X = (\lambda n. A - X n)$

lemma *pos-mat-lim-is-pos-ax*:
fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A :: \text{complex mat}$ **and** $m :: \text{nat}$
assumes *limX*: *limit-mat* $X A m$ **and** *posX*: $\exists k. \forall n \geq k. \text{positive } (X n)$
shows *positive A*
 $\langle \text{proof} \rangle$

lemma *minus-mat-limit*:
fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A :: \text{complex mat}$ **and** $m :: \text{nat}$ **and** $B :: \text{complex mat}$
assumes *dimB*: $B \in \text{carrier-mat } m m$ **and** *limX*: *limit-mat* $X A m$
shows *limit-mat* (*mat-seq-minus* $X B$) ($A - B$) m
 $\langle \text{proof} \rangle$

lemma *mat-minus-limit*:
fixes $X :: \text{nat} \Rightarrow \text{complex mat}$ **and** $A :: \text{complex mat}$ **and** $m :: \text{nat}$ **and** $B :: \text{complex mat}$
assumes *dimA*: $A \in \text{carrier-mat } m m$ **and** *limX*: *limit-mat* $X A m$
shows *limit-mat* (*minus-mat-seq* $B X$) ($B - A$) m
 $\langle \text{proof} \rangle$

lemma *lowner-lub-form*:
lowner-is-lub (*mat dim dim* $(\lambda (i, j). (\text{lim } (\lambda n. (f n) \$\$ (i, j))))$)
 $\langle \text{proof} \rangle$

Lowner partial order is a complete partial order.

lemma *lowner-lub-exists*: $\exists M. \text{lowner-is-lub } M$
 $\langle \text{proof} \rangle$

lemma *lowner-lub-unique*: $\exists! M. \text{lowner-is-lub } M$
 $\langle \text{proof} \rangle$

definition *lowner-lub* :: *complex mat* **where**
lowner-lub = (*THE* $M. \text{lowner-is-lub } M$)

lemma *lowner-lub-prop*: *lowner-is-lub* *lowner-lub*
 $\langle \text{proof} \rangle$

lemma *lowner-lub-is-limit*:
limit-mat $f \text{lowner-lub dim}$
 $\langle \text{proof} \rangle$

lemma *lowner-lub-trace*:
assumes $\forall n. \text{trace } (f n) \leq x$
shows *trace* *lowner-lub* $\leq x$
 $\langle \text{proof} \rangle$

lemma *lower-lub-is-positive*:
shows *positive lower-lub*
 ⟨*proof*⟩

end

2.3 Finite sum of matrices

Add f in the interval $[0, n)$

fun *matrix-sum* :: $\text{nat} \Rightarrow (\text{nat} \Rightarrow 'b::\text{semiring-1 mat}) \Rightarrow \text{nat} \Rightarrow 'b \text{ mat}$ **where**
matrix-sum $d f 0 = 0_m d d$
 | *matrix-sum* $d f (\text{Suc } n) = f n + \text{matrix-sum } d f n$

definition *matrix-inf-sum* :: $\text{nat} \Rightarrow (\text{nat} \Rightarrow \text{complex mat}) \Rightarrow \text{complex mat}$ **where**
matrix-inf-sum $d f = \text{matrix-seq.lower-lub } (\lambda n. \text{matrix-sum } d f n)$

lemma *matrix-sum-dim*:
fixes $f :: \text{nat} \Rightarrow 'b::\text{semiring-1 mat}$
shows $(\bigwedge k. k < n \implies f k \in \text{carrier-mat } d d) \implies \text{matrix-sum } d f n \in \text{carrier-mat } d d$
 ⟨*proof*⟩

lemma *matrix-sum-cong*:
fixes $f :: \text{nat} \Rightarrow 'b::\text{semiring-1 mat}$
shows $(\bigwedge k. k < n \implies f k = f' k) \implies \text{matrix-sum } d f n = \text{matrix-sum } d f' n$
 ⟨*proof*⟩

lemma *matrix-sum-add*:
fixes $f :: \text{nat} \Rightarrow 'b::\text{semiring-1 mat}$ **and** $g :: \text{nat} \Rightarrow 'b::\text{semiring-1 mat}$ **and** h
 :: $\text{nat} \Rightarrow 'b::\text{semiring-1 mat}$
shows $(\bigwedge k. k < n \implies f k \in \text{carrier-mat } d d) \implies (\bigwedge k. k < n \implies g k \in \text{carrier-mat } d d) \implies$
 $(\bigwedge k. k < n \implies f k = g k + h k) \implies \text{matrix-sum } d f n = \text{matrix-sum } d g n$
 $+ \text{matrix-sum } d h n$
 ⟨*proof*⟩

lemma *matrix-sum-smult*:
fixes $f :: \text{nat} \Rightarrow 'b::\text{semiring-1 mat}$
shows $(\bigwedge k. k < n \implies f k \in \text{carrier-mat } d d) \implies$
 $\text{matrix-sum } d (\lambda k. c \cdot_m f k) n = c \cdot_m \text{matrix-sum } d f n$
 ⟨*proof*⟩

lemma *matrix-sum-remove*:
fixes $f :: \text{nat} \Rightarrow 'b::\text{semiring-1 mat}$
assumes $j: j < n$
and $df: (\bigwedge k. k < n \implies f k \in \text{carrier-mat } d d)$
and $f': (\bigwedge k. f' k = (\text{if } k = j \text{ then } 0_m d d \text{ else } f k))$
shows $\text{matrix-sum } d f n = f j + \text{matrix-sum } d f' n$
 ⟨*proof*⟩

lemma *matrix-sum-Suc-remove-head*:

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

shows $(\bigwedge k. k < n + 1 \implies f k \in \text{carrier-mat } d \ d) \implies$

$\text{matrix-sum } d \ f \ (n + 1) = f \ 0 + \text{matrix-sum } d \ (\lambda k. f \ (k + 1)) \ n$

$\langle \text{proof} \rangle$

lemma *matrix-sum-positive*:

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

shows $(\bigwedge k. k < n \implies f k \in \text{carrier-mat } d \ d) \implies (\bigwedge k. k < n \implies \text{positive } (f k))$

$\implies \text{positive } (\text{matrix-sum } d \ f \ n)$

$\langle \text{proof} \rangle$

lemma *matrix-sum-mult-right*:

shows $(\bigwedge k. k < n \implies f k \in \text{carrier-mat } d \ d) \implies A \in \text{carrier-mat } d \ d$

$\implies \text{matrix-sum } d \ (\lambda k. (f k) * A) \ n = \text{matrix-sum } d \ (\lambda k. f k) \ n * A$

$\langle \text{proof} \rangle$

lemma *matrix-sum-add-distrib*:

shows $(\bigwedge k. k < n \implies f k \in \text{carrier-mat } d \ d) \implies (\bigwedge k. k < n \implies g k \in \text{carrier-mat } d \ d)$

$\implies \text{matrix-sum } d \ (\lambda k. (f k) + (g k)) \ n = \text{matrix-sum } d \ f \ n + \text{matrix-sum } d \ g \ n$

$\langle \text{proof} \rangle$

lemma *matrix-sum-minus-distrib*:

fixes $f \ g :: \text{nat} \Rightarrow \text{complex mat}$

shows $(\bigwedge k. k < n \implies f k \in \text{carrier-mat } d \ d) \implies (\bigwedge k. k < n \implies g k \in \text{carrier-mat } d \ d)$

$\implies \text{matrix-sum } d \ (\lambda k. (f k) - (g k)) \ n = \text{matrix-sum } d \ f \ n - \text{matrix-sum } d \ g \ n$

$\langle \text{proof} \rangle$

lemma *matrix-sum-shift-Suc*:

shows $(\bigwedge k. k < (\text{Suc } n) \implies f k \in \text{carrier-mat } d \ d)$

$\implies \text{matrix-sum } d \ f \ (\text{Suc } n) = f \ 0 + \text{matrix-sum } d \ (\lambda k. f \ (\text{Suc } k)) \ n$

$\langle \text{proof} \rangle$

lemma *lower-le-matrix-sum*:

fixes $f \ g :: \text{nat} \Rightarrow \text{complex mat}$

shows $(\bigwedge k. k < n \implies f k \in \text{carrier-mat } d \ d) \implies (\bigwedge k. k < n \implies g k \in \text{carrier-mat } d \ d)$

$\implies (\bigwedge k. k < n \implies f k \leq_L g k)$

$\implies \text{matrix-sum } d \ f \ n \leq_L \text{matrix-sum } d \ g \ n$

$\langle \text{proof} \rangle$

lemma *lower-lub-add*:

assumes $\text{matrix-seq } d \ f \ \text{matrix-seq } d \ g \ \forall \ n. \text{trace } (f \ n + g \ n) \leq 1$

shows $\text{matrix-seq.lower-lub } (\lambda n. f \ n + g \ n) = \text{matrix-seq.lower-lub } f + \text{matrix-seq.lower-lub } g$

$\langle \text{proof} \rangle$

lemma *lower-lub-scale*:

fixes $c :: \text{real}$

assumes $\text{matrix-seq } d f \ \forall n. \text{trace } (c \cdot_m f n) \leq 1 \ c \geq 0$

shows $\text{matrix-seq.lower-lub } (\lambda n. c \cdot_m f n) = c \cdot_m \text{matrix-seq.lower-lub } f$
 $\langle \text{proof} \rangle$

lemma *trace-matrix-sum-linear*:

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

shows $(\bigwedge k. k < n \Rightarrow f k \in \text{carrier-mat } d d) \Rightarrow \text{trace } (\text{matrix-sum } d f n) =$
 $\text{sum } (\lambda k. \text{trace } (f k)) \ \{0..<n\}$
 $\langle \text{proof} \rangle$

lemma *matrix-sum-distrib-left*:

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

shows $P \in \text{carrier-mat } d d \Rightarrow (\bigwedge k. k < n \Rightarrow f k \in \text{carrier-mat } d d) \Rightarrow$
 $\text{matrix-sum } d (\lambda k. P * (f k)) \ n = P * (\text{matrix-sum } d f n)$
 $\langle \text{proof} \rangle$

2.4 Measurement

definition *measurement* $:: \text{nat} \Rightarrow \text{nat} \Rightarrow (\text{nat} \Rightarrow \text{complex mat}) \Rightarrow \text{bool}$ **where**

$\text{measurement } d \ n \ M \longleftrightarrow (\forall j < n. M j \in \text{carrier-mat } d d)$

$\wedge \text{matrix-sum } d (\lambda j. (\text{adjoint } (M j)) * M j) \ n = 1_m \ d$

lemma *measurement-dim*:

assumes $\text{measurement } d \ n \ M$

shows $\bigwedge k. k < n \Rightarrow (M k) \in \text{carrier-mat } d d$

$\langle \text{proof} \rangle$

lemma *measurement-id2*:

assumes $\text{measurement } d \ 2 \ M$

shows $\text{adjoint } (M 0) * M 0 + \text{adjoint } (M 1) * M 1 = 1_m \ d$

$\langle \text{proof} \rangle$

Result of measurement on ρ by matrix M

definition *measurement-res* $:: \text{complex mat} \Rightarrow \text{complex mat} \Rightarrow \text{complex mat}$ **where**

$\text{measurement-res } M \ \rho = M * \rho * \text{adjoint } M$

lemma *add-positive-le-reduce1*:

assumes $dA: A \in \text{carrier-mat } n \ n$ **and** $dB: B \in \text{carrier-mat } n \ n$ **and** $dC: C \in$
 $\text{carrier-mat } n \ n$

and $pB: \text{positive } B$ **and** $le: A + B \leq_L C$

shows $A \leq_L C$

$\langle \text{proof} \rangle$

lemma *add-positive-le-reduce2*:

assumes $dA: A \in \text{carrier-mat } n \ n$ **and** $dB: B \in \text{carrier-mat } n \ n$ **and** $dC: C \in$
 $\text{carrier-mat } n \ n$

and $pB: \text{positive } B$ **and** $le: B + A \leq_L C$

shows $A \leq_L C$
<proof>

lemma *measurement-le-one-mat*:
assumes *measurement* $d\ n\ f$
shows $\bigwedge j. j < n \implies \text{adjoint } (f\ j) * f\ j \leq_L 1_m\ d$
<proof>

lemma *pdo-close-under-measurement*:
fixes $M\ \varrho :: \text{complex mat}$
assumes $dM: M \in \text{carrier-mat } n\ n$ **and** $dr: \varrho \in \text{carrier-mat } n\ n$
and $pdr: \text{partial-density-operator } \varrho$
and $le: \text{adjoint } M * M \leq_L 1_m\ n$
shows *partial-density-operator* $(M * \varrho * \text{adjoint } M)$
<proof>

lemma *trace-measurement*:
assumes $m: \text{measurement } d\ n\ M$ **and** $dA: A \in \text{carrier-mat } d\ d$
shows *trace* $(\text{matrix-sum } d\ (\lambda k. (M\ k) * A * \text{adjoint } (M\ k))\ n) = \text{trace } A$
<proof>

lemma *mat-inc-seq-positive-transform*:
assumes $dfn: \bigwedge n. f\ n \in \text{carrier-mat } d\ d$
and $inc: \bigwedge n. f\ n \leq_L f\ (\text{Suc } n)$
shows $\bigwedge n. f\ n - f\ 0 \in \text{carrier-mat } d\ d$ **and** $\bigwedge n. (f\ n - f\ 0) \leq_L (f\ (\text{Suc } n) - f\ 0)$
<proof>

lemma *mat-inc-seq-lub*:
assumes $dfn: \bigwedge n. f\ n \in \text{carrier-mat } d\ d$
and $inc: \bigwedge n. f\ n \leq_L f\ (\text{Suc } n)$
and $ub: \bigwedge n. f\ n \leq_L A$
shows $\exists B. \text{lower-is-lub } f\ B \wedge \text{limit-mat } f\ B\ d$
<proof>

end

3 Quantum programs

theory *Quantum-Program*
imports *Matrix-Limit*
begin

3.1 Syntax

Datatype for quantum programs

datatype *com* =
SKIP

```

| Utrans complex mat
| Seq com com (-; / - [60, 61] 60)
| Measure nat nat  $\Rightarrow$  complex mat com list
| While nat  $\Rightarrow$  complex mat com

```

A state corresponds to the density operator

type-synonym *state* = *complex mat*

List of dimensions of quantum states

```

locale state-sig =
  fixes dims :: nat list
begin

```

definition *d* :: *nat* **where**

d = *prod-list dims*

Wellformedness of commands

```

fun well-com :: com  $\Rightarrow$  bool where
  well-com SKIP = True
| well-com (Utrans U) = (U  $\in$  carrier-mat d d  $\wedge$  unitary U)
| well-com (Seq S1 S2) = (well-com S1  $\wedge$  well-com S2)
| well-com (Measure n M S) =
  (measurement d n M  $\wedge$  length S = n  $\wedge$  list-all well-com S)
| well-com (While M S) =
  (measurement d 2 M  $\wedge$  well-com S)

```

3.2 Denotational semantics

Denotation of going through the while loop *n* times

```

fun denote-while-n-iter :: complex mat  $\Rightarrow$  complex mat  $\Rightarrow$  (state  $\Rightarrow$  state)  $\Rightarrow$  nat
 $\Rightarrow$  state  $\Rightarrow$  state where
  denote-while-n-iter M0 M1 DS 0  $\rho$  =  $\rho$ 
| denote-while-n-iter M0 M1 DS (Suc n)  $\rho$  = denote-while-n-iter M0 M1 DS n (DS
(M1 *  $\rho$  * adjoint M1))

```

```

fun denote-while-n :: complex mat  $\Rightarrow$  complex mat  $\Rightarrow$  (state  $\Rightarrow$  state)  $\Rightarrow$  nat  $\Rightarrow$ 
state  $\Rightarrow$  state where
  denote-while-n M0 M1 DS n  $\rho$  = M0 * denote-while-n-iter M0 M1 DS n  $\rho$  *
adjoint M0

```

```

fun denote-while-n-comp :: complex mat  $\Rightarrow$  complex mat  $\Rightarrow$  (state  $\Rightarrow$  state)  $\Rightarrow$  nat
 $\Rightarrow$  state  $\Rightarrow$  state where
  denote-while-n-comp M0 M1 DS n  $\rho$  = M1 * denote-while-n-iter M0 M1 DS n  $\rho$ 
* adjoint M1

```

lemma *denote-while-n-iter-assoc*:

```

denote-while-n-iter M0 M1 DS (Suc n)  $\rho$  = DS (M1 * (denote-while-n-iter M0
M1 DS n  $\rho$ ) * adjoint M1)
<proof>

```

lemma *denote-while-n-iter-dim:*

$\varrho \in \text{carrier-mat } m \ m \implies \text{partial-density-operator } \varrho \implies M1 \in \text{carrier-mat } m \ m$
 $\implies \text{adjoint } M1 * M1 \leq_L 1_m \ m$
 $\implies (\bigwedge \varrho. \varrho \in \text{carrier-mat } m \ m \implies \text{partial-density-operator } \varrho \implies DS \ \varrho \in \text{carrier-mat } m \ m \wedge \text{partial-density-operator } (DS \ \varrho))$
 $\implies \text{denote-while-n-iter } M0 \ M1 \ DS \ n \ \varrho \in \text{carrier-mat } m \ m \wedge \text{partial-density-operator } (\text{denote-while-n-iter } M0 \ M1 \ DS \ n \ \varrho)$
 $\langle \text{proof} \rangle$

lemma *pdo-denote-while-n-iter:*

$\varrho \in \text{carrier-mat } m \ m \implies \text{partial-density-operator } \varrho \implies M1 \in \text{carrier-mat } m \ m$
 $\implies \text{adjoint } M1 * M1 \leq_L 1_m \ m$
 $\implies (\bigwedge \varrho. \varrho \in \text{carrier-mat } m \ m \wedge \text{partial-density-operator } \varrho \implies \text{partial-density-operator } (DS \ \varrho))$
 $\implies (\bigwedge \varrho. \varrho \in \text{carrier-mat } m \ m \wedge \text{partial-density-operator } \varrho \implies DS \ \varrho \in \text{carrier-mat } m \ m)$
 $\implies \text{partial-density-operator } (\text{denote-while-n-iter } M0 \ M1 \ DS \ n \ \varrho)$
 $\langle \text{proof} \rangle$

Denotation of while is simply the infinite sum of `denote_while_n`

definition *denote-while* :: $\text{complex mat} \Rightarrow \text{complex mat} \Rightarrow (\text{state} \Rightarrow \text{state}) \Rightarrow \text{state} \Rightarrow \text{state}$ **where**

$\text{denote-while } M0 \ M1 \ DS \ \varrho = \text{matrix-inf-sum } d \ (\lambda n. \text{denote-while-n } M0 \ M1 \ DS \ n \ \varrho)$

lemma *denote-while-n-dim:*

assumes $\varrho \in \text{carrier-mat } d \ d$
 $M0 \in \text{carrier-mat } d \ d$
 $M1 \in \text{carrier-mat } d \ d$
 $\text{partial-density-operator } \varrho$
 $\bigwedge \varrho'. \varrho' \in \text{carrier-mat } d \ d \implies \text{partial-density-operator } \varrho' \implies \text{positive } (DS \ \varrho')$
 $\wedge \text{trace } (DS \ \varrho') \leq \text{trace } \varrho' \wedge DS \ \varrho' \in \text{carrier-mat } d \ d$
shows $\text{denote-while-n } M0 \ M1 \ DS \ n \ \varrho \in \text{carrier-mat } d \ d$
 $\langle \text{proof} \rangle$

lemma *denote-while-n-sum-dim:*

assumes $\varrho \in \text{carrier-mat } d \ d$
 $M0 \in \text{carrier-mat } d \ d$
 $M1 \in \text{carrier-mat } d \ d$
 $\text{partial-density-operator } \varrho$
 $\bigwedge \varrho'. \varrho' \in \text{carrier-mat } d \ d \implies \text{partial-density-operator } \varrho' \implies \text{positive } (DS \ \varrho')$
 $\wedge \text{trace } (DS \ \varrho') \leq \text{trace } \varrho' \wedge DS \ \varrho' \in \text{carrier-mat } d \ d$
shows $\text{matrix-sum } d \ (\lambda n. \text{denote-while-n } M0 \ M1 \ DS \ n \ \varrho) \ n \in \text{carrier-mat } d \ d$
 $\langle \text{proof} \rangle$

lemma *trace-decrease-mul-adj:*

assumes *pdo*: $\text{partial-density-operator } \varrho$ **and** *dimr*: $\varrho \in \text{carrier-mat } d \ d$
and *dimx*: $x \in \text{carrier-mat } d \ d$ **and** *un*: $\text{adjoint } x * x \leq_L 1_m \ d$

shows $\text{trace } (x * \varrho * \text{adjoint } x) \leq \text{trace } \varrho$
 ⟨proof⟩

lemma *denote-while-n-positive*:

assumes $\text{dim0}: M0 \in \text{carrier-mat } d \ d$ **and** $\text{dim1}: M1 \in \text{carrier-mat } d \ d$ **and**
un: $\text{adjoint } M1 * M1 \leq_L 1_m \ d$
and $DS: \bigwedge \varrho. \varrho \in \text{carrier-mat } d \ d \implies \text{partial-density-operator } \varrho \implies \text{positive}$
 $(DS \ \varrho) \wedge \text{trace } (DS \ \varrho) \leq \text{trace } \varrho \wedge DS \ \varrho \in \text{carrier-mat } d \ d$
shows $\text{partial-density-operator } \varrho \wedge \varrho \in \text{carrier-mat } d \ d \implies \text{positive } (\text{denote-while-n } M0 \ M1 \ DS \ n \ \varrho)$
 ⟨proof⟩

lemma *denote-while-n-sum-positive*:

assumes $\text{dim0}: M0 \in \text{carrier-mat } d \ d$ **and** $\text{dim1}: M1 \in \text{carrier-mat } d \ d$ **and**
un: $\text{adjoint } M1 * M1 \leq_L 1_m \ d$
and $DS: \bigwedge \varrho. \varrho \in \text{carrier-mat } d \ d \implies \text{partial-density-operator } \varrho \implies \text{positive}$
 $(DS \ \varrho) \wedge \text{trace } (DS \ \varrho) \leq \text{trace } \varrho \wedge DS \ \varrho \in \text{carrier-mat } d \ d$
and $\text{pdo}: \text{partial-density-operator } \varrho$ **and** $r: \varrho \in \text{carrier-mat } d \ d$
shows $\text{positive } (\text{matrix-sum } d \ (\lambda n. \text{denote-while-n } M0 \ M1 \ DS \ n \ \varrho) \ n)$
 ⟨proof⟩

lemma *trace-measure2-id*:

assumes $dM0: M0 \in \text{carrier-mat } n \ n$ **and** $dM1: M1 \in \text{carrier-mat } n \ n$
and $\text{id}: \text{adjoint } M0 * M0 + \text{adjoint } M1 * M1 = 1_m \ n$
and $dA: A \in \text{carrier-mat } n \ n$
shows $\text{trace } (M0 * A * \text{adjoint } M0) + \text{trace } (M1 * A * \text{adjoint } M1) = \text{trace } A$
 ⟨proof⟩

lemma *measurement-lowner-le-one1*:

assumes $\text{dim0}: M0 \in \text{carrier-mat } d \ d$ **and** $\text{dim1}: M1 \in \text{carrier-mat } d \ d$ **and** $\text{id}: \text{adjoint } M0 * M0 + \text{adjoint } M1 * M1 = 1_m \ d$
shows $\text{adjoint } M1 * M1 \leq_L 1_m \ d$
 ⟨proof⟩

lemma *denote-while-n-sum-trace*:

assumes $\text{dim0}: M0 \in \text{carrier-mat } d \ d$ **and** $\text{dim1}: M1 \in \text{carrier-mat } d \ d$ **and** $\text{id}: \text{adjoint } M0 * M0 + \text{adjoint } M1 * M1 = 1_m \ d$
and $DS: \bigwedge \varrho. \varrho \in \text{carrier-mat } d \ d \implies \text{partial-density-operator } \varrho \implies \text{positive}$
 $(DS \ \varrho) \wedge \text{trace } (DS \ \varrho) \leq \text{trace } \varrho \wedge DS \ \varrho \in \text{carrier-mat } d \ d$
and $r: \varrho \in \text{carrier-mat } d \ d$
and $\text{pdor}: \text{partial-density-operator } \varrho$
shows $\text{trace } (\text{matrix-sum } d \ (\lambda n. \text{denote-while-n } M0 \ M1 \ DS \ n \ \varrho) \ n) \leq \text{trace } \varrho$
 ⟨proof⟩

lemma *denote-while-n-sum-partial-density*:

assumes $\text{dim0}: M0 \in \text{carrier-mat } d \ d$ **and** $\text{dim1}: M1 \in \text{carrier-mat } d \ d$ **and** $\text{id}: \text{adjoint } M0 * M0 + \text{adjoint } M1 * M1 = 1_m \ d$
and $DS: \bigwedge \varrho. \varrho \in \text{carrier-mat } d \ d \implies \text{partial-density-operator } \varrho \implies \text{positive}$
 $(DS \ \varrho) \wedge \text{trace } (DS \ \varrho) \leq \text{trace } \varrho \wedge DS \ \varrho \in \text{carrier-mat } d \ d$

and *pdo*: partial-density-operator ρ **and** *r*: $\rho \in \text{carrier-mat } d \ d$
shows (partial-density-operator (matrix-sum d (λn . denote-while- n $M0 \ M1 \ DS \ n$ ρ) n))
⟨proof⟩

lemma *denote-while-n-sum-lowner-le*:

assumes *dim0*: $M0 \in \text{carrier-mat } d \ d$ **and** *dim1*: $M1 \in \text{carrier-mat } d \ d$ **and** *id*:
adjoint $M0 * M0 + \text{adjoint } M1 * M1 = 1_m \ d$
and *DS*: $\bigwedge \rho$. $\rho \in \text{carrier-mat } d \ d \implies \text{partial-density-operator } \rho \implies \text{positive}$
 $(DS \ \rho) \wedge \text{trace } (DS \ \rho) \leq \text{trace } \rho \wedge DS \ \rho \in \text{carrier-mat } d \ d$
and *pdo*: partial-density-operator ρ **and** *dimr*: $\rho \in \text{carrier-mat } d \ d$
shows (matrix-sum d (λn . denote-while- n $M0 \ M1 \ DS \ n \ \rho$) $n \leq_L$ matrix-sum d
(λn . denote-while- n $M0 \ M1 \ DS \ n \ \rho$) (Suc n))
⟨proof⟩

lemma *loowner-is-lub-matrix-sum*:

assumes *dim0*: $M0 \in \text{carrier-mat } d \ d$ **and** *dim1*: $M1 \in \text{carrier-mat } d \ d$ **and** *id*:
adjoint $M0 * M0 + \text{adjoint } M1 * M1 = 1_m \ d$
and *DS*: $\bigwedge \rho$. $\rho \in \text{carrier-mat } d \ d \implies \text{partial-density-operator } \rho \implies \text{positive}$
 $(DS \ \rho) \wedge \text{trace } (DS \ \rho) \leq \text{trace } \rho \wedge DS \ \rho \in \text{carrier-mat } d \ d$
and *pdo*: partial-density-operator ρ **and** *dimr*: $\rho \in \text{carrier-mat } d \ d$
shows matrix-seq.loowner-is-lub (matrix-sum d (λn . denote-while- n $M0 \ M1 \ DS \ n \ \rho$)) (matrix-seq.loowner-lub (matrix-sum d (λn . denote-while- n $M0 \ M1 \ DS \ n \ \rho$)))
⟨proof⟩

lemma *denote-while-dim-positive*:

assumes *dim0*: $M0 \in \text{carrier-mat } d \ d$ **and** *dim1*: $M1 \in \text{carrier-mat } d \ d$ **and** *id*:
adjoint $M0 * M0 + \text{adjoint } M1 * M1 = 1_m \ d$
and *DS*: $\bigwedge \rho$. $\rho \in \text{carrier-mat } d \ d \implies \text{partial-density-operator } \rho \implies \text{positive}$
 $(DS \ \rho) \wedge \text{trace } (DS \ \rho) \leq \text{trace } \rho \wedge DS \ \rho \in \text{carrier-mat } d \ d$
and *pdo*: partial-density-operator ρ **and** *dimr*: $\rho \in \text{carrier-mat } d \ d$
shows
denote-while $M0 \ M1 \ DS \ \rho \in \text{carrier-mat } d \ d \wedge \text{positive}$ (denote-while $M0 \ M1$
 $DS \ \rho$) $\wedge \text{trace}$ (denote-while $M0 \ M1 \ DS \ \rho$) $\leq \text{trace } \rho$
⟨proof⟩

definition *denote-measure* :: $\text{nat} \Rightarrow (\text{nat} \Rightarrow \text{complex mat}) \Rightarrow ((\text{state} \Rightarrow \text{state}) \text{ list})$
 $\Rightarrow \text{state} \Rightarrow \text{state}$ **where**

denote-measure $n \ M \ DS \ \rho = \text{matrix-sum } d$ (λk . $(DS!k) ((M \ k) * \rho * \text{adjoint } (M \ k))$) n

lemma *denote-measure-dim*:

assumes $\rho \in \text{carrier-mat } d \ d$
measurement $d \ n \ M$
 $\bigwedge \rho' \ k$. $\rho' \in \text{carrier-mat } d \ d \implies k < n \implies (DS!k) \ \rho' \in \text{carrier-mat } d \ d$
shows
denote-measure $n \ M \ DS \ \rho \in \text{carrier-mat } d \ d$
⟨proof⟩

lemma *measure-well-com*:
assumes *well-com* (*Measure n M S*)
shows $\bigwedge k. k < n \implies \text{well-com } (S ! k)$
 $\langle \text{proof} \rangle$

Semantics of commands

fun *denote* :: *com* \Rightarrow *state* \Rightarrow *state* **where**
denote *SKIP* $\varrho = \varrho$
 $|$ *denote* (*Utrans U*) $\varrho = U * \varrho * \text{adjoint } U$
 $|$ *denote* (*Seq S1 S2*) $\varrho = \text{denote } S2 (\text{denote } S1 \varrho)$
 $|$ *denote* (*Measure n M S*) $\varrho = \text{denote-measure } n M (\text{map } \text{denote } S) \varrho$
 $|$ *denote* (*While M S*) $\varrho = \text{denote-while } (M 0) (M 1) (\text{denote } S) \varrho$

lemma *denote-measure-expand*:
assumes *m*: $m \leq n$ **and** *wc*: *well-com* (*Measure n M S*)
shows *denote* (*Measure m M S*) $\varrho = \text{matrix-sum } d (\lambda k. \text{denote } (S!k) ((M k) * \varrho * \text{adjoint } (M k))) m$
 $\langle \text{proof} \rangle$

lemma *matrix-sum-trace-le*:
fixes *f* :: *nat* \Rightarrow *complex mat* **and** *g* :: *nat* \Rightarrow *complex mat*
assumes ($\bigwedge k. k < n \implies f k \in \text{carrier-mat } d d$)
 $(\bigwedge k. k < n \implies g k \in \text{carrier-mat } d d)$
 $(\bigwedge k. k < n \implies \text{trace } (f k) \leq \text{trace } (g k))$
shows $\text{trace } (\text{matrix-sum } d f n) \leq \text{trace } (\text{matrix-sum } d g n)$
 $\langle \text{proof} \rangle$

lemma *map-denote-positive-trace-dim*:
assumes *well-com* (*Measure x1 x2a x3a*)
 $x4 \in \text{carrier-mat } d d$
partial-density-operator $x4$
 $\bigwedge x3aa \varrho. x3aa \in \text{set } x3a \implies \text{well-com } x3aa \implies \varrho \in \text{carrier-mat } d d \implies$
partial-density-operator ϱ
 $\implies \text{positive } (\text{denote } x3aa \varrho) \wedge \text{trace } (\text{denote } x3aa \varrho) \leq \text{trace } \varrho \wedge \text{denote } x3aa$
 $\varrho \in \text{carrier-mat } d d$
shows $\forall k < x1. \text{positive } ((\text{map } \text{denote } x3a ! k) (x2a k * x4 * \text{adjoint } (x2a k)))$
 $\wedge ((\text{map } \text{denote } x3a ! k) (x2a k * x4 * \text{adjoint } (x2a k))) \in \text{carrier-mat}$
 $d d$
 $\wedge \text{trace } ((\text{map } \text{denote } x3a ! k) (x2a k * x4 * \text{adjoint } (x2a k))) \leq \text{trace}$
 $(x2a k * x4 * \text{adjoint } (x2a k))$
 $\langle \text{proof} \rangle$

lemma *denote-measure-positive-trace-dim*:
assumes *well-com* (*Measure x1 x2a x3a*)
 $x4 \in \text{carrier-mat } d d$
partial-density-operator $x4$
 $\bigwedge x3aa \varrho. x3aa \in \text{set } x3a \implies \text{well-com } x3aa \implies \varrho \in \text{carrier-mat } d d \implies$
partial-density-operator ϱ

\implies *positive* (denote $x3aa$ ρ) \wedge *trace* (denote $x3aa$ ρ) \leq *trace* ρ \wedge denote $x3aa$ $\rho \in$ *carrier-mat* d d

shows *positive* (denote (Measure $x1$ $x2a$ $x3a$) $x4$) \wedge *trace* (denote (Measure $x1$ $x2a$ $x3a$) $x4$) \leq *trace* $x4$

\wedge (denote (Measure $x1$ $x2a$ $x3a$) $x4$) \in *carrier-mat* d d
 ⟨*proof*⟩

lemma *denote-positive-trace-dim*:

well-com $S \implies \rho \in$ *carrier-mat* d $d \implies$ *partial-density-operator* ρ
 \implies (*positive* (denote S ρ) \wedge *trace* (denote S ρ) \leq *trace* ρ \wedge denote S $\rho \in$ *carrier-mat* d d)
 ⟨*proof*⟩

lemma *denote-dim-pdo*:

well-com $S \implies \rho \in$ *carrier-mat* d $d \implies$ *partial-density-operator* ρ
 \implies (denote S $\rho \in$ *carrier-mat* d d) \wedge (*partial-density-operator* (denote S ρ))
 ⟨*proof*⟩

lemma *denote-dim*:

well-com $S \implies \rho \in$ *carrier-mat* d $d \implies$ *partial-density-operator* ρ
 \implies (denote S $\rho \in$ *carrier-mat* d d)
 ⟨*proof*⟩

lemma *denote-trace*:

well-com $S \implies \rho \in$ *carrier-mat* d $d \implies$ *partial-density-operator* ρ
 \implies *trace* (denote S ρ) \leq *trace* ρ
 ⟨*proof*⟩

lemma *denote-partial-density-operator*:

assumes *well-com* S *partial-density-operator* ρ $\rho \in$ *carrier-mat* d d
shows *partial-density-operator* (denote S ρ)
 ⟨*proof*⟩

lemma *denote-while-n-sum-mat-seq*:

assumes $\rho \in$ *carrier-mat* d d **and**
 $x1$ $0 \in$ *carrier-mat* d d **and**
 $x1$ $1 \in$ *carrier-mat* d d **and**
partial-density-operator ρ **and**
 wc : *well-com* $x2$ **and** mea : *measurement* d 2 $x1$
shows *matrix-seq* d (*matrix-sum* d (λn . *denote-while-n* ($x1$ 0) ($x1$ 1) (denote $x2$)
 n ρ))
 ⟨*proof*⟩

lemma *denote-while-n-add*:

assumes $M0$: $x1$ $0 \in$ *carrier-mat* d d **and**
 $M1$: $x1$ $1 \in$ *carrier-mat* d d **and**
 wc : *well-com* $x2$ **and** mea : *measurement* d 2 $x1$ **and**
 DS : ($\bigwedge \rho_1 \rho_2$. $\rho_1 \in$ *carrier-mat* d $d \implies \rho_2 \in$ *carrier-mat* d $d \implies$ *partial-density-operator*

$\varrho_1 \implies$
partial-density-operator $\varrho_2 \implies \text{trace} (\varrho_1 + \varrho_2) \leq 1 \implies \text{denote } x2 (\varrho_1 + \varrho_2)$
 $= \text{denote } x2 \varrho_1 + \text{denote } x2 \varrho_2$
shows $\varrho_1 \in \text{carrier-mat } d \ d \implies \varrho_2 \in \text{carrier-mat } d \ d \implies \text{partial-density-operator}$
 $\varrho_1 \implies \text{partial-density-operator } \varrho_2 \implies \text{trace} (\varrho_1 + \varrho_2) \leq 1 \implies$
 $\text{denote-while-n } (x1 \ 0) (x1 \ 1) (\text{denote } x2) \ k (\varrho_1 + \varrho_2) = \text{denote-while-n } (x1 \ 0)$
 $(x1 \ 1) (\text{denote } x2) \ k \ \varrho_1 + \text{denote-while-n } (x1 \ 0) (x1 \ 1) (\text{denote } x2) \ k \ \varrho_2$
 ⟨proof⟩

lemma *denote-while-add:*

assumes $r1: \varrho_1 \in \text{carrier-mat } d \ d$ **and**
 $r2: \varrho_2 \in \text{carrier-mat } d \ d$ **and**
 $M0: x1 \ 0 \in \text{carrier-mat } d \ d$ **and**
 $M1: x1 \ 1 \in \text{carrier-mat } d \ d$ **and**
 $pdo1: \text{partial-density-operator } \varrho_1$ **and**
 $pdo2: \text{partial-density-operator } \varrho_2$ **and** $tr12: \text{trace} (\varrho_1 + \varrho_2) \leq 1$ **and**
 $wc: \text{well-com } x2$ **and** $mea: \text{measurement } d \ 2 \ x1$ **and**
 $DS: (\bigwedge \varrho_1 \ \varrho_2. \varrho_1 \in \text{carrier-mat } d \ d \implies \varrho_2 \in \text{carrier-mat } d \ d \implies \text{partial-density-operator}$
 $\varrho_1 \implies$
partial-density-operator $\varrho_2 \implies \text{trace} (\varrho_1 + \varrho_2) \leq 1 \implies \text{denote } x2 (\varrho_1 + \varrho_2)$
 $= \text{denote } x2 \ \varrho_1 + \text{denote } x2 \ \varrho_2$
shows
 $\text{denote-while } (x1 \ 0) (x1 \ 1) (\text{denote } x2) (\varrho_1 + \varrho_2) = \text{denote-while } (x1 \ 0) (x1 \ 1)$
 $(\text{denote } x2) \ \varrho_1 + \text{denote-while } (x1 \ 0) (x1 \ 1) (\text{denote } x2) \ \varrho_2$
 ⟨proof⟩

lemma *denote-add:*

$\text{well-com } S \implies \varrho_1 \in \text{carrier-mat } d \ d \implies \varrho_2 \in \text{carrier-mat } d \ d \implies$
partial-density-operator $\varrho_1 \implies \text{partial-density-operator } \varrho_2 \implies \text{trace} (\varrho_1 + \varrho_2)$
 $\leq 1 \implies$
 $\text{denote } S (\varrho_1 + \varrho_2) = \text{denote } S \ \varrho_1 + \text{denote } S \ \varrho_2$
 ⟨proof⟩

lemma *multfact:*

fixes $c:: \text{real}$ **and** $a:: \text{complex}$ **and** $b:: \text{complex}$
assumes $c \geq 0$ $a \leq b$
shows $c * a \leq c * b$
 ⟨proof⟩

lemma *denote-while-n-scale:*

fixes $c:: \text{real}$
assumes $c \geq 0$
 $\text{measurement } d \ 2 \ x1 \ \text{well-com } x2$
 $(\bigwedge \varrho. \varrho \in \text{carrier-mat } d \ d \implies \text{partial-density-operator } \varrho \implies \text{trace} (c \cdot_m \ \varrho) \leq 1$
 \implies
 $\text{denote } x2 (c \cdot_m \ \varrho) = c \cdot_m \ \text{denote } x2 \ \varrho$
shows $\varrho \in \text{carrier-mat } d \ d \implies \text{partial-density-operator } \varrho \implies \text{trace} (c \cdot_m \ \varrho) \leq$
 $1 \implies$

$\text{denote-while-}n (x1\ 0) (x1\ 1) (\text{denote } x2) n (\text{complex-of-real } c \cdot_m \varrho) = c \cdot_m$
 $(\text{denote-while-}n (x1\ 0) (x1\ 1) (\text{denote } x2) n \varrho)$
 <proof>

lemma *denote-while-scale:*

fixes $c :: \text{real}$
assumes $\varrho \in \text{carrier-mat } d\ d$
 $\text{partial-density-operator } \varrho$
 $\text{trace } (c \cdot_m \varrho) \leq 1\ c \geq 0$
 $\text{measurement } d\ 2\ x1\ \text{well-com } x2$
 $(\bigwedge \varrho. \varrho \in \text{carrier-mat } d\ d \implies \text{partial-density-operator } \varrho \implies \text{trace } (c \cdot_m \varrho) \leq 1$
 \implies
 $\text{denote } x2 (c \cdot_m \varrho) = c \cdot_m \text{denote } x2 \varrho)$
shows $\text{denote-while } (x1\ 0) (x1\ 1) (\text{denote } x2) (c \cdot_m \varrho) = c \cdot_m \text{denote-while } (x1$
 $0) (x1\ 1) (\text{denote } x2) \varrho$
 <proof>

lemma *denote-scale:*

fixes $c :: \text{real}$
assumes $c \geq 0$
shows $\text{well-com } S \implies \varrho \in \text{carrier-mat } d\ d \implies \text{partial-density-operator } \varrho \implies$
 $\text{trace } (c \cdot_m \varrho) \leq 1 \implies \text{denote } S (c \cdot_m \varrho) = c \cdot_m \text{denote } S \varrho$
 <proof>

lemma *limit-mat-denote-while-n:*

assumes $wc: \text{well-com } (\text{While } M\ S)$ **and** $dr: \varrho \in \text{carrier-mat } d\ d$ **and** $pdor:$
 $\text{partial-density-operator } \varrho$
shows $\text{limit-mat } (\text{matrix-sum } d (\lambda k. \text{denote-while-}n (M\ 0) (M\ 1) (\text{denote } S) k$
 $\varrho)) (\text{denote } (\text{While } M\ S) \varrho) d$
 <proof>

end

end

4 Partial state

theory *Partial-State*

imports *Quantum-Program Deep-Learning.Tensor-Matricization*

begin

lemma *nths-intersection-eq:*

assumes $\{0..<\text{length } xs\} \subseteq A$
shows $\text{nths } xs\ B = \text{nths } xs\ (A \cap B)$
 <proof>

lemma *nths-minus-eq:*

assumes $\{0..<\text{length } xs\} \subseteq A$
shows $\text{nths } xs\ (-B) = \text{nths } xs\ (A - B)$

<proof>

lemma *nths-split-complement-eq*:

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

and $\{0..<length\ xs\} \subseteq A \cup B$

shows $nths\ xs\ A = nths\ xs\ (-B)$

<proof>

lemma *lt-set-card-lt*:

fixes $A :: nat\ set$

assumes *finite* A **and** $x \in A$

shows $card\ \{y.\ y \in A \wedge y < x\} < card\ A$

<proof>

definition *ind-in-set* **where**

ind-in-set $A\ x = card\ \{i.\ i \in A \wedge i < x\}$

lemma *bij-ind-in-set-bound*:

fixes $M :: nat$ **and** $v0 :: nat\ set$

assumes $\bigwedge x.\ f\ x = card\ \{y.\ y \in v0 \wedge y < x\}$

shows *bij-betw* $f\ (\{0..<M\} \cap v0)\ \{0..<card\ (\{0..<M\} \cap v0)\}$

<proof>

lemma *ind-in-set-bound*:

fixes $A :: nat\ set$ **and** $M\ N :: nat$

assumes $N \geq M$

shows *ind-in-set* $A\ N \notin (\textit{ind-in-set}\ A\ (\{0..<M\} \cap A))$

<proof>

lemma *bij-minus-subset*:

bij-betw $f\ A\ B \implies C \subseteq A \implies (f\ 'A) - (f\ 'C) = f\ '(A - C)$

<proof>

lemma *ind-in-set-minus-subset-bound*:

fixes $A\ B :: nat\ set$ **and** $M :: nat$

assumes $B \subseteq A$

shows $(\textit{ind-in-set}\ A\ (\{0..<M\} \cap A)) - (\textit{ind-in-set}\ A\ 'B) = (\textit{ind-in-set}\ A\ '(\{0..<M\} \cap A) \cap (\textit{ind-in-set}\ A\ '(A - B)))$

<proof>

lemma *ind-in-set-iff*:

fixes $A\ B :: nat\ set$

assumes $x \in A$ **and** $B \subseteq A$

shows *ind-in-set* $A\ x \in (\textit{ind-in-set}\ A\ 'B) = (x \in B)$

<proof>

lemma *nths-reencode-eq*:

assumes $B \subseteq A$

shows $nths (nths\ xs\ A) (ind-in-set\ A\ 'B) = nths\ xs\ B$
 ⟨proof⟩

lemma *nths-reencode-eq-comp*:

assumes $B \subseteq A$

shows $nths (nths\ xs\ A) (-\ ind-in-set\ A\ 'B) = nths\ xs\ (A - B)$
 ⟨proof⟩

lemma *nths-prod-list-split*:

fixes $A :: nat\ set$ **and** $xs :: nat\ list$

assumes $B \subseteq A$

shows $prod-list\ (nths\ xs\ A) = (prod-list\ (nths\ xs\ B)) * (prod-list\ (nths\ xs\ (A - B)))$
 ⟨proof⟩

4.1 Encodings

lemma *digit-encode-take*:

$take\ n\ (digit-encode\ ds\ a) = digit-encode\ (take\ n\ ds)\ a$
 ⟨proof⟩

lemma *nths-minus-upt-eq-drop*:

$nths\ l\ (-\{..\lt n\}) = drop\ n\ l$

⟨proof⟩

lemma *digit-encode-drop*:

$drop\ n\ (digit-encode\ ds\ a) = digit-encode\ (drop\ n\ ds)\ (a\ div\ (prod-list\ (take\ n\ ds)))$

⟨proof⟩

List of active variables in the partial state

locale *partial-state* = *state-sig* +

fixes $vars :: nat\ set$

context *partial-state*

begin

Dimensions of active variables

abbreviation $avars :: nat\ set$ **where**

$avars \equiv \{0..\lt length\ dims\}$

definition $dims1 :: nat\ list$ **where**

$dims1 = nths\ dims\ vars$

definition $dims2 :: nat\ list$ **where**

$dims2 = nths\ dims\ (-vars)$

lemma *dims1-alter*:

assumes $avars \subseteq A$

shows $dims1 = nths\ dims\ (A \cap vars)$

<proof>

lemma *dims2-alter*:

assumes $avars \subseteq A$

shows $dims2 = nth\ dims\ (A - vars)$

<proof>

Total dimension for the active variables

definition $d1 :: nat$ **where**

$d1 = prod\ list\ dims1$

Total dimension for the non-active variables

definition $d2 :: nat$ **where**

$d2 = prod\ list\ dims2$

Translating dimension in d to dimension in $d1$

definition $encode1 :: nat \Rightarrow nat$ **where**

$encode1\ i = digit\ decode\ dims1\ (nth\ (digit\ encode\ dims\ i)\ vars)$

lemma *encode1-alter*:

assumes $avars \subseteq A$

shows $encode1\ i = digit\ decode\ dims1\ (nth\ (digit\ encode\ dims\ i)\ (A \cap vars))$

<proof>

Translating dimension in d to dimension in $d2$

definition $encode2 :: nat \Rightarrow nat$ **where**

$encode2\ i = digit\ decode\ dims2\ (nth\ (digit\ encode\ dims\ i)\ (-vars))$

lemma *encode2-alter*:

assumes $avars \subseteq A$

shows $encode2\ i = digit\ decode\ dims2\ (nth\ (digit\ encode\ dims\ i)\ (A - vars))$

<proof>

lemma *encode1-lt* [*simp*]:

assumes $i < d$

shows $encode1\ i < d1$

<proof>

lemma *encode2-lt* [*simp*]:

assumes $i < d$

shows $encode2\ i < d2$

<proof>

Given dimensions in $d1$ and $d2$, form dimension in d

fun $encode12 :: nat \times nat \Rightarrow nat$ **where**

$encode12\ (i, j) = digit\ decode\ dims\ (weave\ vars\ (digit\ encode\ dims1\ i)\ (digit\ encode\ dims2\ j))$

declare $encode12.simps$ [*simp del*]

lemma *encode12-inv*:
assumes $k < d$
shows $\text{encode12} (\text{encode1 } k, \text{encode2 } k) = k$
 $\langle \text{proof} \rangle$

lemma *encode12-inv1*:
assumes $i < d1 \ j < d2$
shows $\text{encode1} (\text{encode12} (i, j)) = i$
 $\langle \text{proof} \rangle$

lemma *encode12-inv2*:
assumes $i < d1 \ j < d2$
shows $\text{encode2} (\text{encode12} (i, j)) = j$
 $\langle \text{proof} \rangle$

lemma *encode12-lt*:
assumes $i < d1 \ j < d2$
shows $\text{encode12} (i, j) < d$
 $\langle \text{proof} \rangle$

lemma *sum-encode*: $(\sum i = 0..<d1. \sum j = 0..<d2. f i j) = \text{sum} (\lambda k. f (\text{encode1 } k) (\text{encode2 } k)) \{0..<d\}$
 $\langle \text{proof} \rangle$

4.2 Tensor product of vectors and matrices

Given vector $v1$ of dimension $d1$, and vector $v2$ of dimension $d2$, form the tensor vector of dimension $d1 * d2 = d$

definition *tensor-vec* :: $'a::\text{times } \text{vec} \Rightarrow 'a \ \text{vec} \Rightarrow 'a \ \text{vec}$ **where**
 $\text{tensor-vec } v1 \ v2 = \text{Matrix.vec } d (\lambda i. v1 \ \$ \ \text{encode1 } i * v2 \ \$ \ \text{encode2 } i)$

lemma *tensor-vec-dim* [*simp*]:
 $\text{dim-vec} (\text{tensor-vec } v1 \ v2) = d$
 $\langle \text{proof} \rangle$

lemma *tensor-vec-carrier*:
 $\text{tensor-vec } v1 \ v2 \in \text{carrier-vec } d$
 $\langle \text{proof} \rangle$

lemma *tensor-vec-eval*:
assumes $i < d$
shows $\text{tensor-vec } v1 \ v2 \ \$ \ i = v1 \ \$ \ \text{encode1 } i * v2 \ \$ \ \text{encode2 } i$
 $\langle \text{proof} \rangle$

lemma *tensor-vec-add1*:
fixes $v1 \ v2 \ v3 :: 'a::\text{comm-ring } \text{vec}$
assumes $v1 \in \text{carrier-vec } d1$
and $v2 \in \text{carrier-vec } d1$
and $v3 \in \text{carrier-vec } d2$

shows $\text{tensor-vec } (v1 + v2) v3 = \text{tensor-vec } v1 v3 + \text{tensor-vec } v2 v3$
 ⟨proof⟩

lemma *tensor-vec-add2*:

fixes $v1 v2 v3 :: 'a::\text{comm-ring } \text{vec}$

assumes $v1 \in \text{carrier-vec } d1$

and $v2 \in \text{carrier-vec } d2$

and $v3 \in \text{carrier-vec } d2$

shows $\text{tensor-vec } v1 (v2 + v3) = \text{tensor-vec } v1 v2 + \text{tensor-vec } v1 v3$

⟨proof⟩

Given $d1$ -by- $d1$ matrix $m1$, and $d2$ -by- $d2$ matrix $m2$, form d -by- d matrix

definition *tensor-mat* $:: 'a::\text{comm-ring-1 } \text{mat} \Rightarrow 'a \text{ mat} \Rightarrow 'a \text{ mat}$ **where**

$\text{tensor-mat } m1 m2 = \text{Matrix.mat } d d (\lambda(i,j).$

$m1 \ \$\$ (\text{encode1 } i, \text{encode1 } j) * m2 \ \$\$ (\text{encode2 } i, \text{encode2 } j))$

lemma *tensor-mat-dim-row* [*simp*]:

$\text{dim-row } (\text{tensor-mat } m1 m2) = d$

⟨proof⟩

lemma *tensor-mat-dim-col* [*simp*]:

$\text{dim-col } (\text{tensor-mat } m1 m2) = d$

⟨proof⟩

lemma *tensor-mat-carrier*:

$\text{tensor-mat } m1 m2 \in \text{carrier-mat } d d$

⟨proof⟩

lemma *tensor-mat-eval*:

assumes $i < d \ j < d$

shows $\text{tensor-mat } m1 m2 \ \$\$ (i, j) = m1 \ \$\$ (\text{encode1 } i, \text{encode1 } j) * m2 \ \$\$ (\text{encode2 } i, \text{encode2 } j)$

⟨proof⟩

lemma *tensor-mat-zero1*:

shows $\text{tensor-mat } (0_m \ d1 \ d1) m1 = 0_m \ d \ d$

⟨proof⟩

lemma *tensor-mat-zero2*:

shows $\text{tensor-mat } m1 (0_m \ d2 \ d2) = 0_m \ d \ d$

⟨proof⟩

lemma *tensor-mat-add1*:

assumes $m1 \in \text{carrier-mat } d1 \ d1$

and $m2 \in \text{carrier-mat } d1 \ d1$

and $m3 \in \text{carrier-mat } d2 \ d2$

shows $\text{tensor-mat } (m1 + m2) m3 = \text{tensor-mat } m1 m3 + \text{tensor-mat } m2 m3$

⟨proof⟩

lemma *tensor-mat-add2*:

assumes $m1 \in \text{carrier-mat } d1 \ d1$

and $m2 \in \text{carrier-mat } d2 \ d2$

and $m3 \in \text{carrier-mat } d2 \ d2$

shows $\text{tensor-mat } m1 \ (m2 + m3) = \text{tensor-mat } m1 \ m2 + \text{tensor-mat } m1 \ m3$

<proof>

lemma *tensor-mat-minus1*:

assumes $m1 \in \text{carrier-mat } d1 \ d1$

and $m2 \in \text{carrier-mat } d1 \ d1$

and $m3 \in \text{carrier-mat } d2 \ d2$

shows $\text{tensor-mat } (m1 - m2) \ m3 = \text{tensor-mat } m1 \ m3 - \text{tensor-mat } m2 \ m3$

<proof>

lemma *tensor-mat-matrix-sum2*:

assumes $m1 \in \text{carrier-mat } d1 \ d1$

shows $(\bigwedge k. k < n \implies f \ k \in \text{carrier-mat } d2 \ d2)$

$\implies \text{matrix-sum } d \ (\lambda k. \text{tensor-mat } m1 \ (f \ k)) \ n = \text{tensor-mat } m1 \ (\text{matrix-sum } d2 \ f \ n)$

<proof>

lemma *tensor-mat-scale1*:

assumes $m1 \in \text{carrier-mat } d1 \ d1$

and $m2 \in \text{carrier-mat } d2 \ d2$

shows $\text{tensor-mat } (a \cdot_m m1) \ m2 = a \cdot_m \text{tensor-mat } m1 \ m2$

<proof>

lemma *tensor-mat-scale2*:

assumes $m1 \in \text{carrier-mat } d1 \ d1$

and $m2 \in \text{carrier-mat } d2 \ d2$

shows $\text{tensor-mat } m1 \ (a \cdot_m m2) = a \cdot_m \text{tensor-mat } m1 \ m2$

<proof>

lemma *tensor-mat-trace*:

assumes $m1 \in \text{carrier-mat } d1 \ d1$

and $m2 \in \text{carrier-mat } d2 \ d2$

shows $\text{trace } (\text{tensor-mat } m1 \ m2) = \text{trace } m1 * \text{trace } m2$

<proof>

lemma *tensor-mat-id*:

$\text{tensor-mat } (1_m \ d1) \ (1_m \ d2) = 1_m \ d$

<proof>

lemma *tensor-mat-mult-vec*:

assumes $m1 \in \text{carrier-mat } d1 \ d1$

and $m2 \in \text{carrier-mat } d2 \ d2$

and $v1 \in \text{carrier-vec } d1$

and $v2 \in \text{carrier-vec } d2$

shows $\text{tensor-vec } (m1 *_{\nu} v1) \ (m2 *_{\nu} v2) = \text{tensor-mat } m1 \ m2 *_{\nu} \text{tensor-vec } v1$

v2
<proof>

lemma *tensor-mat-mult:*

assumes $m1 \in \text{carrier-mat } d1 \ d1$
and $m2 \in \text{carrier-mat } d1 \ d1$
and $m3 \in \text{carrier-mat } d2 \ d2$
and $m4 \in \text{carrier-mat } d2 \ d2$
shows $\text{tensor-mat } (m1 * m2) (m3 * m4) = \text{tensor-mat } m1 \ m3 * \text{tensor-mat } m2 \ m4$
<proof>

lemma *tensor-mat-adjoint:*

assumes $m1 \in \text{carrier-mat } d1 \ d1$
and $m2 \in \text{carrier-mat } d2 \ d2$
shows $\text{adjoint } (\text{tensor-mat } m1 \ m2) = \text{tensor-mat } (\text{adjoint } m1) (\text{adjoint } m2)$
<proof>

lemma *tensor-mat-hermitian:*

assumes $m1 \in \text{carrier-mat } d1 \ d1$
and $m2 \in \text{carrier-mat } d2 \ d2$
and *hermitian* $m1$
and *hermitian* $m2$
shows *hermitian* $(\text{tensor-mat } m1 \ m2)$
<proof>

lemma *tensor-mat-unitary:*

assumes $m1 \in \text{carrier-mat } d1 \ d1$
and $m2 \in \text{carrier-mat } d2 \ d2$
and *unitary* $m1$
and *unitary* $m2$
shows *unitary* $(\text{tensor-mat } m1 \ m2)$
<proof>

lemma *tensor-mat-positive:*

assumes $m1 \in \text{carrier-mat } d1 \ d1$
and $m2 \in \text{carrier-mat } d2 \ d2$
and *positive* $m1$
and *positive* $m2$
shows *positive* $(\text{tensor-mat } m1 \ m2)$
<proof>

lemma *tensor-mat-positive-le:*

assumes $m1 \in \text{carrier-mat } d1 \ d1$
and $m2 \in \text{carrier-mat } d2 \ d2$
and *positive* $m1$
and *positive* $m2$
and $m1 \leq_L A$
and $m2 \leq_L B$

shows $\text{tensor-mat } m1 \ m2 \leq_L \text{ tensor-mat } A \ B$
<proof>

lemma *tensor-mat-le-one*:
assumes $m1 \in \text{carrier-mat } d1 \ d1$
and $m2 \in \text{carrier-mat } d2 \ d2$
and *positive* $m1$
and *positive* $m2$
and $m1 \leq_L 1_m \ d1$
and $m2 \leq_L 1_m \ d2$
shows $\text{tensor-mat } m1 \ m2 \leq_L 1_m \ d$
<proof>

4.3 Extension of matrices

definition *mat-extension* :: $'a::\text{comm-ring-1}$ *mat* $\Rightarrow 'a$ *mat* **where**
 $\text{mat-extension } m = \text{tensor-mat } m \ (1_m \ d2)$

lemma *mat-extension-carrier*:
 $\text{mat-extension } m \in \text{carrier-mat } d \ d$
<proof>

lemma *mat-extension-add*:
assumes $m1 \in \text{carrier-mat } d1 \ d1$
and $m2 \in \text{carrier-mat } d1 \ d1$
shows $\text{mat-extension } (m1 + m2) = \text{mat-extension } m1 + \text{mat-extension } m2$
<proof>

lemma *mat-extension-trace*:
assumes $m \in \text{carrier-mat } d1 \ d1$
shows $\text{trace } (\text{mat-extension } m) = d2 * \text{trace } m$
<proof>

lemma *mat-extension-id*:
 $\text{mat-extension } (1_m \ d1) = 1_m \ d$
<proof>

lemma *mat-extension-mult*:
assumes $m1 \in \text{carrier-mat } d1 \ d1$
and $m2 \in \text{carrier-mat } d1 \ d1$
shows $\text{mat-extension } (m1 * m2) = \text{mat-extension } m1 * \text{mat-extension } m2$
<proof>

lemma *mat-extension-hermitian*:
assumes $m \in \text{carrier-mat } d1 \ d1$
and *hermitian* m
shows *hermitian* $(\text{mat-extension } m)$
<proof>

lemma *mat-extension-unitary*:
assumes $m \in \text{carrier-mat } d1 \ d1$
and *unitary* m
shows *unitary* (*mat-extension* m)
 $\langle \text{proof} \rangle$

end

abbreviation *tensor-mat* \equiv *partial-state.tensor-mat*
abbreviation *mat-extension* \equiv *partial-state.mat-extension*

context *state-sig*
begin

Swapping the order of matrices, as well as switching vars, should have no effect

lemma *tensor-mat-comm*:
assumes $\text{vars1} \cap \text{vars2} = \{\}$
and $\{0..<\text{length } \text{dims}\} \subseteq \text{vars1} \cup \text{vars2}$
and $m1 \in \text{carrier-mat } (\text{prod-list } (\text{nths } \text{dims } \text{vars1})) (\text{prod-list } (\text{nths } \text{dims } \text{vars1}))$
and $m2 \in \text{carrier-mat } (\text{prod-list } (\text{nths } \text{dims } \text{vars2})) (\text{prod-list } (\text{nths } \text{dims } \text{vars2}))$
shows *tensor-mat* *dims* *vars1* $m1$ $m2$ = *tensor-mat* *dims* *vars2* $m2$ $m1$
 $\langle \text{proof} \rangle$
end

4.4 Partial tensor product

In this context, we assume two disjoint sets of variables, and define the tensor product of two matrices on these variables

locale *partial-state2* = *state-sig* +
fixes *vars1* :: *nat set*
and *vars2* :: *nat set*
assumes *disjoint*: $\text{vars1} \cap \text{vars2} = \{\}$

begin

definition *vars0* :: *nat set* **where**
 $\text{vars0} = \text{vars1} \cup \text{vars2}$

definition *dims0* :: *nat list* **where**
 $\text{dims0} = \text{nths } \text{dims } \text{vars0}$

definition *dims1* :: *nat list* **where**
 $\text{dims1} = \text{nths } \text{dims } \text{vars1}$

definition *dims2* :: *nat list* **where**
 $\text{dims2} = \text{nths } \text{dims } \text{vars2}$

definition *d0* :: *nat* **where**

$d0 = \text{prod-list } \text{dims0}$

definition $d1 :: \text{nat}$ **where**

$d1 = \text{prod-list } \text{dims1}$

definition $d2 :: \text{nat}$ **where**

$d2 = \text{prod-list } \text{dims2}$

lemma dims-product :

$d0 = d1 * d2$

$\langle \text{proof} \rangle$

Locations of variables in vars1 relative to vars0 . For example: if $\text{vars0} = 0,1,2,4,5,6,9$ and $\text{vars1} = 1,4,6,9$, then $\text{vars1}'$ should be $1,3,5,6$.

definition $\text{vars1}' :: \text{nat set}$ **where**

$\text{vars1}' = (\text{ind-in-set } \text{vars0}) \text{ ' vars1}$

definition $\text{vars2}' :: \text{nat set}$ **where**

$\text{vars2}' = (\text{ind-in-set } \text{vars0}) \text{ ' vars2}$

lemma $\text{vars1}'I$:

$x \in \text{vars1} \implies \text{card } \{y \in \text{vars0}. y < x\} \in \text{vars1}'$

$\langle \text{proof} \rangle$

lemma $\text{vars1}'D$:

$i \in \text{vars1}' \implies \exists x \in \text{vars1}. \text{card } \{y \in \text{vars0}. y < x\} = i$

$\langle \text{proof} \rangle$

Main property of $\text{vars1}'$

lemma ind-in-set-bij :

$\text{bij-betw } (\text{ind-in-set } \text{vars0}) (\{0..<\text{length } \text{dims}\} \cap \text{vars0}) \{0..<\text{card } (\{0..<\text{length } \text{dims}\} \cap \text{vars0})\}$

$\langle \text{proof} \rangle$

lemma length-dims0 :

$\text{length } \text{dims0} = \text{card } (\{0..<\text{length } \text{dims}\} \cap \text{vars0})$

$\langle \text{proof} \rangle$

lemma $\text{length-dims0-minus-vars2}'\text{-is-vars1}'$:

$\{0..<\text{length } \text{dims0}\} - \text{vars2}' = \{0..<\text{length } \text{dims0}\} \cap \text{vars1}'$

$\langle \text{proof} \rangle$

lemma $\text{length-dims0-minus-vars1}'\text{-is-vars2}'$:

$\{0..<\text{length } \text{dims0}\} - \text{vars1}' = \{0..<\text{length } \text{dims0}\} \cap \text{vars2}'$

$\langle \text{proof} \rangle$

lemma $\text{nths-vars1}'$:

$\text{nths } \text{dims0 } \text{vars1}' = \text{dims1}$

$\langle \text{proof} \rangle$

lemma *nths-vars1'-comp*:
 $nths\ dims0\ (-vars2') = dims1$
 ⟨proof⟩

lemma *nths-vars2'*:
 $nths\ dims0\ (-vars1') = dims2$
 ⟨proof⟩

lemma *nths-vars2'-comp*:
 $nths\ dims0\ (vars2') = dims2$
 ⟨proof⟩

lemma *ptensor-encode1-encode2*:
 $partial-state.encode1\ dims0\ vars1' = partial-state.encode2\ dims0\ vars2'$
 ⟨proof⟩

lemma *ptensor-encode2-encode1*:
 $partial-state.encode1\ dims0\ vars2' = partial-state.encode2\ dims0\ vars1'$
 ⟨proof⟩

Given vector $v1$ of dimension $d1$, and vector $v2$ of dimension $d2$, form the tensor vector of dimension $d1 * d2 = d0$

definition *ptensor-vec* :: 'a::times vec \Rightarrow 'a vec \Rightarrow 'a vec **where**
 $ptensor-vec\ v1\ v2 = partial-state.tensor-vec\ dims0\ vars1'\ v1\ v2$

lemma *ptensor-vec-dim [simp]*:
 $dim-vec\ (ptensor-vec\ v1\ v2) = d0$
 ⟨proof⟩

lemma *ptensor-vec-carrier*:
 $ptensor-vec\ v1\ v2 \in carrier-vec\ d0$
 ⟨proof⟩

lemma *ptensor-vec-add*:
fixes $v1\ v2\ v3 :: 'a::comm-ring\ vec$
assumes $v1 \in carrier-vec\ d1$
and $v2 \in carrier-vec\ d1$
and $v3 \in carrier-vec\ d2$
shows $ptensor-vec\ (v1 + v2)\ v3 = ptensor-vec\ v1\ v3 + ptensor-vec\ v2\ v3$
 ⟨proof⟩

definition *ptensor-mat* :: 'a::comm-ring-1 mat \Rightarrow 'a mat \Rightarrow 'a mat **where**
 $ptensor-mat\ m1\ m2 = partial-state.tensor-mat\ dims0\ vars1'\ m1\ m2$

lemma *ptensor-mat-dim-row [simp]*:
 $dim-row\ (ptensor-mat\ m1\ m2) = d0$
 ⟨proof⟩

lemma *ptensor-mat-dim-col* [*simp*]:
 $\text{dim-col } (\text{ptensor-mat } m1 \ m2) = d0$
 ⟨*proof*⟩

lemma *ptensor-mat-carrier*:
 $\text{ptensor-mat } m1 \ m2 \in \text{carrier-mat } d0 \ d0$
 ⟨*proof*⟩

lemma *ptensor-mat-add*:
 assumes $m1 \in \text{carrier-mat } d1 \ d1$
 and $m2 \in \text{carrier-mat } d1 \ d1$
 and $m3 \in \text{carrier-mat } d2 \ d2$
 shows $\text{ptensor-mat } (m1 + m2) \ m3 = \text{ptensor-mat } m1 \ m3 + \text{ptensor-mat } m2 \ m3$
 ⟨*proof*⟩

lemma *ptensor-mat-trace*:
 assumes $m1 \in \text{carrier-mat } d1 \ d1$
 and $m2 \in \text{carrier-mat } d2 \ d2$
 shows $\text{trace } (\text{ptensor-mat } m1 \ m2) = \text{trace } m1 * \text{trace } m2$
 ⟨*proof*⟩

lemma *ptensor-mat-id*:
 $\text{ptensor-mat } (1_m \ d1) \ (1_m \ d2) = 1_m \ d0$
 ⟨*proof*⟩

lemma *ptensor-mat-mult*:
 assumes $m1 \in \text{carrier-mat } d1 \ d1$
 and $m2 \in \text{carrier-mat } d1 \ d1$
 and $m3 \in \text{carrier-mat } d2 \ d2$
 and $m4 \in \text{carrier-mat } d2 \ d2$
 shows $\text{ptensor-mat } (m1 * m2) \ (m3 * m4) = \text{ptensor-mat } m1 \ m3 * \text{ptensor-mat } m2 \ m4$
 ⟨*proof*⟩

lemma *ptensor-mat-mult-vec*:
 assumes $m1 \in \text{carrier-mat } d1 \ d1$
 and $v1 \in \text{carrier-vec } d1$
 and $m2 \in \text{carrier-mat } d2 \ d2$
 and $v2 \in \text{carrier-vec } d2$
 shows $\text{ptensor-vec } (m1 *_v v1) \ (m2 *_v v2) = \text{ptensor-mat } m1 \ m2 *_v \text{ptensor-vec } v1 \ v2$
 ⟨*proof*⟩

4.5 Partial extensions

definition *pmat-extension* :: '*a*::comm-ring-1 mat \Rightarrow '*a* mat **where**
 $\text{pmat-extension } m = \text{ptensor-mat } m \ (1_m \ d2)$

lemma *pmat-extension-carrier*:
pmat-extension $m \in \text{carrier-mat } d0 \ d0$
 ⟨*proof*⟩

lemma *pmat-extension-add*:
assumes $m1 \in \text{carrier-mat } d1 \ d1$
and $m2 \in \text{carrier-mat } d1 \ d1$
shows $\text{pmat-extension } (m1 + m2) = \text{pmat-extension } m1 + \text{pmat-extension } m2$
 ⟨*proof*⟩

lemma *pmat-extension-trace*:
assumes $m \in \text{carrier-mat } d1 \ d1$
shows $\text{trace } (\text{pmat-extension } m) = d2 * \text{trace } m$
 ⟨*proof*⟩

lemma *pmat-extension-id*:
pmat-extension $(1_m \ d1) = 1_m \ d0$
 ⟨*proof*⟩

lemma *pmat-extension-mult*:
assumes $m1 \in \text{carrier-mat } d1 \ d1$
and $m2 \in \text{carrier-mat } d1 \ d1$
shows $\text{pmat-extension } (m1 * m2) = \text{pmat-extension } m1 * \text{pmat-extension } m2$
 ⟨*proof*⟩

end

context *state-sig*
begin

abbreviation *ptensor-vec* $\equiv \text{partial-state2.ptensor-vec}$
abbreviation *ptensor-mat* $\equiv \text{partial-state2.ptensor-mat}$
abbreviation *pmat-extension* $\equiv \text{partial-state2.pmat-extension}$

Key property: commutativity of tensor product

lemma *ptensor-mat-comm*:
fixes $m1 \ m2 :: \text{complex mat}$
assumes $\text{vars1} \cap \text{vars2} = \{\}$
shows $\text{ptensor-mat } \text{dims } \text{vars1 } \text{vars2 } m1 \ m2 = \text{ptensor-mat } \text{dims } \text{vars2 } \text{vars1 } m2$
 $m1$
 ⟨*proof*⟩

Key property: associativity of tensor product

lemma *ind-in-set-mono*:
fixes $a \ b :: \text{nat}$ **and** $A :: \text{nat set}$
assumes $a \in A \ b \in A \ a < b$
shows $\text{ind-in-set } A \ a < \text{ind-in-set } A \ b$
 ⟨*proof*⟩

lemma *ind-in-set-inj*:

fixes $a\ b :: \text{nat}$ **and** $A :: \text{nat set}$

assumes $a \in A\ b \in A$ *ind-in-set* $A\ a = \text{ind-in-set } A\ b$

shows $a = b$

<proof>

lemma *ind-in-set-mono2*:

fixes $a\ b :: \text{nat}$ **and** $A :: \text{nat set}$

assumes $a \in A\ b \in A$ *ind-in-set* $A\ a < \text{ind-in-set } A\ b$

shows $a < b$

<proof>

lemma *ind-in-set-bij-betw*:

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

assumes $B \subseteq A\ c \in B$

shows *bij-betw* (*ind-in-set* A) $\{i \in B.\ i < c\}$ $\{i \in \text{ind-in-set } A\ ' B.\ i < \text{ind-in-set } A\ c\}$

<proof>

lemma *ind-in-set-assoc*:

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

assumes $C \subseteq B\ B \subseteq A$

shows *ind-in-set* (*ind-in-set* $A\ ' B$) $'$ (*ind-in-set* $A\ ' C$) = *ind-in-set* $B\ ' C$

<proof>

lemma *nths-reencode-eq3*:

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

assumes $C \subseteq B\ B \subseteq A$

shows *nths* (*nths* xs (*ind-in-set* $A\ ' B$)) (*ind-in-set* $B\ ' C$) = *nths* xs (*ind-in-set* $A\ ' C$)

<proof>

lemma *nths-assoc-three-A*:

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

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

and $(A \cup B) \cap C = \{\}$

shows *nths* (*nths* xs (*ind-in-set* $(A \cup B \cup C)\ ' (A \cup B)$)) (*ind-in-set* $(A \cup B)\ ' A$)

= *nths* xs (*ind-in-set* $(A \cup B \cup C)\ ' A$)

<proof>

lemma *nths-assoc-three-B*:

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

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

and $(A \cup B) \cap C = \{\}$

shows *nths* (*nths* xs (*ind-in-set* $(A \cup B \cup C)\ ' (A \cup B)$)) (*ind-in-set* $(A \cup B)\ ' B$)

= *nths* (*nths* xs (*ind-in-set* $(A \cup B \cup C)\ ' (B \cup C)$)) (*ind-in-set* $(B \cup C)\ ' B$)

<proof>

$\langle proof \rangle$

lemma *nths-assoc-three-C*:

fixes $A B C :: nat\ set$

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

and $(A \cup B) \cap C = \{\}$

shows $nths\ (nths\ xs\ (ind-in-set\ (A \cup B \cup C)\ ' (B \cup C)))\ (ind-in-set\ (B \cup C)\ ' C)$

$=\ nths\ xs\ (ind-in-set\ (A \cup B \cup C)\ ' C)$

$\langle proof \rangle$

lemma *valid-index-ind-in-set*:

assumes $is \triangleleft nths\ dims\ A\ B \subseteq A$

shows $nths\ is\ (ind-in-set\ A\ ' B) \triangleleft nths\ dims\ B$

$\langle proof \rangle$

lemma *ind-in-set-id*:

fixes $A :: nat\ set$

assumes *finite* A

shows $ind-in-set\ A\ ' A = \{0..< card\ A\}$

$\langle proof \rangle$

lemma *nths-complement-ind-in-set*:

fixes $A B :: nat\ set$

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

$card\ (A \cup B) = length\ xs$

shows $nths\ xs\ (-\ ind-in-set\ (A \cup B)\ ' A) = nths\ xs\ (ind-in-set\ (A \cup B)\ ' B)$

$\langle proof \rangle$

lemma *ind-in-set-inj'*:

fixes $A B :: nat\ set$

assumes $B \subseteq A$

shows $inj-on\ (ind-in-set\ A)\ B$

$\langle proof \rangle$

lemma *ind-in-set-less*:

fixes $x :: nat$ **and** $A :: nat\ set$

assumes *finite* $A\ x \in A$

shows $ind-in-set\ A\ x < card\ A$

$\langle proof \rangle$

lemma *ptensor-mat-assoc*:

assumes $vars1 \cap vars2 = \{\}$

and $(vars1 \cup vars2) \cap vars3 = \{\}$

and $vars1 \cup vars2 \cup vars3 \subseteq \{0..< length\ dims\}$

shows $ptensor-mat\ dims\ (vars1 \cup vars2)\ vars3\ (ptensor-mat\ dims\ vars1\ vars2\ m1\ m2)\ m3 =$

$ptensor-mat\ dims\ vars1\ (vars2 \cup vars3)\ m1\ (ptensor-mat\ dims\ vars2\ vars3\ m2\ m3)$

<proof>

Some simple consequences of associativity

lemma *pmat-extension-assoc*:

assumes $vars1 \cap vars2 = \{\}$

and $(vars1 \cup vars2) \cap vars3 = \{\}$

and $vars1 \cup vars2 \cup vars3 \subseteq \{0..< length\ dims\}$

shows $pmat-extension\ dims\ vars1\ (vars2 \cup vars3)\ m =$

$pmat-extension\ dims\ (vars1 \cup vars2)\ vars3\ (pmat-extension\ dims\ vars1$

$vars2\ m)$

<proof>

end

4.6 Commands on subset of variables

context *state-sig*

begin

definition *Utrans-P* :: $nat\ set \Rightarrow complex\ mat \Rightarrow com$ **where**

$Utrans-P\ vars\ U = Utrans\ (mat-extension\ dims\ vars\ U)$

lemma *well-com-Utrans-P*:

assumes $U \in carrier-mat\ (prod-list\ (nth\ dims\ vars))\ (prod-list\ (nth\ dims\ vars))$

and *unitary* U

shows *well-com* $(Utrans-P\ vars\ U)$

<proof>

definition *Measure-P* :: $nat\ set \Rightarrow nat \Rightarrow (nat \Rightarrow complex\ mat) \Rightarrow com\ list \Rightarrow com$ **where**

$Measure-P\ vars\ n\ Ps\ Cs = Measure\ n\ (\lambda n. mat-extension\ dims\ vars\ (Ps\ n))\ Cs$

definition *While-P* :: $nat\ set \Rightarrow complex\ mat \Rightarrow complex\ mat \Rightarrow com \Rightarrow com$ **where**

$While-P\ vars\ M0\ M1\ C = While\ (\lambda n.$

$if\ n = 0\ then\ mat-extension\ dims\ vars\ M0$

$else\ if\ n = 1\ then\ mat-extension\ dims\ vars\ M1$

$else\ undefined)\ C$

end

end

5 Standard gates

theory *Gates*

imports *Complex-Matrix*

begin

Pauli matrices

definition *sigma-x* :: complex mat **where**
sigma-x = mat-of-rows-list 2 [[0, 1], [1, 0]]

definition *sigma-y* :: complex mat **where**
sigma-y = mat-of-rows-list 2 [[0, -i], [i, 0]]

definition *sigma-z* :: complex mat **where**
sigma-z = mat-of-rows-list 2 [[1, 0], [0, -1]]

Hadamard matrices

definition *hadamard* :: complex mat **where**
hadamard = mat 2 2 ($\lambda(i, j)$. if ($i = 0 \vee j = 0$) then 1 / csqrt 2 else - 1 / sqrt 2)

lemma *hadamard-dim*:
hadamard \in carrier-mat 2 2
 <proof>

lemma *hermitian-hadamard*:
hermitian hadamard
 <proof>

lemma *csqrt-2-sq*:
 complex-of-real (sqrt 2) * complex-of-real (sqrt 2) = 2
 <proof>

lemma *sum-le-2*:
 $\bigwedge(f :: nat \Rightarrow complex)$. sum f {0..<2} = f 0 + f 1
 <proof>

lemma *unitary-hadamard*:
unitary hadamard
 <proof>

The matrix [0 0 .. 0 1 1 0 .. 0 0 0 1 .. 0 0 0 0 .. 1 0] implements $i := i + 1$ in the last variable.

definition *mat-incr* :: nat \Rightarrow complex mat **where**
mat-incr n = mat n n ($\lambda(i, j)$. if $i = 0$ then (if $j = n - 1$ then 1 else 0) else (if $i = j + 1$ then 1 else 0))

lemma *mat-incr-dim*:
mat-incr n \in carrier-mat n n
 <proof>

lemma *adjoint-mat-incr*:
 adjoint (mat-incr n) = mat n n ($\lambda(i, j)$. if $j = 0$ then (if $i = n - 1$ then 1 else 0) else (if $j = i + 1$ then 1 else 0))
 <proof>

lemma *mat-incr-mult-adjoint-mat-incr*:
shows $\text{mat-incr } n * (\text{adjoint } (\text{mat-incr } n)) = 1_m \ n$
 $\langle \text{proof} \rangle$

lemma *unitary-mat-incr*:
unitary $(\text{mat-incr } n)$
 $\langle \text{proof} \rangle$

end

6 Partial and total correctness

theory *Quantum-Hoare*
imports *Quantum-Program*
begin

context *state-sig*
begin

definition *density-states* :: *state set where*
 $\text{density-states} = \{\varrho \in \text{carrier-mat } d \ d. \text{ partial-density-operator } \varrho\}$

lemma *denote-density-states*:
 $\varrho \in \text{density-states} \implies \text{well-com } S \implies \text{denote } S \ \varrho \in \text{density-states}$
 $\langle \text{proof} \rangle$

definition *is-quantum-predicate* :: *complex mat \implies bool where*
 $\text{is-quantum-predicate } P \iff P \in \text{carrier-mat } d \ d \wedge \text{positive } P \wedge P \leq_L 1_m \ d$

lemma *trace-measurement2*:
assumes m : *measurement* $n \ 2 \ M$ **and** dA : $A \in \text{carrier-mat } n \ n$
shows $\text{trace } ((M \ 0) * A * \text{adjoint } (M \ 0)) + \text{trace } ((M \ 1) * A * \text{adjoint } (M \ 1))$
 $= \text{trace } A$
 $\langle \text{proof} \rangle$

lemma *qp-close-under-unitary-operator*:
fixes $U \ P$:: *complex mat*
assumes dU : $U \in \text{carrier-mat } d \ d$
and u : *unitary* U
and qp : *is-quantum-predicate* P
shows *is-quantum-predicate* $(\text{adjoint } U * P * U)$
 $\langle \text{proof} \rangle$

lemma *qps-after-measure-is-qp*:
assumes m : *measurement* $d \ n \ M$ **and** qpk : $\bigwedge k. k < n \implies \text{is-quantum-predicate}$
 $(P \ k)$
shows *is-quantum-predicate* $(\text{matrix-sum } d \ (\lambda k. \text{adjoint } (M \ k) * P \ k * M \ k) \ n)$
 $\langle \text{proof} \rangle$

definition *hoare-total-correct* :: *complex mat* \Rightarrow *com* \Rightarrow *complex mat* \Rightarrow *bool* (\models_t $\{(1-)\}/ (-)/ \{(1-)\}$ 50) **where**
 $\models_t \{P\} S \{Q\} \longleftrightarrow (\forall \varrho \in \text{density-states. } \text{trace } (P * \varrho) \leq \text{trace } (Q * \text{denote } S \varrho))$

definition *hoare-partial-correct* :: *complex mat* \Rightarrow *com* \Rightarrow *complex mat* \Rightarrow *bool* (\models_p $\{(1-)\}/ (-)/ \{(1-)\}$ 50) **where**
 $\models_p \{P\} S \{Q\} \longleftrightarrow (\forall \varrho \in \text{density-states. } \text{trace } (P * \varrho) \leq \text{trace } (Q * \text{denote } S \varrho) + (\text{trace } \varrho - \text{trace } (\text{denote } S \varrho)))$

lemma *total-implies-partial*:
assumes *S*: *well-com S*
and *total*: $\models_t \{P\} S \{Q\}$
shows $\models_p \{P\} S \{Q\}$
 $\langle \text{proof} \rangle$

lemma *predicate-prob-positive*:
assumes $0_m \ d \ d \leq_L P$
and $\varrho \in \text{density-states}$
shows $0 \leq \text{trace } (P * \varrho)$
 $\langle \text{proof} \rangle$

lemma *total-pre-zero*:
assumes *S*: *well-com S*
and *Q*: *is-quantum-predicate Q*
shows $\models_t \{0_m \ d \ d\} S \{Q\}$
 $\langle \text{proof} \rangle$

lemma *partial-post-identity*:
assumes *S*: *well-com S*
and *P*: *is-quantum-predicate P*
shows $\models_p \{P\} S \{1_m \ d\}$
 $\langle \text{proof} \rangle$

6.1 Weakest liberal preconditions

definition *is-weakest-liberal-precondition* :: *complex mat* \Rightarrow *com* \Rightarrow *complex mat* \Rightarrow *bool* **where**
is-weakest-liberal-precondition *W S P* \longleftrightarrow
is-quantum-predicate *W* $\wedge \models_p \{W\} S \{P\} \wedge (\forall Q. \text{is-quantum-predicate } Q \longrightarrow \models_p \{Q\} S \{P\} \longrightarrow Q \leq_L W)$

definition *wlp-measure* :: *nat* \Rightarrow (*nat* \Rightarrow *complex mat*) \Rightarrow ((*complex mat* \Rightarrow *complex mat*) *list*) \Rightarrow *complex mat* \Rightarrow *complex mat* **where**
wlp-measure *n M WS P* = *matrix-sum* *d* ($\lambda k. \text{adjoint } (M \ k) * ((WS!k) \ P) * (M \ k)$) *n*

fun *wlp-while-n* :: *complex mat* \Rightarrow *complex mat* \Rightarrow (*complex mat* \Rightarrow *complex mat*)
 \Rightarrow *nat* \Rightarrow *complex mat* \Rightarrow *complex mat* **where**
wlp-while-n *M0 M1 WS 0 P* = $1_m d$
| *wlp-while-n* *M0 M1 WS (Suc n) P* = *adjoint M0 * P * M0* + *adjoint M1 * (WS*
(*wlp-while-n* *M0 M1 WS n P*) ** M1*)

lemma *measurement2-leq-one-mat*:

assumes *dP*: $P \in \text{carrier-mat } d \ d$ **and** *dQ*: $Q \in \text{carrier-mat } d \ d$
and *leP*: $P \leq_L 1_m \ d$ **and** *leQ*: $Q \leq_L 1_m \ d$ **and** *m*: *measurement d 2 M*
shows (*adjoint (M 0) * P * (M 0)* + *adjoint (M 1) * Q * (M 1)*) $\leq_L 1_m \ d$
<proof>

lemma *wlp-while-n-close*:

assumes *close*: $\bigwedge P. \text{is-quantum-predicate } P \Longrightarrow \text{is-quantum-predicate } (\text{WS } P)$
and *m*: *measurement d 2 M* **and** *qpP*: *is-quantum-predicate P*
shows *is-quantum-predicate (wlp-while-n (M 0) (M 1) WS k P)*
<proof>

lemma *wlp-while-n-mono*:

assumes $\bigwedge P. \text{is-quantum-predicate } P \Longrightarrow \text{is-quantum-predicate } (\text{WS } P)$
and $\bigwedge P \ Q. \text{is-quantum-predicate } P \Longrightarrow \text{is-quantum-predicate } Q \Longrightarrow P \leq_L Q$
 $\Longrightarrow \text{WS } P \leq_L \text{WS } Q$
and *measurement d 2 M*
and *is-quantum-predicate P*
and *is-quantum-predicate Q*
and $P \leq_L Q$
shows (*wlp-while-n (M 0) (M 1) WS k P*) \leq_L (*wlp-while-n (M 0) (M 1) WS k*
Q)
<proof>

definition *wlp-while* :: *complex mat* \Rightarrow *complex mat* \Rightarrow (*complex mat* \Rightarrow *complex*
mat) \Rightarrow *complex mat* \Rightarrow *complex mat* **where**

wlp-while *M0 M1 WS P* = (*THE Q. limit-mat* ($\lambda n. \text{wlp-while-n } M0 \ M1 \ WS \ n$
P) *Q d*)

lemma *wlp-while-exists*:

assumes $\bigwedge P. \text{is-quantum-predicate } P \Longrightarrow \text{is-quantum-predicate } (\text{WS } P)$
and $\bigwedge P \ Q. \text{is-quantum-predicate } P \Longrightarrow \text{is-quantum-predicate } Q \Longrightarrow P \leq_L Q$
 $\Longrightarrow \text{WS } P \leq_L \text{WS } Q$
and *m*: *measurement d 2 M*
and *qpP*: *is-quantum-predicate P*
shows *is-quantum-predicate (wlp-while (M 0) (M 1) WS P)*
 $\wedge (\forall n. (\text{wlp-while } (M 0) (M 1) WS \ n \ P) \leq_L (\text{wlp-while-n } (M 0) (M 1) WS \ n$
P))
 $\wedge (\forall W'. (\forall n. W' \leq_L (\text{wlp-while-n } (M 0) (M 1) WS \ n \ P)) \longrightarrow W' \leq_L$
(*wlp-while (M 0) (M 1) WS P*))
 $\wedge \text{limit-mat } (\lambda n. \text{wlp-while-n } (M 0) (M 1) WS \ n \ P) (\text{wlp-while } (M 0) (M 1)$
WS P) d
<proof>

lemma *wlp-while-mono*:

assumes $\bigwedge P. \text{is-quantum-predicate } P \implies \text{is-quantum-predicate } (WS\ P)$
and $\bigwedge P\ Q. \text{is-quantum-predicate } P \implies \text{is-quantum-predicate } Q \implies P \leq_L Q$
 $\implies WS\ P \leq_L WS\ Q$
and *measurement* $d \geq M$
and *is-quantum-predicate* P
and *is-quantum-predicate* Q
and $P \leq_L Q$
shows $wlp\text{-while } (M\ 0)\ (M\ 1)\ WS\ P \leq_L wlp\text{-while } (M\ 0)\ (M\ 1)\ WS\ Q$
 $\langle \text{proof} \rangle$

fun *wlp* :: *com* \Rightarrow *complex mat* \Rightarrow *complex mat* **where**

wlp *SKIP* $P = P$
 $| \text{wlp } (Utrans\ U) P = \text{adjoint } U * P * U$
 $| \text{wlp } (Seq\ S1\ S2) P = \text{wlp } S1\ (\text{wlp } S2\ P)$
 $| \text{wlp } (Measure\ n\ M\ S) P = \text{wlp-measure } n\ M\ (\text{map } \text{wlp } S) P$
 $| \text{wlp } (While\ M\ S) P = \text{wlp-while } (M\ 0)\ (M\ 1)\ (\text{wlp } S) P$

lemma *wlp-measure-expand-m*:

assumes $m: m \leq n$ **and** $wc: \text{well-com } (Measure\ n\ M\ S)$
shows $wlp\ (Measure\ m\ M\ S) P = \text{matrix-sum } d\ (\lambda k. \text{adjoint } (M\ k) * (\text{wlp } (S!k) P) * (M\ k))\ m$
 $\langle \text{proof} \rangle$

lemma *wlp-measure-expand*:

assumes $wc: \text{well-com } (Measure\ n\ M\ S)$
shows $wlp\ (Measure\ n\ M\ S) P = \text{matrix-sum } d\ (\lambda k. \text{adjoint } (M\ k) * (\text{wlp } (S!k) P) * (M\ k))\ n$
 $\langle \text{proof} \rangle$

lemma *wlp-mono-and-close*:

shows $\text{well-com } S \implies \text{is-quantum-predicate } P \implies \text{is-quantum-predicate } Q \implies P \leq_L Q$
 $\implies \text{is-quantum-predicate } (\text{wlp } S\ P) \wedge \text{wlp } S\ P \leq_L \text{wlp } S\ Q$
 $\langle \text{proof} \rangle$

lemma *wlp-close*:

assumes $wc: \text{well-com } S$ **and** $qp: \text{is-quantum-predicate } P$
shows $\text{is-quantum-predicate } (\text{wlp } S\ P)$
 $\langle \text{proof} \rangle$

lemma *wlp-soundness*:

$\text{well-com } S \implies$
 $(\bigwedge P. (\text{is-quantum-predicate } P \implies$
 $(\forall \varrho \in \text{density-states. } \text{trace } (\text{wlp } S\ P * \varrho) = \text{trace } (P * (\text{denote } S\ \varrho)) + \text{trace } \varrho - \text{trace } (\text{denote } S\ \varrho))))$
 $\langle \text{proof} \rangle$

lemma *denote-while-split*:

assumes *wc*: *well-com* (*While M S*) **and** *dsr*: $\rho \in \text{density-states}$
shows $\text{denote } (\text{While } M \ S) \ \rho = (M \ 0) * \rho * \text{adjoint } (M \ 0) + \text{denote } (\text{While } M \ S) \ (\text{denote } S \ (M \ 1 * \rho * \text{adjoint } (M \ 1)))$
<proof>

lemma *wlp-while-split*:

assumes *wc*: *well-com* (*While M S*) **and** *qpP*: *is-quantum-predicate P*
shows $\text{wlp } (\text{While } M \ S) \ P = \text{adjoint } (M \ 0) * P * (M \ 0) + \text{adjoint } (M \ 1) * (\text{wlp } S \ (\text{wlp } (\text{While } M \ S) \ P)) * (M \ 1)$
<proof>

lemma *wlp-is-weakest-liberal-precondition*:

assumes *well-com S* **and** *is-quantum-predicate P*
shows *is-weakest-liberal-precondition* ($\text{wlp } S \ P$) $S \ P$
<proof>

6.2 Hoare triples for partial correctness

inductive *hoare-partial* :: *complex mat* \Rightarrow *com* \Rightarrow *complex mat* \Rightarrow *bool* (\vdash_p ($\{(1-)\}$ / $(-)/\{(1-)\}$) 50) **where**

is-quantum-predicate P $\Longrightarrow \vdash_p \{P\} \text{SKIP} \{P\}$
 $|$ *is-quantum-predicate P* $\Longrightarrow \vdash_p \{\text{adjoint } U * P * U\} \text{Utrans } U \{P\}$
 $|$ *is-quantum-predicate P* \Longrightarrow *is-quantum-predicate Q* \Longrightarrow *is-quantum-predicate R*
 \Longrightarrow
 $\vdash_p \{P\} \ S1 \ \{Q\} \Longrightarrow \vdash_p \{Q\} \ S2 \ \{R\} \Longrightarrow$
 $\vdash_p \{P\} \ \text{Seq } S1 \ S2 \ \{R\}$
 $|$ ($\bigwedge k. k < n \Longrightarrow$ *is-quantum-predicate* ($P \ k$)) \Longrightarrow *is-quantum-predicate Q* \Longrightarrow
 $(\bigwedge k. k < n \Longrightarrow \vdash_p \{P \ k\} \ S \ ! \ k \ \{Q\}) \Longrightarrow$
 $\vdash_p \{\text{matrix-sum } d \ (\lambda k. \text{adjoint } (M \ k) * P \ k * M \ k) \ n\} \ \text{Measure } n \ M \ S \ \{Q\}$
 $|$ *is-quantum-predicate P* \Longrightarrow *is-quantum-predicate Q* \Longrightarrow
 $\vdash_p \{Q\} \ S \ \{\text{adjoint } (M \ 0) * P * M \ 0 + \text{adjoint } (M \ 1) * Q * M \ 1\} \Longrightarrow$
 $\vdash_p \{\text{adjoint } (M \ 0) * P * M \ 0 + \text{adjoint } (M \ 1) * Q * M \ 1\} \ \text{While } M \ S \ \{P\}$
 $|$ *is-quantum-predicate P* \Longrightarrow *is-quantum-predicate Q* \Longrightarrow *is-quantum-predicate P'*
 \Longrightarrow *is-quantum-predicate Q'* \Longrightarrow
 $P \leq_L P' \Longrightarrow \vdash_p \{P'\} \ S \ \{Q'\} \Longrightarrow Q' \leq_L Q \Longrightarrow \vdash_p \{P\} \ S \ \{Q\}$

theorem *hoare-partial-sound*:

$\vdash_p \{P\} \ S \ \{Q\} \Longrightarrow \text{well-com } S \Longrightarrow \models_p \{P\} \ S \ \{Q\}$
<proof>

lemma *wlp-complete*:

$\text{well-com } S \Longrightarrow \text{is-quantum-predicate } P \Longrightarrow \vdash_p \{\text{wlp } S \ P\} \ S \ \{P\}$
<proof>

theorem *hoare-partial-complete*:

$\models_p \{P\} \ S \ \{Q\} \Longrightarrow \text{well-com } S \Longrightarrow \text{is-quantum-predicate } P \Longrightarrow \text{is-quantum-predicate } Q \Longrightarrow \vdash_p \{P\} \ S \ \{Q\}$
<proof>

6.3 Consequences of completeness

lemma *hoare-patual-seq-assoc-sem*:

shows $\models_p \{A\} (S1 ;; S2) ;; S3 \{B\} \longleftrightarrow \models_p \{A\} S1 ;; (S2 ;; S3) \{B\}$
<proof>

lemma *hoare-patual-seq-assoc*:

assumes *well-com S1 and well-com S2 and well-com S3*

and *is-quantum-predicate A and is-quantum-predicate B*

shows $\vdash_p \{A\} (S1 ;; S2) ;; S3 \{B\} \longleftrightarrow \vdash_p \{A\} S1 ;; (S2 ;; S3) \{B\}$
<proof>

end

end

7 Grover's algorithm

theory *Grover*

imports *Partial-State Gates Quantum-Hoare*

begin

7.1 Basic definitions

locale *grover-state* =

fixes $n :: \text{nat}$

and $f :: \text{nat} \Rightarrow \text{bool}$

assumes $n: n > 1$

and $\text{dim}M: \text{card} \{i. i < (2::\text{nat})^{\wedge} n \wedge f i\} > 0$

$\text{card} \{i. i < (2::\text{nat})^{\wedge} n \wedge f i\} < (2::\text{nat})^{\wedge} n$

begin

definition *N* **where**

$N = (2::\text{nat})^{\wedge} n$

definition *M* **where**

$M = \text{card} \{i. i < N \wedge f i\}$

lemma *N-ge-0 [simp]*: $0 < N$ *<proof>*

lemma *M-ge-0 [simp]*: $0 < M$ *<proof>*

lemma *M-neq-0 [simp]*: $M \neq 0$ *<proof>*

lemma *M-le-N [simp]*: $M < N$ *<proof>*

lemma *M-not-ge-N [simp]*: $\neg M \geq N$ *<proof>*

definition $\psi :: \text{complex vec}$ **where**

$\psi = \text{Matrix.vec } N (\lambda i. 1 / \text{sqrt } N)$

lemma ψ -dim [simp]:

$\psi \in \text{carrier-vec } N$

$\text{dim-vec } \psi = N$

$\langle \text{proof} \rangle$

lemma ψ -eval:

$i < N \implies \psi \$ i = 1 / \text{sqrt } N$

$\langle \text{proof} \rangle$

lemma ψ -inner:

$\text{inner-prod } \psi \psi = 1$

$\langle \text{proof} \rangle$

lemma ψ -norm:

$\text{vec-norm } \psi = 1$

$\langle \text{proof} \rangle$

definition α :: complex vec **where**

$\alpha = \text{Matrix.vec } N (\lambda i. \text{if } f \ i \ \text{then } 0 \ \text{else } 1 / \text{sqrt } (N - M))$

lemma α -dim [simp]:

$\alpha \in \text{carrier-vec } N$

$\text{dim-vec } \alpha = N$

$\langle \text{proof} \rangle$

lemma α -eval:

$i < N \implies \alpha \$ i = (\text{if } f \ i \ \text{then } 0 \ \text{else } 1 / \text{sqrt } (N - M))$

$\langle \text{proof} \rangle$

lemma α -inner:

$\text{inner-prod } \alpha \alpha = 1$

$\langle \text{proof} \rangle$

definition β :: complex vec **where**

$\beta = \text{Matrix.vec } N (\lambda i. \text{if } f \ i \ \text{then } 1 / \text{sqrt } M \ \text{else } 0)$

lemma β -dim [simp]:

$\beta \in \text{carrier-vec } N$

$\text{dim-vec } \beta = N$

$\langle \text{proof} \rangle$

lemma β -eval:

$i < N \implies \beta \$ i = (\text{if } f \ i \ \text{then } 1 / \text{sqrt } M \ \text{else } 0)$

$\langle \text{proof} \rangle$

lemma β -inner:

$\text{inner-prod } \beta \beta = 1$

<proof>

lemma *alpha-beta-orth:*

inner-prod $\alpha \beta = 0$

<proof>

lemma *beta-alpha-orth:*

inner-prod $\beta \alpha = 0$

<proof>

definition $\vartheta :: \text{real where}$

$\vartheta = 2 * \arccos (\text{sqrt} ((N - M) / N))$

lemma *cos-theta-div-2:*

$\cos (\vartheta / 2) = \text{sqrt} ((N - M) / N)$

<proof>

lemma *sin-theta-div-2:*

$\sin (\vartheta / 2) = \text{sqrt} (M / N)$

<proof>

lemma *ϑ -neq-0:*

$\vartheta \neq 0$

<proof>

abbreviation *ccos where* $\text{ccos } \varphi \equiv \text{complex-of-real } (\cos \varphi)$

abbreviation *csin where* $\text{csin } \varphi \equiv \text{complex-of-real } (\sin \varphi)$

lemma *ψ -eq:*

$\psi = \text{ccos} (\vartheta / 2) \cdot_v \alpha + \text{csin} (\vartheta / 2) \cdot_v \beta$

<proof>

lemma *psi-inner-alpha:*

inner-prod $\psi \alpha = \text{ccos} (\vartheta / 2)$

<proof>

lemma *psi-inner-beta:*

inner-prod $\psi \beta = \text{csin} (\vartheta / 2)$

<proof>

definition *alpha-l :: nat \Rightarrow complex where*

alpha-l $l = \text{ccos} ((l + 1 / 2) * \vartheta)$

lemma *alpha-l-real:*

alpha-l $l \in \text{Reals}$

<proof>

lemma *cnj-alpha-l:*

conjugate (*alpha-l* l) = *alpha-l* l

<proof>

definition *beta-l* :: *nat* \Rightarrow *complex* **where**
 $beta-l\ l = c\sin\ ((l + 1 / 2) * \vartheta)$

lemma *beta-l-real*:
 $beta-l\ l \in Reals$
<proof>

lemma *cnj-beta-l*:
 $conjugate\ (beta-l\ l) = beta-l\ l$
<proof>

lemma *csin-ccos-squared-add*:
 $ccos\ (a::real) * ccos\ a + csin\ a * csin\ a = 1$
<proof>

lemma *alpha-l-beta-l-add-norm*:
 $alpha-l\ l * alpha-l\ l + beta-l\ l * beta-l\ l = 1$
<proof>

definition *psi-l* **where**
 $psi-l\ l = (alpha-l\ l) \cdot_v \alpha + (beta-l\ l) \cdot_v \beta$

lemma *psi-l-dim*:
 $psi-l\ l \in carrier-vec\ N$
<proof>

lemma *inner-psi-l*:
 $inner-prod\ (psi-l\ l)\ (psi-l\ l) = 1$
<proof>

abbreviation *proj* :: *complex vec* \Rightarrow *complex mat* **where**
 $proj\ v \equiv outer-prod\ v\ v$

definition *psi'-l* **where**
 $psi'-l\ l = (alpha-l\ l) \cdot_v \alpha - (beta-l\ l) \cdot_v \beta$

lemma *psi'-l-dim*:
 $psi'-l\ l \in carrier-vec\ N$
<proof>

definition *proj-psi'-l* **where**
 $proj-psi'-l\ l = proj\ (psi'-l\ l)$

lemma *proj-psi'-dim*:
 $proj-psi'-l\ l \in carrier-mat\ N\ N$
<proof>

lemma *psi-inner-psi'-l*:

$$\text{inner-prod } \psi \text{ (psi'-l l)} = (\text{alpha-l l} * \text{ccos } (\vartheta / 2) - \text{beta-l l} * \text{csin } (\vartheta / 2))$$

<proof>

lemma *double-ccos-square*:

$$2 * \text{ccos } (a::\text{real}) * \text{ccos } a = \text{ccos } (2 * a) + 1$$

<proof>

lemma *double-csin-square*:

$$2 * \text{csin } (a::\text{real}) * \text{csin } a = 1 - \text{ccos } (2 * a)$$

<proof>

lemma *csin-double*:

$$2 * \text{csin } (a::\text{real}) * \text{ccos } a = \text{csin}(2 * a)$$

<proof>

lemma *ccos-add*:

$$\text{ccos } (x + y) = \text{ccos } x * \text{ccos } y - \text{csin } x * \text{csin } y$$

<proof>

lemma *alpha-l-Suc-l-derive*:

$$2 * (\text{alpha-l l} * \text{ccos } (\vartheta / 2) - \text{beta-l l} * \text{csin } (\vartheta / 2)) * \text{ccos } (\vartheta / 2) - \text{alpha-l l}$$
$$= \text{alpha-l } (l + 1)$$

(**is** ?lhs = ?rhs)

<proof>

lemma *csin-add*:

$$\text{csin } (x + y) = \text{ccos } x * \text{csin } y + \text{csin } x * \text{ccos } y$$

<proof>

lemma *beta-l-Suc-l-derive*:

$$2 * (\text{alpha-l l} * \text{ccos } (\vartheta / 2) - (\text{beta-l l} * \text{csin } (\vartheta / 2))) * \text{csin } (\vartheta / 2) + \text{beta-l l}$$
$$= \text{beta-l } (l + 1)$$

(**is** ?lhs = ?rhs)

<proof>

lemma *psi-l-Suc-l-derive*:

$$2 * (\text{alpha-l l} * \text{ccos } (\vartheta / 2) - \text{beta-l l} * \text{csin } (\vartheta / 2)) \cdot_v \psi - \text{psi'-l l} = \text{psi-l } (l$$
$$+ 1)$$

(**is** ?lhs = ?rhs)

<proof>

7.2 Grover operator

Oracle O

definition *proj-O* :: complex mat **where**

$$\text{proj-O} = \text{mat } N N \ (\lambda(i, j). \text{if } i = j \text{ then (if } f \ i \text{ then } 1 \text{ else } 0) \text{ else } 0)$$

lemma *proj-O-dim*:

proj-O \in *carrier-mat* $N N$
(proof)

lemma *proj-O-mult-alpha*:
proj-O $*_v$ $\alpha =$ *zero-vec* N
(proof)

lemma *proj-O-mult-beta*:
proj-O $*_v$ $\beta = \beta$
(proof)

definition *mat-O* :: *complex mat where*
mat-O = *mat* $N N$ ($\lambda(i,j)$. if $i = j$ then (if f i then -1 else 1) else 0)

lemma *mat-O-dim*:
mat-O \in *carrier-mat* $N N$
(proof)

lemma *mat-O-mult-alpha*:
mat-O $*_v$ $\alpha = \alpha$
(proof)

lemma *mat-O-mult-beta*:
mat-O $*_v$ $\beta = - \beta$
(proof)

lemma *hermitian-mat-O*:
hermitian *mat-O*
(proof)

lemma *unitary-mat-O*:
unitary *mat-O*
(proof)

definition *mat-Ph* :: *complex mat where*
mat-Ph = *mat* $N N$ ($\lambda(i,j)$. if $i = j$ then if $i = 0$ then 1 else -1 else 0)

lemma *hermitian-mat-Ph*:
hermitian *mat-Ph*
(proof)

lemma *unitary-mat-Ph*:
unitary *mat-Ph*
(proof)

definition *mat-G'* :: *complex mat where*
mat-G' = *mat* $N N$ ($\lambda(i,j)$. if $i = j$ then $2 / N - 1$ else $2 / N$)

Geometrically, the Grover operator G is a rotation

definition *mat-G* :: *complex mat* **where**
mat-G = *mat-G'* * *mat-O*

end

7.3 State of Grover's algorithm

The dimensions are [2, 2, ..., 2, n]. We work with a very special case as in the paper

locale *grover-state-sig* = *grover-state* + *state-sig* +
fixes *R* :: *nat*
fixes *K* :: *nat*
assumes *dims-def*: *dims* = *replicate* *n* 2 @ [*K*]
assumes *R*: $R = pi / (2 * \vartheta) - 1 / 2$
assumes *K*: $K > R$

begin

lemma *K-gt-0*:

$K > 0$

<proof>

Bits *q0* to *q_(n-1)*

definition *vars1* :: *nat set* **where**

vars1 = {0 ..< *n*}

Bit *r*

definition *vars2* :: *nat set* **where**

vars2 = {*n*}

lemma *length-dims*:

length *dims* = *n* + 1

<proof>

lemma *dims-nth-lt-n*:

$l < n \implies nth\ dims\ l = 2$

<proof>

lemma *nths-Suc-n-dims*:

nths *dims* {0..<(Suc *n*)} = *dims*

<proof>

interpretation *ps2-P*: *partial-state2* *dims* *vars1* *vars2*

<proof>

interpretation *ps-P*: *partial-state* *ps2-P.dims0* *ps2-P.vars1'* *<proof>*

abbreviation *tensor-P* **where**

tensor-P *A* *B* \equiv *ps2-P.ptensor-mat* *A* *B*

lemma *tensor-P-dim*:
tensor-P A B ∈ carrier-mat d d
⟨proof⟩

lemma *dims-nths-le-n*:
assumes $l \leq n$
shows *nths dims* {0..*l*} = *replicate l 2*
⟨proof⟩

lemma *dims-nths-one-lt-n*:
assumes $l < n$
shows *nths dims* {*l*} = [*2*]
⟨proof⟩

lemma *dims-vars1*:
nths dims vars1 = replicate n 2
⟨proof⟩

lemma *nths-rep-2-n*:
nths (replicate n 2) {n} = []
⟨proof⟩

lemma *dims-vars2*:
nths dims vars2 = [K]
⟨proof⟩

lemma *d-vars1*:
prod-list (nths dims vars1) = N
⟨proof⟩

lemma *ps2-P-dims0*:
ps2-P.dims0 = dims
⟨proof⟩

lemma *ps2-P-vars1'*:
ps2-P.vars1' = vars1
⟨proof⟩

lemma *ps2-P-d0*:
ps2-P.d0 = d
⟨proof⟩

lemma *ps2-P-d1*:
ps2-P.d1 = N
⟨proof⟩

lemma *ps2-P-d2*:
ps2-P.d2 = K

<proof>

lemma *ps-P-d*:

ps-P.d = d

<proof>

lemma *ps-P-d1*:

ps-P.d1 = N

<proof>

lemma *ps-P-d2*:

ps-P.d2 = K

<proof>

lemma *nths-uminus-vars1*:

nths dims (- vars1) = nths dims vars2

<proof>

lemma *tensor-P-mult*:

assumes *m1 ∈ carrier-mat (2[^]n) (2[^]n)*

and *m2 ∈ carrier-mat (2[^]n) (2[^]n)*

and *m3 ∈ carrier-mat K K*

and *m4 ∈ carrier-mat K K*

shows *(tensor-P m1 m3) * (tensor-P m2 m4) = tensor-P (m1 * m2) (m3 * m4)*

<proof>

lemma *mat-ext-vars1*:

shows *mat-extension dims vars1 A = tensor-P A (1_m K)*

<proof>

lemma *Utrans-P-is-tensor-P1*:

Utrans-P vars1 A = Utrans (tensor-P A (1_m K))

<proof>

lemma *nths-dims-uminus-vars2*:

nths dims (-vars2) = nths dims vars1

<proof>

lemma *mat-ext-vars2*:

assumes *A ∈ carrier-mat K K*

shows *mat-extension dims vars2 A = tensor-P (1_m N) A*

<proof>

lemma *Utrans-P-is-tensor-P2*:

assumes *A ∈ carrier-mat K K*

shows *Utrans-P vars2 A = Utrans (tensor-P (1_m N) A)*

<proof>

7.4 Grover's algorithm

Apply hadamard operator to first n variables

definition *hadamard-on-i* :: *nat* \Rightarrow *complex mat* **where**
hadamard-on-i *i* = *pmat-extension dims {i} (vars1 - {i}) hadamard*
declare *hadamard-on-i-def* [*simp*]

fun *hadamard-n* :: *nat* \Rightarrow *com* **where**
hadamard-n 0 = *SKIP*
| *hadamard-n* (*Suc i*) = *hadamard-n i* ;; *Utrans (tensor-P (hadamard-on-i i) (1_m K))*

Body of the loop

definition *D* :: *com* **where**
D = *Utrans-P vars1 mat-O* ;;
hadamard-n n ;;
Utrans-P vars1 mat-Ph ;;
hadamard-n n ;;
Utrans-P vars2 (mat-incr K)

lemma *unitary-ex-mat-O*:
unitary (tensor-P mat-O (1_m K))
 \langle *proof* \rangle

lemma *unitary-ex-mat-Ph*:
unitary (tensor-P mat-Ph (1_m K))
 \langle *proof* \rangle

lemma *unitary-hadamard-on-i*:
assumes $k < n$
shows *unitary (hadamard-on-i k)*
 \langle *proof* \rangle

lemma *unitary-exhadamard-on-i*:
assumes $k < n$
shows *unitary (tensor-P (hadamard-on-i k) (1_m K))*
 \langle *proof* \rangle

lemma *hadamard-on-i-dim*:
assumes $k < n$
shows *hadamard-on-i k* \in *carrier-mat N N*
 \langle *proof* \rangle

lemma *well-com-hadamard-k*:
 $k \leq n \implies$ *well-com (hadamard-n k)*
 \langle *proof* \rangle

lemma *well-com-hadamard-n*:
well-com (hadamard-n n)

<proof>

lemma *well-com-mat-O:*
well-com (Utrans-P vars1 mat-O)
<proof>

lemma *well-com-mat-Ph:*
well-com (Utrans-P vars1 mat-Ph)
<proof>

lemma *unitary-exmat-incr:*
unitary (tensor-P (1_m N) (mat-incr K))
<proof>

lemma *well-com-mat-incr:*
well-com (Utrans-P vars2 (mat-incr K))
<proof>

lemma *well-com-D: well-com D*
<proof>

Test at while loop

definition *M0 :: complex mat where*
M0 = mat K K ($\lambda(i,j).$ if $i = j \wedge i \geq R$ then 1 else 0)

lemma *hermitian-M0:*
hermitian M0
<proof>

lemma *M0-dim:*
M0 \in carrier-mat K K
<proof>

lemma *M0-mult-M0:*
*M0 * M0 = M0*
<proof>

definition *M1 :: complex mat where*
M1 = mat K K ($\lambda(i,j).$ if $i = j \wedge i < R$ then 1 else 0)

lemma *M1-dim:*
M1 \in carrier-mat K K
<proof>

lemma *hermitian-M1:*
hermitian M1
<proof>

lemma *M1-mult-M1:*

$M1 * M1 = M1$
 ⟨proof⟩

lemma *M1-add-M0*:
 $M1 + M0 = 1_m K$
 ⟨proof⟩

Test at the end

definition *testN* :: $nat \Rightarrow complex\ mat$ **where**
 $testN\ k = mat\ N\ N\ (\lambda(i,j).\ if\ i = k \wedge j = k\ then\ 1\ else\ 0)$

lemma *hermitian-testN*:
 $hermitian\ (testN\ k)$
 ⟨proof⟩

lemma *testN-mult-testN*:
 $testN\ k * testN\ k = testN\ k$
 ⟨proof⟩

lemma *testN-dim*:
 $testN\ k \in carrier\ mat\ N\ N$
 ⟨proof⟩

definition *test-fst-k* :: $nat \Rightarrow complex\ mat$ **where**
 $test\ fst\ k\ k = mat\ N\ N\ (\lambda(i,j).\ if\ (i = j \wedge i < k)\ then\ 1\ else\ 0)$

lemma *sum-test-k*:
assumes $m \leq N$
shows $matrix\ sum\ N\ (\lambda k.\ testN\ k)\ m = test\ fst\ k\ m$
 ⟨proof⟩

lemma *test-fst-kN*:
 $test\ fst\ k\ N = 1_m\ N$
 ⟨proof⟩

lemma *matrix-sum-tensor-P1*:
 $(\bigwedge k.\ k < m \implies g\ k \in carrier\ mat\ N\ N) \implies (A \in carrier\ mat\ K\ K) \implies$
 $matrix\ sum\ d\ (\lambda k.\ tensor\ P\ (g\ k)\ A)\ m = tensor\ P\ (matrix\ sum\ N\ g\ m)\ A$
 ⟨proof⟩

Grover's algorithm. Assume we start in the zero state

definition *Grover* :: com **where**
 $Grover = hadamard\ n\ n\ ;;$
 $While\ P\ vars2\ M0\ M1\ D\ ;;$
 $Measure\ P\ vars1\ N\ testN\ (replicate\ N\ SKIP)$

lemma *well-com-if*:
 $well\ com\ (Measure\ P\ vars1\ N\ testN\ (replicate\ N\ SKIP))$
 ⟨proof⟩

lemma *well-com-while*:
well-com (*While-P vars2 M0 M1 D*)
 ⟨*proof*⟩

lemma *well-com-Grover*:
well-com Grover
 ⟨*proof*⟩

7.5 Correctness

Pre-condition: assume in the zero state

definition *ket-pre* :: *complex vec* **where**
ket-pre = *Matrix.vec N* ($\lambda k. \text{if } k = 0 \text{ then } 1 \text{ else } 0$)

lemma *ket-pre-dim*:
ket-pre \in *carrier-vec N* ⟨*proof*⟩

definition *pre* :: *complex mat* **where**
pre = *proj ket-pre*

lemma *pre-dim*:
pre \in *carrier-mat N N*
 ⟨*proof*⟩

lemma *norm-pre*:
inner-prod ket-pre ket-pre = 1
 ⟨*proof*⟩

lemma *pre-trace*:
trace pre = 1
 ⟨*proof*⟩

lemma *positive-pre*:
positive pre
 ⟨*proof*⟩

lemma *pre-le-one*:
pre $\leq_L 1_m N$
 ⟨*proof*⟩

Post-condition: should be in a state i with $f\ i = 1$

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

lemma *post-dim*:
post \in *carrier-mat N N*
 ⟨*proof*⟩

lemma *hermitian-post*:

hermitian post

<proof>

Hoare triples of initialization

definition *ket-zero* :: *complex vec* **where**

ket-zero = *Matrix.vec* 2 ($\lambda k. \text{if } k = 0 \text{ then } 1 \text{ else } 0$)

lemma *ket-zero-dim*:

ket-zero \in *carrier-vec* 2 *<proof>*

definition *proj-zero* **where**

proj-zero = *proj ket-zero*

definition *ket-one* **where**

ket-one = *Matrix.vec* 2 ($\lambda k. \text{if } k = 1 \text{ then } 1 \text{ else } 0$)

definition *proj-one* **where**

proj-one = *proj ket-one*

definition *ket-plus* **where**

ket-plus = *Matrix.vec* 2 ($\lambda k. 1 / \text{csqrt } 2$)

lemma *ket-plus-dim*:

ket-plus \in *carrier-vec* 2 *<proof>*

lemma *ket-plus-eval* [*simp*]:

$i < 2 \implies \text{ket-plus } \$ i = 1 / \text{csqrt } 2$

<proof>

lemma *csqrt-2-sq* [*simp*]:

complex-of-real (*sqrt* 2) * *complex-of-real* (*sqrt* 2) = 2

<proof>

lemma *ket-plus-tensor-n*:

partial-state.tensor-vec [2, 2] {0} *ket-plus ket-plus* = *Matrix.vec* 4 ($\lambda k. 1 / 2$)

<proof>

definition *proj-plus* **where**

proj-plus = *proj ket-plus*

lemma *hadamard-on-zero*:

hadamard *_v *ket-zero* = *ket-plus*

<proof>

fun *exH-k* :: *nat* \Rightarrow *complex mat* **where**

exH-k 0 = *hadamard-on-i* 0

| *exH-k* (*Suc* k) = *exH-k* k * *hadamard-on-i* (*Suc* k)

fun $H-k :: nat \Rightarrow complex\ mat$ **where**

$H-k\ 0 = hadamard$

| $H-k\ (Suc\ k) = ptensor-mat\ dims\ \{0..<Suc\ k\}\ \{Suc\ k\}\ (H-k\ k)\ hadamard$

lemma $H-k-dim$:

$k < n \implies H-k\ k \in carrier-mat\ (2^{Suc\ k})\ (2^{Suc\ k})$

$\langle proof \rangle$

lemma $exH-k-eq-H-k$:

$k < n \implies exH-k\ k = pmat-extension\ dims\ \{0..<(Suc\ k)\}\ \{(Suc\ k)..<n\}\ (H-k\ k)$

$\langle proof \rangle$

lemma $mult-exH-k-left$:

assumes $Suc\ k < n$

shows $hadamard-on-i\ (Suc\ k) * exH-k\ k = exH-k\ (Suc\ k)$

$\langle proof \rangle$

lemma $exH-eq-H$:

$exH-k\ (n - 1) = H-k\ (n - 1)$

$\langle proof \rangle$

fun $ket-zero-k :: nat \Rightarrow complex\ vec$ **where**

$ket-zero-k\ 0 = ket-zero$

| $ket-zero-k\ (Suc\ k) = ptensor-vec\ dims\ \{0..<(Suc\ k)\}\ \{Suc\ k\}\ (ket-zero-k\ k)$
 $ket-zero$

lemma $ket-zero-k-dim$:

assumes $k < n$

shows $ket-zero-k\ k \in carrier-vec\ (2^{Suc\ k})$

$\langle proof \rangle$

fun $ket-plus-k$ **where**

$ket-plus-k\ 0 = ket-plus$

| $ket-plus-k\ (Suc\ k) = ptensor-vec\ dims\ \{0..<(Suc\ k)\}\ \{Suc\ k\}\ (ket-plus-k\ k)$
 $ket-plus$

lemma $ket-plus-k-dim$:

assumes $k < n$

shows $ket-plus-k\ k \in carrier-vec\ (2^{Suc\ k})$

$\langle proof \rangle$

lemma $H-k-ket-zero-k$:

$k < n \implies (H-k\ k) *_v\ (ket-zero-k\ k) = (ket-plus-k\ k)$

$\langle proof \rangle$

lemma $encode1-replicate-2$:

$partial-state.encode1\ (replicate\ (Suc\ k)\ 2)\ \{0..<k\}\ i = i\ mod\ (2 \wedge k)$

<proof>

lemma *encode2-replicate-2*:

assumes $i < 2^{\wedge} \text{Suc } k$

shows $\text{partial-state.encode2 } (\text{replicate } (\text{Suc } k) 2) \{0..<k\} i = i \text{ div } (2^{\wedge} k)$

<proof>

lemma *ket-zero-k-decode*:

$k < n \implies \text{ket-zero-k } k = \text{Matrix.vec } (2^{\wedge}(\text{Suc } k)) (\lambda k. \text{if } k = 0 \text{ then } 1 \text{ else } 0)$

<proof>

lemma *ket-plus-k-decode*:

$k < n \implies \text{ket-plus-k } k = \text{Matrix.vec } (2^{\wedge}(\text{Suc } k)) (\lambda l. 1 / \text{csqrt } (2^{\wedge}(\text{Suc } k)))$

<proof>

lemma *exH-k-mult-pre-is-psi*:

$\text{exH-k } (n - 1) *_v \text{ket-pre} = \psi$

<proof>

definition *ket-k* :: $\text{nat} \implies \text{complex vec}$ **where**

$\text{ket-k } x = \text{Matrix.vec } K (\lambda k. \text{if } k = x \text{ then } 1 \text{ else } 0)$

lemma *ket-k-dim*:

$\text{ket-k } k \in \text{carrier-vec } K$

<proof>

lemma *mat-incr-mult-ket-k*:

$k < K \implies (\text{mat-incr } K) *_v (\text{ket-k } k) = (\text{ket-k } ((k + 1) \text{ mod } K))$

<proof>

definition *proj-k* **where**

$\text{proj-k } x = \text{proj } (\text{ket-k } x)$

lemma *proj-k-dim*:

$\text{proj-k } k \in \text{carrier-mat } K K$

<proof>

lemma *norm-ket-k-lt-K*:

$k < K \implies \text{inner-prod } (\text{ket-k } k) (\text{ket-k } k) = 1$

<proof>

lemma *norm-ket-k-ge-K*:

$k \geq K \implies \text{inner-prod } (\text{ket-k } k) (\text{ket-k } k) = 0$

<proof>

lemma *norm-ket-k*:

$\text{inner-prod } (\text{ket-k } k) (\text{ket-k } k) \leq 1$

<proof>

lemma *proj-k-mat*:

assumes $k < K$

shows $\text{proj-k } k = \text{mat } K \ K \ (\lambda(i, j). \text{if } (i = j \wedge i = k) \text{ then } 1 \text{ else } 0)$

<proof>

lemma *positive-proj-k*:

positive ($\text{proj-k } k$)

<proof>

lemma *proj-k-le-one*:

$(\text{proj-k } k) \leq_L 1_m \ K$

<proof>

definition *proj-psi* **where**

$\text{proj-psi} = \text{proj } \psi$

lemma *proj-psi-dim*:

$\text{proj-psi} \in \text{carrier-mat } N \ N$

<proof>

lemma *norm-psi*:

inner-prod $\psi \ \psi = 1$

<proof>

lemma *proj-psi-mat*:

$\text{proj-psi} = \text{mat } N \ N \ (\lambda k. 1 / N)$

<proof>

lemma *hermitian-proj-psi*:

hermitian proj-psi

<proof>

lemma *hermitian-exproj-psi*:

hermitian ($\text{tensor-P } \text{proj-psi} \ (1_m \ K)$)

<proof>

lemma *proj-psi-is-projection*:

$\text{proj-psi} * \text{proj-psi} = \text{proj-psi}$

<proof>

lemma *proj-psi-trace*:

trace (proj-psi) = 1

<proof>

lemma *positive-proj-psi*:

positive (proj-psi)

<proof>

lemma *proj-psi-le-one*:

$(\text{proj-psi}) \leq_L 1_m N$
 $\langle \text{proof} \rangle$

lemma *hermitian-hadamard-on-k*:
 assumes $k < n$
 shows *hermitian (hadamard-on-i k)*
 $\langle \text{proof} \rangle$

lemma *hermitian-H-k*:
 $k < n \implies \text{hermitian } (H-k \ k)$
 $\langle \text{proof} \rangle$

lemma *unitary-H-k*:
 $k < n \implies \text{unitary } (H-k \ k)$
 $\langle \text{proof} \rangle$

lemma *exH-k-dim*:
 shows $k < n \implies \text{exH-k } k \in \text{carrier-mat } N \ N$
 $\langle \text{proof} \rangle$

lemma *exH-n-dim*:
 shows $\text{exH-k } (n - 1) \in \text{carrier-mat } N \ N$
 $\langle \text{proof} \rangle$

lemma *unitary-exH-k*:
 shows $k < n \implies \text{unitary } (\text{exH-k } k)$
 $\langle \text{proof} \rangle$

lemma *hermitian-exH-n*:
 hermitian (exH-k (n - 1))
 $\langle \text{proof} \rangle$

lemma *exH-k-mult-psi-is-pre*:
 $\text{exH-k } (n - 1) *_v \psi = \text{ket-pre}$
 $\langle \text{proof} \rangle$

fun *exexH-k* :: $\text{nat} \Rightarrow \text{complex mat}$ **where**
 $\text{exexH-k } k = \text{tensor-P } (\text{exH-k } k) (1_m \ K)$

lemma *unitary-exexH-k*:
 $k < n \implies \text{unitary } (\text{exexH-k } k)$
 $\langle \text{proof} \rangle$

lemma *exexH-k-dim*:
 $k < n \implies \text{exexH-k } k \in \text{carrier-mat } d \ d$
 $\langle \text{proof} \rangle$

lemma *hoare-seq-utrans*:
 fixes $P :: \text{complex mat}$

assumes *unitary U1 and unitary U2 and is-quantum-predicate P*
and $dU1: U1 \in \text{carrier-mat } d \ d$ **and** $dU2: U2 \in \text{carrier-mat } d \ d$
shows
 \vdash_p
 $\{\text{adjoint } (U2 * U1) * P * (U2 * U1)\}$
 $\text{Utrans } U1;; \text{Utrans } U2$
 $\{P\}$
 $\langle \text{proof} \rangle$

lemma *qp-close-after-exexH-k:*
fixes $P :: \text{complex mat}$
assumes *is-quantum-predicate P*
shows $k < n \implies \text{is-quantum-predicate } (\text{adjoint } (\text{exexH-k } k) * P * \text{exexH-k } k)$
 $\langle \text{proof} \rangle$

lemma *hoare-hadamard-n:*
fixes $P :: \text{complex mat}$
shows $\text{is-quantum-predicate } P \implies k < n \implies$
 \vdash_p
 $\{\text{adjoint } (\text{exexH-k } k) * P * \text{exexH-k } k\}$
 $\text{hadamard-n } (\text{Suc } k)$
 $\{P\}$
 $\langle \text{proof} \rangle$

lemma *qp-pre:*
 $\text{is-quantum-predicate } (\text{tensor-P pre } (\text{proj-k } 0))$
 $\langle \text{proof} \rangle$

lemma *qp-init-post:*
 $\text{is-quantum-predicate } (\text{tensor-P proj-psi } (\text{proj-k } 0))$
 $\langle \text{proof} \rangle$

lemma *tensor-P-adjoint-left-right:*
assumes $m1 \in \text{carrier-mat } N \ N$ **and** $m2 \in \text{carrier-mat } K \ K$ **and** $m3 \in \text{carrier-mat } N \ N$ **and** $m4 \in \text{carrier-mat } K \ K$
shows $\text{adjoint } (\text{tensor-P } m1 \ m2) * \text{tensor-P } m3 \ m4 * \text{tensor-P } m1 \ m2 = \text{tensor-P } (\text{adjoint } m1 * m3 * m1) (\text{adjoint } m2 * m4 * m2)$
 $\langle \text{proof} \rangle$

abbreviation *exH-n where*
 $\text{exH-n} \equiv \text{exH-k } (n - 1)$

lemma *hoare-triple-init:*
 \vdash_p
 $\{\text{tensor-P pre } (\text{proj-k } 0)\}$
 $\text{hadamard-n } n$
 $\{\text{tensor-P proj-psi } (\text{proj-k } 0)\}$
 $\langle \text{proof} \rangle$

Hoare triples of while loop

definition *proj-psi-l* **where**

$$\text{proj-psi-l } l = \text{proj } (\text{psi-l } l)$$

lemma *positive-psi-l*:

$$k < K \implies \text{positive } (\text{proj-psi-l } k)$$

<proof>

lemma *hermitian-proj-psi-l*:

$$k < K \implies \text{hermitian } (\text{proj-psi-l } k)$$

<proof>

definition *P'* **where**

$$P' = \text{tensor-P } (\text{proj-psi-l } R) (\text{proj-k } R)$$

lemma *proj-psi-l-dim*:

$$\text{proj-psi-l } l \in \text{carrier-mat } N \ N$$

<proof>

definition *Q* :: *complex mat* **where**

$$Q = \text{matrix-sum } d \ (\lambda l. \text{tensor-P } (\text{proj-psi-l } l) (\text{proj-k } l)) \ R$$

lemma *psi-l-le-id*:

$$\text{shows } \text{proj-psi-l } l \leq_L 1_m \ N$$

<proof>

lemma *positive-proj-psi-l*:

$$\text{shows } \text{positive } (\text{proj-psi-l } l)$$

<proof>

definition *proj-fst-k* :: *nat* \Rightarrow *complex mat* **where**

$$\text{proj-fst-k } k = \text{mat } K \ K \ (\lambda(i, j). \text{if } (i = j \wedge i < k) \text{ then } 1 \text{ else } 0)$$

lemma *hermitian-proj-fst-k*:

$$\text{adjoint } (\text{proj-fst-k } k) = \text{proj-fst-k } k$$

<proof>

lemma *proj-fst-k-is-projection*:

$$\text{proj-fst-k } k * \text{proj-fst-k } k = \text{proj-fst-k } k$$

<proof>

lemma *positive-proj-fst-k*:

$$\text{positive } (\text{proj-fst-k } k)$$

<proof>

lemma *proj-fst-k-le-one*:

$$\text{proj-fst-k } k \leq_L 1_m \ K$$

<proof>

lemma *sum-proj-k*:

assumes $m \leq K$
shows $\text{matrix-sum } K (\lambda k. \text{proj-k } k) m = \text{proj-fst-k } m$
 $\langle \text{proof} \rangle$

lemma $\text{proj-psi-proj-k-le-exproj-k}$:
shows $\text{tensor-P } (\text{proj-psi-l } k) (\text{proj-k } l) \leq_L \text{tensor-P } (1_m N) (\text{proj-k } l)$
 $\langle \text{proof} \rangle$

definition $Q1 :: \text{complex mat where}$
 $Q1 = \text{matrix-sum } d (\lambda l. \text{tensor-P } (\text{proj-psi'-l } l) (\text{proj-k } l)) R$

lemma $\text{tensor-P-left-right-partial1}$:
assumes $m1 \in \text{carrier-mat } N N$ **and** $m2 \in \text{carrier-mat } N N$ **and** $m3 \in \text{carrier-mat } K K$ **and** $m4 \in \text{carrier-mat } N N$
shows $\text{tensor-P } m1 (1_m K) * \text{tensor-P } m2 m3 * \text{tensor-P } m4 (1_m K) = \text{tensor-P } (m1 * m2 * m4) m3$
 $\langle \text{proof} \rangle$

lemma $\text{tensor-P-left-right-partial2}$:
assumes $m1 \in \text{carrier-mat } K K$ **and** $m2 \in \text{carrier-mat } K K$ **and** $m3 \in \text{carrier-mat } N N$ **and** $m4 \in \text{carrier-mat } K K$
shows $\text{tensor-P } (1_m N) m1 * \text{tensor-P } m3 m2 * \text{tensor-P } (1_m N) m4 = \text{tensor-P } m3 (m1 * m2 * m4)$
 $\langle \text{proof} \rangle$

lemma $\text{matrix-sum-mult-left-right}$:
fixes $A B :: \text{complex mat}$
assumes $dg: (\bigwedge k. k < l \implies g k \in \text{carrier-mat } m m)$
and $dA: A \in \text{carrier-mat } m m$ **and** $dB: B \in \text{carrier-mat } m m$
shows $\text{matrix-sum } m (\lambda k. A * g k * B) l = A * \text{matrix-sum } m g l * B$
 $\langle \text{proof} \rangle$

lemma mat-O-split :
 $\text{mat-O} = 1_m N - 2 \cdot_m \text{proj-O}$
 $\langle \text{proof} \rangle$

lemma mat-O-mult-psi'-l :
 $\text{mat-O} *_v (\text{psi'-l } l) = \text{psi-l } l$
 $\langle \text{proof} \rangle$

lemma mat-O-times-Q1 :
 $\text{adjoint } (\text{tensor-P } \text{mat-O } (1_m K)) * Q1 * (\text{tensor-P } \text{mat-O } (1_m K)) = Q$
 $\langle \text{proof} \rangle$

definition $Q2 \text{ where}$
 $Q2 = \text{matrix-sum } d (\lambda l. \text{tensor-P } (\text{proj-psi-l } (l + 1)) (\text{proj-k } l)) R$

lemma $Q2\text{-dim}$:
 $Q2 \in \text{carrier-mat } d d$

$\langle \text{proof} \rangle$

lemma *Q2-le-one*:

$Q2 \leq_L 1_m d$
 $\langle \text{proof} \rangle$

lemma *qp-Q2*:

is-quantum-predicate *Q2*
 $\langle \text{proof} \rangle$

lemma *pre-mat*:

$\text{pre} = \text{mat } N N (\lambda(i, j). \text{if } i = j \wedge i = 0 \text{ then } 1 \text{ else } 0)$
 $\langle \text{proof} \rangle$

lemma *mat-Ph-split*:

$\text{mat-Ph} = 2 \cdot_m \text{pre} - 1_m N$
 $\langle \text{proof} \rangle$

lemma *H-Ph-H*:

$\text{exexH-k } (n-1) * \text{tensor-P mat-Ph } (1_m K) * \text{exexH-k } (n-1) = 2 \cdot_m \text{tensor-P}$
 $\text{proj-psi } (1_m K) - 1_m d$
 $\langle \text{proof} \rangle$

lemma *hermitian-proj-psi-minus-1*:

hermitian $(2 \cdot_m \text{proj-psi} - 1_m N)$
 $\langle \text{proof} \rangle$

lemma *unitary-proj-psi-minus-1*:

unitary $(2 \cdot_m \text{proj-psi} - 1_m N)$
 $\langle \text{proof} \rangle$

lemma *proj-psi-minus-1-mult-psi'-l*:

$(2 \cdot_m \text{proj-psi} - 1_m N) *_v \text{psi}'-l l = \text{psi}-l (l+1)$
 $\langle \text{proof} \rangle$

lemma *proj-psi-minus-1-mult-psi-Suc-l*:

$(2 \cdot_m \text{proj-psi} - 1_m N) *_v \text{psi}-l (l+1) = \text{psi}'-l l$
 $\langle \text{proof} \rangle$

lemma *exproj-psi-minus-1-tensor*:

$(2 \cdot_m \text{tensor-P proj-psi } (1_m K)) - 1_m d = \text{tensor-P } (2 \cdot_m \text{proj-psi} - (1_m N))$
 $(1_m K)$
 $\langle \text{proof} \rangle$

lemma *unitary-exproj-psi-minus-1*:

unitary $(2 \cdot_m \text{tensor-P proj-psi } (1_m K) - 1_m d)$
 $\langle \text{proof} \rangle$

lemma *proj-psi-minus-1-Q2*:

adjoint $(2 \cdot_m \text{tensor-}P \text{proj-psi } (1_m K) - 1_m d) * Q2 * (2 \cdot_m \text{tensor-}P \text{proj-psi } (1_m K) - 1_m d) = Q1$
 ⟨proof⟩

lemma qp-Q1:
is-quantum-predicate Q1
 ⟨proof⟩

lemma qp-Q:
is-quantum-predicate Q
 ⟨proof⟩

lemma hoare-triple-D1:
 \vdash_p
 $\{Q\}$
Utrans-P vars1 mat-O
 $\{Q1\}$
 ⟨proof⟩

lemma hoare-triple-D2:
 \vdash_p
 $\{Q1\}$
hadamard-n n ;;
Utrans-P vars1 mat-Ph ;;
hadamard-n n
 $\{Q2\}$
 ⟨proof⟩

definition exM0 where
 $exM0 = \text{tensor-}P (1_m N) M0$

lemma M0-mult-ket-k-R:
 $M0 *_v \text{ket-k } R = \text{ket-k } R$
 ⟨proof⟩

lemma exP0-P':
 adjoint $exM0 * P' * exM0 = P'$
 ⟨proof⟩

definition exM1 where
 $exM1 = \text{tensor-}P (1_m N) M1$

lemma M1-mult-ket-k:
assumes $k < R$
shows $M1 *_v \text{ket-k } k = \text{ket-k } k$
 ⟨proof⟩

lemma exP1-Q:
 adjoint $exM1 * Q * exM1 = Q$

$\langle proof \rangle$

lemma *qp-P'*:

is-quantum-predicate P'

$\langle proof \rangle$

lemma *P'-add-Q*:

$P' + Q = \text{matrix-sum } d (\lambda l. \text{tensor-P } (\text{proj-psi-l } l) (\text{proj-k } l)) (R + 1)$

$\langle proof \rangle$

lemma *positive-Qk*:

positive (tensor-P (proj-psi-l l) (proj-k l))

$\langle proof \rangle$

lemma *P'-Q-dim*:

$P' + Q \in \text{carrier-mat } d \ d$

$\langle proof \rangle$

lemma *P'-add-Q-le-one*:

$P' + Q \leq_L 1_m \ d$

$\langle proof \rangle$

lemma *qp-P'-Q*:

is-quantum-predicate (P' + Q)

$\langle proof \rangle$

lemma *Q2-leq-lemma*:

$\text{tensor-P } (1_m \ N) (\text{mat-incr } K) * Q2 * \text{adjoint } (\text{tensor-P } (1_m \ N) (\text{mat-incr } K))$
 $\leq_L P' + Q$

$\langle proof \rangle$

lemma *Q2-leq*:

$Q2 \leq_L \text{adjoint } (\text{tensor-P } (1_m \ N) (\text{mat-incr } K)) * (P' + Q) * \text{tensor-P } (1_m \ N)$
 $(\text{mat-incr } K)$

$\langle proof \rangle$

lemma *hoare-triple-D3*:

\vdash_p

$\{Q2\}$

$U\text{trans-P vars2 } (\text{mat-incr } K)$

$\{\text{adjoint } \text{exM0} * P' * \text{exM0} + \text{adjoint } \text{exM1} * Q * \text{exM1}\}$

$\langle proof \rangle$

lemma *qp-D3-post*:

*is-quantum-predicate (adjoint exM0 * P' * exM0 + adjoint exM1 * Q * exM1)*

$\langle proof \rangle$

lemma *hoare-triple-D*:

\vdash_p

$\{Q\}$
 D
 $\{adjoint\ exM0 * P' * exM0 + adjoint\ exM1 * Q * exM1\}$
 $\langle proof \rangle$

lemma *psi-is-psi-l0*:

$\psi = psi-l\ 0$
 $\langle proof \rangle$

lemma *proj-psi-is-proj-psi-l0*:

$proj-psi = proj-psi-l\ 0$
 $\langle proof \rangle$

lemma *lowner-le-Q*:

$tensor-P\ proj-psi\ (proj-k\ 0) \leq_L\ adjoint\ exM0 * P' * exM0 + adjoint\ exM1 * Q$
 $*\ exM1$
 $\langle proof \rangle$

lemma *hoare-triple-while*:

\vdash_p
 $\{adjoint\ exM0 * P' * exM0 + adjoint\ exM1 * Q * exM1\}$
 $While-P\ vars2\ M0\ M1\ D$
 $\{P'\}$
 $\langle proof \rangle$

lemma *R-and-a-half- ϑ* :

$(R + 1/2) * \vartheta = pi / 2$
 $\langle proof \rangle$

lemma *psi-lR-is-beta*:

$psi-l\ R = \beta$
 $\langle proof \rangle$

lemma *post-mult-beta*:

$post *_{\nu} \beta = \beta$
 $\langle proof \rangle$

lemma *post-mult-post*:

$post * post = post$
 $\langle proof \rangle$

lemma *post-mult-proj-psi-lR*:

$post * proj-psi-l\ R = proj-psi-l\ R$
 $\langle proof \rangle$

lemma *proj-psi-lR-mult-post*:

$proj-psi-l\ R * post = proj-psi-l\ R$
 $\langle proof \rangle$

lemma *proj-psi-lR-mult-proj-psi-lR*:
 $proj\text{-}psi\text{-}l\ R * proj\text{-}psi\text{-}l\ R = proj\text{-}psi\text{-}l\ R$
 ⟨proof⟩

lemma *proj-psi-lR-le-post*:
 $proj\text{-}psi\text{-}l\ R \leq_L post$
 ⟨proof⟩

lemma *P'-le-post-R*:
 $P' \leq_L (tensor\text{-}P\ post\ (proj\text{-}k\ R))$
 ⟨proof⟩

lemma *positive-post*:
 $positive\ post$
 ⟨proof⟩

lemma *lowner-le-P'*:
 $P' \leq_L tensor\text{-}P\ post\ (1_m\ K)$
 ⟨proof⟩

lemma *post-mult-testNk*:
assumes $f\ k$
shows $post * (testN\ k) = testN\ k$
 ⟨proof⟩

lemma *post-mult-testNk-neg*:
assumes $\neg f\ k$
shows $post * testN\ k = 0_m\ N\ N$
 ⟨proof⟩

lemma *testN-post1*:
 $f\ k \implies adjoint\ (testN\ k) * post * testN\ k = testN\ k$
 ⟨proof⟩

lemma *testN-post2*:
 $\neg f\ k \implies adjoint\ (testN\ k) * post * testN\ k = 0_m\ N\ N$
 ⟨proof⟩

definition *post-fst-k* :: $nat \Rightarrow complex\ mat$ **where**
 $post\text{-}fst\text{-}k\ k = mat\ N\ N\ (\lambda(i, j). \text{if } (i = j \wedge f\ i \wedge i < k) \text{ then } 1 \text{ else } 0)$

lemma *post-fst-kN*:
 $post\text{-}fst\text{-}k\ N = post$
 ⟨proof⟩

lemma *post-fst-k-Suc*:
 $f\ i \implies post\text{-}fst\text{-}k\ (Suc\ i) = testN\ i + post\text{-}fst\text{-}k\ i$
 ⟨proof⟩

lemma *post-fst-k-Suc-neg*:

$\neg f\ i \implies \text{post-fst-k}\ (\text{Suc}\ i) = \text{post-fst-k}\ i$

$\langle \text{proof} \rangle$

lemma *testN-sum*:

$\text{matrix-sum}\ N\ (\lambda k. \text{adjoint}\ (\text{testN}\ k) * \text{post} * \text{testN}\ k)\ N = \text{post}$

$\langle \text{proof} \rangle$

lemma *tensor-P-testN-sum*:

$\text{matrix-sum}\ d\ (\lambda k. \text{adjoint}\ (\text{tensor-P}\ (\text{testN}\ k)\ (1_m\ K)) * \text{tensor-P}\ \text{post}\ (1_m\ K))$
 $* \text{tensor-P}\ (\text{testN}\ k)\ (1_m\ K)\ N =$

$\text{tensor-P}\ \text{post}\ (1_m\ K)$

$\langle \text{proof} \rangle$

lemma *post-le-one*:

$\text{post} \leq_L 1_m\ N$

$\langle \text{proof} \rangle$

lemma *qp-post*:

$\text{is-quantum-predicate}\ (\text{tensor-P}\ \text{post}\ (1_m\ K))$

$\langle \text{proof} \rangle$

lemma *hoare-triple-if*:

\vdash_p

$\{\text{tensor-P}\ \text{post}\ (1_m\ K)\}$

$\text{Measure-P}\ \text{vars1}\ N\ \text{testN}\ (\text{replicate}\ N\ \text{SKIP})$

$\{\text{tensor-P}\ \text{post}\ (1_m\ K)\}$

$\langle \text{proof} \rangle$

theorem *grover-partial-deduct*:

\vdash_p

$\{\text{tensor-P}\ \text{pre}\ (\text{proj-k}\ 0)\}$

Grover

$\{\text{tensor-P}\ \text{post}\ (1_m\ K)\}$

$\langle \text{proof} \rangle$

theorem *grover-partial-correct*:

\vDash_p

$\{\text{tensor-P}\ \text{pre}\ (\text{proj-k}\ 0)\}$

Grover

$\{\text{tensor-P}\ \text{post}\ (1_m\ K)\}$

$\langle \text{proof} \rangle$

end

end

References

- [1] M. Ying. Floyd–Hoare logic for quantum programs. *ACM Transactions on Programming Languages and Systems*, 33(6):19:1–19:49, 2011.