

Projective-Measurements

Mnacho

February 6, 2026

Contents

1 Preliminaries	1
1.1 Misc	1
1.2 Unifying notions between Isabelle Marries Dirac and QHL-Prover	4
1.3 Types to sets lemmata transfers	5
2 Linear algebra complements	7
2.1 Additional properties of matrices	7
2.2 Complements on complex matrices	10
2.3 Tensor product complements	20
2.4 Fixed carrier matrices locale	25
2.5 A locale for square matrices	27
2.6 Projectors	40
2.7 Rank 1 projection	43
3 Projective measurements	53
3.1 First definitions	54
3.2 Measurements with observables	59
3.2.1 On the diagonal elements of a matrix	59
3.2.2 Construction of measurement outcomes	64
3.2.3 Projective measurement associated with an observable	70
3.2.4 Properties on the spectrum of a Hermitian matrix	76
3.2.5 Similar properties for eigenvalues rather than spectrum indices	86
4 The CHSH inequality	91
4.1 Inequality statement	91
4.2 Properties of specific observables	110
4.2.1 Ket 0, Ket 1 and the corresponding projectors	110
4.2.2 The X and Z matrices and two of their combinations	112
4.2.3 No local hidden variable	118

theory *Linear-Algebra-Complements* **imports**

Isabelle-Marries-Dirac.Tensor
Isabelle-Marries-Dirac.More-Tensor
QHLProver.Gates
HOL-Types-To-Sets.Group-On-With
HOL-Probability.Probability

begin

hide-const(**open**) *S*

1 Preliminaries

1.1 Misc

lemma *mult-real-cpx*:

fixes *a::complex*

fixes *b::complex*

assumes *a ∈ Reals*

shows $a * (\text{Re } b) = \text{Re } (a * b)$ **using** *assms*

by (*metis Reals-cases complex.exhaust complex.sel(1) complex-of-real-mult-Complex of-real-mult*)

lemma *fct-bound*:

fixes *f::complex ⇒ real*

assumes $f(-1) + f 1 = 1$

and $0 \leq f 1$

and $0 \leq f (-1)$

shows $-1 \leq f 1 - f(-1) \wedge f 1 - f(-1) \leq 1$

proof

have $f 1 - f(-1) = 1 - f(-1) - f(-1)$ **using** *assms* **by** *simp*

also have $\dots \geq -1$ **using** *assms* **by** *simp*

finally show $-1 \leq f 1 - f(-1)$.

next

have $f(-1) - f 1 = 1 - f 1 - f 1$ **using** *assms* **by** *simp*

also have $\dots \geq -1$ **using** *assms* **by** *simp*

finally have $-1 \leq f(-1) - f 1$.

thus $f 1 - f (-1) \leq 1$ **by** *simp*

qed

lemma *fct-bound'*:

fixes *f::complex ⇒ real*

assumes $f(-1) + f 1 = 1$

and $0 \leq f 1$

and $0 \leq f (-1)$

shows $|f 1 - f(-1)| \leq 1$ **using** *assms* *fct-bound* **by** *auto*

lemma *pos-sum-1-le*:

assumes *finite I*
and $\forall i \in I. (0::real) \leq f i$
and $(\sum_{i \in I} f i) = 1$
and $j \in I$
shows $f j \leq 1$
proof (*rule ccontr*)
assume $\neg f j \leq 1$
hence $1 < f j$ **by** *simp*
hence $1 < (\sum_{i \in I} f i)$ **using** *assms* **by** (*metis* $\langle \neg f j \leq 1 \rangle$ *sum-nonneg-leq-bound*)

thus *False* **using** *assms* **by** *simp*
qed

lemma *last-subset*:
assumes $A \subseteq \{a, b\}$
and $a \neq b$
and $A \neq \{a, b\}$
and $A \neq \{\}$
and $A \neq \{a\}$
shows $A = \{b\}$ **using** *assms* **by** *blast*

lemma *disjoint-Un*:
assumes *disjoint-family-on A (insert x F)*
and $x \notin F$
shows $(A x) \cap (\bigcup_{a \in F} A a) = \{\}$
proof –
have $(A x) \cap (\bigcup_{a \in F} A a) = (\bigcup_{i \in F} (A x) \cap A i)$ **using** *Int-UN-distrib* **by** *simp*
also have $\dots = (\bigcup_{i \in F} \{\})$ **using** *assms disjoint-family-onD* **by** *fastforce*
also have $\dots = \{\}$ **using** *SUP-bot-conv(2)* **by** *simp*
finally show *?thesis* .
qed

lemma *sum-but-one*:
assumes $\forall j < (n::nat). j \neq i \longrightarrow f j = (0::'a::ring)$
and $i < n$
shows $(\sum_{j \in \{0 ..< n\}} f j * g j) = f i * g i$
proof –
have $sum (\lambda x. f x * g x) ((\{0 ..< n\} - \{i\}) \cup \{i\}) = sum (\lambda x. f x * g x) (\{0 ..< n\} - \{i\}) +$
 $sum (\lambda x. f x * g x) \{i\}$ **by** (*rule sum.union-disjoint, auto*)
also have $\dots = sum (\lambda x. f x * g x) \{i\}$ **using** *assms* **by** *auto*
also have $\dots = f i * g i$ **by** *simp*
finally have $sum (\lambda x. f x * g x) ((\{0 ..< n\} - \{i\}) \cup \{i\}) = f i * g i$.
moreover have $\{0 ..< n\} = (\{0 ..< n\} - \{i\}) \cup \{i\}$ **using** *assms* **by** *auto*
ultimately show *?thesis* **by** *simp*
qed

lemma *sum-2-elems*:

assumes $I = \{a, b\}$
and $a \neq b$
shows $(\sum a \in I. f a) = f a + f b$
proof –
have $(\sum a \in I. f a) = (\sum a \in (\text{insert } a \{b\}). f a)$ **using** *assms* **by** *simp*
also have $\dots = f a + (\sum a \in \{b\}. f a)$
proof (*rule sum.insert*)
show *finite* $\{b\}$ **by** *simp*
show $a \notin \{b\}$ **using** *assms* **by** *simp*
qed
also have $\dots = f a + f b$ **by** *simp*
finally show *?thesis* .
qed

lemma *sum-4-elems*:
shows $(\sum i < (4 :: \text{nat}). f i) = f 0 + f 1 + f 2 + f 3$
proof –
have $(\sum i < (4 :: \text{nat}). f i) = (\sum i < (3 :: \text{nat}). f i) + f 3$
by (*metis Suc-numeral semiring-norm(2) semiring-norm(8) sum.lessThan-Suc*)
moreover have $(\sum i < (3 :: \text{nat}). f i) = (\sum i < (2 :: \text{nat}). f i) + f 2$
by (*metis Suc-1 add-2-eq-Suc' nat-1-add-1 numeral-code(3) numerals(1) one-plus-numeral-commute sum.lessThan-Suc*)
moreover have $(\sum i < (2 :: \text{nat}). f i) = (\sum i < (1 :: \text{nat}). f i) + f 1$
by (*metis Suc-1 sum.lessThan-Suc*)
ultimately show *?thesis* **by** *simp*
qed

lemma *disj-family-sum*:
shows $\text{finite } I \implies \text{disjoint-family-on } A \ I \implies (\bigwedge i. i \in I \implies \text{finite } (A \ i)) \implies$
 $(\sum i \in (\bigcup n \in I. A \ n). f i) = (\sum n \in I. (\sum i \in A \ n. f i))$
proof (*induct rule:finite-induct*)
case *empty*
then show *?case* **by** *simp*
next
case (*insert x F*)
hence *disjoint-family-on* $A \ F$
by (*meson disjoint-family-on-mono subset-insertI*)
have $(\bigcup n \in (\text{insert } x \ F). A \ n) = A \ x \cup (\bigcup n \in F. A \ n)$ **using** *insert* **by** *simp*
hence $(\sum i \in (\bigcup n \in (\text{insert } x \ F). A \ n). f i) = (\sum i \in (A \ x \cup (\bigcup n \in F. A \ n)). f i)$ **by** *simp*
also have $\dots = (\sum i \in A \ x. f i) + (\sum i \in (\bigcup n \in F. A \ n). f i)$
by (*rule sum.union-disjoint, (simp add: insert disjoint-Un)+*)
also have $\dots = (\sum i \in A \ x. f i) + (\sum n \in F. \text{sum } f \ (A \ n))$ **using** $\langle \text{disjoint-family-on } A \ F \rangle$
by (*simp add: insert*)
also have $\dots = (\sum n \in (\text{insert } x \ F). \text{sum } f \ (A \ n))$ **using** *insert* **by** *simp*
finally show *?case* .
qed

```

lemma integrable-real-mult-right:
  fixes c::real
  assumes integrable M f
  shows integrable M ( $\lambda w. c * f w$ )
proof (cases c = 0)
  case True
  thus ?thesis by simp
next
  case False
  thus ?thesis using integrable-mult-right[of c] assms by simp
qed

```

1.2 Unifying notions between Isabelle Marries Dirac and QHLProver

```

lemma mult-conj-cmod-square:
  fixes z::complex
  shows z * conjugate z = (cmod z)2
proof -
  have z * conjugate z = (Re z)2 + (Im z)2 using complex-mult-conj by auto
  also have ... = (cmod z)2 unfolding cmod-def by simp
  finally show ?thesis .
qed

```

```

lemma vec-norm-sq-cpx-vec-length-sq:
  shows (vec-norm v)2 = (cpx-vec-length v)2
proof -
  have (vec-norm v)2 = inner-prod v v unfolding vec-norm-def using power2-csqr by blast
  also have ... = ( $\sum i < dim-vec v. (cmod (Matrix.vec-index v i))$ )2 unfolding Matrix.scalar-prod-def
  proof -
    have  $\bigwedge i. i < dim-vec v \implies Matrix.vec-index v i * conjugate (Matrix.vec-index v i) = (cmod (Matrix.vec-index v i))$ 2 using mult-conj-cmod-square by simp
    thus ( $\sum i = 0..<dim-vec (conjugate v). Matrix.vec-index v i * Matrix.vec-index (conjugate v) i$ ) = ( $\sum i < dim-vec v. (cmod (Matrix.vec-index v i))$ )2
      by (simp add: lessThan-atLeast0)
  qed
  finally show (vec-norm v)2 = (cpx-vec-length v)2 unfolding cpx-vec-length-def by (simp add: sum-nonneg)
qed

```

```

lemma vec-norm-eq-cpx-vec-length:
  shows vec-norm v = cpx-vec-length v using vec-norm-sq-cpx-vec-length-sq
  by (metis cpx-vec-length-inner-prod inner-prod-csqr power2-csqr vec-norm-def)

```

```

lemma cpx-vec-length-square:

```

shows $\|v\|^2 = (\sum_{i=0..<dim-vec\ v} (cmod\ (Matrix.vec-index\ v\ i))^2)$ **unfolding**
cpx-vec-length-def

by (*simp add: lessThan-atLeast0 sum-nonneg*)

lemma *state-qbit-norm-sq*:

assumes $v \in state-qbit\ n$

shows $(cpx-vec-length\ v)^2 = 1$

proof –

have $cpx-vec-length\ v = 1$ **using** *assms* **unfolding** *state-qbit-def* **by** *simp*

thus *?thesis* **by** *simp*

qed

lemma *dagger-adjoint*:

shows $dagger\ M = Complex-Matrix.adjoint\ M$ **unfolding** *dagger-def* *Complex-Matrix.adjoint-def*

by (*simp add: cong-mat*)

1.3 Types to sets lemmata transfers

context *ab-group-add-on-with* **begin**

context includes *lifting-syntax* **assumes** $ltd: \exists (Rep::'s \Rightarrow 'a) (Abs::'a \Rightarrow 's).$

type-definition *Rep* *Abs* *S* **begin**

interpretation *local-typedef-ab-group-add-on-with* *pls* *z* *mns* *um* *S* *TYPE('s)* **by**
unfold-locales *fact*

lemmas *lt-sum-union-disjoint = sum.union-disjoint*

[*var-simplified* *explicit-ab-group-add*,

unoverload-type 'c,

OF type.comm-monoid-add-axioms,

untransferred]

lemmas *lt-disj-family-sum = disj-family-sum*

[*var-simplified* *explicit-ab-group-add*,

unoverload-type 'd,

OF type.comm-monoid-add-axioms,

untransferred]

lemmas *lt-sum-reindex-cong = sum.reindex-cong*

[*var-simplified* *explicit-ab-group-add*,

unoverload-type 'd,

OF type.comm-monoid-add-axioms,

untransferred]

end

lemmas *sum-with-union-disjoint =*

lt-sum-union-disjoint

[*cancel-type-definition*,

OF carrier-ne,

simplified pred-fun-def, *simplified*]

lemmas *disj-family-sum-with* =
lt-disj-family-sum
 [*cancel-type-definition*,
OF carrier-ne,
simplified pred-fun-def, simplified]

lemmas *sum-with-reindex-cong* =
lt-sum-reindex-cong
 [*cancel-type-definition*,
OF carrier-ne,
simplified pred-fun-def, simplified]

end

lemma (*in comm-monoid-add-on-with*) *sum-with-cong'*:
 shows $finite\ I \implies (\bigwedge i. i \in I \implies A\ i = B\ i) \implies (\bigwedge i. i \in I \implies A\ i \in S) \implies$
 $(\bigwedge i. i \in I \implies B\ i \in S) \implies sum-with\ pls\ z\ A\ I = sum-with\ pls\ z\ B\ I$
proof (*induct rule: finite-induct*)
 case *empty*
 then show ?*case* by *simp*
next
 case (*insert x F*)
 have $sum-with\ pls\ z\ A\ (insert\ x\ F) = pls\ (A\ x)\ (sum-with\ pls\ z\ A\ F)$ **using** *insert*

sum-with-insert[of A] **by** (*simp add: image-subset-iff*)
 also have $\dots = pls\ (B\ x)\ (sum-with\ pls\ z\ B\ F)$ **using** *insert* **by** *simp*
 also have $\dots = sum-with\ pls\ z\ B\ (insert\ x\ F)$ **using** *insert sum-with-insert[of B]*
by (*simp add: image-subset-iff*)
 finally show ?*case* .
qed

2 Linear algebra complements

2.1 Additional properties of matrices

lemma *smult-one*:
 shows $(1::'a::monoid-mult) \cdot_m A = A$ **by** (*simp add: eq-matI*)

lemma *times-diag-index*:
 fixes $A::'a::comm-ring\ Matrix.mat$
 assumes $A \in carrier-mat\ n\ n$
 and $B \in carrier-mat\ n\ n$
 and *diagonal-mat B*
 and $j < n$
 and $i < n$
 shows $Matrix.vec-index\ (Matrix.row\ (A*B)\ j)\ i = diag-mat\ B\ !\ i * A\ \$\$ (j, i)$
proof –
 have $Matrix.vec-index\ (Matrix.row\ (A*B)\ j)\ i = (A*B)\ \$\$ (j, i)$

using *Matrix.row-def*[of $A*B$] *assms* **by** *simp*
also have ... = *Matrix.scalar-prod* (*Matrix.row* A j) (*Matrix.col* B i) **using** *assms*

times-mat-def[of A] **by** *simp*
also have ... = *Matrix.scalar-prod* (*Matrix.col* B i) (*Matrix.row* A j)
using *comm-scalar-prod*[of *Matrix.row* A j n] *assms* **by** *auto*
also have ... = (*Matrix.vec-index* (*Matrix.col* B i) i) * (*Matrix.vec-index* (*Matrix.row* A j) i)
unfolding *Matrix.scalar-prod-def*
proof (*rule sum-but-one*)
show $i < \text{dim-vec}$ (*Matrix.row* A j) **using** *assms* **by** *simp*
show $\forall ia < \text{dim-vec}$ (*Matrix.row* A j). $ia \neq i \longrightarrow \text{Matrix.vec-index}$ (*Matrix.col* B i) $ia = 0$ **using** *assms*
by (*metis carrier-matD(1) carrier-matD(2) diagonal-mat-def index-col index-row(2)*)
qed
also have ... = B $\$(i, i)$ * (*Matrix.vec-index* (*Matrix.row* A j) i) **using** *assms*
by *auto*
also have ... = *diag-mat* B ! i * (*Matrix.vec-index* (*Matrix.row* A j) i) **unfolding**
diag-mat-def
using *assms* **by** *simp*
also have ... = *diag-mat* B ! i * A $\$(j, i)$ **using** *assms* **by** *simp*
finally show *?thesis* .
qed

lemma *inner-prod-adjoint-comp*:

assumes ($U::'a::\text{conjugatable-field}$ *Matrix.mat*) \in *carrier-mat* n n
and ($V::'a::\text{conjugatable-field}$ *Matrix.mat*) \in *carrier-mat* n n
and $i < n$
and $j < n$
shows *Complex-Matrix.inner-prod* (*Matrix.col* V i) (*Matrix.col* U j) =
(*Complex-Matrix.adjoint* V) * U $\$(i, j)$
proof –
have *Complex-Matrix.inner-prod* (*Matrix.col* V i) (*Matrix.col* U j) =
Matrix.scalar-prod (*Matrix.col* U j) (*Matrix.row* (*Complex-Matrix.adjoint* V) i)
using *adjoint-row*[of i V] *assms* **by** *simp*
also have ... = *Matrix.scalar-prod* (*Matrix.row* (*Complex-Matrix.adjoint* V) i)
(*Matrix.col* U j)
by (*metis adjoint-row assms(1) assms(2) assms(3) carrier-matD(1) carrier-matD(2) Matrix.col-dim*
conjugate-vec-sprod-comm)
also have ... = ((*Complex-Matrix.adjoint* V) * U) $\$(i, j)$ **using** *assms*
by (*simp add:times-mat-def*)
finally show *?thesis* .
qed

lemma *mat-unit-vec-col*:

assumes ($A::'a::\text{conjugatable-field}$ *Matrix.mat*) \in *carrier-mat* n n

and $i < n$
shows $A *_v (\text{unit-vec } n \ i) = \text{Matrix.col } A \ i$
proof
show $\text{dim-vec } (A *_v \text{unit-vec } n \ i) = \text{dim-vec } (\text{Matrix.col } A \ i)$ **using** *assms* **by**
simp
show $\bigwedge j. j < \text{dim-vec } (\text{Matrix.col } A \ i) \implies \text{Matrix.vec-index } (A *_v \text{unit-vec } n \ i)$
 $j =$
 $\text{Matrix.vec-index } (\text{Matrix.col } A \ i) \ j$
proof –
fix j
assume $j < \text{dim-vec } (\text{Matrix.col } A \ i)$
hence $\text{Matrix.vec-index } (A *_v \text{unit-vec } n \ i) \ j =$
 $\text{Matrix.scalar-prod } (\text{Matrix.row } A \ j) (\text{unit-vec } n \ i)$ **unfolding** *mult-mat-vec-def*
by *simp*
also have $\dots = \text{Matrix.scalar-prod } (\text{unit-vec } n \ i) (\text{Matrix.row } A \ j)$ **using**
comm-scalar-prod
assms **by** *auto*
also have $\dots = (\text{Matrix.vec-index } (\text{unit-vec } n \ i) \ i) * (\text{Matrix.vec-index } (\text{Matrix.row}$
 $A \ j) \ i)$
unfolding *Matrix.scalar-prod-def*
proof (*rule sum-but-one*)
show $i < \text{dim-vec } (\text{Matrix.row } A \ j)$ **using** *assms* **by** *auto*
show $\forall ia < \text{dim-vec } (\text{Matrix.row } A \ j). ia \neq i \longrightarrow \text{Matrix.vec-index } (\text{unit-vec}$
 $n \ i) \ ia = 0$
using *assms* **unfolding** *unit-vec-def* **by** *auto*
qed
also have $\dots = (\text{Matrix.vec-index } (\text{Matrix.row } A \ j) \ i)$ **using** *assms* **by** *simp*
also have $\dots = A \ \$\$ (j, i)$ **using** *assms* $\langle j < \text{dim-vec } (\text{Matrix.col } A \ i) \rangle$ **by** *simp*
also have $\dots = \text{Matrix.vec-index } (\text{Matrix.col } A \ i) \ j$ **using** *assms* $\langle j < \text{dim-vec}$
 $(\text{Matrix.col } A \ i) \rangle$ **by** *simp*
finally show $\text{Matrix.vec-index } (A *_v \text{unit-vec } n \ i) \ j =$
 $\text{Matrix.vec-index } (\text{Matrix.col } A \ i) \ j .$
qed
qed

lemma *mat-prod-unit-vec-cong*:

assumes $(A :: 'a :: \text{conjugatable-field } \text{Matrix.mat}) \in \text{carrier-mat } n \ n$
and $B \in \text{carrier-mat } n \ n$
and $\bigwedge i. i < n \implies A *_v (\text{unit-vec } n \ i) = B *_v (\text{unit-vec } n \ i)$
shows $A = B$
proof
show $\text{dim-row } A = \text{dim-row } B$ **using** *assms* **by** *simp*
show $\text{dim-col } A = \text{dim-col } B$ **using** *assms* **by** *simp*
show $\bigwedge i j. i < \text{dim-row } B \implies j < \text{dim-col } B \implies A \ \$\$ (i, j) = B \ \$\$ (i, j)$
proof –
fix $i \ j$
assume $ij: i < \text{dim-row } B \ j < \text{dim-col } B$
hence $A \ \$\$ (i, j) = \text{Matrix.vec-index } (\text{Matrix.col } A \ j) \ i$ **using** *assms* **by** *simp*
also have $\dots = \text{Matrix.vec-index } (A *_v (\text{unit-vec } n \ j)) \ i$ **using** *mat-unit-vec-col* [of

A] *ij* *assms*
 by *simp*
 also have ... = *Matrix.vec-index* (*B* *_v (*unit-vec* *n* *j*)) *i* using *assms ij* by
simp
 also have ... = *Matrix.vec-index* (*Matrix.col* *B* *j*) *i* using *mat-unit-vec-col ij*
assms by *simp*
 also have ... = *B* \$\$ (*i,j*) using *assms ij* by *simp*
 finally show *A* \$\$ (*i, j*) = *B* \$\$ (*i, j*) .
 qed
 qed

lemma *smult-smult-times*:

fixes *a*::'*a*::*semigroup-mult*
 shows *a*·_m (*k*·_m *A*) = (*a* * *k*)·_m *A*
 proof
 show *r*:*dim-row* (*a*·_m (*k*·_m *A*)) = *dim-row* (*a* * *k*·_m *A*) by *simp*
 show *c*:*dim-col* (*a*·_m (*k*·_m *A*)) = *dim-col* (*a* * *k*·_m *A*) by *simp*
 show $\bigwedge i j. i < \text{dim-row } (a * k \cdot_m A) \implies$
 $j < \text{dim-col } (a * k \cdot_m A) \implies (a \cdot_m (k \cdot_m A)) \text{ $$ } (i, j) = (a * k \cdot_m A) \text{ $$}$
 (*i, j*)
 proof –
 fix *i j*
 assume *i* < *dim-row* (*a* * *k*·_m *A*) and *j* < *dim-col* (*a* * *k*·_m *A*) note *ij* =
 this
 hence (*a*·_m (*k*·_m *A*)) \$\$ (*i, j*) = *a* * (*k*·_m *A*) \$\$ (*i, j*) by *simp*
 also have ... = *a* * (*k* * *A* \$\$ (*i,j*)) using *ij* by *simp*
 also have ... = (*a* * *k*) * *A* \$\$ (*i,j*)
 by (*simp add: semigroup-mult-class.mult.assoc*)
 also have ... = (*a* * *k*·_m *A*) \$\$ (*i, j*) using *r c ij* by *simp*
 finally show (*a*·_m (*k*·_m *A*)) \$\$ (*i, j*) = (*a* * *k*·_m *A*) \$\$ (*i, j*) .
 qed
 qed

lemma *mat-minus-minus*:

fixes *A* :: '*a* :: *ab-group-add Matrix.mat*
 assumes *A* ∈ *carrier-mat* *n m*
 and *B* ∈ *carrier-mat* *n m*
 and *C* ∈ *carrier-mat* *n m*
 shows *A* - (*B* - *C*) = *A* - *B* + *C*
 proof
 show *dim-row* (*A* - (*B* - *C*)) = *dim-row* (*A* - *B* + *C*) using *assms* by *simp*
 show *dim-col* (*A* - (*B* - *C*)) = *dim-col* (*A* - *B* + *C*) using *assms* by *simp*
 show $\bigwedge i j. i < \text{dim-row } (A - B + C) \implies j < \text{dim-col } (A - B + C) \implies$
 $(A - (B - C)) \text{ $$ } (i, j) = (A - B + C) \text{ $$ } (i, j)$
 proof –
 fix *i j*
 assume *i* < *dim-row* (*A* - *B* + *C*) and *j* < *dim-col* (*A* - *B* + *C*) note *ij* =
 this
 have (*A* - (*B* - *C*)) \$\$ (*i, j*) = (*A* \$\$ (*i,j*) - *B* \$\$ (*i,j*) + *C* \$\$ (*i,j*)) using

ij *assms* **by** *simp*
also have ... = $(A - B + C) \text{ \$(\$ (i, j))}$ **using** *assms ij* **by** *simp*
finally show $(A - (B - C)) \text{ \$(\$ (i, j))} = (A - B + C) \text{ \$(\$ (i, j))}$.
qed
qed

2.2 Complements on complex matrices

lemma *hermitian-square*:
assumes *hermitian M*
shows $M \in \text{carrier-mat } (\text{dim-row } M) (\text{dim-row } M)$
proof –
have $\text{dim-col } M = \text{dim-row } M$ **using** *assms* **unfolding** *hermitian-def* *adjoint-def*
by (*metis adjoint-dim-col*)
thus *?thesis* **by** *auto*
qed

lemma *hermitian-add*:
assumes $A \in \text{carrier-mat } n \ n$
and $B \in \text{carrier-mat } n \ n$
and *hermitian A*
and *hermitian B*
shows *hermitian* $(A + B)$ **unfolding** *hermitian-def*
by (*metis adjoint-add assms hermitian-def*)

lemma *hermitian-minus*:
assumes $A \in \text{carrier-mat } n \ n$
and $B \in \text{carrier-mat } n \ n$
and *hermitian A*
and *hermitian B*
shows *hermitian* $(A - B)$ **unfolding** *hermitian-def*
by (*metis adjoint-minus assms hermitian-def*)

lemma *hermitian-smult*:
fixes $a::\text{real}$
fixes $A::\text{complex Matrix.mat}$
assumes $A \in \text{carrier-mat } n \ n$
and *hermitian A*
shows *hermitian* $(a \cdot_m A)$
proof –
have $\text{dim: Complex-Matrix.adjoint } A \in \text{carrier-mat } n \ n$ **using** *assms* **by** (*simp*
add: adjoint-dim)
{
fix $i \ j$
assume $i < n$ **and** $j < n$
hence $\text{Complex-Matrix.adjoint } (a \cdot_m A) \text{ \$(\$ (i,j))} = a * (\text{Complex-Matrix.adjoint}$
 $A \text{ \$(\$ (i,j))})$
using *adjoint-scale[of a A]* *assms* **by** *simp*
also have ... = $a * (A \text{ \$(\$ (i,j))})$ **using** *assms* **unfolding** *hermitian-def* **by** *simp*

```

    also have ... = (a ·m A) $$ (i,j) using ‹i < n› ‹j < n› assms by simp
    finally have Complex-Matrix.adjoint (a ·m A) $$ (i,j) = (a ·m A) $$ (i,j) .
  }
  thus ?thesis using dim assms unfolding hermitian-def by auto
qed

lemma unitary-eigenvalues-norm-square:
  fixes U::complex Matrix.mat
  assumes unitary U
  and U ∈ carrier-mat n n
  and eigenvalue U k
shows conjugate k * k = 1
proof -
  have ∃ v. eigenvector U v k using assms unfolding eigenvalue-def by simp
  from this obtain v where eigenvector U v k by auto
  define vn where vn = vec-normalize v
  have eigenvector U vn k using normalize-keep-eigenvector ‹eigenvector U v k›
    using assms(2) eigenvector-def vn-def by blast
  have vn ∈ carrier-vec n
    using ‹eigenvector U v k› assms(2) eigenvector-def normalized-vec-dim vn-def
  by blast
  have Complex-Matrix.inner-prod vn vn = 1 using ‹vn = vec-normalize v› ‹eigen-
vector U v k›
    eigenvector-def normalized-vec-norm by blast
  hence conjugate k * k = conjugate k * k * Complex-Matrix.inner-prod vn vn by
simp
  also have ... = conjugate k * Complex-Matrix.inner-prod vn (k ·v vn)
  proof -
    have k * Complex-Matrix.inner-prod vn vn = Complex-Matrix.inner-prod vn
(k ·v vn)
    using inner-prod-smult-left[of vn n vn k] ‹vn ∈ carrier-vec n› by simp
    thus ?thesis by simp
  qed
  also have ... = Complex-Matrix.inner-prod (k ·v vn) (k ·v vn)
    using inner-prod-smult-right[of vn n - k] by (simp add: ‹vn ∈ carrier-vec n›)
  also have ... = Complex-Matrix.inner-prod (U *v vn) (U *v vn)
    using ‹eigenvector U vn k› unfolding eigenvector-def by simp
  also have ... =
    Complex-Matrix.inner-prod (Complex-Matrix.adjoint U *v (U *v vn)) vn
    using adjoint-def-alter[of U *v vn n vn n U] assms
  by (metis ‹eigenvector U vn k› carrier-matD(1) carrier-vec-dim-vec dim-mult-mat-vec

    eigenvector-def)
  also have ... = Complex-Matrix.inner-prod vn vn
  proof -
    have Complex-Matrix.adjoint U *v (U *v vn) = (Complex-Matrix.adjoint U *
U) *v vn
    using assms
    by (metis ‹eigenvector U vn k› adjoint-dim assoc-mult-mat-vec carrier-matD(1)

```

eigenvector-def)
also have ... = vn **using** *assms* **unfolding** *unitary-def* *inverts-mat-def*
by (*metis* \langle *eigenvector* U vn k \rangle *assms*(1) *eigenvector-def* *one-mult-mat-vec*
unitary-simps(1))
finally show ?*thesis* **by** *simp*
qed
also have ... = 1 **using** \langle *vn* = *vec-normalize* v \rangle \langle *eigenvector* U v k \rangle *eigenvector-def*
normalized-vec-norm **by** *blast*
finally show ?*thesis* .
qed

lemma *outer-prod-smult-left*:

fixes $v::\text{complex Matrix.vec}$
shows *outer-prod* ($a \cdot_v v$) w = $a \cdot_m$ *outer-prod* $v w$
proof –
define *paw* **where** *paw* = *outer-prod* ($a \cdot_v v$) w
define *apw* **where** *apw* = $a \cdot_m$ *outer-prod* $v w$
have *paw* = *apw*
proof
have *dim-row* *paw* = *dim-vec* v **unfolding** *paw-def* **using** *outer-prod-dim*
by (*metis* *carrier-matD*(1) *carrier-vec-dim-vec* *index-smult-vec*(2))
also have ... = *dim-row* *apw* **unfolding** *apw-def* **using** *outer-prod-dim*
by (*metis* *carrier-matD*(1) *carrier-vec-dim-vec* *index-smult-mat*(2))
finally show *dr*: *dim-row* *paw* = *dim-row* *apw* .
have *dim-col* *paw* = *dim-vec* w **unfolding** *paw-def* **using** *outer-prod-dim*
using *carrier-vec-dim-vec* **by** *blast*
also have ... = *dim-col* *apw* **unfolding** *apw-def* **using** *outer-prod-dim*
by (*metis* *apw-def* *carrier-matD*(2) *carrier-vec-dim-vec* *smult-carrier-mat*)
finally show *dc*: *dim-col* *paw* = *dim-col* *apw* .
show $\bigwedge i j. i < \text{dim-row } apw \implies j < \text{dim-col } apw \implies paw \ \$\$ (i, j) = apw \ \$\$$
 (i, j)
proof –
fix $i j$
assume $i < \text{dim-row } apw$ **and** $j < \text{dim-col } apw$ **note** $ij = \text{this}$
hence $paw \ \$\$ (i, j) = a * (\text{Matrix.vec-index } v \ i) * \text{conj } (\text{Matrix.vec-index } w \ j)$
using *dr* *dc* **unfolding** *paw-def* *outer-prod-def* **by** *simp*
also have ... = $apw \ \$\$ (i, j)$ **using** *dr* *dc* *ij* **unfolding** *apw-def* *outer-prod-def*
by *simp*
finally show $paw \ \$\$ (i, j) = apw \ \$\$ (i, j)$.
qed
qed
thus ?*thesis* **unfolding** *paw-def* *apw-def* **by** *simp*
qed

lemma *outer-prod-smult-right*:

fixes $v::\text{complex Matrix.vec}$
shows *outer-prod* $v (a \cdot_v w) = (\text{conjugate } a) \cdot_m$ *outer-prod* $v w$
proof –

```

define paw where paw = outer-prod v (a ·v w)
define apw where apw = (conjugate a) ·m outer-prod v w
have paw = apw
proof
  have dim-row paw = dim-vec v unfolding paw-def using outer-prod-dim
    by (metis carrier-matD(1) carrier-vec-dim-vec)
  also have ... = dim-row apw unfolding apw-def using outer-prod-dim
    by (metis carrier-matD(1) carrier-vec-dim-vec index-smult-mat(2))
  finally show dr: dim-row paw = dim-row apw .
  have dim-col paw = dim-vec w unfolding paw-def using outer-prod-dim
    using carrier-vec-dim-vec by (metis carrier-matD(2) index-smult-vec(2))
  also have ... = dim-col apw unfolding apw-def using outer-prod-dim
    by (metis apw-def carrier-matD(2) carrier-vec-dim-vec smult-carrier-mat)
  finally show dc: dim-col paw = dim-col apw .
  show  $\bigwedge i j. i < \text{dim-row } apw \implies j < \text{dim-col } apw \implies paw \text{ } \$\$ (i, j) = apw \text{ } \$\$$ 
(i, j)
  proof –
    fix i j
    assume i < dim-row apw and j < dim-col apw note ij = this
    hence paw  $\text{ } \$\$ (i, j) = (\text{conjugate } a) * (\text{Matrix.vec-index } v \ i) * \text{cnj } (\text{Matrix.vec-index } w \ j)$ 
    using dr dc unfolding paw-def outer-prod-def by simp
    also have ... = apw  $\text{ } \$\$ (i, j)$  using dr dc ij unfolding apw-def outer-prod-def
by simp
    finally show paw  $\text{ } \$\$ (i, j) = apw \text{ } \$\$ (i, j)$  .
  qed
qed
thus ?thesis unfolding paw-def apw-def by simp
qed

lemma outer-prod-add-left:
  fixes v::complex Matrix.vec
  assumes dim-vec v = dim-vec x
  shows outer-prod (v + x) w = outer-prod v w + (outer-prod x w)
proof –
  define paw where paw = outer-prod (v+x) w
  define apw where apw = outer-prod v w + (outer-prod x w)
  have paw = apw
  proof
    have rv: dim-row paw = dim-vec v unfolding paw-def using outer-prod-dim
      assms
    by (metis carrier-matD(1) carrier-vec-dim-vec index-add-vec(2) paw-def)
    also have ... = dim-row apw unfolding apw-def using outer-prod-dim assms
    by (metis carrier-matD(1) carrier-vec-dim-vec index-add-mat(2))
    finally show dr: dim-row paw = dim-row apw .
    have cw: dim-col paw = dim-vec w unfolding paw-def using outer-prod-dim
      assms
    using carrier-vec-dim-vec by (metis carrier-matD(2))
    also have ... = dim-col apw unfolding apw-def using outer-prod-dim

```

```

    by (metis apw-def carrier-matD(2) carrier-vec-dim-vec add-carrier-mat)
  finally show dc: dim-col paw = dim-col apw .
  show  $\bigwedge i j. i < \text{dim-row } apw \implies j < \text{dim-col } apw \implies paw \ \$\$ (i, j) = apw \ \$\$$ 
(i, j)
  proof -
    fix i j
    assume i < dim-row apw and j < dim-col apw note ij = this
    hence paw  $\ \$\$ (i, j) = (\text{Matrix.vec-index } v \ i + \text{Matrix.vec-index } x \ i) *$ 
      cnj (Matrix.vec-index w j)
    using dr dc unfolding paw-def outer-prod-def by simp
    also have ... = Matrix.vec-index v i * cnj (Matrix.vec-index w j) +
      Matrix.vec-index x i * cnj (Matrix.vec-index w j)
    by (simp add: ring-class.ring-distrib(2))
    also have ... = (outer-prod v w)  $\ \$\$ (i, j) + (\text{outer-prod } x \ w) \ \$\$ (i, j)$ 
    using rv cw dr dc ij assms unfolding outer-prod-def by auto
    also have ... = apw  $\ \$\$ (i, j)$  using dr dc ij unfolding apw-def outer-prod-def
  by simp
  finally show paw  $\ \$\$ (i, j) = apw \ \$\$ (i, j)$  .
  qed
  qed
  thus ?thesis unfolding paw-def apw-def by simp
  qed

```

lemma outer-prod-add-right:

```

  fixes v::complex Matrix.vec
  assumes dim-vec w = dim-vec x
  shows outer-prod v (w + x) = outer-prod v w + (outer-prod v x)
  proof -
    define paw where paw = outer-prod v (w+x)
    define apw where apw = outer-prod v w + (outer-prod v x)
    have paw = apw
    proof
      have rv: dim-row paw = dim-vec v unfolding paw-def using outer-prod-dim
      assms
      by (metis carrier-matD(1) carrier-vec-dim-vec index-add-vec(2) paw-def)
      also have ... = dim-row apw unfolding apw-def using outer-prod-dim assms
      by (metis carrier-matD(1) carrier-vec-dim-vec index-add-mat(2))
      finally show dr: dim-row paw = dim-row apw .
      have cw: dim-col paw = dim-vec w unfolding paw-def using outer-prod-dim
      assms
      using carrier-vec-dim-vec
      by (metis carrier-matD(2) index-add-vec(2) paw-def)
      also have ... = dim-col apw unfolding apw-def using outer-prod-dim
      by (metis assms carrier-matD(2) carrier-vec-dim-vec index-add-mat(3))
      finally show dc: dim-col paw = dim-col apw .
      show  $\bigwedge i j. i < \text{dim-row } paw \implies j < \text{dim-col } paw \implies paw \ \$\$ (i, j) = apw \ \$\$$ 
(i, j)
      proof -
        fix i j

```

assume $i < \dim\text{-row } apw$ **and** $j < \dim\text{-col } apw$ **note** $ij = \text{this}$
hence $paw \text{ } \$\$ (i,j) = \text{Matrix.vec-index } v \ i * \text{cnj } (\text{Matrix.vec-index } w \ j + (\text{Matrix.vec-index } x \ j))$
using $dr \ dc$ **unfolding** $paw\text{-def}$ $outer\text{-prod-def}$ **by** $simp$
also have $\dots = \text{Matrix.vec-index } v \ i * \text{cnj } (\text{Matrix.vec-index } w \ j) + \text{Matrix.vec-index } v \ i * \text{cnj } (\text{Matrix.vec-index } x \ j)$
by $(simp \ add: \ ring\text{-class.ring-distrib}(1))$
also have $\dots = (outer\text{-prod } v \ w) \ \$\$ (i,j) + (outer\text{-prod } v \ x) \ \$\$ (i,j)$
using $rv \ cw \ dr \ dc \ ij \ assms$ **unfolding** $outer\text{-prod-def}$ **by** $auto$
also have $\dots = apw \ \$\$ (i,j)$ **using** $dr \ dc \ ij$ **unfolding** $apw\text{-def}$ $outer\text{-prod-def}$
by $simp$
finally show $paw \ \$\$ (i, j) = apw \ \$\$ (i, j) .$
qed
qed
thus $?thesis$ **unfolding** $paw\text{-def}$ $apw\text{-def}$ **by** $simp$
qed

lemma $outer\text{-prod-minus-left}$:

fixes $v::\text{complex } \text{Matrix.vec}$
assumes $\dim\text{-vec } v = \dim\text{-vec } x$
shows $outer\text{-prod } (v - x) \ w = outer\text{-prod } v \ w - (outer\text{-prod } x \ w)$
proof $-$
define paw **where** $paw = outer\text{-prod } (v-x) \ w$
define apw **where** $apw = outer\text{-prod } v \ w - (outer\text{-prod } x \ w)$
have $paw = apw$
proof
have $rv: \dim\text{-row } paw = \dim\text{-vec } v$ **unfolding** $paw\text{-def}$ **using** $outer\text{-prod-dim}$
 $assms$
by $(metis \ carrier\text{-matD}(1) \ carrier\text{-vec-dim-vec} \ index\text{-minus-vec}(2) \ paw\text{-def})$
also have $\dots = \dim\text{-row } apw$ **unfolding** $apw\text{-def}$ **using** $outer\text{-prod-dim}$ $assms$
by $(metis \ carrier\text{-matD}(1) \ carrier\text{-vec-dim-vec} \ index\text{-minus-mat}(2))$
finally show $dr: \dim\text{-row } paw = \dim\text{-row } apw .$
have $cw: \dim\text{-col } paw = \dim\text{-vec } w$ **unfolding** $paw\text{-def}$ **using** $outer\text{-prod-dim}$
 $assms$
using $carrier\text{-vec-dim-vec}$ **by** $(metis \ carrier\text{-matD}(2))$
also have $\dots = \dim\text{-col } apw$ **unfolding** $apw\text{-def}$ **using** $outer\text{-prod-dim}$
by $(metis \ apw\text{-def} \ carrier\text{-matD}(2) \ carrier\text{-vec-dim-vec} \ minus\text{-carrier-mat})$
finally show $dc: \dim\text{-col } paw = \dim\text{-col } apw .$
show $\bigwedge i \ j. \ i < \dim\text{-row } apw \implies j < \dim\text{-col } apw \implies paw \ \$\$ (i, j) = apw \ \$\$ (i, j)$
proof $-$
fix $i \ j$
assume $i < \dim\text{-row } apw$ **and** $j < \dim\text{-col } apw$ **note** $ij = \text{this}$
hence $paw \ \$\$ (i,j) = (\text{Matrix.vec-index } v \ i - \text{Matrix.vec-index } x \ i) * \text{cnj } (\text{Matrix.vec-index } w \ j)$
using $dr \ dc$ **unfolding** $paw\text{-def}$ $outer\text{-prod-def}$ **by** $simp$
also have $\dots = \text{Matrix.vec-index } v \ i * \text{cnj } (\text{Matrix.vec-index } w \ j) - \text{Matrix.vec-index } x \ i * \text{cnj } (\text{Matrix.vec-index } w \ j)$
by $(simp \ add: \ ring\text{-class.ring-distrib})$

also have ... = (outer-prod v w) \$\$ (i,j) - (outer-prod x w) \$\$ (i,j)
using rv cw dr dc ij *assms* **unfolding** outer-prod-def **by** auto
also have ... = apw \$\$ (i,j) **using** dr dc ij **unfolding** apw-def outer-prod-def
by simp
finally show paw \$\$ (i, j) = apw \$\$ (i, j) .
qed
qed
thus ?thesis **unfolding** paw-def apw-def **by** simp
qed

lemma outer-prod-minus-right:

fixes v::complex Matrix.vec
assumes dim-vec w = dim-vec x
shows outer-prod v (w - x) = outer-prod v w - (outer-prod v x)
proof -
define paw **where** paw = outer-prod v (w-x)
define apw **where** apw = outer-prod v w - (outer-prod v x)
have paw = apw
proof
have rv: dim-row paw = dim-vec v **unfolding** paw-def **using** outer-prod-dim
assms
by (metis carrier-matD(1) carrier-vec-dim-vec paw-def)
also have ... = dim-row apw **unfolding** apw-def **using** outer-prod-dim *assms*
by (metis carrier-matD(1) carrier-vec-dim-vec index-minus-mat(2))
finally show dr: dim-row paw = dim-row apw .
have cw: dim-col paw = dim-vec w **unfolding** paw-def **using** outer-prod-dim
assms
using carrier-vec-dim-vec
by (metis carrier-matD(2) index-minus-vec(2) paw-def)
also have ... = dim-col apw **unfolding** apw-def **using** outer-prod-dim
by (metis *assms* carrier-matD(2) carrier-vec-dim-vec index-minus-mat(3))
finally show dc: dim-col paw = dim-col apw .
show $\bigwedge i j. i < \text{dim-row apw} \implies j < \text{dim-col apw} \implies \text{paw } \$\$ (i, j) = \text{apw } \$\$ (i, j)$
proof -
fix i j
assume i < dim-row apw **and** j < dim-col apw **note** ij = this
hence paw \$\$ (i,j) = Matrix.vec-index v i *
(cnj (Matrix.vec-index w j - (Matrix.vec-index x j)))
using dr dc **unfolding** paw-def outer-prod-def **by** simp
also have ... = Matrix.vec-index v i * cnj (Matrix.vec-index w j) -
Matrix.vec-index v i * cnj (Matrix.vec-index x j)
by (simp add: ring-class.ring-distrib)
also have ... = (outer-prod v w) \$\$ (i,j) - (outer-prod v x) \$\$ (i,j)
using rv cw dr dc ij *assms* **unfolding** outer-prod-def **by** auto
also have ... = apw \$\$ (i,j) **using** dr dc ij **unfolding** apw-def outer-prod-def
by simp
finally show paw \$\$ (i, j) = apw \$\$ (i, j) .
qed

qed
 thus ?thesis unfolding paw-def apw-def by simp
 qed

lemma outer-minus-minus:

fixes a::complex Matrix.vec

assumes dim-vec a = dim-vec b

and dim-vec u = dim-vec v

shows outer-prod (a - b) (u - v) = outer-prod a u - outer-prod a v -
 outer-prod b u + outer-prod b v

proof -

have outer-prod (a - b) (u - v) = outer-prod a (u - v)

- outer-prod b (u - v) using outer-prod-minus-left assms by simp

also have ... = outer-prod a u - outer-prod a v -

outer-prod b (u - v) using assms outer-prod-minus-right by simp

also have ... = outer-prod a u - outer-prod a v -

(outer-prod b u - outer-prod b v) using assms outer-prod-minus-right by simp

also have ... = outer-prod a u - outer-prod a v -

outer-prod b u + outer-prod b v

proof (rule mat-minus-minus)

show outer-prod b u ∈ carrier-mat (dim-vec b) (dim-vec u) by simp

show outer-prod b v ∈ carrier-mat (dim-vec b) (dim-vec u) using assms by

simp

show outer-prod a u - outer-prod a v ∈ carrier-mat (dim-vec b) (dim-vec u)

using assms

by (metis carrier-vecI minus-carrier-mat outer-prod-dim)

qed

finally show ?thesis .

qed

lemma outer-trace-inner:

assumes A ∈ carrier-mat n n

and dim-vec u = n

and dim-vec v = n

shows Complex-Matrix.trace (outer-prod u v * A) = Complex-Matrix.inner-prod
 v (A *_v u)

proof -

have Complex-Matrix.trace (outer-prod u v * A) = Complex-Matrix.trace (A *
 outer-prod u v)

proof (rule trace-comm)

show A ∈ carrier-mat n n using assms by simp

show outer-prod u v ∈ carrier-mat n n using assms

by (metis carrier-vec-dim-vec outer-prod-dim)

qed

also have ... = Complex-Matrix.inner-prod v (A *_v u) using trace-outer-prod-right[of
 A n u v]

assms carrier-vec-dim-vec by metis

finally show ?thesis .

qed

lemma *zero-hermitian*:

shows *hermitian* $(0_m \ n \ n)$ **unfolding** *hermitian-def*
by (*metis adjoint-minus hermitian-def hermitian-one minus-r-inv-mat one-carrier-mat*)

lemma *trace-1*:

shows *Complex-Matrix.trace* $((1_m \ n)::\text{complex Matrix.mat}) = (n::\text{complex})$ **using**
one-mat-def
by (*simp add: Complex-Matrix.trace-def Matrix.mat-def*)

lemma *trace-add*:

assumes *square-mat A*
and *square-mat B*
and *dim-row A = dim-row B*
shows *Complex-Matrix.trace* $(A + B) = \text{Complex-Matrix.trace } A + \text{Complex-Matrix.trace } B$
using *assms* **by** (*simp add: Complex-Matrix.trace-def sum.distrib*)

lemma *bra-vec-carrier*:

shows *bra-vec* $v \in \text{carrier-mat } 1 \ (\text{dim-vec } v)$
proof –
have *dim-row* $(\text{ket-vec } v) = \text{dim-vec } v$ **unfolding** *ket-vec-def* **by** *simp*
thus *?thesis* **using** *bra-bra-vec[of v] bra-def[of ket-vec v]* **by** *simp*
qed

lemma *mat-mult-ket-carrier*:

assumes $A \in \text{carrier-mat } n \ m$
shows $A * |v\rangle \in \text{carrier-mat } n \ 1$ **using** *assms*
by (*metis bra-bra-vec bra-vec-carrier carrier-matD(1) carrier-matI dagger-of-ket-is-bra*
dim-row-of-dagger index-mult-mat(2) index-mult-mat(3))

lemma *mat-mult-ket*:

assumes $A \in \text{carrier-mat } n \ m$
and *dim-vec* $v = m$
shows $A * |v\rangle = |A *_v v\rangle$
proof –
have *rn*: *dim-row* $(A * |v\rangle) = n$ **unfolding** *times-mat-def* **using** *assms* **by** *simp*
have *co*: *dim-col* $|A *_v v\rangle = 1$ **using** *assms* **unfolding** *ket-vec-def* **by** *simp*
have *cov*: *dim-col* $|v\rangle = 1$ **using** *assms* **unfolding** *ket-vec-def* **by** *simp*
have *er*: *dim-row* $(A * |v\rangle) = \text{dim-row } |A *_v v\rangle$ **using** *assms*
by (*metis bra-bra-vec bra-vec-carrier carrier-matD(2) dagger-of-ket-is-bra dim-col-of-dagger*
dim-mult-mat-vec index-mult-mat(2))
have *ec*: *dim-col* $(A * |v\rangle) = \text{dim-col } |A *_v v\rangle$ **using** *assms*
by (*metis carrier-matD(2) index-mult-mat(3) mat-mult-ket-carrier*)
{
 fix $i::\text{nat}$

```

fix j::nat
assume i < n
and j < 1
hence j = 0 by simp
have (A * |v>) $$ (i,0) = Matrix.scalar-prod (Matrix.row A i) (Matrix.col |v)
0)
  using times-mat-def[of A] ⟨i < n⟩ rn cov by simp
  also have ... = Matrix.scalar-prod (Matrix.row A i) v using ket-vec-col by
simp
  also have ... = |A *_v v⟩ $$ (i,j) unfolding mult-mat-vec-def
  using ⟨i < n⟩ ⟨j = 0⟩ assms(1) by auto
} note idx = this
have A * |v⟩ = Matrix.mat n 1 (λ(i, j). Matrix.scalar-prod (Matrix.row A i)
(Matrix.col |v⟩ j))
  using assms unfolding times-mat-def ket-vec-def by simp
  also have ... = |A *_v v⟩ using er ec idx rn co by auto
  finally show ?thesis .
qed

```

lemma unitary-density:

```

assumes density-operator R
and unitary U
and R ∈ carrier-mat n n
and U ∈ carrier-mat n n
shows density-operator (U * R * (Complex-Matrix.adjoint U)) unfolding den-
sity-operator-def
proof (intro conjI)
  show Complex-Matrix.positive (U * R * Complex-Matrix.adjoint U)
  proof (rule positive-close-under-left-right-mult-adjoint)
    show U ∈ carrier-mat n n using assms by simp
    show R ∈ carrier-mat n n using assms by simp
    show Complex-Matrix.positive R using assms unfolding density-operator-def
by simp
  qed
  have Complex-Matrix.trace (U * R * Complex-Matrix.adjoint U) =
    Complex-Matrix.trace (Complex-Matrix.adjoint U * U * R)
  using trace-comm[of U * R n Complex-Matrix.adjoint U] assms
  by (metis adjoint-dim mat-assoc-test(10))
  also have ... = Complex-Matrix.trace R using assms by simp
  also have ... = 1 using assms unfolding density-operator-def by simp
  finally show Complex-Matrix.trace (U * R * Complex-Matrix.adjoint U) = 1 .
qed

```

2.3 Tensor product complements

lemma tensor-vec-dim[simp]:

shows dim-vec (tensor-vec u v) = dim-vec u * (dim-vec v)

proof –

have length (mult.vec-vec-Tensor (*) (list-of-vec u) (list-of-vec v)) =

```

    length (list-of-vec u) * length (list-of-vec v)
  using mult.vec-vec-Tensor-length[of 1::real (*) list-of-vec u list-of-vec v]
  by (simp add: Matrix-Tensor.mult-def)
  thus ?thesis unfolding tensor-vec-def by simp
qed

lemma index-tensor-vec[simp]:
  assumes 0 < dim-vec v
  and i < dim-vec u * dim-vec v
  shows vec-index (tensor-vec u v) i =
    vec-index u (i div (dim-vec v)) * vec-index v (i mod dim-vec v)
  proof -
    have m: Matrix-Tensor.mult (1::complex) (*) by (simp add: Matrix-Tensor.mult-def)

    have length (list-of-vec v) = dim-vec v using assms by simp
    hence vec-index (tensor-vec u v) i = (*) (vec-index u (i div dim-vec v)) (vec-index
  v (i mod dim-vec v))
      unfolding tensor-vec-def using mult.vec-vec-Tensor-elements assms m
      by (metis (mono-tags, lifting) length-greater-0-conv length-list-of-vec list-of-vec-index

          mult.vec-vec-Tensor-elements vec-of-list-index)
    thus ?thesis by simp
  qed

lemma outer-prod-tensor-comm:
  fixes a::complex Matrix.vec
  fixes u::complex Matrix.vec
  assumes 0 < dim-vec a
  and 0 < dim-vec b
  shows outer-prod (tensor-vec u v) (tensor-vec a b) = tensor-mat (outer-prod u a)
  (outer-prod v b)
  proof -
    define ot where ot = outer-prod (tensor-vec u v) (tensor-vec a b)
    define to where to = tensor-mat (outer-prod u a) (outer-prod v b)
    define dv where dv = dim-vec v
    define db where db = dim-vec b
    have ot = to
    proof
      have ro: dim-row ot = dim-vec u * dim-vec v unfolding ot-def outer-prod-def
    by simp
      have dim-row to = dim-row (outer-prod u a) * dim-row (outer-prod v b)
      unfolding to-def by simp
      also have ... = dim-vec u * dim-vec v using outer-prod-dim
      by (metis carrier-matD(1) carrier-vec-dim-vec)
      finally have rt: dim-row to = dim-vec u * dim-vec v .
      show dim-row ot = dim-row to using ro rt by simp
      have co: dim-col ot = dim-vec a * dim-vec b unfolding ot-def outer-prod-def
    by simp
      have dim-col to = dim-col (outer-prod u a) * dim-col (outer-prod v b) unfolding

```

to-def **by** *simp*
also have ... = *dim-vec a * dim-vec b* **using** *outer-prod-dim*
by (*metis carrier-matD(2) carrier-vec-dim-vec*)
finally have *ct: dim-col to = dim-vec a * dim-vec b .*
show *dim-col ot = dim-col to* **using** *co ct* **by** *simp*
show $\bigwedge i j. i < \text{dim-row to} \implies j < \text{dim-col to} \implies \text{ot } \$\$ (i, j) = \text{to } \$\$ (i, j)$
proof –
fix *i j*
assume *i < dim-row to* **and** *j < dim-col to* **note** *ij = this*
have *ot* $\$\$ (i, j) = \text{Matrix.vec-index (tensor-vec u v) } i * (\text{conjugate (Matrix.vec-index (tensor-vec a b) } j))$
unfolding *ot-def outer-prod-def* **using** *ij rt ct* **by** *simp*
also have ... = *vec-index u (i div dv) * vec-index v (i mod dv) **
(conjugate (Matrix.vec-index (tensor-vec a b) j)) **using** *ij rt assms*
unfolding *dv-def*
by (*metis index-tensor-vec less-nat-zero-code nat-0-less-mult-iff neq0-conv*)
also have ... = *vec-index u (i div dv) * vec-index v (i mod dv) **
*(conjugate (vec-index a (j div db) * vec-index b (j mod db)))* **using** *ij ct*
assms
unfolding *db-def* **by** *simp*
also have ... = *vec-index u (i div dv) * vec-index v (i mod dv) **
*(conjugate (vec-index a (j div db))) * (conjugate (vec-index b (j mod db)))*
by *simp*
also have ... = *vec-index u (i div dv) * (conjugate (vec-index a (j div db))) **
*vec-index v (i mod dv) * (conjugate (vec-index b (j mod db)))* **by** *simp*
also have ... = (*outer-prod u a*) $\$\$ (i \text{ div } dv, j \text{ div } db) *
*vec-index v (i mod dv) * (conjugate (vec-index b (j mod db)))*
proof –
have *i div dv < dim-vec u* **using** *ij rt* **unfolding** *dv-def*
by (*simp add: less-mult-imp-div-less*)
moreover have *j div db < dim-vec a* **using** *ij ct assms* **unfolding** *db-def*
by (*simp add: less-mult-imp-div-less*)
ultimately have *vec-index u (i div dv) * (conjugate (vec-index a (j div db)))*
=
(outer-prod u a) \$\$ (i div dv, j div db) **unfolding** *outer-prod-def* **by** *simp*
thus *?thesis* **by** *simp*
qed
also have ... = (*outer-prod u a*) $\$\$ (i \text{ div } dv, j \text{ div } db) *$
(outer-prod v b) \$\$ (i mod dv, j mod db)
proof –
have *i mod dv < dim-vec v* **using** *ij rt* **unfolding** *dv-def*
using *assms mod-less-divisor*
by (*metis less-nat-zero-code mult commute neq0-conv times-nat.simps(1)*)
moreover have *j mod db < dim-vec b* **using** *ij ct assms* **unfolding** *db-def*
by (*simp add: less-mult-imp-div-less*)
ultimately have *vec-index v (i mod dv) * (conjugate (vec-index b (j mod*
db))) =
(outer-prod v b) \$\$ (i mod dv, j mod db) **unfolding** *outer-prod-def* **by** *simp*
thus *?thesis* **by** *simp*$

qed
also have ... = *tensor-mat* (*outer-prod u a*) (*outer-prod v b*) \$\$ (i, j)
proof (*rule index-tensor-mat[symmetric]*)
show *dim-row* (*outer-prod u a*) = *dim-vec u* **unfolding** *outer-prod-def* **by**
simp
show *dim-row* (*outer-prod v b*) = *dv* **unfolding** *outer-prod-def dv-def* **by**
simp
show *dim-col* (*outer-prod v b*) = *db* **unfolding** *db-def outer-prod-def* **by**
simp
show $i < \text{dim-vec } u * dv$ **unfolding** *dv-def* **using** *ij rt* **by** *simp*
show *dim-col* (*outer-prod u a*) = *dim-vec a* **unfolding** *outer-prod-def* **by**
simp
show $j < \text{dim-vec } a * db$ **unfolding** *db-def* **using** *ij ct* **by** *simp*
show $0 < \text{dim-vec } a$ **using** *assms* **by** *simp*
show $0 < db$ **unfolding** *db-def* **using** *assms* **by** *simp*
qed
finally show *ot* \$\$ (i, j) = *to* \$\$ (i, j) **unfolding** *to-def* .
qed
qed
thus *?thesis* **unfolding** *ot-def to-def* **by** *simp*
qed

lemma *tensor-mat-adjoint*:

assumes $m1 \in \text{carrier-mat } r1 \ c1$
and $m2 \in \text{carrier-mat } r2 \ c2$
and $0 < c1$
and $0 < c2$
and $0 < r1$
and $0 < r2$
shows *Complex-Matrix.adjoint* (*tensor-mat m1 m2*) =
tensor-mat (*Complex-Matrix.adjoint m1*) (*Complex-Matrix.adjoint m2*)
apply (*rule eq-matI, auto*)
proof –
fix $i \ j$
assume $i < \text{dim-col } m1 * \text{dim-col } m2$ **and** $j < \text{dim-row } m1 * \text{dim-row } m2$ **note**
ij = this
have $c1: \text{dim-col } m1 = c1$ **using** *assms* **by** *simp*
have $r1: \text{dim-row } m1 = r1$ **using** *assms* **by** *simp*
have $c2: \text{dim-col } m2 = c2$ **using** *assms* **by** *simp*
have $r2: \text{dim-row } m2 = r2$ **using** *assms* **by** *simp*
have *Complex-Matrix.adjoint* ($m1 \otimes m2$) \$\$ (i, j) = *conjugate* (($m1 \otimes m2$)
\$\$ (j, i))
using *ij* **by** (*simp add: adjoint-eval*)
also have ... = *conjugate* ($m1$ \$\$ (j div r2, i div c2) * $m2$ \$\$ (j mod r2, i mod
c2))
proof –
have ($m1 \otimes m2$) \$\$ (j, i) = $m1$ \$\$ (j div r2, i div c2) * $m2$ \$\$ (j mod r2, i
mod c2)
proof (*rule index-tensor-mat[of m1 r1 c1 m2 r2 c2 j i]*, (*auto simp add: assms*

$ij\ c1\ c2\ r1\ r2))$
show $j < r1 * r2$ **using** $ij\ r1\ r2$ **by** *simp*
show $i < c1 * c2$ **using** $ij\ c1\ c2$ **by** *simp*
qed
thus *?thesis* **by** *simp*
qed
also have $\dots = \text{conjugate } (m1\ \$\$ (j\ \text{div } r2, i\ \text{div } c2)) * \text{conjugate } (m2\ \$\$ (j\ \text{mod } r2, i\ \text{mod } c2))$
by *simp*
also have $\dots = (\text{Complex-Matrix.adjoint } m1)\ \$\$ (i\ \text{div } c2, j\ \text{div } r2) * \text{conjugate } (m2\ \$\$ (j\ \text{mod } r2, i\ \text{mod } c2))$
by *(metis adjoint-eval c2 ij less-mult-imp-div-less r2)*
also have $\dots = (\text{Complex-Matrix.adjoint } m1)\ \$\$ (i\ \text{div } c2, j\ \text{div } r2) * (\text{Complex-Matrix.adjoint } m2)\ \$\$ (i\ \text{mod } c2, j\ \text{mod } r2)$
using $\langle 0 < c2 \rangle \langle 0 < r2 \rangle$ **by** *(simp add: adjoint-eval c2 r2)*
also have $\dots = (\text{tensor-mat } (\text{Complex-Matrix.adjoint } m1)\ (\text{Complex-Matrix.adjoint } m2))\ \$\$ (i, j)$
proof *(rule index-tensor-mat[symmetric], (simp add: ij c1 c2 r1 r2) +)*
show $i < c1 * c2$ **using** $ij\ c1\ c2$ **by** *simp*
show $j < r1 * r2$ **using** $ij\ r1\ r2$ **by** *simp*
show $0 < r1$ **using** *assms* **by** *simp*
show $0 < r2$ **using** *assms* **by** *simp*
qed
finally show $\text{Complex-Matrix.adjoint } (m1 \otimes m2)\ \$\$ (i, j) = (\text{Complex-Matrix.adjoint } m1 \otimes \text{Complex-Matrix.adjoint } m2)\ \$\$ (i, j)$.
qed

lemma *index-tensor-mat'*:
assumes $0 < \text{dim-col } A$
and $0 < \text{dim-col } B$
and $i < \text{dim-row } A * \text{dim-row } B$
and $j < \text{dim-col } A * \text{dim-col } B$
shows $(A \otimes B)\ \$\$ (i, j) = A\ \$\$ (i\ \text{div } (\text{dim-row } B), j\ \text{div } (\text{dim-col } B)) * B\ \$\$ (i\ \text{mod } (\text{dim-row } B), j\ \text{mod } (\text{dim-col } B))$
by *(rule index-tensor-mat, (simp add: assms)+)*

lemma *tensor-mat-carrier*:
shows $\text{tensor-mat } U\ V \in \text{carrier-mat } (\text{dim-row } U * \text{dim-row } V)\ (\text{dim-col } U * \text{dim-col } V)$ **by** *auto*

lemma *tensor-mat-id*:
assumes $0 < d1$
and $0 < d2$
shows $\text{tensor-mat } (1_m\ d1)\ (1_m\ d2) = 1_m\ (d1 * d2)$
proof *(rule eq-matI, auto)*
show $\text{tensor-mat } (1_m\ d1)\ (1_m\ d2)\ \$\$ (i, i) = 1$ **if** $i < (d1 * d2)$ **for** i
using *that index-tensor-mat'[of 1_m d1 1_m d2]*
by *(simp add: assms less-mult-imp-div-less)*

next
show $\text{tensor-mat } (1_m \ d1) (1_m \ d2) \ \S\S (i, j) = 0$ **if** $i < d1 * d2$ $j < d1 * d2$ $i \neq j$ **for** $i \ j$
using *that index-tensor-mat[of $1_m \ d1 \ d1 \ d1 \ 1_m \ d2 \ d2 \ d2 \ i \ j$]*
by (*metis* *assms*(1) *assms*(2) *index-one-mat*(1) *index-one-mat*(2) *index-one-mat*(3))

less-mult-imp-div-less mod-less-divisor mult-div-mod-eq mult-not-zero
qed

lemma *tensor-mat-hermitian*:
assumes $A \in \text{carrier-mat } n \ n$
and $B \in \text{carrier-mat } n' \ n'$
and $0 < n$
and $0 < n'$
and *hermitian* A
and *hermitian* B
shows *hermitian* ($\text{tensor-mat } A \ B$) **using** *assms* **by** (*metis* *hermitian-def* *tensor-mat-adjoint*)

lemma *tensor-mat-unitary*:
assumes *Complex-Matrix.unitary* U
and *Complex-Matrix.unitary* V
and $0 < \text{dim-row } U$
and $0 < \text{dim-row } V$
shows *Complex-Matrix.unitary* ($\text{tensor-mat } U \ V$)
proof –
define UI **where** $UI = \text{tensor-mat } U \ V$
have *Complex-Matrix.adjoint* $UI =$
 $\text{tensor-mat } (\text{Complex-Matrix.adjoint } U) (\text{Complex-Matrix.adjoint } V)$ **unfolding**
 $UI\text{-def}$
proof (*rule tensor-mat-adjoint*)
show $U \in \text{carrier-mat } (\text{dim-row } U) (\text{dim-row } U)$ **using** *assms* **unfolding**
 $\text{Complex-Matrix.unitary-def}$
by *simp*
show $V \in \text{carrier-mat } (\text{dim-row } V) (\text{dim-row } V)$ **using** *assms* **unfolding**
 $\text{Complex-Matrix.unitary-def}$
by *simp*
show $0 < \text{dim-row } V$ **using** *assms* **by** *simp*
show $0 < \text{dim-row } U$ **using** *assms* **by** *simp*
show $0 < \text{dim-row } V$ **using** *assms* **by** *simp*
show $0 < \text{dim-row } U$ **using** *assms* **by** *simp*
qed
hence $UI * (\text{Complex-Matrix.adjoint } UI) =$
 $\text{tensor-mat } (U * \text{Complex-Matrix.adjoint } U) (V * \text{Complex-Matrix.adjoint } V)$
using *mult-distr-tensor*[of $U \ \text{Complex-Matrix.adjoint } U \ V \ \text{Complex-Matrix.adjoint } V$]
unfolding $UI\text{-def}$
by (*metis* (*no-types*, *lifting*) *Complex-Matrix.unitary-def* *adjoint-dim-col* *adjoint-dim-row*)

```

    assms carrier-matD(2) )
  also have ... = tensor-mat ( $1_m$  (dim-row  $U$ )) ( $1_m$  (dim-row  $V$ )) using assms
unitary-simps(2)
    by (metis Complex-Matrix.unitary-def)
  also have ... = ( $1_m$  (dim-row  $U * \text{dim-row } V$ )) using tensor-mat-id assms by
simp
  finally have  $UI * (\text{Complex-Matrix.adjoint } UI) = (1_m (\text{dim-row } U * \text{dim-row } V))$  .
  hence inverts-mat  $UI$  (Complex-Matrix.adjoint  $UI$ ) unfolding inverts-mat-def
UI-def by simp
  thus ?thesis using assms unfolding Complex-Matrix.unitary-def UI-def
  by (metis carrier-matD(2)) carrier-matI dim-col-tensor-mat dim-row-tensor-mat)
qed

```

2.4 Fixed carrier matrices locale

We define a locale for matrices with a fixed number of rows and columns, and define a finite sum operation on this locale. The `Type_To_Sets` transfer tools then permits to obtain lemmata on the locale for free.

```

locale fixed-carrier-mat =
  fixes fc-mats::'a::field Matrix.mat set
  fixes dimR dimC
  assumes fc-mats-carrier: fc-mats = carrier-mat dimR dimC
begin

  sublocale semigroup-add-on-with fc-mats (+)
  proof (unfold-locales)
    show  $\bigwedge a b. a \in \text{fc-mats} \implies b \in \text{fc-mats} \implies a + b \in \text{fc-mats}$  using fc-mats-carrier
  by simp
    show  $\bigwedge a b c. a \in \text{fc-mats} \implies b \in \text{fc-mats} \implies c \in \text{fc-mats} \implies a + b + c = a + (b + c)$ 
    using fc-mats-carrier by simp
  qed

  sublocale ab-semigroup-add-on-with fc-mats (+)
  proof (unfold-locales)
    show  $\bigwedge a b. a \in \text{fc-mats} \implies b \in \text{fc-mats} \implies a + b = b + a$  using fc-mats-carrier

    by (simp add: comm-add-mat)
  qed

  sublocale comm-monoid-add-on-with fc-mats (+) 0_m dimR dimC
  proof (unfold-locales)
    show  $0_m \text{ dimR dimC} \in \text{fc-mats}$  using fc-mats-carrier by simp
    show  $\bigwedge a. a \in \text{fc-mats} \implies 0_m \text{ dimR dimC} + a = a$  using fc-mats-carrier by
simp
  qed

  sublocale ab-group-add-on-with fc-mats (+) 0_m dimR dimC (-) uminus

```

proof (*unfold-locale*)
show $\bigwedge a. a \in \text{fc-mats} \implies - a + a = 0_m \text{ dimR dimC}$ **using** *fc-mats-carrier*
by *simp*
show $\bigwedge a b. a \in \text{fc-mats} \implies b \in \text{fc-mats} \implies a - b = a + - b$ **using**
fc-mats-carrier
by (*simp add: add-uminus-minus-mat*)
show $\bigwedge a. a \in \text{fc-mats} \implies - a \in \text{fc-mats}$ **using** *fc-mats-carrier* **by** *simp*
qed
end

lemma (*in fixed-carrier-mat*) *smult-mem*:
assumes $A \in \text{fc-mats}$
shows $a \cdot_m A \in \text{fc-mats}$ **using** *fc-mats-carrier* *assms* **by** *auto*

definition (*in fixed-carrier-mat*) *sum-mat* **where**
 $\text{sum-mat } A \ I = \text{sum-with } (+) \ (0_m \text{ dimR dimC}) \ A \ I$

lemma (*in fixed-carrier-mat*) *sum-mat-empty[simp]*:
shows $\text{sum-mat } A \ \{\} = 0_m \text{ dimR dimC}$ **unfolding** *sum-mat-def* **by** *simp*

lemma (*in fixed-carrier-mat*) *sum-mat-carrier*:
shows $(\bigwedge i. i \in I \implies (A \ i) \in \text{fc-mats}) \implies \text{sum-mat } A \ I \in \text{carrier-mat dimR dimC}$
unfolding *sum-mat-def* **using** *sum-with-mem[of A I]* *fc-mats-carrier* **by** *auto*

lemma (*in fixed-carrier-mat*) *sum-mat-insert*:
assumes $A \ x \in \text{fc-mats}$ $A \ ' I \subseteq \text{fc-mats}$
and A : *finite I* **and** x : $x \notin I$
shows $\text{sum-mat } A \ (\text{insert } x \ I) = A \ x + \text{sum-mat } A \ I$ **unfolding** *sum-mat-def*
using *assms* *sum-with-insert[of A x I]* **by** *simp*

2.5 A locale for square matrices

locale *cpx-sq-mat* = *fixed-carrier-mat fc-mats::complex Matrix.mat set* **for** *fc-mats*
+
assumes *dim-eq: dimR = dimC*
and *npos: 0 < dimR*

lemma (*in cpx-sq-mat*) *one-mem*:
shows $1_m \text{ dimR} \in \text{fc-mats}$ **using** *fc-mats-carrier* *dim-eq* **by** *simp*

lemma (*in cpx-sq-mat*) *square-mats*:
assumes $A \in \text{fc-mats}$
shows *square-mat A* **using** *fc-mats-carrier* *dim-eq* *assms* **by** *simp*

lemma (*in cpx-sq-mat*) *cpx-sq-mat-mult*:
assumes $A \in \text{fc-mats}$
and $B \in \text{fc-mats}$
shows $A * B \in \text{fc-mats}$

proof –

have $\dim\text{-row } (A * B) = \dim R$ **using** *assms fc-mats-carrier* **by** *simp*
moreover have $\dim\text{-col } (A * B) = \dim R$ **using** *assms fc-mats-carrier dim-eq*
by *simp*
ultimately show *?thesis* **using** *fc-mats-carrier carrier-mat-def dim-eq* **by** *auto*
qed

lemma (in *cpx-sq-mat*) *sum-mat-distrib-left*:

shows $\text{finite } I \implies R \in \text{fc-mats} \implies (\bigwedge i. i \in I \implies (A\ i) \in \text{fc-mats}) \implies$
 $\text{sum-mat } (\lambda i. R * (A\ i))\ I = R * (\text{sum-mat } A\ I)$

proof (*induct rule: finite-induct*)

case *empty*

hence *a*: $\text{sum-mat } (\lambda i. R * (A\ i))\ \{\} = 0_m\ \dim R\ \dim C$ **unfolding** *sum-mat-def*
by *simp*

have $\text{sum-mat } A\ \{\} = 0_m\ \dim R\ \dim C$ **unfolding** *sum-mat-def* **by** *simp*

hence $R * (\text{sum-mat } A\ \{\}) = 0_m\ \dim R\ \dim C$ **using** *fc-mats-carrier*
right-mult-zero-mat[of *R dimR dimC dimC*] *empty dim-eq* **by** *simp*

thus *?case* **using** *a* **by** *simp*

next

case (*insert x F*)

hence $\text{sum-mat } (\lambda i. R * A\ i)\ (\text{insert } x\ F) = R * (A\ x) + \text{sum-mat } (\lambda i. R * A\ i)$
i) *F*

using *sum-mat-insert*[of $\lambda i. R * A\ i\ x\ F$] **by** (*simp add: image-subsetI fc-mats-carrier*
dim-eq)

also have $\dots = R * (A\ x) + R * (\text{sum-mat } A\ F)$ **using** *insert* **by** *simp*

also have $\dots = R * (A\ x + (\text{sum-mat } A\ F))$

by (*metis dim-eq fc-mats-carrier insert.prem1 insert.prem2 insertCI*
mult-add-distrib-mat
sum-mat-carrier)

also have $\dots = R * \text{sum-mat } A\ (\text{insert } x\ F)$

proof –

have $A\ x + (\text{sum-mat } A\ F) = \text{sum-mat } A\ (\text{insert } x\ F)$

by (*rule sum-mat-insert[symmetric]*, (*auto simp add: insert*))

thus *?thesis* **by** *simp*

qed

finally show *?case* .

qed

lemma (in *cpx-sq-mat*) *sum-mat-distrib-right*:

shows $\text{finite } I \implies R \in \text{fc-mats} \implies (\bigwedge i. i \in I \implies (A\ i) \in \text{fc-mats}) \implies$
 $\text{sum-mat } (\lambda i. (A\ i) * R)\ I = (\text{sum-mat } A\ I) * R$

proof (*induct rule: finite-induct*)

case *empty*

hence *a*: $\text{sum-mat } (\lambda i. (A\ i) * R)\ \{\} = 0_m\ \dim R\ \dim C$ **unfolding** *sum-mat-def*
by *simp*

have $\text{sum-mat } A\ \{\} = 0_m\ \dim R\ \dim C$ **unfolding** *sum-mat-def* **by** *simp*

hence $(\text{sum-mat } A\ \{\}) * R = 0_m\ \dim R\ \dim C$ **using** *fc-mats-carrier right-mult-zero-mat*[of
R]

dim-eq empty **by** *simp*

```

thus ?case using a by simp
next
  case (insert x F)
  have a: ( $\lambda i. A\ i * R$ ) '  $F \subseteq \text{fc-mats}$  using insert cpx-sq-mat-mult
    by (simp add: image-subsetI)
  have  $A\ x * R \in \text{fc-mats}$  using insert
    by (metis insertI1 local.fc-mats-carrier mult-carrier-mat dim-eq)
  hence  $\text{sum-mat } (\lambda i. A\ i * R) (\text{insert } x\ F) = (A\ x) * R + \text{sum-mat } (\lambda i. A\ i * R) F$  using insert a
    using sum-mat-insert[of  $\lambda i. A\ i * R\ x\ F$ ] by (simp add: image-subsetI local.fc-mats-carrier)
  also have  $\dots = (A\ x) * R + (\text{sum-mat } A\ F) * R$  using insert by simp
  also have  $\dots = (A\ x + (\text{sum-mat } A\ F)) * R$ 
  proof (rule add-mult-distrib-mat[symmetric])
    show  $A\ x \in \text{carrier-mat } \text{dim}R\ \text{dim}C$  using insert fc-mats-carrier by simp
    show  $\text{sum-mat } A\ F \in \text{carrier-mat } \text{dim}R\ \text{dim}C$  using insert fc-mats-carrier sum-mat-carrier by blast
    show  $R \in \text{carrier-mat } \text{dim}C\ \text{dim}C$  using insert fc-mats-carrier dim-eq by simp
  qed
  also have  $\dots = \text{sum-mat } A (\text{insert } x\ F) * R$ 
  proof -
    have  $A\ x + (\text{sum-mat } A\ F) = \text{sum-mat } A (\text{insert } x\ F)$ 
      by (rule sum-mat-insert[symmetric], (auto simp add: insert))
    thus ?thesis by simp
  qed
finally show ?case .
qed

```

```

lemma (in cpx-sq-mat) trace-sum-mat:
  fixes  $A::'b \Rightarrow \text{complex Matrix.mat}$ 
  shows  $\text{finite } I \implies (\bigwedge i. i \in I \implies (A\ i) \in \text{fc-mats}) \implies$ 
     $\text{Complex-Matrix.trace } (\text{sum-mat } A\ I) = (\sum i \in I. \text{Complex-Matrix.trace } (A\ i))$ 
unfolding sum-mat-def
proof (induct rule: finite-induct)
  case empty
  then show ?case using trace-zero dim-eq by simp

```

```

next
  case (insert x F)
  have  $\text{Complex-Matrix.trace } (\text{sum-with } (+) (0_m\ \text{dim}R\ \text{dim}C) A (\text{insert } x\ F)) =$ 
     $\text{Complex-Matrix.trace } (A\ x + \text{sum-with } (+) (0_m\ \text{dim}R\ \text{dim}C) A\ F)$ 
    using sum-with-insert[of  $A\ x\ F$ ] insert by (simp add: image-subset-iff dim-eq)
  also have  $\dots = \text{Complex-Matrix.trace } (A\ x) +$ 
     $\text{Complex-Matrix.trace } (\text{sum-with } (+) (0_m\ \text{dim}R\ \text{dim}C) A\ F)$  using trace-add square-mats insert
    by (metis carrier-matD(1) fc-mats-carrier image-subset-iff insert-iff sum-with-mem)

  also have  $\dots = \text{Complex-Matrix.trace } (A\ x) + (\sum i \in F. \text{Complex-Matrix.trace } (A\ i))$ 
    using insert by simp

```

also have $\dots = (\sum_{i \in (\text{insert } x F)}. \text{Complex-Matrix.trace } (A i))$
using *sum-with-insert*[of $A x F$] *insert* **by** (*simp add: image-subset-iff*)
finally show *?case* .
qed

lemma (in *cpx-sq-mat*) *cpx-sq-mat-smult*:
assumes $A \in \text{fc-mats}$
shows $x \cdot_m A \in \text{fc-mats}$
using *assms fc-mats-carrier* **by** *auto*

lemma (in *cpx-sq-mat*) *mult-add-distrib-right*:
assumes $A \in \text{fc-mats } B \in \text{fc-mats } C \in \text{fc-mats}$
shows $A * (B + C) = A * B + A * C$
using *assms fc-mats-carrier mult-add-distrib-mat dim-eq* **by** *simp*

lemma (in *cpx-sq-mat*) *mult-add-distrib-left*:
assumes $A \in \text{fc-mats } B \in \text{fc-mats } C \in \text{fc-mats}$
shows $(B + C) * A = B * A + C * A$
using *assms fc-mats-carrier add-mult-distrib-mat dim-eq* **by** *simp*

lemma (in *cpx-sq-mat*) *mult-sum-mat-distrib-left*:
shows $\text{finite } I \implies (\bigwedge i. i \in I \implies (A i) \in \text{fc-mats}) \implies B \in \text{fc-mats} \implies$
 $(\text{sum-mat } (\lambda i. B * (A i)) I) = B * (\text{sum-mat } A I)$
proof (*induct rule: finite-induct*)
case *empty*
hence $\text{sum-mat } A \{\} = 0_m \text{ dimR dimC}$ **using** *sum-mat-empty* **by** *simp*
hence $B * (\text{sum-mat } A \{\}) = 0_m \text{ dimR dimC}$ **using** *empty* **by** (*simp add:*
fc-mats-carrier dim-eq)
moreover have $\text{sum-mat } (\lambda i. B * (A i)) \{\} = 0_m \text{ dimR dimC}$ **using** *sum-mat-empty*[of
 $\lambda i. B * (A i)$]
by *simp*
ultimately show *?case* **by** *simp*

next
case (*insert x F*)
have $\text{sum-mat } (\lambda i. B * (A i)) (\text{insert } x F) = B * (A x) + \text{sum-mat } (\lambda i. B * (A$
 $i)) F$
using *sum-with-insert*[of $\lambda i. B * (A i) x F$] *insert*
by (*simp add: image-subset-iff local.sum-mat-def cpx-sq-mat-mult*)
also have $\dots = B * (A x) + B * (\text{sum-mat } A F)$ **using** *insert* **by** *simp*
also have $\dots = B * (A x + (\text{sum-mat } A F))$
proof (*rule mult-add-distrib-right[symmetric]*)
show $B \in \text{fc-mats}$ **using** *insert* **by** *simp*
show $A x \in \text{fc-mats}$ **using** *insert* **by** *simp*
show $\text{sum-mat } A F \in \text{fc-mats}$ **using** *insert* **by** (*simp add: fc-mats-carrier*
sum-mat-carrier)
qed
also have $\dots = B * (\text{sum-mat } A (\text{insert } x F))$ **using** *sum-with-insert*[of $A x F$]
insert
by (*simp add: image-subset-iff sum-mat-def*)

finally show ?case .
qed

lemma (in *cpx-sq-mat*) *mult-sum-mat-distrib-right*:

shows $finite\ I \implies (\bigwedge i. i \in I \implies (A\ i) \in fc\ mats) \implies B \in fc\ mats \implies$
 $(sum\ mat\ (\lambda i. (A\ i) * B)\ I) = (sum\ mat\ A\ I) * B$

proof (induct rule: *finite-induct*)

case *empty*

hence $sum\ mat\ A\ \{\} = 0_m\ dimR\ dimC$ using *sum-mat-empty* by *simp*

hence $(sum\ mat\ A\ \{\}) * B = 0_m\ dimR\ dimC$ using *empty* by (*simp add: fc-mats-carrier dim-eq*)

moreover have $sum\ mat\ (\lambda i. (A\ i) * B)\ \{\} = 0_m\ dimR\ dimC$ by *simp*

ultimately show ?case by *simp*

next

case (*insert x F*)

have $sum\ mat\ (\lambda i. (A\ i) * B)\ (insert\ x\ F) = (A\ x) * B + sum\ mat\ (\lambda i. (A\ i) * B)\ F$

using *sum-with-insert*[of $\lambda i. (A\ i) * B\ x\ F$] *insert*

by (*simp add: image-subset-iff local.sum-mat-def cpx-sq-mat-mult*)

also have $\dots = (A\ x) * B + (sum\ mat\ A\ F) * B$ using *insert* by *simp*

also have $\dots = (A\ x + (sum\ mat\ A\ F)) * B$

proof (rule *mult-add-distrib-left*[*symmetric*])

show $B \in fc\ mats$ using *insert* by *simp*

show $A\ x \in fc\ mats$ using *insert* by *simp*

show $sum\ mat\ A\ F \in fc\ mats$ using *insert* by (*simp add: fc-mats-carrier sum-mat-carrier*)

qed

also have $\dots = (sum\ mat\ A\ (insert\ x\ F)) * B$ using *sum-with-insert*[of $A\ x\ F$] *insert*

by (*simp add: image-subset-iff sum-mat-def*)

finally show ?case .

qed

lemma (in *cpx-sq-mat*) *trace-sum-mat-mat-distrib*:

assumes *finite I*

and $\bigwedge i. i \in I \implies B\ i \in fc\ mats$

and $A \in fc\ mats$

and $C \in fc\ mats$

shows $(\sum i \in I. Complex\ Matrix.trace(A * (B\ i) * C)) =$

$Complex\ Matrix.trace(A * (sum\ mat\ B\ I) * C)$

proof –

have *seq*: $sum\ mat\ (\lambda i. A * (B\ i) * C)\ I = A * (sum\ mat\ B\ I) * C$

proof –

have $sum\ mat\ (\lambda i. A * (B\ i) * C)\ I = (sum\ mat\ (\lambda i. A * (B\ i))\ I) * C$

proof (rule *mult-sum-mat-distrib-right*)

show *finite I* using *assms* by *simp*

show $C \in fc\ mats$ using *assms* by *simp*

show $\bigwedge i. i \in I \implies A * B\ i \in fc\ mats$ using *assms cpx-sq-mat-mult* by *simp*

qed

moreover have $\text{sum-mat } (\lambda i. A * (B i)) I = A * (\text{sum-mat } B I)$
by *(rule mult-sum-mat-distrib-left, (auto simp add: assms))*
ultimately show $\text{sum-mat } (\lambda i. A * (B i) * C) I = A * (\text{sum-mat } B I) * C$
by *simp*
qed
have $(\sum i \in I. \text{Complex-Matrix.trace}(A * (B i) * C)) =$
 $\text{Complex-Matrix.trace } (\text{sum-mat } (\lambda i. A * (B i) * C) I)$
proof *(rule trace-sum-mat[symmetric])*
show *finite I using assms by simp*
fix i
assume $i \in I$
thus $A * B i * C \in \text{fc-mats}$ **using** *assms* **by** *(simp add: cpx-sq-mat-mult)*
qed
also have $\dots = \text{Complex-Matrix.trace } (A * (\text{sum-mat } B I) * C)$ **using** *seq* **by**
simp
finally show *?thesis .*
qed

definition *(in cpx-sq-mat)* **zero-col where**
 $\text{zero-col } U = (\lambda i. \text{if } i < \text{dimR} \text{ then } \text{Matrix.col } U i \text{ else } 0_v \text{ dimR})$

lemma *(in cpx-sq-mat)* **zero-col-dim:**
assumes $U \in \text{fc-mats}$
shows $\text{dim-vec } (\text{zero-col } U i) = \text{dimR}$ **using** *assms* **fc-mats-carrier unfolding**
 zero-col-def **by** *simp*

lemma *(in cpx-sq-mat)* **zero-col-col:**
assumes $i < \text{dimR}$
shows $\text{zero-col } U i = \text{Matrix.col } U i$ **using** *assms* **unfolding** zero-col-def **by**
simp

lemma *(in cpx-sq-mat)* **sum-mat-index:**
shows $\text{finite } I \implies (\bigwedge i. i \in I \implies (A i) \in \text{fc-mats}) \implies i < \text{dimR} \implies j < \text{dimC}$
 \implies
 $(\text{sum-mat } (\lambda k. (A k)) I) \$\$ (i,j) = (\sum k \in I. (A k) \$\$ (i,j))$

proof *(induct rule: finite-induct)*
case *empty*
thus *?case* **unfolding** sum-mat-def **by** *simp*
next
case *(insert x F)*
hence $(\text{sum-mat } (\lambda k. (A k)) (\text{insert } x F)) \$\$ (i,j) =$
 $(A x + (\text{sum-mat } (\lambda k. (A k)) F)) \$\$ (i,j)$ **using** *insert sum-mat-insert[of A]*
by *(simp add: image-subsetI local.fc-mats-carrier)*
also have $\dots = (A x) \$\$ (i,j) + (\text{sum-mat } (\lambda k. (A k)) F) \$\$ (i,j)$ **using** *insert*
 $\text{sum-mat-carrier[of F A]}$ **fc-mats-carrier** **by** *simp*
also have $\dots = (A x) \$\$ (i,j) + (\sum k \in F. (A k) \$\$ (i,j))$ **using** *insert* **by** *simp*
also have $\dots = (\sum k \in (\text{insert } x F). (A k) \$\$ (i,j))$ **using** *insert* **by** *simp*
finally show *?case .*

qed

lemma (in *cpx-sq-mat*) *sum-mat-cong*:

shows $\text{finite } I \implies (\bigwedge i. i \in I \implies A \ i = B \ i) \implies (\bigwedge i. i \in I \implies A \ i \in \text{fc-mats}) \implies$

$(\bigwedge i. i \in I \implies B \ i \in \text{fc-mats}) \implies \text{sum-mat } A \ I = \text{sum-mat } B \ I$

proof (induct rule: *finite-induct*)

case *empty*

then show *?case* **by** *simp*

next

case (*insert x F*)

have $\text{sum-mat } A \ (\text{insert } x \ F) = A \ x + \text{sum-mat } A \ F$ **using** *insert sum-mat-insert[of A]*

by (*simp add: image-subset-iff*)

also have $\dots = B \ x + \text{sum-mat } B \ F$ **using** *insert by simp*

also have $\dots = \text{sum-mat } B \ (\text{insert } x \ F)$ **using** *insert sum-mat-insert[of B]*

by (*simp add: image-subset-iff*)

finally show *?case* .

qed

lemma (in *cpx-sq-mat*) *smult-sum-mat*:

shows $\text{finite } I \implies (\bigwedge i. i \in I \implies A \ i \in \text{fc-mats}) \implies a \cdot_m \text{sum-mat } A \ I =$
 $\text{sum-mat } (\lambda i. a \cdot_m (A \ i)) \ I$

proof (induct rule: *finite-induct*)

case *empty*

then show *?case* **by** *simp*

next

case (*insert x F*)

have $a \cdot_m \text{sum-mat } A \ (\text{insert } x \ F) = a \cdot_m (A \ x + \text{sum-mat } A \ F)$ **using** *insert sum-mat-insert[of A]*

by (*simp add: image-subset-iff*)

also have $\dots = a \cdot_m A \ x + a \cdot_m (\text{sum-mat } A \ F)$ **using** *insert*

by (*metis add-smult-distrib-left-mat fc-mats-carrier insert-iff sum-mat-carrier*)

also have $\dots = a \cdot_m A \ x + \text{sum-mat } (\lambda i. a \cdot_m (A \ i)) \ F$ **using** *insert by simp*

also have $\dots = \text{sum-mat } (\lambda i. a \cdot_m (A \ i)) \ (\text{insert } x \ F)$ **using** *insert*

sum-mat-insert[of (\lambda i. a \cdot_m (A \ i))] **by** (*simp add: image-subset-iff cpx-sq-mat-smult*)

finally show *?case* .

qed

lemma (in *cpx-sq-mat*) *zero-sum-mat*:

shows $\text{finite } I \implies \text{sum-mat } (\lambda i. ((0_m \ \text{dimR} \ \text{dimR})::\text{complex Matrix.mat})) \ I =$
 $((0_m \ \text{dimR} \ \text{dimR})::\text{complex Matrix.mat})$

proof (induct rule: *finite-induct*)

case *empty*

then show *?case* **using** *dim-eq sum-mat-empty* **by** *auto*

next

case (*insert x F*)

have $\text{sum-mat } (\lambda i. ((0_m \ \text{dimR} \ \text{dimR})::\text{complex Matrix.mat})) \ (\text{insert } x \ F) =$
 $0_m \ \text{dimR} \ \text{dimR} + \text{sum-mat } (\lambda i. 0_m \ \text{dimR} \ \text{dimR}) \ F$

using *insert dim-eq zero-mem sum-mat-insert*[of $\lambda i. ((0_m \text{ dimR dimR})::\text{complex Matrix.mat})$]
by *fastforce*
also have ... = $((0_m \text{ dimR dimR})::\text{complex Matrix.mat})$ **using** *insert by auto*
finally show ?case .
qed

lemma (in *cpx-sq-mat*) *sum-mat-adjoint*:
shows $\text{finite } I \implies (\bigwedge i. i \in I \implies A \ i \in \text{fc-mats}) \implies$
 $\text{Complex-Matrix.adjoint } (\text{sum-mat } A \ I) = \text{sum-mat } (\lambda i. \text{Complex-Matrix.adjoint } (A \ i)) \ I$
proof (*induct rule: finite-induct*)
case *empty*
then show ?case **using** *zero-hermitian*[of *dimR*]
by (*metis (no-types) dim-eq hermitian-def sum-mat-empty*)
next
case (*insert x F*)
have $\text{Complex-Matrix.adjoint } (\text{sum-mat } A \ (\text{insert } x \ F)) =$
 $\text{Complex-Matrix.adjoint } (A \ x + \text{sum-mat } A \ F)$ **using** *insert sum-mat-insert*[of
 A]
by (*simp add: image-subset-iff*)
also have ... = $\text{Complex-Matrix.adjoint } (A \ x) + \text{Complex-Matrix.adjoint } (\text{sum-mat } A \ F)$
proof (*rule adjoint-add*)
show $A \ x \in \text{carrier-mat } \text{dimR } \text{dimC}$ **using** *insert fc-mats-carrier* **by** *simp*
show $\text{sum-mat } A \ F \in \text{carrier-mat } \text{dimR } \text{dimC}$ **using** *insert fc-mats-carrier*
sum-mat-carrier[of F]
by *simp*
qed
also have ... = $\text{Complex-Matrix.adjoint } (A \ x) + \text{sum-mat } (\lambda i. \text{Complex-Matrix.adjoint } (A \ i)) \ F$
using *insert by simp*
also have ... = $\text{sum-mat } (\lambda i. \text{Complex-Matrix.adjoint } (A \ i)) \ (\text{insert } x \ F)$
proof (*rule sum-mat-insert*[*symmetric*], (*auto simp add: insert*))
show $\text{Complex-Matrix.adjoint } (A \ x) \in \text{fc-mats}$ **using** *insert fc-mats-carrier*
dim-eq
by (*simp add: adjoint-dim*)
show $\bigwedge i. i \in F \implies \text{Complex-Matrix.adjoint } (A \ i) \in \text{fc-mats}$ **using** *insert*
fc-mats-carrier dim-eq
by (*simp add: adjoint-dim*)
qed
finally show ?case .
qed

lemma (in *cpx-sq-mat*) *sum-mat-hermitian*:
assumes *finite I*
and $\forall i \in I. \text{hermitian } (A \ i)$
and $\forall i \in I. A \ i \in \text{fc-mats}$
shows *hermitian (sum-mat A I)*

proof –
have $\text{Complex-Matrix.adjoint (sum-mat A I) = sum-mat } (\lambda i. \text{Complex-Matrix.adjoint (A i)}) I$
using *assms sum-mat-adjoint[of I]* **by** *simp*
also have $\dots = \text{sum-mat A I}$
proof (*rule sum-mat-cong, (auto simp add: assms)*)
show $\bigwedge i. i \in I \implies \text{Complex-Matrix.adjoint (A i) = A i}$ **using** *assms*
unfolding *hermitian-def* **by** *simp*
show $\bigwedge i. i \in I \implies \text{Complex-Matrix.adjoint (A i)} \in \text{fc-mats}$ **using** *assms*
fc-mats-carrier dim-eq
by (*simp add: adjoint-dim*)
qed
finally show *?thesis* **unfolding** *hermitian-def* .
qed

lemma (*in cpx-sq-mat*) *sum-mat-positive*:
shows $\text{finite I} \implies (\bigwedge i. i \in I \implies \text{Complex-Matrix.positive (A i)}) \implies$
 $(\bigwedge i. i \in I \implies \text{A i} \in \text{fc-mats}) \implies \text{Complex-Matrix.positive (sum-mat A I)}$
proof (*induct rule: finite-induct*)
case empty
then show *?case* **using** *positive-zero[of dimR]* **by** (*metis (no-types) dim-eq sum-mat-empty*)
next
case (insert x F)
hence $\text{sum-mat A (insert x F) = A x + (sum-mat A F)}$ **using** *sum-mat-insert[of A]*
by (*simp add: image-subset-iff*)
moreover have $\text{Complex-Matrix.positive (A x + (sum-mat A F))}$
proof (*rule positive-add, (auto simp add: insert)*)
show $\text{A x} \in \text{carrier-mat dimR dimR}$ **using** *insert fc-mats-carrier dim-eq* **by**
simp
show $\text{sum-mat A F} \in \text{carrier-mat dimR dimR}$ **using** *insert sum-mat-carrier dim-eq*
by (*metis insertCI*)
qed
ultimately show $\text{Complex-Matrix.positive (sum-mat A (insert x F))}$ **by** *simp*
qed

lemma (*in cpx-sq-mat*) *sum-mat-left-ortho-zero*:
shows $\text{finite I} \implies$
 $(\bigwedge i. i \in I \implies \text{A i} \in \text{fc-mats}) \implies (B \in \text{fc-mats}) \implies$
 $(\bigwedge i. i \in I \implies \text{A i} * B = (0_m \text{ dimR dimR})) \implies$
 $(\text{sum-mat A I}) * B = 0_m \text{ dimR dimR}$
proof (*induct rule: finite-induct*)
case empty
then show *?case* **using** *dim-eq*
by (*metis finite.intros(1) sum-mat-empty mult-sum-mat-distrib-right*)
next
case (insert x F)

have $(\text{sum-mat } A (\text{insert } x F)) * B =$
 $(A x + \text{sum-mat } A F) * B$ **using** *insert sum-mat-insert[of A]*
by *(simp add: image-subset-iff)*
also have $\dots = A x * B + \text{sum-mat } A F * B$
proof *(rule add-mult-distrib-mat)*
show $A x \in \text{carrier-mat } \dim R \dim C$ **using** *insert fc-mats-carrier* **by** *simp*
show $\text{sum-mat } A F \in \text{carrier-mat } \dim R \dim C$ **using** *insert*
by *(metis insert-iff local.fc-mats-carrier sum-mat-carrier)*
show $B \in \text{carrier-mat } \dim C \dim R$ **using** *insert fc-mats-carrier dim-eq* **by** *simp*
qed
also have $\dots = A x * B + (0_m \dim R \dim R)$ **using** *insert* **by** *simp*
also have $\dots = 0_m \dim R \dim R$ **using** *insert* **by** *simp*
finally show *?case .*
qed

lemma *(in cpx-sq-mat) sum-mat-right-ortho-zero:*

shows *finite I* \implies
 $(\bigwedge i. i \in I \implies A i \in \text{fc-mats}) \implies (B \in \text{fc-mats}) \implies$
 $(\bigwedge i. i \in I \implies B * A i = (0_m \dim R \dim R)) \implies$
 $B * (\text{sum-mat } A I) = 0_m \dim R \dim R$
proof *(induct rule:finite-induct)*
case *empty*
then show *?case* **using** *dim-eq*
by *(metis finite.intros(1) sum-mat-empty mult-sum-mat-distrib-left)*
next

case *(insert x F)*
have $B * (\text{sum-mat } A (\text{insert } x F)) =$
 $B * (A x + \text{sum-mat } A F)$ **using** *insert sum-mat-insert[of A]*
by *(simp add: image-subset-iff)*
also have $\dots = B * A x + B * \text{sum-mat } A F$
proof *(rule mult-add-distrib-mat)*
show $A x \in \text{carrier-mat } \dim R \dim C$ **using** *insert fc-mats-carrier* **by** *simp*
show $\text{sum-mat } A F \in \text{carrier-mat } \dim R \dim C$ **using** *insert*
by *(metis insert-iff local.fc-mats-carrier sum-mat-carrier)*
show $B \in \text{carrier-mat } \dim C \dim R$ **using** *insert fc-mats-carrier dim-eq* **by** *simp*
qed
also have $\dots = B * A x + (0_m \dim R \dim R)$ **using** *insert* **by** *simp*
also have $\dots = 0_m \dim R \dim R$ **using** *insert* **by** *simp*
finally show *?case .*
qed

lemma *(in cpx-sq-mat) sum-mat-ortho-square:*

shows *finite I* $\implies (\bigwedge i. i \in I \implies ((A i)::\text{complex Matrix.mat}) * (A i) = A i)$
 \implies
 $(\bigwedge i. i \in I \implies A i \in \text{fc-mats}) \implies$
 $(\bigwedge i j. i \in I \implies j \in I \implies i \neq j \implies A i * (A j) = (0_m \dim R \dim R)) \implies$
 $(\text{sum-mat } A I) * (\text{sum-mat } A I) = (\text{sum-mat } A I)$
proof *(induct rule:finite-induct)*
case *empty*

then show *?case* **using** *dim-eq*
by (*metis fc-mats-carrier right-mult-zero-mat sum-mat-empty zero-mem*)
next
case (*insert x F*)
have (*sum-mat A (insert x F)*) * (*sum-mat A (insert x F)*) =
(*A x + sum-mat A F*) * (*A x + sum-mat A F*) **using** *insert sum-mat-insert[of A]*
by (*simp add: ‹ $\bigwedge i. i \in \text{insert } x F \implies A i * A i = A i$ › image-subset-iff*)
also have ... = *A x * (A x + sum-mat A F) + sum-mat A F * (A x + sum-mat A F)*
proof (*rule add-mult-distrib-mat*)
show *A x* \in *carrier-mat dimR dimC* **using** *insert fc-mats-carrier* **by** *simp*
show *sum-mat A F* \in *carrier-mat dimR dimC* **using** *insert*
by (*metis insert-iff local.fc-mats-carrier sum-mat-carrier*)
thus *A x + sum-mat A F* \in *carrier-mat dimC dimC* **using** *insert dim-eq* **by**
simp
qed
also have ... = *A x * A x + A x * (sum-mat A F) + sum-mat A F * (A x + sum-mat A F)*
proof –
have *A x * (A x + sum-mat A F)* = *A x * A x + A x * (sum-mat A F)*
using *dim-eq insert.premis(2) mult-add-distrib-right sum-mat-carrier*
by (*metis fc-mats-carrier insertI1 subsetD subset-insertI*)
thus *?thesis* **by** *simp*
qed
also have ... = *A x * A x + A x * (sum-mat A F) + sum-mat A F * A x + sum-mat A F * (sum-mat A F)*
proof –
have *sum-mat A F * (A x + local.sum-mat A F)* =
*sum-mat A F * A x + local.sum-mat A F * local.sum-mat A F*
using *insert dim-eq add-assoc add-mem mult-add-distrib-right cpx-sq-mat-mult sum-mat-carrier*
by (*metis fc-mats-carrier insertI1 subsetD subset-insertI*)
hence *A x * A x + A x * sum-mat A F + sum-mat A F * (A x + sum-mat A F)* =
*A x * A x + A x * sum-mat A F + (sum-mat A F * A x + sum-mat A F * sum-mat A F)* **by** *simp*
also have ... = *A x * A x + A x * sum-mat A F + sum-mat A F * A x + sum-mat A F * sum-mat A F*
proof (*rule assoc-add-mat[symmetric]*)
show *A x * A x + A x * sum-mat A F* \in *carrier-mat dimR dimR* **using**
sum-mat-carrier insert
dim-eq fc-mats-carrier **by** (*metis add-mem cpx-sq-mat-mult insertCI*)
show *sum-mat A F * A x* \in *carrier-mat dimR dimR* **using** *sum-mat-carrier insert*
dim-eq fc-mats-carrier **by** (*metis cpx-sq-mat-mult insertCI*)
show *sum-mat A F * sum-mat A F* \in *carrier-mat dimR dimR* **using**
sum-mat-carrier insert
dim-eq fc-mats-carrier **by** (*metis cpx-sq-mat-mult insertCI*)

qed
finally show ?thesis .
qed
also have ... = A x + sum-mat A F
proof -
have A x * A x = A x using insert by simp
moreover have sum-mat A F * sum-mat A F = sum-mat A F using insert
by simp
moreover have A x * (sum-mat A F) = 0_m dimR dimR
proof -
have A x * (sum-mat A F) = sum-mat (λi. A x * (A i)) F
by (rule sum-mat-distrib-left[symmetric], (simp add: insert)+)
also have ... = sum-mat (λi. 0_m dimR dimR) F
proof (rule sum-mat-cong, (auto simp add: insert zero-mem))
show $\bigwedge i. i \in F \implies A x * A i = 0_m \dimR \dimR$ using insert by auto
show $\bigwedge i. i \in F \implies A x * A i \in \text{fc-mats}$ using insert cpx-sq-mat-mult by
auto
show $\bigwedge i. i \in F \implies 0_m \dimR \dimR \in \text{fc-mats}$ using zero-mem dim-eq by
simp
qed
also have ... = 0_m dimR dimR using zero-sum-mat insert by simp
finally show ?thesis .
qed
moreover have sum-mat A F * A x = 0_m dimR dimR
proof -
have sum-mat A F * A x = sum-mat (λi. A i * (A x)) F
by (rule sum-mat-distrib-right[symmetric], (simp add: insert)+)
also have ... = sum-mat (λi. 0_m dimR dimR) F
proof (rule sum-mat-cong, (auto simp add: insert zero-mem))
show $\bigwedge i. i \in F \implies A i * A x = 0_m \dimR \dimR$ using insert by auto
show $\bigwedge i. i \in F \implies A i * A x \in \text{fc-mats}$ using insert cpx-sq-mat-mult by
auto
show $\bigwedge i. i \in F \implies 0_m \dimR \dimR \in \text{fc-mats}$ using zero-mem dim-eq by
simp
qed
also have ... = 0_m dimR dimR using zero-sum-mat insert by simp
finally show ?thesis .
qed
ultimately show ?thesis using add-commute add-zero insert.premis(2) zero-mem
dim-eq by auto
qed
also have ... = sum-mat A (insert x F) using insert sum-mat-insert[of A x F]
by (simp add: $\langle \bigwedge i. i \in \text{insert } x F \implies A i * A i = A i \rangle \text{image-subsetI}$)
finally show ?case .
qed

lemma diagonal-unit-vec:

assumes B ∈ carrier-mat n n
and diagonal-mat (B::complex Matrix.mat)

shows $B *_v (\text{unit-vec } n \ i) = B \ \$\$ (i,i) \cdot_v (\text{unit-vec } n \ i)$
proof –
define $v :: \text{complex Matrix.vec}$ **where** $v = \text{unit-vec } n \ i$
have $B *_v v = \text{Matrix.vec } n \ (\lambda \ i. \text{Matrix.scalar-prod } (\text{Matrix.row } B \ i) \ v)$
using *assms unfolding mult-mat-vec-def by simp*
also have $\dots = \text{Matrix.vec } n \ (\lambda \ i. B \ \$\$ (i,i) * \text{Matrix.vec-index } v \ i)$
proof –
have $\forall i < n. (\text{Matrix.scalar-prod } (\text{Matrix.row } B \ i) \ v = B \ \$\$ (i,i) * \text{Matrix.vec-index } v \ i)$
proof (*intro allI impI*)
fix i
assume $i < n$
have $(\text{Matrix.scalar-prod } (\text{Matrix.row } B \ i) \ v) =$
 $(\sum j \in \{0 \dots n\}. \text{Matrix.vec-index } (\text{Matrix.row } B \ i) \ j * \text{Matrix.vec-index } v \ j)$ **using** *assms*
unfolding *Matrix.scalar-prod-def v-def by simp*
also have $\dots = \text{Matrix.vec-index } (\text{Matrix.row } B \ i) \ i * \text{Matrix.vec-index } v \ i$
proof (*rule sum-but-one*)
show $\forall j < n. j \neq i \longrightarrow \text{Matrix.vec-index } (\text{Matrix.row } B \ i) \ j = 0$
proof (*intro allI impI*)
fix j
assume $j < n$ **and** $j \neq i$
hence $\text{Matrix.vec-index } (\text{Matrix.row } B \ i) \ j = B \ \$\$ (i,j)$ **using** $\langle i < n \rangle \langle j < n \rangle$ *assms*
by *auto*
also have $\dots = 0$ **using** *assms* $\langle i < n \rangle \langle j < n \rangle \langle j \neq i \rangle$ **unfolding** *diagonal-mat-def by simp*
finally show $\text{Matrix.vec-index } (\text{Matrix.row } B \ i) \ j = 0$.
qed
show $i < n$ **using** $\langle i < n \rangle$.
qed
also have $\dots = B \ \$\$ (i,i) * \text{Matrix.vec-index } v \ i$ **using** *assms* $\langle i < n \rangle$ **by** *auto*
finally show $(\text{Matrix.scalar-prod } (\text{Matrix.row } B \ i) \ v) = B \ \$\$ (i,i) * \text{Matrix.vec-index } v \ i$.
qed
thus *?thesis by auto*
qed
also have $\dots = B \ \$\$ (i,i) \cdot_v v$ **unfolding** *v-def unit-vec-def by auto*
finally have $B *_v v = B \ \$\$ (i,i) \cdot_v v$.
thus *?thesis unfolding v-def by simp*
qed

lemma *mat-vec-mult-assoc:*

assumes $A \in \text{carrier-mat } n \ p$
and $B \in \text{carrier-mat } p \ q$
and $\text{dim-vec } v = q$
shows $A *_v (B *_v v) = (A * B) *_v v$ **using** *assms by auto*

lemma (*in cpx-sq-mat*) *similar-eigenvectors:*

assumes $A \in \text{fc-mats}$
and $B \in \text{fc-mats}$
and $P \in \text{fc-mats}$
and $\text{similar-mat-wit } A \ B \ P \ (\text{Complex-Matrix.adjoint } P)$
and $\text{diagonal-mat } B$
and $i < n$
shows $A *_v (P *_v (\text{unit-vec } \dim R \ i)) = B \ \$\$ (i,i) \cdot_v (P *_v (\text{unit-vec } \dim R \ i))$
proof –
have $A *_v (P *_v (\text{unit-vec } \dim R \ i)) =$
 $(P * B * (\text{Complex-Matrix.adjoint } P)) *_v (P *_v (\text{unit-vec } \dim R \ i))$
using $\text{assms unfolding similar-mat-wit-def by metis}$
also have $\dots = P * B * (\text{Complex-Matrix.adjoint } P) * P *_v (\text{unit-vec } \dim R \ i)$
proof ($\text{rule mat-vec-mult-assoc[of- } \dim R \ \dim R], (\text{auto simp add: assms fc-mats-carrier})$)
show $P \in \text{carrier-mat } \dim R \ \dim R$ **using** $\text{assms fc-mats-carrier dim-eq by simp}$
show $P * B * \text{Complex-Matrix.adjoint } P \in \text{carrier-mat } \dim R \ \dim R$
using $\text{assms fc-mats-carrier by auto}$
qed
also have $\dots = P * B * ((\text{Complex-Matrix.adjoint } P) * P) *_v (\text{unit-vec } \dim R \ i)$
using assms dim-eq
by ($\text{smt (verit) fc-mats-carrier mat-assoc-test(1) similar-mat-witD2(6) similar-mat-wit-sym}$)
also have $\dots = P * B *_v (\text{unit-vec } \dim R \ i)$
proof –
have $(\text{Complex-Matrix.adjoint } P) * P = 1_m \ \dim R$ **using** $\text{assms dim-eq unfolding similar-mat-wit-def}$
by ($\text{simp add: fc-mats-carrier}$)
thus $?thesis$ **using** $\text{assms(2) local.fc-mats-carrier dim-eq by auto}$
qed
also have $\dots = P *_v (B *_v (\text{unit-vec } \dim R \ i))$ **using** $\text{mat-vec-mult-assoc assms fc-mats-carrier}$
 dim-eq by simp
also have $\dots = P *_v (B \ \$\$ (i,i) \cdot_v (\text{unit-vec } \dim R \ i))$ **using** $\text{assms diagonal-unit-vec}$
 $\text{fc-mats-carrier dim-eq by simp}$
also have $\dots = B \ \$\$ (i,i) \cdot_v (P *_v (\text{unit-vec } \dim R \ i))$
proof (rule mult-mat-vec)
show $P \in \text{carrier-mat } \dim R \ \dim C$ **using** $\text{assms fc-mats-carrier by simp}$
show $\text{unit-vec } \dim R \ i \in \text{carrier-vec } \dim C$ **using** dim-eq by simp
qed
finally show $?thesis$.
qed

2.6 Projectors

definition projector **where**
 $\text{projector } M \iff (\text{hermitian } M \wedge M * M = M)$

lemma $\text{projector-hermitian}$:
assumes $\text{projector } M$

```

shows hermitian  $M$  using assms unfolding projector-def by simp

lemma zero-projector[simp]:
  shows projector  $(0_m\ n\ n)$  unfolding projector-def
proof
  show hermitian  $(0_m\ n\ n)$  using zero-hermitian[of n] by simp
  show  $0_m\ n\ n * 0_m\ n\ n = 0_m\ n\ n$  by simp
qed

lemma projector-square-eq:
  assumes projector  $M$ 
  shows  $M * M = M$  using assms unfolding projector-def by simp

lemma projector-positive:
  assumes projector  $M$ 
  shows Complex-Matrix.positive  $M$ 
proof (rule positive-if-decomp)
  show  $M \in \text{carrier-mat } (\text{dim-row } M) (\text{dim-row } M)$  using assms projector-hermitian
hermitian-square
    by auto
next
  have  $M = \text{Complex-Matrix.adjoint } M$  using assms projector-hermitian[of M]
    unfolding hermitian-def by simp
  hence  $M * \text{Complex-Matrix.adjoint } M = M * M$  by simp
  also have  $\dots = M$  using assms projector-square-eq by auto
  finally have  $M * \text{Complex-Matrix.adjoint } M = M$  .
  thus  $\exists Ma. Ma * \text{Complex-Matrix.adjoint } Ma = M$  by auto
qed

lemma projector-collapse-trace:
  assumes projector  $(P::\text{complex Matrix.mat})$ 
  and  $P \in \text{carrier-mat } n\ n$ 
  and  $R \in \text{carrier-mat } n\ n$ 
shows Complex-Matrix.trace  $(P * R * P) = \text{Complex-Matrix.trace } (R * P)$ 
proof –
  have Complex-Matrix.trace  $(R * P) = \text{Complex-Matrix.trace } (P * R)$  using
trace-comm assms by auto
  also have  $\dots = \text{Complex-Matrix.trace } ((P * P) * R)$  using assms projector-square-eq[of
P] by simp
  also have  $\dots = \text{Complex-Matrix.trace } (P * (P * R))$  using assms by auto
  also have  $\dots = \text{Complex-Matrix.trace } (P * R * P)$  using trace-comm[of P n P
* R] assms by auto
  finally have Complex-Matrix.trace  $(R * P) = \text{Complex-Matrix.trace } (P * R * P)$  .
  thus ?thesis by simp
qed

lemma positive-proj-trace:
  assumes projector  $(P::\text{complex Matrix.mat})$ 

```

```

    and Complex-Matrix.positive R
    and P ∈ carrier-mat n n
    and R ∈ carrier-mat n n
shows Complex-Matrix.trace (R * P) ≥ 0
proof -
  have Complex-Matrix.trace (R * P) = Complex-Matrix.trace ((P * R) * P)
    using assms projector-collapse-trace by auto
  also have ... = Complex-Matrix.trace ((P * R) * (Complex-Matrix.adjoint P))
    using assms projector-hermitian[of P]
    unfolding hermitian-def by simp
  also have ... ≥ 0
  proof (rule positive-trace)
    show P * R * Complex-Matrix.adjoint P ∈ carrier-mat n n using assms by
    auto
    show Complex-Matrix.positive (P * R * Complex-Matrix.adjoint P)
      by (rule positive-close-under-left-right-mult-adjoint[of - n], (auto simp add:
      assms))
    qed
    finally show ?thesis .
  qed
qed

```

```

lemma trace-proj-pos-real:
  assumes projector (P::complex Matrix.mat)
  and Complex-Matrix.positive R
  and P ∈ carrier-mat n n
  and R ∈ carrier-mat n n
shows Re (Complex-Matrix.trace (R * P)) = Complex-Matrix.trace (R * P)
proof -
  have Complex-Matrix.trace (R * P) ≥ 0 using assms positive-proj-trace by
  simp
  thus ?thesis by (simp add: complex-eqI less-eq-complex-def)
qed

```

```

lemma (in cpx-sq-mat) trace-sum-mat-proj-pos-real:
  fixes f::'a ⇒ real
  assumes finite I
  and ∀ i ∈ I. projector (P i)
  and Complex-Matrix.positive R
  and ∀ i ∈ I. P i ∈ fc-mats
  and R ∈ fc-mats
shows Complex-Matrix.trace (R * (sum-mat (λi. f i ·m (P i)) I)) =
  Re (Complex-Matrix.trace (R * (sum-mat (λi. f i ·m (P i)) I)))
proof -
  have sm: ⋀x. x ∈ I ⇒ Complex-Matrix.trace (f x ·m (R * P x)) =
  f x * Complex-Matrix.trace (R * P x)
  proof -
    fix i
    assume i ∈ I
    show Complex-Matrix.trace (f i ·m (R * P i)) = f i * Complex-Matrix.trace (R

```

```

* P i)
  proof (rule trace-smult)
    show R * P i ∈ carrier-mat dimR dimR using assms cpx-sq-mat-mult
fc-mats-carrier ⟨i ∈ I⟩
    dim-eq by simp
  qed
  qed
  have sw:  $\bigwedge x. x \in I \implies R * (f x \cdot_m P x) = f x \cdot_m (R * P x)$ 
  proof -
    fix i
    assume i ∈ I
    show R * (f i \cdot_m P i) = f i \cdot_m (R * P i)
    proof (rule mult-smult-distrib)
      show R ∈ carrier-mat dimR dimR using assms fc-mats-carrier dim-eq by
simp
      show P i ∈ carrier-mat dimR dimR using assms ⟨i ∈ I⟩ fc-mats-carrier dim-eq
by simp
    qed
  qed
  have dr:  $Complex-Matrix.trace (R * (sum-mat (\lambda i. f i \cdot_m (P i)) I)) =$ 
 $Complex-Matrix.trace (sum-mat (\lambda i. (R * (f i \cdot_m (P i)))) I)$ 
    using sum-mat-distrib-left[of I] assms by (simp add: cpx-sq-mat-smult)
  also have trs:  $\dots = (\sum i \in I. Complex-Matrix.trace (R * (f i \cdot_m (P i))))$ 
  proof (rule trace-sum-mat, (simp add: assms))
    show  $\bigwedge i. i \in I \implies R * (f i \cdot_m P i) \in fc-mats$  using assms
    by (simp add: cpx-sq-mat-smult cpx-sq-mat-mult)
  qed
  also have  $\dots = (\sum i \in I. Complex-Matrix.trace (f i \cdot_m (R * (P i))))$ 
    by (rule sum.cong, (simp add: sw)+)
  also have  $\dots = (\sum i \in I. f i * Complex-Matrix.trace (R * (P i)))$ 
    by (rule sum.cong, (simp add: sm)+)
  also have  $\dots = (\sum i \in I. complex-of-real (f i * Re (Complex-Matrix.trace (R *$ 
(P i))))))
  proof (rule sum.cong, simp)
    show  $\bigwedge x. x \in I \implies$ 
 $complex-of-real (f x) * Complex-Matrix.trace (R * P x) =$ 
 $complex-of-real (f x * Re (Complex-Matrix.trace (R * P x)))$ 
    proof -
      fix x
      assume x ∈ I
      have  $complex-of-real (f x) * Complex-Matrix.trace (R * P x) =$ 
 $complex-of-real (f x) * complex-of-real (Re (Complex-Matrix.trace (R * P$ 
x)))
      using assms sum.cong[of I I] fc-mats-carrier trace-proj-pos-real ⟨x ∈ I⟩
dim-eq by auto
      also have  $\dots = complex-of-real (f x * Re (Complex-Matrix.trace (R * P x)))$ 
by simp
    finally show  $complex-of-real (f x) * Complex-Matrix.trace (R * P x) =$ 
 $complex-of-real (f x * Re (Complex-Matrix.trace (R * P x))) .$ 

```

qed
qed
also have ... = $(\sum_{i \in I} f i * \text{Re} (\text{Complex-Matrix.trace} (R * (P i))))$ **by** *simp*
also have ... = $(\sum_{i \in I} \text{Re} (\text{Complex-Matrix.trace} (f i \cdot_m (R * (P i)))))$
proof –
have $(\sum_{i \in I} f i * \text{Re} (\text{Complex-Matrix.trace} (R * (P i)))) =$
 $(\sum_{i \in I} \text{Re} (\text{Complex-Matrix.trace} (f i \cdot_m (R * (P i)))))$
by (*rule sum.cong, (simp add: sm)+*)
thus ?thesis **by** *simp*
qed
also have ... = $(\sum_{i \in I} \text{Re} (\text{Complex-Matrix.trace} (R * (f i \cdot_m (P i)))))$
proof –
have $\bigwedge i. i \in I \implies f i \cdot_m (R * (P i)) = R * (f i \cdot_m (P i))$ **using** *sw* **by** *simp*
thus ?thesis **by** *simp*
qed
also have ... = $\text{Re} (\sum_{i \in I} (\text{Complex-Matrix.trace} (R * (f i \cdot_m (P i)))))$ **by**
simp
also have ... = $\text{Re} (\text{Complex-Matrix.trace} (\text{sum-mat} (\lambda i. R * (f i \cdot_m (P i))) I))$
using *trs* **by** *simp*
also have ... = $\text{Re} (\text{Complex-Matrix.trace} (R * (\text{sum-mat} (\lambda i. f i \cdot_m (P i))) I))$
using *dr* **by** *simp*
finally show ?thesis .
qed

2.7 Rank 1 projection

definition *rank-1-proj* **where**
rank-1-proj $v = \text{outer-prod } v \ v$

lemma *rank-1-proj-square-mat*:
shows *square-mat* (*rank-1-proj* v) **using** *outer-prod-dim* **unfolding** *rank-1-proj-def*
by (*metis carrier-matD(1) carrier-matD(2) carrier-vec-dim-vec square-mat.simps*)

lemma *rank-1-proj-dim*[*simp*]:
shows *dim-row* (*rank-1-proj* v) = *dim-vec* v **using** *outer-prod-dim* **unfolding**
rank-1-proj-def
using *carrier-vecI* **by** *blast*

lemma *rank-1-proj-carrier*[*simp*]:
shows *rank-1-proj* $v \in \text{carrier-mat} (\text{dim-vec } v) (\text{dim-vec } v)$ **using** *outer-prod-dim*
unfolding *rank-1-proj-def* **using** *carrier-vecI* **by** *blast*

lemma *rank-1-proj-coord*:
assumes $i < \text{dim-vec } v$
and $j < \text{dim-vec } v$
shows (*rank-1-proj* v) \$\$ (i, j) = \text{Matrix.vec-index } v \ i * (\text{cnj} (\text{Matrix.vec-index } v \ j)) **using** *assms*
unfolding *rank-1-proj-def* *outer-prod-def* **by** *auto*

lemma *rank-1-proj-adjoint*:
shows $\text{Complex-Matrix.adjoint} (\text{rank-1-proj } (v::\text{complex Matrix.vec})) = \text{rank-1-proj } v$
proof
show $\text{dim-row} (\text{Complex-Matrix.adjoint} (\text{rank-1-proj } v)) = \text{dim-row} (\text{rank-1-proj } v)$
using *rank-1-proj-square-mat* **by** *auto*
thus $\text{dim-col} (\text{Complex-Matrix.adjoint} (\text{rank-1-proj } v)) = \text{dim-col} (\text{rank-1-proj } v)$
by *auto*
fix $i\ j$
assume $i < \text{dim-row} (\text{rank-1-proj } v)$ **and** $j < \text{dim-col} (\text{rank-1-proj } v)$ **note** $ij = \text{this}$
have $\text{Complex-Matrix.adjoint} (\text{rank-1-proj } v) \ \$\$ (i, j) = \text{conjugate} ((\text{rank-1-proj } v) \ \$\$ (j, i))$
using $ij \langle \text{dim-col} (\text{Complex-Matrix.adjoint} (\text{rank-1-proj } v)) = \text{dim-col} (\text{rank-1-proj } v) \rangle$
adjoint-eval **by** *fastforce*
also have $\dots = \text{conjugate} (\text{Matrix.vec-index } v\ j * (\text{cnj} (\text{Matrix.vec-index } v\ i)))$
using *rank-1-proj-coord*
by $(\text{metis} \langle \text{dim-col} (\text{Complex-Matrix.adjoint} (\text{rank-1-proj } v)) = \text{dim-col} (\text{rank-1-proj } v) \rangle$
adjoint-dim-col rank-1-proj-dim)
also have $\dots = \text{Matrix.vec-index } v\ i * (\text{cnj} (\text{Matrix.vec-index } v\ j))$ **by** *simp*
also have $\dots = \text{rank-1-proj } v \ \$\$ (i, j)$ **using** ij *rank-1-proj-coord*
by $(\text{metis} \langle \text{dim-col} (\text{Complex-Matrix.adjoint} (\text{rank-1-proj } v)) = \text{dim-col} (\text{rank-1-proj } v) \rangle$
adjoint-dim-col rank-1-proj-dim)
finally show $\text{Complex-Matrix.adjoint} (\text{rank-1-proj } v) \ \$\$ (i, j) = \text{rank-1-proj } v \ \$\$ (i, j)$.
qed

lemma *rank-1-proj-hermitian*:
shows $\text{hermitian} (\text{rank-1-proj } (v::\text{complex Matrix.vec}))$ **using** *rank-1-proj-adjoint*

unfolding *hermitian-def* **by** *simp*

lemma *rank-1-proj-trace*:
assumes $\|v\| = 1$
shows $\text{Complex-Matrix.trace} (\text{rank-1-proj } v) = 1$
proof –
have $\text{Complex-Matrix.trace} (\text{rank-1-proj } v) = \text{inner-prod } v\ v$ **using** *trace-outer-prod*

unfolding *rank-1-proj-def* **using** *carrier-vecI* **by** *blast*
also have $\dots = (\text{vec-norm } v)^2$ **unfolding** *vec-norm-def* **using** *power2-csqrt* **by** *presburger*
also have $\dots = \|v\|^2$ **using** *vec-norm-sq-cpx-vec-length-sq* **by** *simp*
also have $\dots = 1$ **using** *assms* **by** *simp*
finally show *?thesis* .

qed

lemma *rank-1-proj-mat-col*:

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

and $i < n$

and $j < n$

and $k < n$

shows $(\text{rank-1-proj } (\text{Matrix.col } A \ i)) \ \$\$ \ (j, k) = A \ \$\$ \ (j, i) * \text{conjugate } (A \ \$\$ \ (k, i))$

proof –

have $(\text{rank-1-proj } (\text{Matrix.col } A \ i)) \ \$\$ \ (j, k) = \text{Matrix.vec-index } (\text{Matrix.col } A \ i) \ j *$

$\text{conjugate } (\text{Matrix.vec-index } (\text{Matrix.col } A \ i) \ k)$ **using** *index-outer-prod*[of *Matrix.col A i n Matrix.col A i n*]

assms unfolding rank-1-proj-def by simp

also have $\dots = A \ \$\$ \ (j, i) * \text{conjugate } (A \ \$\$ \ (k, i))$ **using** *assms by simp*

finally show *?thesis* .

qed

lemma (in *cpx-sq-mat*) *weighted-sum-rank-1-proj-unitary-index*:

assumes $A \in \text{fc-mats}$

and $B \in \text{fc-mats}$

and *diagonal-mat B*

and *Complex-Matrix.unitary A*

and $j < \text{dim}R$

and $k < \text{dim}R$

shows $(\text{sum-mat } (\lambda i. (\text{diag-mat } B)!i \cdot_m \text{rank-1-proj } (\text{Matrix.col } A \ i)) \ \{.. < \text{dim}R\}) \ \$\$ \ (j, k) =$

$(A * B * (\text{Complex-Matrix.adjoint } A)) \ \$\$ \ (j, k)$

proof –

have $(\text{sum-mat } (\lambda i. (\text{diag-mat } B)!i \cdot_m \text{rank-1-proj } (\text{Matrix.col } A \ i)) \ \{.. < \text{dim}R\}) \ \$\$ \ (j, k) =$

$(\sum_{i \in \{.. < \text{dim}R\}} ((\text{diag-mat } B)!i \cdot_m \text{rank-1-proj } (\text{Matrix.col } A \ i)) \ \$\$ \ (j, k))$

proof (*rule sum-mat-index, (auto simp add: fc-mats-carrier assms)*)

show $\bigwedge i. i < \text{dim}R \implies (\text{diag-mat } B)!i \cdot_m \text{rank-1-proj } (\text{Matrix.col } A \ i) \in \text{carrier-mat } \text{dim}R \ \text{dim}C$

using *rank-1-proj-carrier assms fc-mats-carrier dim-eq*

by (*metis smult-carrier-mat zero-col-col zero-col-dim*)

show $k < \text{dim}C$ **using** *assms dim-eq by simp*

qed

also have $\dots = (\sum_{i \in \{.. < \text{dim}R\}} (\text{diag-mat } B)!i * A \ \$\$ \ (j, i) * \text{conjugate } (A \ \$\$ \ (k, i)))$

proof (*rule sum.cong, simp*)

have $\text{idx: } \bigwedge x. x \in \{.. < \text{dim}R\} \implies (\text{rank-1-proj } (\text{Matrix.col } A \ x)) \ \$\$ \ (j, k) =$

$A \ \$\$ \ (j, x) * \text{conjugate } (A \ \$\$ \ (k, x))$ **using** *rank-1-proj-mat-col assms fc-mats-carrier dim-eq*

by *blast*

show $\bigwedge x. x \in \{.. < \text{dim}R\} \implies ((\text{diag-mat } B)!x \cdot_m \text{rank-1-proj } (\text{Matrix.col } A \ x)) \ \$\$ \ (j, k) =$

```

    (diag-mat B)!x * A $$ (j, x) * conjugate (A $$ (k, x))
  proof -
    fix x
    assume x ∈ {..m rank-1-proj (Matrix.col A x)) $$ (j, k) =
      (diag-mat B)!x * (rank-1-proj (Matrix.col A x)) $$ (j, k)
    proof (rule index-smult-mat)
      show j < dim-row (rank-1-proj (Matrix.col A x)) using ⟨x ∈ {..m rank-1-proj (Matrix.col A x)) $$ (j, k) =
      (diag-mat B)!x * A $$ (j, x) * conjugate (A $$ (k, x)) using idx ⟨x ∈ {..<
dimR}⟩ by simp
    qed
  qed
  also have ... = Matrix.scalar-prod (Matrix.col (Complex-Matrix.adjoint A) k)
(Matrix.row (A*B) j)
    unfolding Matrix.scalar-prod-def
  proof (rule sum.cong)
    show {..

```

also have ... = $(A * B * (\text{Complex-Matrix.adjoint } A)) \text{ \$\$ } (j, k)$
proof –
have $\text{Matrix.mat } (\text{dim-row } (A * B)) (\text{dim-col } (\text{Complex-Matrix.adjoint } A))$
 $(\lambda(i, j). \text{Matrix.scalar-prod } (\text{Matrix.row } (A * B) i) (\text{Matrix.col } (\text{Complex-Matrix.adjoint } A) j)) \text{ \$\$}$
 $(j, k) = \text{Matrix.scalar-prod } (\text{Matrix.row } (A * B) j) (\text{Matrix.col } (\text{Complex-Matrix.adjoint } A) k)$
using *assms fc-mats-carrier by simp*
also have ... = $\text{Matrix.scalar-prod } (\text{Matrix.col } (\text{Complex-Matrix.adjoint } A) k)$
 $(\text{Matrix.row } (A * B) j)$
using *assms comm-scalar-prod[of Matrix.row (A*B) j dimR] fc-mats-carrier dim-eq*
by (*metis adjoint-dim Matrix.col-carrier-vec row-carrier-vec cpx-sq-mat-mult*)

thus *?thesis using assms fc-mats-carrier unfolding times-mat-def by simp*
qed
finally show *?thesis .*
qed

lemma (in *cpx-sq-mat*) *weighted-sum-rank-1-proj-unitary*:

assumes $A \in \text{fc-mats}$
and $B \in \text{fc-mats}$
and *diagonal-mat B*
and *Complex-Matrix.unitary A*
shows $(\text{sum-mat } (\lambda i. (\text{diag-mat } B)!i \cdot_m \text{rank-1-proj } (\text{Matrix.col } A i)) \{.. < \text{dimR}\})$
 $=$
 $(A * B * (\text{Complex-Matrix.adjoint } A))$
proof
show $\text{dim-row } (\text{sum-mat } (\lambda i. \text{diag-mat } B ! i \cdot_m \text{rank-1-proj } (\text{Matrix.col } A i)) \{.. < \text{dimR}\}) =$
 $\text{dim-row } (A * B * \text{Complex-Matrix.adjoint } A)$ **using** *assms fc-mats-carrier dim-eq*
by (*metis (no-types, lifting) carrier-matD(1) cpx-sq-mat-smult dim-col index-mult-mat(2)*)
 $\text{rank-1-proj-carrier sum-mat-carrier}$
show $\text{dim-col } (\text{local.sum-mat } (\lambda i. \text{diag-mat } B ! i \cdot_m \text{rank-1-proj } (\text{Matrix.col } A i)) \{.. < \text{dimR}\}) =$
 $\text{dim-col } (A * B * \text{Complex-Matrix.adjoint } A)$ **using** *assms fc-mats-carrier dim-eq*
by (*metis (mono-tags, lifting) adjoint-dim-col carrier-matD(1) carrier-matI dim-col index-mult-mat(3) index-smult-mat(2) index-smult-mat(3) rank-1-proj-dim rank-1-proj-square-mat square-mat.elims(2) square-mats sum-mat-carrier*)
show $\bigwedge i j. i < \text{dim-row } (A * B * \text{Complex-Matrix.adjoint } A) \implies$
 $j < \text{dim-col } (A * B * \text{Complex-Matrix.adjoint } A) \implies$
 $\text{local.sum-mat } (\lambda i. \text{diag-mat } B ! i \cdot_m \text{rank-1-proj } (\text{Matrix.col } A i)) \{.. < \text{dimR}\} \text{ \$\$ } (i, j) =$
 $(A * B * \text{Complex-Matrix.adjoint } A) \text{ \$\$ } (i, j)$
using *weighted-sum-rank-1-proj-unitary-index[of A B] dim-eq fc-mats-carrier*
using *assms by auto*

qed

lemma *rank-1-proj-projector*:

assumes $\|v\| = 1$

shows *projector* (*rank-1-proj v*)

proof –

have *Complex-Matrix.adjoint* (*rank-1-proj v*) = *rank-1-proj v* **using** *rank-1-proj-adjoint*
by *simp*

hence *a*: *hermitian* (*rank-1-proj v*) **unfolding** *hermitian-def* **by** *simp*

have *rank-1-proj v* * *rank-1-proj v* = *inner-prod v v* \cdot_m *outer-prod v v* **unfolding**
rank-1-proj-def

using *outer-prod-mult-outer-prod assms* **using** *carrier-vec-dim-vec* **by** *blast*

also have ... = (*vec-norm v*)² \cdot_m *outer-prod v v* **unfolding** *vec-norm-def* **using**
power2-csqrt

by *presburger*

also have ... = (*cpx-vec-length v*)² \cdot_m *outer-prod v v* **using** *vec-norm-sq-cpx-vec-length-sq*
by *simp*

also have ... = *outer-prod v v* **using** *assms state-qbit-norm-sq smult-one*[*of*
outer-prod v v]

by *simp*

also have ... = *rank-1-proj v* **unfolding** *rank-1-proj-def* **by** *simp*

finally show *?thesis* **using** *a* **unfolding** *projector-def* **by** *simp*

qed

lemma *rank-1-proj-positive*:

assumes $\|v\| = 1$

shows *Complex-Matrix.positive* (*rank-1-proj v*)

proof –

have *projector* (*rank-1-proj v*) **using** *assms rank-1-proj-projector* **by** *simp*

thus *?thesis* **using** *projector-positive* **by** *simp*

qed

lemma *rank-1-proj-density*:

assumes $\|v\| = 1$

shows *density-operator* (*rank-1-proj v*) **unfolding** *density-operator-def* **using**
assms

by (*simp add: rank-1-proj-positive rank-1-proj-trace*)

lemma (**in** *cpx-sq-mat*) *sum-rank-1-proj-unitary-index*:

assumes $A \in \text{fc-mats}$

and *Complex-Matrix.unitary* A

and $j < \text{dim}R$

and $k < \text{dim}R$

shows (*sum-mat* ($\lambda i.$ *rank-1-proj* (*Matrix.col A i*)) $\{.. < \text{dim}R\}$) $\$\$ (j,k) = (1_m$
 $\text{dim}R) \$\$ (j,k)$

proof –

have (*sum-mat* ($\lambda i.$ *rank-1-proj* (*Matrix.col A i*)) $\{.. < \text{dim}R\}$) $\$\$ (j,k) =$
 $(\sum_{i \in \{.. < \text{dim}R\}.}$ (*rank-1-proj* (*Matrix.col A i*)) $\$\$ (j,k))$

proof (*rule sum-mat-index, (auto simp add: fc-mats-carrier assms)*)

```

show  $\bigwedge i. i < \dim R \implies \text{rank-1-proj } (\text{Matrix.col } A \ i) \in \text{carrier-mat } \dim R \ \dim C$ 

proof –
  fix  $i$ 
  assume  $i < \dim R$ 
  hence  $\dim\text{-vec } (\text{Matrix.col } A \ i) = \dim R$  using assms fc-mats-carrier by simp
  thus  $\text{rank-1-proj } (\text{Matrix.col } A \ i) \in \text{carrier-mat } \dim R \ \dim C$  using rank-1-proj-carrier
assms
    fc-mats-carrier dim-eq fc-mats-carrier by (metis dim-col fc-mats-carrier)
  qed
  show  $k < \dim C$  using assms dim-eq by simp
  qed
  also have  $\dots = (\sum i \in \{.. < \dim R\}. A \ \$\$ (j, i) * \text{conjugate } (A \ \$\$ (k, i)))$ 
  proof (rule sum.cong, simp)
    show  $\bigwedge x. x \in \{.. < \dim R\} \implies \text{rank-1-proj } (\text{Matrix.col } A \ x) \ \$\$ (j, k) =$ 
       $A \ \$\$ (j, x) * \text{conjugate } (A \ \$\$ (k, x))$ 
    using rank-1-proj-mat-col assms using local.fc-mats-carrier dim-eq by blast
  qed
  also have  $\dots = \text{Matrix.scalar-prod } (\text{Matrix.col } (\text{Complex-Matrix.adjoint } A) \ k)$ 
  (Matrix.row } A \ j)
    unfolding Matrix.scalar-prod-def
  proof (rule sum.cong)
    show  $\{.. < \dim R\} = \{0.. < \dim\text{-vec } (\text{Matrix.row } A \ j)\}$ 
    using assms fc-mats-carrier dim-eq by auto
    show  $\bigwedge x. x \in \{0.. < \dim\text{-vec } (\text{Matrix.row } A \ j)\} \implies$ 
       $A \ \$\$ (j, x) * \text{conjugate } (A \ \$\$ (k, x)) =$ 
       $\text{Matrix.vec-index } ((\text{Matrix.col } (\text{Complex-Matrix.adjoint } A) \ k)) \ x * \text{Matrix.vec-index}$ 
  (Matrix.row } A \ j)  $x$ 
    proof –
      fix  $x$ 
      assume  $x \in \{0.. < \dim\text{-vec } (\text{Matrix.row } A \ j)\}$ 
      hence  $x \in \{0.. < \dim R\}$  using assms fc-mats-carrier dim-eq by simp
      hence  $A \ \$\$ (j, x) = \text{Matrix.vec-index } (\text{Matrix.row } A \ j) \ x$  using  $\langle x \in$ 
   $\{0.. < \dim R\} \rangle$ 
      assms fc-mats-carrier dim-eq by simp
      moreover have  $\text{conjugate } (A \ \$\$ (k, x)) =$ 
       $\text{Matrix.vec-index } ((\text{Matrix.col } (\text{Complex-Matrix.adjoint } A) \ k)) \ x$  using  $\langle x \in$ 
   $\{0.. < \dim R\} \rangle$ 
      assms fc-mats-carrier dim-eq by (simp add: adjoint-col)
      ultimately show  $A \ \$\$ (j, x) * \text{conjugate } (A \ \$\$ (k, x)) =$ 
       $\text{Matrix.vec-index } ((\text{Matrix.col } (\text{Complex-Matrix.adjoint } A) \ k)) \ x *$ 
       $\text{Matrix.vec-index } (\text{Matrix.row } A \ j) \ x$  by simp
    qed
  qed
  also have  $\dots = (A * (\text{Complex-Matrix.adjoint } A)) \ \$\$ (j, k)$ 
  proof –
    have  $\text{Matrix.mat } (\dim\text{-row } A) \ (\dim\text{-col } (\text{Complex-Matrix.adjoint } A))$ 
    ( $\lambda(i, j). \text{Matrix.scalar-prod } (\text{Matrix.row } A \ i) \ (\text{Matrix.col } (\text{Complex-Matrix.adjoint}$ 
   $A) \ j)) \ \$\$$ 

```

$(j, k) = \text{Matrix.scalar-prod} (\text{Matrix.row } A \ j) (\text{Matrix.col} (\text{Complex-Matrix.adjoint } A) \ k)$
using *assms fc-mats-carrier* **by** *simp*
also have $\dots = \text{Matrix.scalar-prod} (\text{Matrix.col} (\text{Complex-Matrix.adjoint } A) \ k)$
 $(\text{Matrix.row } A \ j)$
using *assms comm-scalar-prod[of Matrix.row A j dimR] fc-mats-carrier*
by *(simp add: adjoint-dim dim-eq)*
thus *?thesis* **using** *assms fc-mats-carrier unfolding times-mat-def* **by** *simp*
qed
also have $\dots = (1_m \ \text{dim}R) \ \S\(j,k) **using** *assms dim-eq* **by** *(simp add: fc-mats-carrier)*
finally show *?thesis* .
qed

lemma (in *cpx-sq-mat*) *rank-1-proj-sum-density*:

assumes *finite I*
and $\forall i \in I. \|u \ i\| = 1$
and $\forall i \in I. \text{dim-vec} (u \ i) = \text{dim}R$
and $\forall i \in I. 0 \leq p \ i$
and $(\sum i \in I. p \ i) = 1$
shows *density-operator* $(\text{sum-mat} (\lambda i. p \ i \cdot_m (\text{rank-1-proj} (u \ i))) \ I)$ **unfolding**
density-operator-def
proof
have *Complex-Matrix.trace* $(\text{sum-mat} (\lambda i. p \ i \cdot_m \text{rank-1-proj} (u \ i)) \ I) =$
 $(\sum i \in I. \text{Complex-Matrix.trace} (p \ i \cdot_m \text{rank-1-proj} (u \ i)))$
proof *(rule trace-sum-mat, (simp add: assms))*
show $\bigwedge i. i \in I \implies p \ i \cdot_m \text{rank-1-proj} (u \ i) \in \text{fc-mats}$ **using** *assms smult-mem*
fc-mats-carrier
dim-eq **by** *auto*
qed
also have $\dots = (\sum i \in I. p \ i * \text{Complex-Matrix.trace} (\text{rank-1-proj} (u \ i)))$
proof –
{
fix *i*
assume $i \in I$
hence *Complex-Matrix.trace* $(p \ i \cdot_m \text{rank-1-proj} (u \ i)) =$
 $p \ i * \text{Complex-Matrix.trace} (\text{rank-1-proj} (u \ i))$
using *trace-smult[of rank-1-proj (u i) dimR] assms fc-mats-carrier dim-eq*
by *auto*
}
thus *?thesis* **by** *simp*
qed
also have $\dots = 1$ **using** *assms rank-1-proj-trace assms* **by** *auto*
finally show *Complex-Matrix.trace* $(\text{sum-mat} (\lambda i. p \ i \cdot_m \text{rank-1-proj} (u \ i)) \ I)$
 $= 1$.
next
show *Complex-Matrix.positive* $(\text{sum-mat} (\lambda i. p \ i \cdot_m \text{rank-1-proj} (u \ i)) \ I)$
proof *(rule sum-mat-positive, (simp add: assms))*
fix *i*
assume $i \in I$

thus $p \ i \cdot_m \text{rank-1-proj} (u \ i) \in \text{fc-mats}$ **using** *assms smult-mem fc-mats-carrier dim-eq* **by** *auto*
show *Complex-Matrix.positive* ($p \ i \cdot_m \text{rank-1-proj} (u \ i)$) **using** *assms* $\langle i \in I \rangle$
rank-1-proj-positive positive-smult[of rank-1-proj (u i) dimR] *dim-eq* **by** *auto*
qed
qed

lemma (in *cpx-sq-mat*) *sum-rank-1-proj-unitary*:

assumes $A \in \text{fc-mats}$
and *Complex-Matrix.unitary* A
shows ($\text{sum-mat} (\lambda i. \text{rank-1-proj} (\text{Matrix.col } A \ i)) \{.. < \text{dimR}\}$) = ($1_m \ \text{dimR}$)
proof
show $\text{dim-row} (\text{sum-mat} (\lambda i. \text{rank-1-proj} (\text{Matrix.col } A \ i)) \{.. < \text{dimR}\}) = \text{dim-row} (1_m \ \text{dimR})$
using *assms fc-mats-carrier*
by (*metis carrier-matD(1) dim-col dim-eq index-one-mat(2) rank-1-proj-carrier sum-mat-carrier*)
show $\text{dim-col} (\text{sum-mat} (\lambda i. \text{rank-1-proj} (\text{Matrix.col } A \ i)) \{.. < \text{dimR}\}) = \text{dim-col} (1_m \ \text{dimR})$
using *assms fc-mats-carrier rank-1-proj-carrier sum-mat-carrier*
by (*metis carrier-matD(2) dim-col dim-eq index-one-mat(3) square-mat.simps square-mats*)
show $\bigwedge i \ j. \ i < \text{dim-row} (1_m \ \text{dimR}) \implies$
 $j < \text{dim-col} (1_m \ \text{dimR}) \implies \text{sum-mat} (\lambda i. \text{rank-1-proj} (\text{Matrix.col } A \ i))$
 $\{.. < \text{dimR}\} \ \S\S (i, j) =$
 $1_m \ \text{dimR} \ \S\S (i, j)$
using *assms sum-rank-1-proj-unitary-index* **by** *simp*
qed

lemma (in *cpx-sq-mat*) *rank-1-proj-unitary*:

assumes $A \in \text{fc-mats}$
and *Complex-Matrix.unitary* A
and $j < \text{dimR}$
and $k < \text{dimR}$
shows $\text{rank-1-proj} (\text{Matrix.col } A \ j) * (\text{rank-1-proj} (\text{Matrix.col } A \ k)) =$
 $(1_m \ \text{dimR}) \ \S\S (j, k) \cdot_m (\text{outer-prod} (\text{Matrix.col } A \ j) (\text{Matrix.col } A \ k))$
proof –
have $\text{rank-1-proj} (\text{Matrix.col } A \ j) * (\text{rank-1-proj} (\text{Matrix.col } A \ k)) =$
 $\text{Complex-Matrix.inner-prod} (\text{Matrix.col } A \ j) (\text{Matrix.col } A \ k) \cdot_m \text{outer-prod}$
 $(\text{Matrix.col } A \ j) (\text{Matrix.col } A \ k)$
using *outer-prod-mult-outer-prod assms Matrix.col-dim local.fc-mats-carrier*
unfolding *rank-1-proj-def*
by *blast*
also have $\dots = (\text{Complex-Matrix.adjoint } A * A) \ \S\S (j, k) \cdot_m (\text{outer-prod} (\text{Matrix.col}$
 $A \ j) (\text{Matrix.col } A \ k))$
using *inner-prod-adjoint-comp[of A dimR A] assms fc-mats-carrier dim-eq* **by**
simp

also have ... = $(1_m \text{ dimR}) \text{ \$(\$ (j,k) \cdot_m (outer-prod (Matrix.col A j) (Matrix.col A k))}$ **using** *assms*
unfolding *Complex-Matrix.unitary-def*
by (*metis add-commute assms(2) index-add-mat(2) index-one-mat(2) one-mem unitary-simps(1)*)
finally show *?thesis* .
qed

lemma (in *cpx-sq-mat*) *rank-1-proj-unitary-ne*:

assumes $A \in \text{fc-mats}$
and *Complex-Matrix.unitary A*
and $j < \text{dimR}$
and $k < \text{dimR}$
and $j \neq k$
shows $\text{rank-1-proj (Matrix.col A j)} * (\text{rank-1-proj (Matrix.col A k)}) = (0_m \text{ dimR dimR})$
proof –
have $\text{dim-row } (0 \cdot_m \text{ outer-prod (Matrix.col A j) (Matrix.col A k)}) = \text{dim-row (outer-prod (Matrix.col A j) (Matrix.col A k))}$
by *simp*
also have ... = dimR **using** *assms fc-mats-carrier dim-eq* **by** *auto*
finally have *rw*: $\text{dim-row } (0 \cdot_m \text{ outer-prod (Matrix.col A j) (Matrix.col A k)}) = \text{dimR}$.
have $\text{dim-col } (0 \cdot_m \text{ outer-prod (Matrix.col A j) (Matrix.col A k)}) = \text{dim-col (outer-prod (Matrix.col A j) (Matrix.col A k))}$
by *simp*
also have ... = dimR **using** *assms fc-mats-carrier dim-eq* **by** *auto*
finally have *cl*: $\text{dim-col } (0 \cdot_m \text{ outer-prod (Matrix.col A j) (Matrix.col A k)}) = \text{dimR}$.
have $\text{rank-1-proj (Matrix.col A j)} * (\text{rank-1-proj (Matrix.col A k)}) = (0::\text{complex}) \cdot_m (\text{outer-prod (Matrix.col A j) (Matrix.col A k)})$
using *assms rank-1-proj-unitary* **by** *simp*
also have ... = (0_m dimR dimR)
proof
show $\text{dim-row } (0 \cdot_m \text{ outer-prod (Matrix.col A j) (Matrix.col A k)}) = \text{dim-row } (0_m \text{ dimR dimR})$ **using** *rw* **by** *simp*
next
show $\text{dim-col } (0 \cdot_m \text{ outer-prod (Matrix.col A j) (Matrix.col A k)}) = \text{dim-col } (0_m \text{ dimR dimR})$ **using** *cl* **by** *simp*
next
show $\bigwedge i p. i < \text{dim-row } (0_m \text{ dimR dimR}) \implies p < \text{dim-col } (0_m \text{ dimR dimR}) \implies (0 \cdot_m \text{ outer-prod (Matrix.col A j) (Matrix.col A k)}) \text{ \$(\$ (i, p) = } 0_m \text{ dimR dimR} \text{ \$(\$ (i, p) using } rw \text{ cl by auto}$
qed
finally show *?thesis* .
qed

lemma (in *cpx-sq-mat*) *rank-1-proj-unitary-eq*:

```

    assumes  $A \in fc\text{-}mats$ 
  and  $Complex\text{-}Matrix.unitary\ A$ 
  and  $j < dimR$ 
  shows  $rank\text{-}1\text{-}proj\ (Matrix.col\ A\ j) * (rank\text{-}1\text{-}proj\ (Matrix.col\ A\ j)) = rank\text{-}1\text{-}proj\ (Matrix.col\ A\ j)$ 
  proof -
    have  $rank\text{-}1\text{-}proj\ (Matrix.col\ A\ j) * (rank\text{-}1\text{-}proj\ (Matrix.col\ A\ j)) = (1::complex)$ 
    ·m  $(rank\text{-}1\text{-}proj\ (Matrix.col\ A\ j))$ 
      using assms rank-1-proj-unitary unfolding rank-1-proj-def by simp
      also have  $\dots = (rank\text{-}1\text{-}proj\ (Matrix.col\ A\ j))$  by (simp add: smult-one)
      finally show ?thesis .
  qed

```

end

```

theory Projective-Measurements imports
  Linear-Algebra-Complements

```

begin

3 Projective measurements

In this part we define projective measurements, also referred to as von Neumann measurements. The latter are characterized by a set of orthogonal projectors, which are used to compute the probabilities of measurement outcomes and to represent the state of the system after the measurement.

The state of the system (a density operator in this case) after a measurement is represented by the `density_collapse` function.

3.1 First definitions

We begin by defining a type synonym for couples of measurement values (reals) and the associated projectors (complex matrices).

```

type-synonym measure-outcome = real × complex Matrix.mat

```

The corresponding values and projectors are retrieved thanks to `meas_outcome_val` and `meas_outcome_prj`.

```

definition meas-outcome-val::measure-outcome ⇒ real where
  meas-outcome-val  $Mi = fst\ Mi$ 

```

```

definition meas-outcome-prj::measure-outcome ⇒ complex Matrix.mat where
  meas-outcome-prj  $Mi = snd\ Mi$ 

```

We define a predicate for projective measurements. A projective measurement is characterized by the number p of possible measure outcomes, and a function M mapping outcome i to the corresponding `measure_outcome`.

definition (in `cpx-sq-mat`) `proj-measurement::nat ⇒ (nat ⇒ measure_outcome) ⇒ bool` **where**

```

proj-measurement n M ←→
  (inj-on (λi. meas-outcome-val (M i)) {.. $n$ }) ∧
  (∀ j < n. meas-outcome-prj (M j) ∈ fc-mats ∧
   projector (meas-outcome-prj (M j))) ∧
  (∀ i < n. ∀ j < n. i ≠ j →
   meas-outcome-prj (M i) * meas-outcome-prj (M j) = 0m dimR dimR) ∧
  sum-mat (λj. meas-outcome-prj (M j)) {.. $n$ } = 1m dimR

```

lemma (in `cpx-sq-mat`) `proj-measurement-inj`:
assumes `proj-measurement p M`
shows `inj-on (λi. meas-outcome-val (M i)) {.. p }` **using** `assms`
unfolding `proj-measurement-def` **by** `simp`

lemma (in `cpx-sq-mat`) `proj-measurement-carrier`:
assumes `proj-measurement p M`
and `i < p`
shows `meas-outcome-prj (M i) ∈ fc-mats` **using** `assms`
unfolding `proj-measurement-def` **by** `simp`

lemma (in `cpx-sq-mat`) `proj-measurement-ortho`:
assumes `proj-measurement p M`
and `i < p`
and `j < p`
and `i ≠ j`
shows `meas-outcome-prj (M i) * meas-outcome-prj (M j) = 0m dimR dimR` **using** `assms`
unfolding `proj-measurement-def` **by** `simp`

lemma (in `cpx-sq-mat`) `proj-measurement-id`:
assumes `proj-measurement p M`
shows `sum-mat (λj. meas-outcome-prj (M j)) {.. p } = 1m dimR` **using** `assms`
unfolding `proj-measurement-def` **by** `simp`

lemma (in `cpx-sq-mat`) `proj-measurement-square`:
assumes `proj-measurement p M`
and `i < p`
shows `meas-outcome-prj (M i) ∈ fc-mats` **using** `assms` **unfolding** `proj-measurement-def`
by `simp`

lemma (in `cpx-sq-mat`) `proj-measurement-proj`:
assumes `proj-measurement p M`
and `i < p`
shows `projector (meas-outcome-prj (M i))` **using** `assms` **unfolding** `proj-measurement-def`
by `simp`

We define the probability of obtaining a measurement outcome: this is a positive number and the sum over all the measurement outcomes is 1.

definition (in *cpx-sq-mat*) *meas-outcome-prob* :: *complex Matrix.mat* \Rightarrow
*(nat \Rightarrow real \times complex Matrix.mat) \Rightarrow nat \Rightarrow complex **where**
meas-outcome-prob *R M i* = *Complex-Matrix.trace* (*R** (*meas-outcome-prj* (*M i*)))*

lemma (in *cpx-sq-mat*) *meas-outcome-prob-real*:

assumes *R* \in *fc-mats*

and *density-operator* *R*

and *proj-measurement* *n M*

and *i* < *n*

shows *meas-outcome-prob* *R M i* \in \mathbb{R}

proof –

have *complex-of-real* (*Re* (*Complex-Matrix.trace* (*R** *meas-outcome-prj* (*M i*))))

=

Complex-Matrix.trace (*R** *meas-outcome-prj* (*M i*))

proof (*rule trace-proj-pos-real*)

show *R* \in *carrier-mat* *dimR dimR* **using** *assms fc-mats-carrier dim-eq* **by** *simp*

show *Complex-Matrix.positive* *R* **using** *assms unfolding density-operator-def*

by *simp*

show *projector* (*meas-outcome-prj* (*M i*)) **using** *assms proj-measurement-proj*

by *simp*

show *meas-outcome-prj* (*M i*) \in *carrier-mat* *dimR dimR* **using** *assms proj-measurement-carrier fc-mats-carrier dim-eq* **by** *simp*

qed

thus *?thesis* **unfolding** *meas-outcome-prob-def* **by** (*metis Reals-of-real*)

qed

lemma (in *cpx-sq-mat*) *meas-outcome-prob-pos*:

assumes *R* \in *fc-mats*

and *density-operator* *R*

and *proj-measurement* *n M*

and *i* < *n*

shows $0 \leq$ *meas-outcome-prob* *R M i* **unfolding** *meas-outcome-prob-def*

proof (*rule positive-proj-trace*)

show *R* \in *carrier-mat* *dimR dimR* **using** *assms fc-mats-carrier dim-eq* **by** *simp*

show *Complex-Matrix.positive* *R* **using** *assms unfolding density-operator-def*

by *simp*

show *projector* (*meas-outcome-prj* (*M i*)) **using** *assms proj-measurement-proj*

by *simp*

show *meas-outcome-prj* (*M i*) \in *carrier-mat* *dimR dimR* **using** *assms proj-measurement-carrier fc-mats-carrier dim-eq* **by** *simp*

qed

lemma (in *cpx-sq-mat*) *meas-outcome-prob-sum*:

assumes *density-operator* *R*

and *R* \in *fc-mats*

and *proj-measurement* *n M*

shows $(\sum j \in \{..< n\}. \text{meas-outcome-prob } R M j) = 1$

```

proof –
  have ( $\sum j \in \{..< n\}$ . Complex-Matrix.trace ( $R * (\text{meas-outcome-prj } (M j))$ )) =
    Complex-Matrix.trace (sum-mat ( $\lambda j$ .  $R * (\text{meas-outcome-prj } (M j))$ )  $\{..< n\}$ )
  proof (rule trace-sum-mat[symmetric], auto)
    fix  $j$ 
    assume  $j < n$ 
    thus  $R * \text{meas-outcome-prj } (M j) \in \text{fc-mats}$  using cpx-sq-mat-mult assms
      unfolding proj-measurement-def by simp
    qed
  also have ... = Complex-Matrix.trace ( $R * (\text{sum-mat } (\lambda j$ . (meas-outcome-prj ( $M j$ ))  $\{..< n\}$ )))
  proof –
    have sum-mat ( $\lambda j$ .  $R * (\text{meas-outcome-prj } (M j))$ )  $\{..< n\}$  =
       $R * (\text{sum-mat } (\lambda j$ . (meas-outcome-prj ( $M j$ ))  $\{..< n\}$ )
    proof (rule mult-sum-mat-distrib-left, (auto simp add: assms))
      fix  $j$ 
      assume  $j < n$ 
      thus meas-outcome-prj ( $M j$ )  $\in \text{fc-mats}$  using assms unfolding proj-measurement-def
    by simp
    qed
    thus ?thesis by simp
  qed
  also have ... = Complex-Matrix.trace ( $R * 1_m \text{ dim}R$ ) using assms unfolding
proj-measurement-def
  by simp
  also have ... = Complex-Matrix.trace  $R$  using assms by (simp add: fc-mats-carrier
dim-eq)
  also have ... = 1 using assms unfolding density-operator-def by simp
  finally show ?thesis unfolding meas-outcome-prob-def .
qed

```

We introduce the maximally mixed density operator. Intuitively, this density operator corresponds to a uniform distribution of the states of an orthonormal basis. This operator will be used to define the density operator after a measurement for the measure outcome probabilities equal to zero.

definition *max-mix-density* :: *nat* \Rightarrow *complex Matrix.mat* **where**
max-mix-density $n = ((1::\text{real})/n) \cdot_m (1_m n)$

lemma *max-mix-density-carrier*:

shows *max-mix-density* $n \in \text{carrier-mat } n n$ **unfolding** *max-mix-density-def* **by** *simp*

lemma *max-mix-is-density*:

assumes $0 < n$

shows *density-operator* (*max-mix-density* n) **unfolding** *density-operator-def* *max-mix-density-def*

proof

have *Complex-Matrix.trace* (*complex-of-real* $((1::\text{real})/n) \cdot_m 1_m n$) =
complex-of-real $((1::\text{real})/n) * (\text{Complex-Matrix.trace } ((1_m n)::\text{complex Matrix.mat}))$

using *one-carrier-mat trace-smult*[of $(1_m \ n)::\text{complex Matrix.mat}$] **by** *blast*
also have $\dots = (\text{complex-of-real } ((1::\text{real})/n)) * (\text{real } n)$ **using** *trace-1*[of n]
by *simp*
also have $\dots = 1$ **using** *assms* **by** *simp*
finally show $\text{Complex-Matrix.trace } (\text{complex-of-real } ((1::\text{real})/n) \cdot_m 1_m \ n) =$
 1 .
next
show $\text{Complex-Matrix.positive } (\text{complex-of-real } (1 / \text{real } n) \cdot_m 1_m \ n)$
by (*rule positive-smult*, (*auto simp add: positive-one less-eq-complex-def*))
qed

lemma (*in cpx-sq-mat*) *max-mix-density-square*:
shows $\text{max-mix-density } \text{dim}R \in \text{fc-mats}$ **unfolding** *max-mix-density-def*
using *fc-mats-carrier dim-eq* **by** *simp*

Given a measurement outcome, **density_collapse** represents the resulting density operator. In practice only the measure outcomes with nonzero probabilities are of interest; we (arbitrarily) collapse the density operator for zero-probability outcomes to the maximally mixed density operator.

definition *density-collapse* $::\text{complex Matrix.mat} \Rightarrow \text{complex Matrix.mat} \Rightarrow \text{complex Matrix.mat}$ **where**
density-collapse $R \ P = (\text{if } ((\text{Complex-Matrix.trace } (R * P)) = 0) \text{ then } (\text{max-mix-density } (\text{dim-row } R))$
 $\text{else } ((1::\text{real}) / ((\text{Complex-Matrix.trace } (R * P)))) \cdot_m (P * R * P))$

lemma *density-collapse-carrier*:
assumes $0 < \text{dim-row } R$
and $P \in \text{carrier-mat } n \ n$
and $R \in \text{carrier-mat } n \ n$
shows $(\text{density-collapse } R \ P) \in \text{carrier-mat } n \ n$
proof (*cases* ($\text{Complex-Matrix.trace } (R * P) = 0$)
case *True*
hence $\text{density-collapse } R \ P = \text{max-mix-density } (\text{dim-row } R)$ **unfolding** *density-collapse-def* **by** *simp*
then show *?thesis* **using** *max-mix-is-density assms max-mix-density-carrier* **by** *auto*
next
case *False*
hence $\text{density-collapse } R \ P = \text{complex-of-real } 1 / \text{Complex-Matrix.trace } (R * P)$
 $\cdot_m (P * R * P)$
unfolding *density-collapse-def* **by** *simp*
thus *?thesis* **using** *assms* **by** *auto*
qed

lemma *density-collapse-operator*:
assumes *projector* P
and *density-operator* R
and $0 < \text{dim-row } R$
and $P \in \text{carrier-mat } n \ n$

and $R \in \text{carrier-mat } n \ n$
shows $\text{density-operator } (\text{density-collapse } R \ P)$
proof $(\text{cases } (\text{Complex-Matrix.trace } (R * P)) = 0)$
 case True
 hence $\text{density-collapse } R \ P = \text{max-mix-density } (\text{dim-row } R)$ **unfolding** $\text{density-collapse-def}$ **by** simp
 then show $?thesis$ **using** $\text{max-mix-is-density assms}$ **by** simp
 next
 case False
 show $?thesis$ **unfolding** $\text{density-operator-def}$
 proof (intro conjI)
 have $\text{Complex-Matrix.positive } ((1 / (\text{Complex-Matrix.trace } (R * P))) \cdot_m (P * R * P))$
 proof $(\text{rule positive-smult})$
 show $P * R * P \in \text{carrier-mat } n \ n$ **using** assms **by** simp
 have $\text{Complex-Matrix.positive } R$ **using** assms **unfolding** $\text{density-operator-def}$
by simp
 hence $0 \leq (\text{Complex-Matrix.trace } (R * P))$ **using** $\text{positive-proj-trace[of } P \ R \ n] \text{ assms}$
 False **by** auto
 hence $0 \leq \text{Re } (\text{Complex-Matrix.trace } (R * P))$ **by** $(\text{simp add: less-eq-complex-def})$
 hence $0 \leq 1 / (\text{Re } (\text{Complex-Matrix.trace } (R * P)))$ **by** simp
 have $\text{Re } (\text{Complex-Matrix.trace } (R * P)) = \text{Complex-Matrix.trace } (R * P)$
 using $\text{assms } \langle \text{Complex-Matrix.positive } R \rangle \text{ trace-proj-pos-real}$ **by** simp
 hence $\text{inv: } 1 / (\text{Complex-Matrix.trace } (R * P)) = 1 / (\text{Re } (\text{Complex-Matrix.trace } (R * P)))$ **by** simp
 thus $0 \leq 1 / (\text{Complex-Matrix.trace } (R * P))$
 using $\langle 0 \leq 1 / (\text{Re } (\text{Complex-Matrix.trace } (R * P))) \rangle$ **by** $(\text{simp add: inv less-eq-complex-def})$
 show $\text{Complex-Matrix.positive } (P * R * P)$ **using** assms
 **positive-close-under-left-right-mult-adjoint[of } P \ n \ R]
 by $(\text{simp add: } \langle \text{Complex-Matrix.positive } R \rangle \text{ hermitian-def projector-def})$
 qed
 thus $\text{Complex-Matrix.positive } (\text{density-collapse } R \ P)$ **using** False
 unfolding $\text{density-collapse-def}$ **by** simp
 next
 have $\text{Complex-Matrix.trace } (\text{density-collapse } R \ P) =$
 $\text{Complex-Matrix.trace } ((1 / (\text{Complex-Matrix.trace } (R * P))) \cdot_m (P * R * P))$
 using False **unfolding** $\text{density-collapse-def}$ **by** simp
 also have $\dots = 1 / (\text{Complex-Matrix.trace } (R * P)) * \text{Complex-Matrix.trace } (P * R * P)$
 using $\text{trace-smult[of } P * R * P \ n] \text{ assms}$ **by** simp
 also have $\dots = 1 / (\text{Complex-Matrix.trace } (R * P)) * \text{Complex-Matrix.trace } (R * P)$
 using $\text{projector-collapse-trace assms}$ **by** simp
 also have $\dots = 1$ **using** False **by** simp
 finally show $\text{Complex-Matrix.trace } (\text{density-collapse } R \ P) = 1$.
 qed**

qed

3.2 Measurements with observables

It is standard in quantum mechanics to represent projective measurements with so-called *observables*. These are Hermitian matrices which encode projective measurements as follows: the eigenvalues of an observable represent the possible projective measurement outcomes, and the associated projectors are the projectors onto the corresponding eigenspaces. The results in this part are based on the spectral theorem, which states that any Hermitian matrix admits an orthonormal basis consisting of eigenvectors of the matrix.

3.2.1 On the diagonal elements of a matrix

We begin by introducing definitions that will be used on the diagonalized version of a Hermitian matrix.

definition *diag-elems* where

$\text{diag-elems } B = \{B\$(i, i) \mid i. i < \text{dim-row } B\}$

Relationship between `diag_elems` and the list `diag_mat`

lemma *diag-elems-set-diag-mat*:

shows $\text{diag-elems } B = \text{set } (\text{diag-mat } B)$ **unfolding** *diag-mat-def* *diag-elems-def*

proof

show $\{B\$(i, i) \mid i. i < \text{dim-row } B\} \subseteq \text{set } (\text{map } (\lambda i. B\$(i, i)) [0..<\text{dim-row } B])$

proof

fix x

assume $x \in \{B\$(i, i) \mid i. i < \text{dim-row } B\}$

hence $\exists i < \text{dim-row } B. x = B\(i, i) **by** *auto*

from this obtain i **where** $i < \text{dim-row } B$ **and** $x = B\$(i, i)$ **by** *auto*

thus $x \in \text{set } (\text{map } (\lambda i. B\$(i, i)) [0..<\text{dim-row } B])$ **by** *auto*

qed

next

show $\text{set } (\text{map } (\lambda i. B\$(i, i)) [0..<\text{dim-row } B]) \subseteq \{B\$(i, i) \mid i. i < \text{dim-row } B\}$

proof

fix x

assume $x \in \text{set } (\text{map } (\lambda i. B\$(i, i)) [0..<\text{dim-row } B])$

thus $x \in \{B\$(i, i) \mid i. i < \text{dim-row } B\}$ **by** *auto*

qed

qed

lemma *diag-elems-finite[simp]*:

shows *finite* $(\text{diag-elems } B)$ **unfolding** *diag-elems-def* **by** *simp*

lemma *diag-elems-mem[simp]*:

assumes $i < \text{dim-row } B$

shows $B \$(i,i) \in \text{diag-elems } B$ **using** *assms* **unfolding** *diag-elems-def* **by** *auto*

When x is a diagonal element of B , `diag_elem_indices` returns the set of diagonal indices of B with value x .

definition *diag-elem-indices* **where**

diag-elem-indices $x B = \{i \mid i. i < \text{dim-row } B \wedge B \$(i,i) = x\}$

lemma *diag-elem-indices-elim*:

assumes $a \in \text{diag-elem-indices } x B$

shows $a < \text{dim-row } B \wedge B \$(a,a) = x$ **using** *assms* **unfolding** *diag-elem-indices-def* **by** *simp*

lemma *diag-elem-indices-itself*:

assumes $i < \text{dim-row } B$

shows $i \in \text{diag-elem-indices } (B \$(i,i)) B$ **using** *assms* **unfolding** *diag-elem-indices-def* **by** *simp*

lemma *diag-elem-indices-finite*:

shows *finite* (*diag-elem-indices* $x B$) **unfolding** *diag-elem-indices-def* **by** *simp*

We can therefore partition the diagonal indices of a matrix B depending on the value of the diagonal elements. If B admits p elements on its diagonal, then we define bijections between its set of diagonal elements and the initial segment $[0..p - 1]$.

definition *dist-el-card* **where**

dist-el-card $B = \text{card } (\text{diag-elems } B)$

definition *diag-idx-to-el* **where**

diag-idx-to-el $B = (\text{SOME } h. \text{bij-betw } h \{.. < \text{dist-el-card } B\} (\text{diag-elems } B))$

definition *diag-el-to-idx* **where**

diag-el-to-idx $B = \text{inv-into } \{.. < \text{dist-el-card } B\} (\text{diag-idx-to-el } B)$

lemma *diag-idx-to-el-bij*:

shows *bij-betw* (*diag-idx-to-el* B) $\{.. < \text{dist-el-card } B\} (\text{diag-elems } B)$

proof –

let $?V = \text{SOME } h. \text{bij-betw } h \{.. < \text{dist-el-card } B\} (\text{diag-elems } B)$

have *vprop*: *bij-betw* $?V \{.. < \text{dist-el-card } B\} (\text{diag-elems } B)$ **using**

someI-ex[*of* $\lambda h. \text{bij-betw } h \{.. < \text{dist-el-card } B\} (\text{diag-elems } B)$]

diag-elems-finite **unfolding** *dist-el-card-def* **using** *bij-betw-from-nat-into-finite*

by *blast*

show *?thesis* **by** (*simp* *add:diag-idx-to-el-def* *vprop*)

qed

lemma *diag-el-to-idx-bij*:

shows *bij-betw* (*diag-el-to-idx* B) (*diag-elems* B) $\{.. < \text{dist-el-card } B\}$

unfolding *diag-el-to-idx-def*

proof (*rule* *bij-betw-inv-into-subset*[*of* - - *diag-elems* B], (*simp* *add:diag-idx-to-el-bij*)+)

show *diag-idx-to-el* $B \{.. < \text{dist-el-card } B\} = \text{diag-elems } B$

by (*simp add: diag-idx-to-el-bij bij-betw-imp-surj-on*)
 qed

lemma *diag-idx-to-el-less-inj*:
 assumes $i < \text{dist-el-card } B$
 and $j < \text{dist-el-card } B$
 and $\text{diag-idx-to-el } B \ i = \text{diag-idx-to-el } B \ j$
 shows $i = j$
proof –
 have $i \in \{.. < \text{dist-el-card } B\}$ **using** *assms* **by** *simp*
 moreover have $j \in \{.. < \text{dist-el-card } B\}$ **using** *assms* **by** *simp*
 moreover have $\text{inj-on } (\text{diag-idx-to-el } B) \ \{.. < \text{dist-el-card } B\}$
 using *diag-idx-to-el-bij[of B]*
 unfolding *bij-betw-def* **by** *simp*
 ultimately show *?thesis* **by** (*meson assms(3) inj-on-contrad*)
 qed

lemma *diag-idx-to-el-less-surj*:
 assumes $x \in \text{diag-elems } B$
 shows $\exists k \in \{.. < \text{dist-el-card } B\}. x = \text{diag-idx-to-el } B \ k$
proof –
 have $\text{diag-idx-to-el } B \ \{.. < \text{dist-el-card } B\} = \text{diag-elems } B$
 using *diag-idx-to-el-bij[of B]* **unfolding** *bij-betw-def* **by** *simp*
 thus *?thesis* **using** *assms* **by** *auto*
 qed

lemma *diag-idx-to-el-img*:
 assumes $k < \text{dist-el-card } B$
 shows $\text{diag-idx-to-el } B \ k \in \text{diag-elems } B$
proof –
 have $\text{diag-idx-to-el } B \ \{.. < \text{dist-el-card } B\} = \text{diag-elems } B$
 using *diag-idx-to-el-bij[of B]* **unfolding** *bij-betw-def* **by** *simp*
 thus *?thesis* **using** *assms* **by** *auto*
 qed

lemma *diag-elems-real*:
 fixes $B::\text{complex Matrix.mat}$
 assumes $\forall i < \text{dim-row } B. B \ \$\$ (i,i) \in \text{Reals}$
 shows $\text{diag-elems } B \subseteq \text{Reals}$
proof
 fix x
 assume $x \in \text{diag-elems } B$
 hence $\exists i < \text{dim-row } B. x = B \ \$\$ (i,i)$ **using** *assms* **unfolding** *diag-elems-def*
 by *auto*
 from *this* **obtain** i **where** $i < \text{dim-row } B \ x = B \ \$\$ (i,i)$ **by** *auto*
 thus $x \in \text{Reals}$ **using** *assms* **by** *simp*
 qed

lemma *diag-elems-Re*:

fixes $B::\text{complex Matrix.mat}$
assumes $\forall i < (\text{dim-row } B). B\$(i,i) \in \text{Reals}$
shows $\{ \text{Re } x \mid x \in \text{diag-elems } B \} = \text{diag-elems } B$
proof
show $\{ \text{complex-of-real } (\text{Re } x) \mid x \in \text{diag-elems } B \} \subseteq \text{diag-elems } B$
proof
fix x
assume $x \in \{ \text{complex-of-real } (\text{Re } x) \mid x \in \text{diag-elems } B \}$
hence $\exists y \in \text{diag-elems } B. x = \text{Re } y$ **by auto**
from this obtain y **where** $y \in \text{diag-elems } B$ **and** $x = \text{Re } y$ **by auto**
hence $y = x$ **using** $\text{assms diag-elems-real[of } B]$ **by auto**
thus $x \in \text{diag-elems } B$ **using** $\langle y \in \text{diag-elems } B \rangle$ **by simp**
qed
show $\text{diag-elems } B \subseteq \{ \text{complex-of-real } (\text{Re } x) \mid x \in \text{diag-elems } B \}$
proof
fix x
assume $x \in \text{diag-elems } B$
hence $\text{Re } x = x$ **using** $\text{assms diag-elems-real[of } B]$ **by auto**
thus $x \in \{ \text{complex-of-real } (\text{Re } x) \mid x \in \text{diag-elems } B \}$ **using** $\langle x \in \text{diag-elems } B \rangle$ **by force**
qed
qed

lemma *diag-idx-to-el-real*:
fixes $B::\text{complex Matrix.mat}$
assumes $\forall i < \text{dim-row } B. B\$(i,i) \in \text{Reals}$
and $i < \text{dist-el-card } B$
shows $\text{Re } (\text{diag-idx-to-el } B \ i) = \text{diag-idx-to-el } B \ i$
proof –
have $\text{diag-idx-to-el } B \ i \in \text{diag-elems } B$ **using** $\text{diag-idx-to-el-img[of } i \ B]$ assms **by simp**
hence $\text{diag-idx-to-el } B \ i \in \text{Reals}$ **using** $\text{diag-elems-real[of } B]$ assms **by auto**
thus ?thesis **by simp**
qed

lemma *diag-elem-indices-empty*:
assumes $B \in \text{carrier-mat dimR dimC}$
and $i < (\text{dist-el-card } B)$
and $j < (\text{dist-el-card } B)$
and $i \neq j$
shows $\text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B \cap$
 $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B) = \{ \}$
proof (*rule ccontr*)
assume $\text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B \cap$
 $\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B \neq \{ \}$
hence $\exists x. x \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B \cap$
 $\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B$ **by auto**
from this obtain x **where**
 $x \text{prop: } x \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B \cap$

$\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B$ **by auto**
hence $B \ \$\$ (x,x) = (\text{diag-idx-to-el } B \ i)$
using $\text{diag-elem-indices-elem}[\text{of } x \ \text{diag-idx-to-el } B \ i]$ **by simp**
moreover have $B \ \$\$ (x,x) = (\text{diag-idx-to-el } B \ j)$
using $\text{diag-elem-indices-elem}[\text{of } x \ \text{diag-idx-to-el } B \ j]$ $xprop$ **by simp**
ultimately have $\text{diag-idx-to-el } B \ i = \text{diag-idx-to-el } B \ j$ **by simp**
hence $i = j$ **using** $\text{diag-idx-to-el-less-inj}$ $assms$ **by auto**
thus False using $assms$ **by simp**
qed

lemma (in $cpx\text{-sq-mat}$) $\text{diag-elem-indices-disjoint}$:
assumes $B \in \text{carrier-mat } \dim R \ \dim C$
shows $\text{disjoint-family-on } (\lambda n. \ \text{diag-elem-indices } (\text{diag-idx-to-el } B \ n) \ B)$
 $\{.. < \text{dist-el-card } B\}$ **unfolding** $\text{disjoint-family-on-def}$
proof (intro ballI impI)
fix $p \ m$
assume $m \in \{.. < \text{dist-el-card } B\}$ **and** $p \in \{.. < \text{dist-el-card } B\}$ **and** $m \neq p$
thus $\text{diag-elem-indices } (\text{diag-idx-to-el } B \ m) \ B \cap$
 $\text{diag-elem-indices } (\text{diag-idx-to-el } B \ p) \ B = \{\}$
using $\text{diag-elem-indices-empty}$ $assms$ $fc\text{-mats-carrier}$ **by simp**
qed

lemma $\text{diag-elem-indices-union}$:
assumes $B \in \text{carrier-mat } \dim R \ \dim C$
shows $(\bigcup i \in \{.. < \text{dist-el-card } B\}. \ \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B) =$
 $\{.. < \dim R\}$
proof
show $(\bigcup i < \text{dist-el-card } B. \ \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B) \subseteq \{.. < \dim R\}$
proof
fix x
assume $x \in (\bigcup i < \text{dist-el-card } B. \ \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B)$
hence $\exists i < \text{dist-el-card } B. \ x \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B$ **by auto**
from this obtain i **where** $i < \text{dist-el-card } B$
 $x \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B$ **by auto**
hence $x < \dim R$ **using** $\text{diag-elem-indices-elem}[\text{of } x \ - \ B]$ $assms$ **by simp**
thus $x \in \{.. < \dim R\}$ **by auto**
qed

next
show $\{.. < \dim R\} \subseteq (\bigcup i < \text{dist-el-card } B. \ \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B)$
proof
fix j
assume $j \in \{.. < \dim R\}$
hence $j < \dim\text{-row } B$ **using** $assms$ **by simp**
hence $B \ \$\$ (j,j) \in \text{diag-elems } B$ **by simp**
hence $\exists k \in \{.. < \text{dist-el-card } B\}. \ B \ \$\$ (j,j) = \text{diag-idx-to-el } B \ k$
using $\text{diag-idx-to-el-less-surj}[\text{of } B \ \$\$ (j,j)]$ **by simp**
from this obtain k **where** $kprop: k \in \{.. < \text{dist-el-card } B\}$
 $B \ \$\$ (j,j) = \text{diag-idx-to-el } B \ k$ **by auto**
hence $j \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ k) \ B$ **using** $\langle j < \dim\text{-row } B \rangle$

```

      diag-elem-indices-itself by fastforce
    thus  $j \in (\bigcup i < \text{dist-el-card } B. \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B)$ 
      using kprop by auto
    qed
  qed

```

3.2.2 Construction of measurement outcomes

The construction of a projective measurement for a hermitian matrix A is based on the Schur decomposition $A = U * B * U^\dagger$, where B is diagonal and U is unitary. The columns of U are normalized and pairwise orthogonal; they are used to construct the projectors associated with a measurement value

definition (in *cpx-sq-mat*) *project-vecs* **where**
project-vecs ($f :: \text{nat} \Rightarrow \text{complex Matrix.vec}$) $N = \text{sum-mat } (\lambda i. \text{rank-1-proj } (f \ i)) \ N$

lemma (in *cpx-sq-mat*) *project-vecs-dim*:

assumes $\forall i \in N. \text{dim-vec } (f \ i) = \text{dim}R$

shows *project-vecs* $f \ N \in \text{fc-mats}$

proof –

have *project-vecs* $f \ N \in \text{carrier-mat } \text{dim}R \ \text{dim}C$ **unfolding** *project-vecs-def*

proof (*rule sum-mat-carrier*)

show $\bigwedge i. i \in N \implies \text{rank-1-proj } (f \ i) \in \text{fc-mats}$ **using** *assms fc-mats-carrier rank-1-proj-dim*

dim-eq rank-1-proj-carrier **by** *fastforce*

qed

thus *?thesis* **using** *fc-mats-carrier* **by** *simp*

qed

definition (in *cpx-sq-mat*) *mk-meas-outcome* **where**

mk-meas-outcome $B \ U = (\lambda i. (\text{Re } (\text{diag-idx-to-el } B \ i)),$

project-vecs $(\lambda i. \text{zero-col } U \ i) \ (\text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B))$

lemma (in *cpx-sq-mat*) *mk-meas-outcome-carrier*:

assumes *Complex-Matrix.unitary* U

and $U \in \text{fc-mats}$

and $B \in \text{fc-mats}$

shows *meas-outcome-prj* $((\text{mk-meas-outcome } B \ U) \ j) \in \text{fc-mats}$

proof –

have *project-vecs* $(\lambda i. \text{zero-col } U \ i) \ (\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B) \in \text{fc-mats}$

using *project-vecs-dim* **by** (*simp add: assms(2) zero-col-dim*)

thus *?thesis* **unfolding** *mk-meas-outcome-def meas-outcome-prj-def* **by** *simp*

qed

lemma (in *cpx-sq-mat*) *mk-meas-outcome-sum-id*:

assumes *Complex-Matrix.unitary* U

and $U \in \text{fc-mats}$

and $B \in \text{fc-mats}$
shows $\text{sum-mat } (\lambda j. \text{meas-outcome-prj } ((\text{mk-meas-outcome } B \ U) \ j))$
 $\{..\langle \text{dist-el-card } B \rangle\} = 1_m \ \text{dim}R$
proof –
have $\text{sum-mat } (\lambda j. \text{meas-outcome-prj } ((\text{mk-meas-outcome } B \ U) \ j)) \{..\langle \text{dist-el-card } B \rangle\} =$
 $\text{sum-mat } (\lambda j. \text{project-vecs } (\lambda i. \text{zero-col } U \ i) \ (\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B))$
 $\{..\langle \text{dist-el-card } B \rangle\}$
unfolding $\text{mk-meas-outcome-def}$ $\text{meas-outcome-prj-def}$ **by** simp
also have $\dots = \text{sum-mat } (\lambda i. \text{rank-1-proj } (\text{zero-col } U \ i))$
 $(\bigcup_{j < \text{dist-el-card } B} \text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B)$
unfolding project-vecs-def sum-mat-def
proof ($\text{rule } \text{disj-family-sum-with}[\text{symmetric}]$)
show $\forall j. \text{rank-1-proj } (\text{zero-col } U \ j) \in \text{fc-mats}$ **using** zero-col-dim fc-mats-carrier
 dim-eq
by ($\text{metis } \text{assms}(2)$) $\text{rank-1-proj-carrier}$
show $\text{finite } \{..\langle \text{dist-el-card } B \rangle\}$ **by** simp
show $\bigwedge i. i \in \{..\langle \text{dist-el-card } B \rangle\} \implies \text{finite } (\text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B)$
using $\text{diag-elem-indices-finite}$ **by** simp
show $\text{disjoint-family-on } (\lambda n. \text{diag-elem-indices } (\text{diag-idx-to-el } B \ n) \ B)$
 $\{..\langle \text{dist-el-card } B \rangle\}$
unfolding $\text{disjoint-family-on-def}$
proof (intro ballI impI)
fix $p \ m$
assume $m \in \{..\langle \text{dist-el-card } B \rangle\}$ **and** $p \in \{..\langle \text{dist-el-card } B \rangle\}$ **and** $m \neq p$
thus $\text{diag-elem-indices } (\text{diag-idx-to-el } B \ m) \ B \cap$
 $\text{diag-elem-indices } (\text{diag-idx-to-el } B \ p) \ B = \{\}$
using $\text{diag-elem-indices-empty}$ assms fc-mats-carrier **by** simp
qed
qed
also have $\dots = \text{sum-mat } (\lambda i. \text{rank-1-proj } (\text{zero-col } U \ i)) \{..\langle \text{dim}R \rangle\}$
using $\text{diag-elem-indices-union}[\text{of } B]$ assms fc-mats-carrier **by** simp
also have $\dots = \text{sum-mat } (\lambda i. \text{rank-1-proj } (\text{Matrix.col } U \ i)) \{..\langle \text{dim}R \rangle\}$
proof ($\text{rule } \text{sum-mat-cong}$, simp)
show $\bigwedge i. i \in \{..\langle \text{dim}R \rangle\} \implies \text{rank-1-proj } (\text{zero-col } U \ i) \in \text{fc-mats}$ **using** dim-eq
by ($\text{metis } \text{assms}(2)$) $\text{local.fc-mats-carrier}$ $\text{rank-1-proj-carrier}$ zero-col-dim
thus $\bigwedge i. i \in \{..\langle \text{dim}R \rangle\} \implies \text{rank-1-proj } (\text{Matrix.col } U \ i) \in \text{fc-mats}$ **using**
 dim-eq
by ($\text{metis } \text{lessThan-iff}$ zero-col-col)
show $\bigwedge i. i \in \{..\langle \text{dim}R \rangle\} \implies \text{rank-1-proj } (\text{zero-col } U \ i) = \text{rank-1-proj } (\text{Matrix.col } U \ i)$
by (simp add: zero-col-col)
qed
also have $\dots = 1_m \ \text{dim}R$ **using** $\text{sum-rank-1-proj-unitary}$ assms **by** simp
finally show $?thesis$.
qed

lemma (in *cpx-sq-mat*) *make-meas-outcome-prj-ortho*:
assumes *Complex-Matrix.unitary* U
and $U \in \text{fc-mats}$
and $B \in \text{fc-mats}$
and $i < \text{dist-el-card } B$
and $j < \text{dist-el-card } B$
and $i \neq j$
shows $\text{meas-outcome-prj } ((\text{mk-meas-outcome } B \ U) \ i) * \text{meas-outcome-prj } ((\text{mk-meas-outcome } B \ U) \ j) = 0_m \ \text{dimR} \ \text{dimR}$
proof –
define P_i **where** $P_i = \text{sum-mat } (\lambda k. \text{rank-1-proj } (\text{zero-col } U \ k))$
 $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B)$
have $\text{sneqi}: \text{meas-outcome-prj } (\text{mk-meas-outcome } B \ U \ i) = P_i$
unfolding $\text{mk-meas-outcome-def project-vecs-def } P_i\text{-def meas-outcome-prj-def}$
by *simp*
define P_j **where** $P_j = \text{sum-mat } (\lambda k. \text{rank-1-proj } (\text{zero-col } U \ k))$
 $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B)$
have $\text{sneqj}: \text{meas-outcome-prj } (\text{mk-meas-outcome } B \ U \ j) = P_j$
unfolding $\text{mk-meas-outcome-def project-vecs-def } P_j\text{-def meas-outcome-prj-def}$
by *simp*
have $P_i * P_j = 0_m \ \text{dimR} \ \text{dimR}$ **unfolding** $P_i\text{-def}$
proof (*rule sum-mat-left-ortho-zero*)
show *finite* $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B)$
using *diag-elem-indices-finite[of - B]* **by** *simp*
show $\text{km}: \bigwedge k. k \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B \implies$
 $\text{rank-1-proj } (\text{zero-col } U \ k) \in \text{fc-mats}$ **using** *zero-col-dim assms fc-mats-carrier*
dim-eq
by (*metis rank-1-proj-carrier*)
show $P_j \in \text{fc-mats}$ **using** *sneqj assms mk-meas-outcome-carrier* **by** *auto*
show $\bigwedge k. k \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B \implies$
 $\text{rank-1-proj } (\text{zero-col } U \ k) * P_j = 0_m \ \text{dimR} \ \text{dimR}$
proof –
fix k
assume $k \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B$
show $\text{rank-1-proj } (\text{zero-col } U \ k) * P_j = 0_m \ \text{dimR} \ \text{dimR}$ **unfolding** $P_j\text{-def}$
proof (*rule sum-mat-right-ortho-zero*)
show *finite* $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B)$
using *diag-elem-indices-finite[of - B]* **by** *simp*
show $\bigwedge i. i \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B \implies$
 $\text{rank-1-proj } (\text{zero-col } U \ i) \in \text{fc-mats}$ **using** *zero-col-dim assms fc-mats-carrier*
dim-eq
by (*metis rank-1-proj-carrier*)
show $\text{rank-1-proj } (\text{zero-col } U \ k) \in \text{fc-mats}$
by (*simp add: km* $\langle k \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B \rangle$)
show $\bigwedge i. i \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B \implies$
 $\text{rank-1-proj } (\text{zero-col } U \ k) * \text{rank-1-proj } (\text{zero-col } U \ i) = 0_m \ \text{dimR} \ \text{dimR}$
proof –
fix m
assume $m \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B$

hence $m \neq k$ **using** $\langle k \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B \rangle$
 $\text{diag-elem-indices-disjoint[of } B \text{]} \text{ fc-mats-carrier } \text{assms}$ **unfolding** *dis-*
joint-family-on-def **by** *auto*
have $\bigwedge i. i \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B \implies i < \text{dimR}$
using $\text{diag-elem-indices-elem } \text{fc-mats-carrier } \text{assms } \text{dim-eq}$ **by** (*metis*
carrier-matD(1))
hence $m < \text{dimR}$ **using** $\langle m \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B \rangle$
by *simp*
have $\bigwedge k. k \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B \implies k < \text{dimR}$
using $\text{diag-elem-indices-elem } \text{fc-mats-carrier } \text{assms } \text{dim-eq}$ **by** (*metis*
carrier-matD(1))
hence $k < \text{dimR}$ **using** $\langle k \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B \rangle$ **by**
simp
show $\text{rank-1-proj } (\text{zero-col } U \ k) * \text{rank-1-proj } (\text{zero-col } U \ m) = 0_m \ \text{dimR}$
 dimR
using $\text{rank-1-proj-unitary-ne[of } U \ k \ m \text{]} \ \text{assms } \langle m < \text{dimR} \rangle \langle k < \text{dimR} \rangle$
by (*metis* $\langle m \neq k \rangle$ *zero-col-col*)
qed
qed
qed
qed
thus *?thesis* **using** *sneqi sneqj* **by** *simp*
qed

lemma (in *cpx-sq-mat*) *make-meas-outcome-projects:*

assumes *Complex-Matrix.unitary* U
and $U \in \text{fc-mats}$
and $B \in \text{fc-mats}$
and $j < \text{dist-el-card } B$
shows $\text{projector } (\text{meas-outcome-prj } ((\text{mk-meas-outcome } B \ U) \ j))$ **unfolding** *pro-*
jector-def
proof
define P_j **where** $P_j = \text{sum-mat } (\lambda i. \text{rank-1-proj } (\text{zero-col } U \ i))$
 $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B)$
have *sneq*: $\text{meas-outcome-prj } (\text{mk-meas-outcome } B \ U \ j) = P_j$
unfolding *mk-meas-outcome-def project-vecs-def Pj-def meas-outcome-prj-def*
by *simp*
moreover **have** *hermitian* P_j **unfolding** *Pj-def*
proof (*rule sum-mat-hermitian*)
show *finite* $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B)$
using *diag-elem-indices-finite[of - B]* **by** *simp*
show $\forall i \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B. \text{hermitian } (\text{rank-1-proj } (\text{zero-col}$
 $U \ i))$
using *rank-1-proj-hermitian* **by** *simp*
show $\forall i \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B. \text{rank-1-proj } (\text{zero-col } U \ i) \in$
 fc-mats
using *zero-col-dim fc-mats-carrier dim-eq* **by** (*metis* *assms(2)* *rank-1-proj-carrier*)
qed
ultimately **show** *hermitian* $(\text{meas-outcome-prj } (\text{mk-meas-outcome } B \ U \ j))$ **by**

```

simp
have  $P_j * P_j = P_j$  unfolding  $P_j$ -def
proof (rule sum-mat-ortho-square)
  show finite (diag-elem-indices (diag-idx-to-el  $B$   $j$ )  $B$ )
    using diag-elem-indices-finite[of  $B$ ] by simp
  show  $\bigwedge i. i \in \text{diag-elem-indices (diag-idx-to-el } B \text{ } j) B \implies$ 
    rank-1-proj (zero-col  $U$   $i$ ) * rank-1-proj (zero-col  $U$   $i$ ) = rank-1-proj (zero-col
 $U$   $i$ )
  proof -
    fix  $i$ 
    assume  $imem: i \in \text{diag-elem-indices (diag-idx-to-el } B \text{ } j) B$ 
    hence  $i < \text{dim}R$  using diag-elem-indices-elem fc-mats-carrier  $assms$  dim-eq
      by (metis carrier-matD(1))
    hence zero-col  $U$   $i$  = Matrix.col  $U$   $i$  using zero-col-col[of  $i$ ] by simp
    hence rank-1-proj (zero-col  $U$   $i$ ) = rank-1-proj (Matrix.col  $U$   $i$ ) by simp
    moreover have rank-1-proj (Matrix.col  $U$   $i$ ) * rank-1-proj (Matrix.col  $U$   $i$ )
=
    rank-1-proj (Matrix.col  $U$   $i$ ) by (rule rank-1-proj-unitary-eq, (auto simp
add:  $assms$   $\langle i < \text{dim}R \rangle$ ))
    ultimately show rank-1-proj (zero-col  $U$   $i$ ) * rank-1-proj (zero-col  $U$   $i$ ) =
    rank-1-proj (zero-col  $U$   $i$ ) by simp
  qed
show  $\bigwedge i. i \in \text{diag-elem-indices (diag-idx-to-el } B \text{ } j) B \implies$ 
  rank-1-proj (zero-col  $U$   $i$ )  $\in$  fc-mats
using zero-col-dim  $assms$  fc-mats-carrier dim-eq by (metis rank-1-proj-carrier)

have  $\bigwedge i. i \in \text{diag-elem-indices (diag-idx-to-el } B \text{ } j) B \implies i < \text{dim}R$ 
using diag-elem-indices-elem fc-mats-carrier  $assms$  dim-eq by (metis car-
rier-matD(1))
thus  $\bigwedge i \text{ } ja.$ 
   $i \in \text{diag-elem-indices (diag-idx-to-el } B \text{ } j) B \implies$ 
   $ja \in \text{diag-elem-indices (diag-idx-to-el } B \text{ } j) B \implies$ 
   $i \neq ja \implies \text{rank-1-proj (zero-col } U \text{ } i) * \text{rank-1-proj (zero-col } U \text{ } ja) = 0_m$ 
 $\text{dim}R \text{ dim}R$ 
using rank-1-proj-unitary-ne by (simp add:  $assms$ (1)  $assms$ (2) zero-col-col)
qed
thus meas-outcome-prj (mk-meas-outcome  $B$   $U$   $j$ ) *
  meas-outcome-prj (mk-meas-outcome  $B$   $U$   $j$ ) =
  meas-outcome-prj (mk-meas-outcome  $B$   $U$   $j$ )
using sneq by simp
qed

lemma (in cpx-sq-mat) mk-meas-outcome-fst-inj:
assumes  $\forall i < (\text{dim-row } B). B\$\$(i,i) \in \text{Reals}$ 
shows inj-on ( $\lambda i. \text{meas-outcome-val ((mk-meas-outcome } B \text{ } U) i)$ )  $\{.. < \text{dist-el-card}$ 
 $B\}$ 
unfolding inj-on-def
proof (intro ballI impI)
fix  $x$   $y$ 

```

assume $x \in \{..<dist-el-card B\}$ **and** $y \in \{..<dist-el-card B\}$
and $meas-outcome-val (mk-meas-outcome B U x) =$
 $meas-outcome-val (mk-meas-outcome B U y)$ **note** $xy = this$
hence $diag-idx-to-el B x = Re (diag-idx-to-el B x)$
using $assms diag-idx-to-el-real$ **by** $simp$
also have $... = Re (diag-idx-to-el B y)$ **using** xy
unfolding $mk-meas-outcome-def meas-outcome-val-def$ **by** $simp$
also have $... = diag-idx-to-el B y$ **using** $assms diag-idx-to-el-real xy$ **by** $simp$
finally have $diag-idx-to-el B x = diag-idx-to-el B y .$
thus $x = y$ **using** $diag-idx-to-el-less-inj xy$ **by** $simp$
qed

lemma (in $cpx-sq-mat$) $mk-meas-outcome-fst-bij$:
assumes $\forall i < (dim-row B). B\$(i,i) \in Reals$
shows $bij-betw (\lambda i. meas-outcome-val ((mk-meas-outcome B U) i)) \{..<dist-el-card B\}$
 $\{Re x | x. x \in diag-elems B\}$
unfolding $bij-betw-def$
proof
have $inj-on (\lambda x. (meas-outcome-val (mk-meas-outcome B U x))) \{..<dist-el-card B\}$
using $assms mk-meas-outcome-fst-inj$ **by** $simp$
moreover have $\{Re x | x. x \in diag-elems B\} = diag-elems B$ **using** $diag-elems-Re[of B]$ $assms$ **by** $simp$
ultimately show $inj-on (\lambda x. meas-outcome-val (mk-meas-outcome B U x)) \{..<dist-el-card B\}$ **by** $simp$
show $(\lambda i. meas-outcome-val (mk-meas-outcome B U i)) ' \{..<dist-el-card B\} =$
 $\{Re x | x. x \in diag-elems B\}$ **unfolding** $meas-outcome-val-def mk-meas-outcome-def$
proof
show $(\lambda i. fst (Re (diag-idx-to-el B i), project-vecs (zero-col U)$
 $(diag-elem-indices (diag-idx-to-el B i) B))) ' \{..<dist-el-card B\} \subseteq$
 $\{Re x | x. x \in diag-elems B\}$
using $diag-idx-to-el-bij[of B]$ $bij-betw-apply$ **by** $fastforce$
show $\{Re x | x. x \in diag-elems B\}$
 $\subseteq (\lambda i. fst (Re (diag-idx-to-el B i),$
 $project-vecs (zero-col U) (diag-elem-indices (diag-idx-to-el B i) B))) ' \{..<dist-el-card B\}$ **using** $diag-idx-to-el-less-surj$ **by** $fastforce$
qed
qed

3.2.3 Projective measurement associated with an observable

definition $eigvals$ where

$eigvals M = (SOME as. char-poly M = (\prod a \leftarrow as. [- a, 1:]) \wedge length as = dim-row M)$

lemma $eigvals-poly-length$:

assumes $(M::complex Matrix.mat) \in carrier-mat n n$

shows $char-poly M = (\prod a \leftarrow (eigvals M). [- a, 1:]) \wedge length (eigvals M) =$

$dim\text{-row } M$
proof –
let $?V = SOME\ as.\ char\text{-poly } M = (\prod a \leftarrow as.\ [:- a, 1:]) \wedge length\ as = dim\text{-row } M$
have $vprop: char\text{-poly } M = (\prod a \leftarrow ?V.\ [:- a, 1:]) \wedge length\ ?V = dim\text{-row } M$
using
 $someI\text{-ex}[of\ \lambda as.\ char\text{-poly } M = (\prod a \leftarrow as.\ [:- a, 1:]) \wedge length\ as = dim\text{-row } M]$
 $complex\text{-mat}\text{-char}\text{-poly}\text{-factorizable}\ assms$ **by** $blast$
show $?thesis$ **by** $(metis\ (no\text{-types})\ eigvals\text{-def}\ vprop)$
qed

We define the spectrum of a matrix A : this is its set of eigenvalues; its elements are roots of the characteristic polynomial of A .

definition $spectrum$ **where**
 $spectrum\ M = set\ (eigvals\ M)$

lemma $spectrum\text{-finite}$:
shows $finite\ (spectrum\ M)$ **unfolding** $spectrum\text{-def}$ **by** $simp$

lemma $spectrum\text{-char}\text{-poly}\text{-root}$:
fixes $A::complex\ Matrix.\ mat$
assumes $A \in carrier\text{-mat } n\ n$
and $k \in spectrum\ A$
shows $poly\ (char\text{-poly } A)\ k = 0$ **using** $eigvals\text{-poly}\text{-length}[of\ A\ n]$ $assms$
unfolding $spectrum\text{-def}$ $eigenvalue\text{-root}\text{-char}\text{-poly}$
by $(simp\ add: linear\text{-poly}\text{-root})$

lemma $spectrum\text{-eigenvalues}$:
fixes $A::complex\ Matrix.\ mat$
assumes $A \in carrier\text{-mat } n\ n$
and $k \in spectrum\ A$
shows $eigenvalue\ A\ k$ **using** $eigenvalue\text{-root}\text{-char}\text{-poly}[of\ A\ n\ k]$
 $spectrum\text{-char}\text{-poly}\text{-root}[of\ A\ n\ k]$ $assms$ **by** $simp$

The main result that is used to construct a projective measurement for a Hermitian matrix is that it is always possible to decompose it as $A = U * B * U^\dagger$, where B is diagonal and only contains real elements, and U is unitary.

definition $hermitian\text{-decomp}$ **where**
 $hermitian\text{-decomp } A\ B\ U \equiv similar\text{-mat}\text{-wit } A\ B\ U\ (Complex\text{-Matrix}.\ adjoint\ U)$
 $\wedge diagonal\text{-mat } B \wedge$
 $diag\text{-mat } B = (eigvals\ A) \wedge unitary\ U \wedge (\forall i < dim\text{-row } B.\ B\ \$(i, i) \in Reals)$

lemma $hermitian\text{-decomp}\text{-sim}$:
assumes $hermitian\text{-decomp } A\ B\ U$
shows $similar\text{-mat}\text{-wit } A\ B\ U\ (Complex\text{-Matrix}.\ adjoint\ U)$ **using** $assms$
unfolding $hermitian\text{-decomp}\text{-def}$ **by** $simp$

lemma *hermitian-decomp-diag-mat*:
assumes *hermitian-decomp* $A B U$
shows *diagonal-mat* B **using** *assms*
unfolding *hermitian-decomp-def* **by** *simp*

lemma *hermitian-decomp-eigenvalues*:
assumes *hermitian-decomp* $A B U$
shows *diag-mat* $B = (\text{eigvals } A)$ **using** *assms*
unfolding *hermitian-decomp-def* **by** *simp*

lemma *hermitian-decomp-unitary*:
assumes *hermitian-decomp* $A B U$
shows *unitary* U **using** *assms*
unfolding *hermitian-decomp-def* **by** *simp*

lemma *hermitian-decomp-real-eigvals*:
assumes *hermitian-decomp* $A B U$
shows $\forall i < \text{dim-row } B. B\$\$(i, i) \in \text{Reals}$ **using** *assms*
unfolding *hermitian-decomp-def* **by** *simp*

lemma *hermitian-decomp-dim-carrier*:
assumes *hermitian-decomp* $A B U$
shows $B \in \text{carrier-mat } (\text{dim-row } A) (\text{dim-col } A)$ **using** *assms*
unfolding *hermitian-decomp-def similar-mat-wit-def*
by (*metis* (*full-types*) *index-mult-mat*(β) *index-one-mat*(β) *insert-subset*)

lemma *similar-mat-wit-dim-row*:
assumes *similar-mat-wit* $A B Q R$
shows $\text{dim-row } B = \text{dim-row } A$ **using** *assms* *Let-def* **unfolding** *similar-mat-wit-def*
by (*meson* *carrier-matD*(1) *insert-subset*)

lemma (**in** *cpx-sq-mat*) *hermitian-schur-decomp*:
assumes *hermitian* A
and $A \in \text{fc-mats}$
obtains $B U$ **where** *hermitian-decomp* $A B U$
proof –
{
 have $es: \text{char-poly } A = (\prod (e :: \text{complex}) \leftarrow (\text{eigvals } A). [:- e, 1:])$
 using *assms* *fc-mats-carrier* *eigvals-poly-length* *dim-eq* **by** *auto*
 obtain $B U Q$ **where** $us: \text{unitary-schur-decomposition } A (\text{eigvals } A) = (B, U, Q)$

 by (*cases* *unitary-schur-decomposition* $A (\text{eigvals } A)$)
 hence $pr: \text{similar-mat-wit } A B U$ (*Complex-Matrix.adjoint* U) \wedge *diagonal-mat* $B \wedge$
 diag-mat $B = (\text{eigvals } A) \wedge$ *unitary* $U \wedge (\forall i < \text{dimR}. B\$\$(i, i) \in \text{Reals})$
 using *hermitian-eigenvalue-real* *assms* *fc-mats-carrier* *es* *dim-eq* **by** *auto*
 moreover **have** $\text{dim-row } B = \text{dimR}$ **using** *assms* *fc-mats-carrier* *dim-eq* *similar-mat-wit-dim-row*[*of* A]
 pr **by** *auto*

ultimately have *hermitian-decomp* $A B U$ **unfolding** *hermitian-decomp-def*
by *simp*
hence $\exists B U. \text{hermitian-decomp } A B U$ **by** *auto*
}
thus *?thesis using that by auto*
qed

lemma (in *cpx-sq-mat*) *hermitian-spectrum-real*:
assumes $A \in \text{fc-mats}$
and *hermitian* A
and $a \in \text{spectrum } A$
shows $a \in \text{Reals}$
proof –
obtain $B U$ **where** *bu: hermitian-decomp* $A B U$ **using** *assms hermitian-schur-decomp*
by *auto*
hence *dimB: B ∈ carrier-mat dimR dimR using assms fc-mats-carrier*
dim-eq hermitian-decomp-dim-carrier[of A] by simp
hence $Bii: \bigwedge i. i < \text{dimR} \implies B\$(i, i) \in \text{Reals}$ **using** *hermitian-decomp-real-eigvals[of*
 $A B U]$
bu assms fc-mats-carrier by simp
have *diag-mat B = (eigvals A) using bu hermitian-decomp-eigenvalues[of A B]*
by *simp*
hence $a \in \text{set } (\text{diag-mat } B)$ **using** *assms unfolding spectrum-def by simp*
hence $\exists i < \text{length } (\text{diag-mat } B). a = \text{diag-mat } B ! i$ **by** (*metis in-set-conv-nth*)
from this obtain i **where** $i < \text{length } (\text{diag-mat } B)$ **and** $a = \text{diag-mat } B ! i$ **by**
auto
hence $a = B \$(i, i)$ **unfolding** *diag-mat-def by simp*
moreover have $i < \text{dimR}$ **using** *dimB <i < length (diag-mat B)> unfolding*
diag-mat-def by simp
ultimately show *?thesis using Bii by simp*
qed

lemma (in *cpx-sq-mat*) *spectrum-ne*:
assumes $A \in \text{fc-mats}$
and *hermitian* A
shows *spectrum* $A \neq \{\}$
proof –
obtain $B U$ **where** *bu: hermitian-decomp* $A B U$ **using** *assms hermitian-schur-decomp*
by *auto*
hence *dimB: B ∈ carrier-mat dimR dimR using assms fc-mats-carrier*
dim-eq hermitian-decomp-dim-carrier[of A] by simp
have *diag-mat B = (eigvals A) using bu hermitian-decomp-eigenvalues[of A B]*
by *simp*
have $B\$(0,0) \in \text{set } (\text{diag-mat } B)$ **using** *dimB npos unfolding diag-mat-def*
by *simp*
hence *set (diag-mat B) ≠ {} by auto*
thus *?thesis unfolding spectrum-def using <diag-mat B = eigvals A> by auto*
qed

```

lemma unitary-hermitian-eigenvalues:
  fixes  $U::\text{complex Matrix.mat}$ 
  assumes unitary  $U$ 
  and hermitian  $U$ 
  and  $U \in \text{carrier-mat } n \ n$ 
  and  $0 < n$ 
  and  $k \in \text{spectrum } U$ 
shows  $k \in \{-1, 1\}$ 
proof -
  have  $\text{cpx-sq-mat } n \ n \ (\text{carrier-mat } n \ n)$  by (unfold-locales, (simp add: assms)+)
  have eigenvalue  $U \ k$  using spectrum-eigenvalues assms by simp
  have  $k \in \text{Reals}$  using assms  $\langle \text{cpx-sq-mat } n \ n \ (\text{carrier-mat } n \ n) \rangle$ 
     $\text{cpx-sq-mat.hermitian-spectrum-real}$  by simp
  hence conjugate  $k = k$  by (simp add: Reals-cnj-iff)
  hence  $k^2 = 1$  using unitary-eigenvalues-norm-square[ $\text{of } U \ n \ k$ ] assms
    by (simp add:  $\langle \text{eigenvalue } U \ k \rangle$  power2-eq-square)
  thus ?thesis using power2-eq-1-iff by auto
qed

```

```

lemma unitary-hermitian-Re-spectrum:
  fixes  $U::\text{complex Matrix.mat}$ 
  assumes unitary  $U$ 
  and hermitian  $U$ 
  and  $U \in \text{carrier-mat } n \ n$ 
  and  $0 < n$ 
  shows  $\{\text{Re } x \mid x \in \text{spectrum } U\} \subseteq \{-1, 1\}$ 
proof
  fix  $y$ 
  assume  $y \in \{\text{Re } x \mid x \in \text{spectrum } U\}$ 
  hence  $\exists x \in \text{spectrum } U. y = \text{Re } x$  by auto
  from this obtain  $x$  where  $x \in \text{spectrum } U$  and  $y = \text{Re } x$  by auto
  hence  $x \in \{-1, 1\}$  using unitary-hermitian-eigenvalues assms by simp
  thus  $y \in \{-1, 1\}$  using  $\langle y = \text{Re } x \rangle$  by auto
qed

```

The projective measurement associated with matrix M is obtained from its Schur decomposition, by considering the number of distinct elements on the resulting diagonal matrix and constructing the projectors associated with each of them.

type-synonym $\text{proj-meas-rep} = \text{nat} \times (\text{nat} \Rightarrow \text{measure-outcome})$

definition $\text{proj-meas-size}::\text{proj-meas-rep} \Rightarrow \text{nat}$ **where**
 $\text{proj-meas-size } P = \text{fst } P$

definition $\text{proj-meas-outcomes}::\text{proj-meas-rep} \Rightarrow (\text{nat} \Rightarrow \text{measure-outcome})$ **where**
 $\text{proj-meas-outcomes } P = \text{snd } P$

definition (in cpx-sq-mat) $\text{make-pm}::\text{complex Matrix.mat} \Rightarrow \text{proj-meas-rep}$ **where**
 $\text{make-pm } A = (\text{let } (B, U, -) = \text{unitary-schur-decomposition } A \ (\text{eigvals } A) \text{ in}$

(*dist-el-card B, mk-meas-outcome B U*)

lemma (in *cpx-sq-mat*) *make-pm-decomp*:

shows *make-pm A = (proj-meas-size (make-pm A), proj-meas-outcomes (make-pm A))*

unfolding *proj-meas-size-def proj-meas-outcomes-def* **by** *simp*

lemma (in *cpx-sq-mat*) *make-pm-proj-measurement*:

assumes *A ∈ fc-mats*

and *hermitian A*

and *make-pm A = (n, M)*

shows *proj-measurement n M*

proof –

have *es: char-poly A = (∏ (e :: complex) ← (eigvals A). [:- e, 1:])*

using *assms fc-mats-carrier eigvals-poly-length dim-eq* **by** *auto*

obtain *B U Q* **where** *us: unitary-schur-decomposition A (eigvals A) = (B, U, Q)*

by (*cases unitary-schur-decomposition A (eigvals A), auto*)

then have *similar-mat-wit A B U (Complex-Matrix.adjoint U) ∧ diagonal-mat B ∧*

diag-mat B = (eigvals A) ∧ unitary U ∧ (∀ i < dimR. B\$\$ (i, i) ∈ Reals)

using *hermitian-eigenvalue-real assms fc-mats-carrier es dim-eq* **by** *auto*

hence *A: A = U * B * (Complex-Matrix.adjoint U)* **and** *dB: diagonal-mat B*

and *Bii: ∧ i. i < dimR ⇒ B\$\$ (i, i) ∈ Reals*

and *dimB: B ∈ carrier-mat dimR dimR* **and** *dimP: U ∈ carrier-mat dimR dimR* **and**

dimAP: Complex-Matrix.adjoint U ∈ carrier-mat dimR dimR **and** *unit: unitary U*

unfolding *similar-mat-wit-def Let-def* **using** *assms fc-mats-carrier* **by** *auto*

hence *mpeq: make-pm A = (dist-el-card B, mk-meas-outcome B U)* **using** *us Let-def*

unfolding *make-pm-def* **by** *simp*

hence *n = dist-el-card B* **using** *assms* **by** *simp*

have *M = mk-meas-outcome B U* **using** *assms mpeq* **by** *simp*

show *?thesis* **unfolding** *proj-measurement-def*

proof (*intro conjI*)

show *inj-on (λ i. meas-outcome-val (M i)) {..<n}*

using *mk-meas-outcome-fst-inj[of B U]*

⟨n = dist-el-card B⟩ ⟨M = mk-meas-outcome B U⟩ Bii fc-mats-carrier dimB

by *auto*

show *∀ j < n. meas-outcome-prj (M j) ∈ fc-mats ∧ projector (meas-outcome-prj (M j))*

proof (*intro allI impI conjI*)

fix *j*

assume *j < n*

show *meas-outcome-prj (M j) ∈ fc-mats* **using** *⟨M = mk-meas-outcome B U⟩ assms*

fc-mats-carrier ⟨j < n⟩ ⟨n = dist-el-card B⟩ dim-eq mk-meas-outcome-carrier

```

      dimB dimP unit by blast
    show projector (meas-outcome-prj (M j)) using make-meas-outcome-projectors

      ⟨M = mk-meas-outcome B U⟩
      fc-mats-carrier ⟨n = dist-el-card B⟩ unit ⟨j < n⟩ dimB dimP dim-eq by
blast
    qed
    show ∀ i < n. ∀ j < n. i ≠ j → meas-outcome-prj (M i) * meas-outcome-prj (M
j) =
      0_m dimR dimR
    proof (intro allI impI)
      fix i
      fix j
      assume i < n and j < n and i ≠ j
      thus meas-outcome-prj (M i) * meas-outcome-prj (M j) = 0_m dimR dimR
      using make-meas-outcome-prj-ortho[of U B i j] assms dimB dimP fc-mats-carrier
dim-eq
      by (simp add: ⟨M = mk-meas-outcome B U⟩ ⟨n = dist-el-card B⟩ unit)
    qed
    show sum-mat (λj. meas-outcome-prj (M j)) {.. $n$ } = 1_m dimR using
      mk-meas-outcome-sum-id
      ⟨M = mk-meas-outcome B U⟩ fc-mats-carrier dim-eq ⟨n = dist-el-card B⟩
unit dimB dimP
      by blast
    qed
  qed

lemma (in cpx-sq-mat) make-pm-proj-measurement':
  assumes A ∈ fc-mats
  and hermitian A
  shows proj-measurement (proj-meas-size (make-pm A)) (proj-meas-outcomes (make-pm
A))
  unfolding proj-meas-size-def proj-meas-outcomes-def
  by (rule make-pm-proj-measurement[of A], (simp add: assms)+)

lemma (in cpx-sq-mat) make-pm-projectors:
  assumes A ∈ fc-mats
  and hermitian A
  and i < proj-meas-size (make-pm A)
  shows projector (meas-outcome-prj (proj-meas-outcomes (make-pm A) i))
  proof -
    have proj-measurement (proj-meas-size (make-pm A)) (proj-meas-outcomes (make-pm
A))
    using assms make-pm-proj-measurement' by simp
    thus ?thesis using proj-measurement-proj assms by simp
  qed

lemma (in cpx-sq-mat) make-pm-square:
  assumes A ∈ fc-mats

```

and *hermitian* A
and $i < \text{proj-meas-size } (\text{make-pm } A)$
shows $\text{meas-outcome-prj } (\text{proj-meas-outcomes } (\text{make-pm } A) i) \in \text{fc-mats}$
proof –
have $\text{proj-measurement } (\text{proj-meas-size } (\text{make-pm } A)) (\text{proj-meas-outcomes } (\text{make-pm } A))$
using *assms make-pm-proj-measurement'* **by** *simp*
thus *?thesis using proj-measurement-square assms* **by** *simp*
qed

3.2.4 Properties on the spectrum of a Hermitian matrix

lemma (in *cpx-sq-mat*) *hermitian-schur-decomp'*:
assumes *hermitian* A
and $A \in \text{fc-mats}$
obtains $B U$ **where** *hermitian-decomp* $A B U \wedge$
 $\text{make-pm } A = (\text{dist-el-card } B, \text{mk-meas-outcome } B U)$
proof –
{
have $es: \text{char-poly } A = (\prod (e :: \text{complex}) \leftarrow (\text{eigvals } A). [:- e, 1:])$
using *assms fc-mats-carrier eigvals-poly-length dim-eq* **by** *auto*
obtain $B U Q$ **where** $us: \text{unitary-schur-decomposition } A (\text{eigvals } A) = (B, U, Q)$

by (*cases unitary-schur-decomposition* $A (\text{eigvals } A)$)
hence $pr: \text{similar-mat-wit } A B U (\text{Complex-Matrix.adjoint } U) \wedge \text{diagonal-mat } B \wedge$
 $\text{diag-mat } B = (\text{eigvals } A) \wedge \text{unitary } U \wedge (\forall i < \text{dimR}. B\$\$(i, i) \in \text{Reals})$
using *hermitian-eigenvalue-real assms fc-mats-carrier es dim-eq* **by** *auto*
moreover **have** $\text{dim-row } B = \text{dimR}$ **using** *assms fc-mats-carrier dim-eq similar-mat-wit-dim-row*[of A]
 pr **by** *auto*
ultimately **have** *hermitian-decomp* $A B U$ **unfolding** *hermitian-decomp-def*
by *simp*
moreover **have** $\text{make-pm } A = (\text{dist-el-card } B, \text{mk-meas-outcome } B U)$ **using**
 us *Let-def*
unfolding *make-pm-def hermitian-decomp-def* **by** *simp*
ultimately **have** $\exists B U. \text{hermitian-decomp } A B U \wedge$
 $\text{make-pm } A = (\text{dist-el-card } B, \text{mk-meas-outcome } B U)$ **by** *auto*
}
thus *?thesis using that* **by** *auto*
qed

lemma (in *cpx-sq-mat*) *spectrum-meas-outcome-val-eq*:
assumes *hermitian* A
and $A \in \text{fc-mats}$
and $\text{make-pm } A = (p, M)$
shows $\text{spectrum } A = (\lambda i. \text{meas-outcome-val } (M i)) \{.. < p\}$
proof –
obtain $B U$ **where** $bu: \text{hermitian-decomp } A B U \wedge$

```

    make-pm A = (dist-el-card B, mk-meas-outcome B U)
    using assms hermitian-schur-decomp'[OF assms(1)] by auto
    hence p = dist-el-card B using assms by simp
    have dimB: B ∈ carrier-mat dimR dimR using hermitian-decomp-dim-carrier[of
A] dim-eq bu assms
      fc-mats-carrier by auto
    have Bvals: diag-mat B = eigvals A using bu hermitian-decomp-eigenvalues[of
A B U] by simp
    have Bii:  $\bigwedge i. i < \dim R \implies B\$(i, i) \in \text{Reals}$  using bu hermitian-decomp-real-eigvals[of
A B U]
      dimB by simp
    have diag-elems B = set (eigvals A) using dimB Bvals diag-elems-set-diag-mat[of
B] by simp
    have M = mk-meas-outcome B U using assms bu by simp
    {
      fix i
      assume i < p
      have meas-outcome-val (M i) = Re (diag-idx-to-el B i)
        using  $\langle M = \text{mk-meas-outcome } B \ U \rangle$ 
      unfolding meas-outcome-val-def mk-meas-outcome-def by simp
      also have ... = diag-idx-to-el B i using diag-idx-to-el-real dimB  $\langle i < p \rangle$ 
         $\langle p = \text{dist-el-card } B \rangle$  Bii by simp
      finally have meas-outcome-val (M i) = diag-idx-to-el B i .
    } note eq = this
    have bij-betw (diag-idx-to-el B)  $\{..<\text{dist-el-card } B\}$  (diag-elems B)
      using diag-idx-to-el-bij[of B] by simp
    hence diag-idx-to-el B  $\{..<p\} = \text{diag-elems } B$  using  $\langle p = \text{dist-el-card } B \rangle$ 
      unfolding bij-betw-def by simp
    thus ?thesis using eq  $\langle \text{diag-elems } B = \text{set } (\text{eigvals } A) \rangle$  unfolding spectrum-def
by auto
qed

```

```

lemma (in cpx-sq-mat) spectrum-meas-outcome-val-eq':
  assumes hermitian A
  and A ∈ fc-mats
  and make-pm A = (p, M)
  shows {Re x | x. x ∈ spectrum A} = (λi. meas-outcome-val (M i))  $\{..<p\}$ 
proof
  show {Re x | x. x ∈ spectrum A} ⊆ (λi. meas-outcome-val (M i))  $\{..<p\}$ 
    using spectrum-meas-outcome-val-eq assms by force
  show (λi. meas-outcome-val (M i))  $\{..<p\} \subseteq \{ \text{Re } x \mid x. x \in \text{spectrum } A \}$ 
    using spectrum-meas-outcome-val-eq assms by force
qed

```

```

lemma (in cpx-sq-mat) make-pm-eigenvalues:
  assumes A ∈ fc-mats
  and hermitian A
  and i < proj-meas-size (make-pm A)
  shows meas-outcome-val (proj-meas-outcomes (make-pm A) i) ∈ spectrum A

```

using *spectrum-meas-outcome-val-eq*[of A] *assms make-pm-decomp* **by** *auto*

lemma (in *cpx-sq-mat*) *make-pm-spectrum*:
assumes $A \in \text{fc-mats}$
and *hermitian* A
and *make-pm* $A = (p, M)$
shows $(\lambda i. \text{meas-outcome-val} (\text{proj-meas-outcomes} (\text{make-pm } A) i)) \text{ ' } \{..< p\} = \text{spectrum } A$
proof
show $(\lambda x. \text{complex-of-real} (\text{meas-outcome-val} (\text{proj-meas-outcomes} (\text{make-pm } A) x))) \text{ ' } \{..< p\} \subseteq \text{spectrum } A$
using *assms make-pm-eigenvalues make-pm-decomp*
by (*metis (mono-tags, lifting) Pair-inject image-subsetI lessThan-iff*)
show $\text{spectrum } A \subseteq (\lambda x. \text{complex-of-real} (\text{meas-outcome-val} (\text{proj-meas-outcomes} (\text{make-pm } A) x))) \text{ ' } \{..< p\}$
by (*metis Pair-inject assms equalityE make-pm-decomp spectrum-meas-outcome-val-eq*)
qed

lemma (in *cpx-sq-mat*) *spectrum-size*:
assumes *hermitian* A
and $A \in \text{fc-mats}$
and *make-pm* $A = (p, M)$
shows $p = \text{card} (\text{spectrum } A)$
proof –
obtain $B U$ **where** *bu: hermitian-decomp* $A B U \wedge \text{make-pm } A = (\text{dist-el-card } B, \text{mk-meas-outcome } B U)$
using *assms hermitian-schur-decomp'*[*OF assms(1)*] **by** *auto*
hence $p = \text{dist-el-card } B$ **using** *assms* **by** *simp*
have $\text{spectrum } A = \text{set} (\text{diag-mat } B)$ **using** *bu hermitian-decomp-eigenvalues*[of $A B U$]
unfolding *spectrum-def* **by** *simp*
also have $\dots = \text{diag-elems } B$ **using** *diag-elems-set-diag-mat*[of B] **by** *simp*
finally have $\text{spectrum } A = \text{diag-elems } B$.
thus *?thesis* **using** $\langle p = \text{dist-el-card } B \rangle$ **unfolding** *dist-el-card-def* **by** *simp*
qed

lemma (in *cpx-sq-mat*) *spectrum-size'*:
assumes *hermitian* A
and $A \in \text{fc-mats}$
shows $\text{proj-meas-size} (\text{make-pm } A) = \text{card} (\text{spectrum } A)$ **using** *spectrum-size*
unfolding *proj-meas-size-def*
by (*metis assms fst-conv surj-pair*)

lemma (in *cpx-sq-mat*) *make-pm-projectors'*:
assumes *hermitian* A
and $A \in \text{fc-mats}$
and $a < \text{card} (\text{spectrum } A)$

shows *projector* (*meas-outcome-prj* (*proj-meas-outcomes* (*make-pm* *A*) *a*))
by (*rule* *make-pm-projectors*, (*simp* *add: assms spectrum-size'*)⁺)

lemma (*in* *cpx-sq-mat*) *meas-outcome-prj-carrier*:
assumes *hermitian* *A*
and *A* ∈ *fc-mats*
and *a* < *card* (*spectrum* *A*)
shows *meas-outcome-prj* (*proj-meas-outcomes* (*make-pm* *A*) *a*) ∈ *fc-mats*
proof (*rule* *proj-measurement-square*)
show *proj-measurement* (*proj-meas-size* (*make-pm* *A*)) (*proj-meas-outcomes* (*make-pm* *A*))
using *make-pm-proj-measurement'* *assms* **by** *simp*
show *a* < *proj-meas-size* (*make-pm* *A*) **using** *assms*
spectrum-size[*of* - *proj-meas-size* (*make-pm* *A*)] *make-pm-decomp*[*of* *A*] **by**
simp
qed

lemma (*in* *cpx-sq-mat*) *meas-outcome-prj-trace-real*:
assumes *hermitian* *A*
and *density-operator* *R*
and *R* ∈ *fc-mats*
and *A* ∈ *fc-mats*
and *a* < *card* (*spectrum* *A*)
shows *Re* (*Complex-Matrix.trace* (*R* * *meas-outcome-prj* (*proj-meas-outcomes* (*make-pm* *A*) *a*))) =
Complex-Matrix.trace (*R* * *meas-outcome-prj* (*proj-meas-outcomes* (*make-pm* *A*) *a*))
proof (*rule* *trace-proj-pos-real*)
show *R* ∈ *carrier-mat* *dimR* *dimR* **using** *fc-mats-carrier* *assms* *dim-eq* **by** *simp*
show *Complex-Matrix.positive* *R* **using** *assms* **unfolding** *density-operator-def*
by *simp*
show *projector* (*meas-outcome-prj* (*proj-meas-outcomes* (*make-pm* *A*) *a*)) **using**
assms
make-pm-projectors' **by** *simp*
show *meas-outcome-prj* (*proj-meas-outcomes* (*make-pm* *A*) *a*) ∈ *carrier-mat*
dimR *dimR*
using *meas-outcome-prj-carrier* *assms*
dim-eq *fc-mats-carrier* **by** *simp*
qed

lemma (*in* *cpx-sq-mat*) *sum-over-spectrum*:
assumes *A* ∈ *fc-mats*
and *hermitian* *A*
and *make-pm* *A* = (*p*, *M*)
shows ($\sum y \in \text{spectrum } A. f y$) = ($\sum i < p. f (\text{meas-outcome-val } (M i))$)
proof (*rule* *sum.reindex-cong*)
show *spectrum* *A* = ($\lambda x. (\text{meas-outcome-val } (M x))$)['] {..*p*}
using *spectrum-meas-outcome-val-eq* *assms* **by** *simp*
show *inj-on* ($\lambda x. \text{complex-of-real } (\text{meas-outcome-val } (M x))$) {..*p*}

proof –
have $\text{inj-on } (\lambda x. (\text{meas-outcome-val } (M x))) \{..<p\}$
using $\text{make-pm-proj-measurement}[of A p M]$ *assms proj-measurement-inj*
by *simp*
thus *?thesis* **by** (*simp add: inj-on-def*)
qed
qed *simp*

lemma (*in cpx-sq-mat*) *sum-over-spectrum'*:
assumes $A \in \text{fc-mats}$
and *hermitian* A
and $\text{make-pm } A = (p, M)$
shows $(\sum y \in \{Re x | x. x \in \text{spectrum } A\}. f y) = (\sum i < p. f (\text{meas-outcome-val } (M i)))$
proof (*rule sum.reindex-cong*)
show $\{Re x | x. x \in \text{spectrum } A\} = (\lambda x. (\text{meas-outcome-val } (M x)))' \{..<p\}$
using *spectrum-meas-outcome-val-eq'* **assms** **by** *simp*
show $\text{inj-on } (\lambda x. (\text{meas-outcome-val } (M x))) \{..<p\}$ **using** $\text{make-pm-proj-measurement}[of A p M]$
assms proj-measurement-inj **by** *simp*
qed *simp*

When a matrix A is decomposed into a projective measurement $\{(\lambda_a, \pi_a)\}$, it can be recovered by the formula $A = \sum \lambda_a \pi_a$.

lemma (*in cpx-sq-mat*) *make-pm-sum*:
assumes $A \in \text{fc-mats}$
and *hermitian* A
and $\text{make-pm } A = (p, M)$
shows $\text{sum-mat } (\lambda i. (\text{meas-outcome-val } (M i)) \cdot_m \text{meas-outcome-prj } (M i)) \{..<p\} = A$
proof –
have $es: \text{char-poly } A = (\prod (e :: \text{complex}) \leftarrow (\text{eigvals } A). [- e, 1:])$
using *assms fc-mats-carrier eigvals-poly-length dim-eq* **by** *auto*
obtain $B U Q$ **where** $us: \text{unitary-schur-decomposition } A (\text{eigvals } A) = (B, U, Q)$

by (*cases unitary-schur-decomposition A (eigvals A), auto*)
then have *similar-mat-wit* $A B U (\text{Complex-Matrix.adjoint } U) \wedge \text{diagonal-mat } B \wedge$
 $\text{diag-mat } B = (\text{eigvals } A)$
 $\wedge \text{unitary } U \wedge (\forall i < \text{dimR}. B\$\$(i, i) \in \text{Reals})$
using *hermitian-eigenvalue-real assms fc-mats-carrier es dim-eq* **by** *auto*
hence $A: A = U * B * (\text{Complex-Matrix.adjoint } U)$ **and** $dB: \text{diagonal-mat } B$
and $Bii: \bigwedge i. i < \text{dimR} \implies B\$\$(i, i) \in \text{Reals}$
and $\text{dimB}: B \in \text{carrier-mat } \text{dimR } \text{dimR}$ **and** $\text{dimP}: U \in \text{carrier-mat } \text{dimR } \text{dimR}$ **and**
 $\text{dimAP}: \text{Complex-Matrix.adjoint } U \in \text{carrier-mat } \text{dimR } \text{dimR}$ **and** *unit: unitary* U

unfolding *similar-mat-wit-def Let-def* **using** *assms fc-mats-carrier* **by** *auto*
hence $mpeq: \text{make-pm } A = (\text{dist-el-card } B, \text{mk-meas-outcome } B U)$ **using** *us*

Let-def
unfolding *make-pm-def by simp*
hence $p = \text{dist-el-card } B$ **using** *assms by simp*
have $M = \text{mk-meas-outcome } B \ U$ **using** *assms mpeq by simp*
have $\text{sum-mat } (\lambda i. \text{meas-outcome-val } (M \ i) \cdot_m \text{meas-outcome-prj } (M \ i)) \{..< p\}$
 $=$
 $\text{sum-mat } (\lambda j. \text{Re } (\text{diag-idx-to-el } B \ j) \cdot_m \text{project-vecs } (\lambda i. \text{zero-col } U \ i)$
 $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B)) \{..< (\text{dist-el-card } B)\}$
using $\langle p = \text{dist-el-card } B \rangle$
 $\langle M = \text{mk-meas-outcome } B \ U \rangle$ **unfolding** *meas-outcome-val-def meas-outcome-prj-def*

mk-meas-outcome-def by simp
also have $\dots = \text{sum-mat}$
 $(\lambda j. \text{sum-mat } (\lambda i. (\text{Re } (\text{diag-idx-to-el } B \ j)) \cdot_m \text{rank-1-proj } (\text{zero-col } U \ i))$
 $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B)) \{..< \text{dist-el-card } B\}$
unfolding *project-vecs-def*
proof (*rule sum-mat-cong*)
show *finite* $\{..< \text{dist-el-card } B\}$ **by** *simp*
show $\bigwedge i. i \in \{..< \text{dist-el-card } B\} \implies$
 $\text{complex-of-real } (\text{Re } (\text{diag-idx-to-el } B \ i)) \cdot_m$
 $\text{sum-mat } (\lambda i. \text{rank-1-proj } (\text{zero-col } U \ i)) (\text{diag-elem-indices } (\text{diag-idx-to-el}$
 $B \ i) \ B)$
 $\in \text{fc-mats}$ **using** *assms fc-mats-carrier dimP project-vecs-def project-vecs-dim*
zero-col-dim
dim-eq by auto
show $\bigwedge i. i \in \{..< \text{dist-el-card } B\} \implies$
 $\text{sum-mat } (\lambda ia. \text{complex-of-real } (\text{Re } (\text{diag-idx-to-el } B \ i)) \cdot_m \text{rank-1-proj}$
 $(\text{zero-col } U \ ia))$
 $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B) \in \text{fc-mats}$ **using** *assms fc-mats-carrier*
dimP
project-vecs-def project-vecs-dim zero-col-dim dim-eq
by (*metis (no-types, lifting) rank-1-proj-carrier cpx-sq-mat-smult sum-mat-carrier*)
show $\bigwedge j. j \in \{..< \text{dist-el-card } B\} \implies$
 $(\text{Re } (\text{diag-idx-to-el } B \ j)) \cdot_m \text{sum-mat } (\lambda i. \text{rank-1-proj } (\text{zero-col } U \ i))$
 $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B) =$
 $\text{sum-mat } (\lambda i. \text{complex-of-real } (\text{Re } (\text{diag-idx-to-el } B \ j)) \cdot_m \text{rank-1-proj } (\text{zero-col}$
 $U \ i))$
 $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B)$
proof –
fix j
assume $j \in \{..< \text{dist-el-card } B\}$
show $(\text{Re } (\text{diag-idx-to-el } B \ j)) \cdot_m \text{sum-mat } (\lambda i. \text{rank-1-proj } (\text{zero-col } U \ i))$
 $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B) =$
 $\text{sum-mat } (\lambda i. (\text{Re } (\text{diag-idx-to-el } B \ j)) \cdot_m \text{rank-1-proj } (\text{zero-col } U \ i))$
 $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B)$
proof (*rule smult-sum-mat*)
show *finite* $(\text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B)$
using *diag-elem-indices-finite[of - B] by simp*
show $\bigwedge i. i \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B \implies$

```

      rank-1-proj (zero-col U i) ∈ fc-mats
      using dim-eq by (metis dimP local.fc-mats-carrier rank-1-proj-carrier
zero-col-dim)
    qed
  qed
  qed
  also have ... = sum-mat
    (λj. sum-mat (λia. (diag-mat B ! ia) ·m rank-1-proj (zero-col U ia))
    (diag-elem-indices (diag-idx-to-el B j) B)) {..dist-el-card B}
  proof (rule sum-mat-cong)
    show finite {..dist-el-card B} by simp
    show ∧i. i ∈ {..dist-el-card B} ⇒
      sum-mat (λia. complex-of-real (Re (diag-idx-to-el B i)) ·m rank-1-proj
(zero-col U ia))
      (diag-elem-indices (diag-idx-to-el B i) B) ∈ fc-mats using assms fc-mats-carrier
dimP
      project-vecs-def project-vecs-dim zero-col-dim dim-eq
    by (metis (no-types, lifting) rank-1-proj-carrier cpx-sq-mat-smult sum-mat-carrier)
    show ∧i. i ∈ {..dist-el-card B} ⇒
      local.sum-mat (λia. (diag-mat B ! ia) ·m rank-1-proj (zero-col U ia))
      (diag-elem-indices (diag-idx-to-el B i) B) ∈ fc-mats using assms fc-mats-carrier
dimP
      project-vecs-def project-vecs-dim zero-col-dim dim-eq
    by (metis (no-types, lifting) rank-1-proj-carrier cpx-sq-mat-smult sum-mat-carrier)
    show ∧i. i ∈ {..dist-el-card B} ⇒
      sum-mat (λia. (Re (diag-idx-to-el B i)) ·m rank-1-proj (zero-col U ia))
      (diag-elem-indices (diag-idx-to-el B i) B) =
      sum-mat (λia. (diag-mat B ! ia) ·m rank-1-proj (zero-col U ia))
      (diag-elem-indices (diag-idx-to-el B i) B)
  proof -
    fix i
    assume i ∈ {..dist-el-card B}
    show sum-mat (λia. (Re (diag-idx-to-el B i)) ·m rank-1-proj (zero-col U ia))
      (diag-elem-indices (diag-idx-to-el B i) B) =
      sum-mat (λia. (diag-mat B ! ia) ·m rank-1-proj (zero-col U ia))
      (diag-elem-indices (diag-idx-to-el B i) B)
  proof (rule sum-mat-cong)
    show finite (diag-elem-indices (diag-idx-to-el B i) B)
      using diag-elem-indices-finite[of - B] by simp
    show ∧ia. ia ∈ diag-elem-indices (diag-idx-to-el B i) B ⇒
      (Re (diag-idx-to-el B i)) ·m rank-1-proj (zero-col U ia) ∈ fc-mats
    using assms fc-mats-carrier dimP project-vecs-def project-vecs-dim zero-col-dim
dim-eq
      by (metis (no-types, lifting) rank-1-proj-carrier cpx-sq-mat-smult)
    show ∧ia. ia ∈ diag-elem-indices (diag-idx-to-el B i) B ⇒
      (diag-mat B !ia) ·m rank-1-proj (zero-col U ia) ∈ fc-mats
    using assms fc-mats-carrier dimP project-vecs-def project-vecs-dim zero-col-dim
dim-eq
      by (metis (no-types, lifting) rank-1-proj-carrier cpx-sq-mat-smult)

```

show $\bigwedge i. ia \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B \implies$
 $(\text{Re } (\text{diag-idx-to-el } B \ i)) \cdot_m \text{rank-1-proj } (\text{zero-col } U \ ia) =$
 $(\text{diag-mat } B \ ! \ ia) \cdot_m \text{rank-1-proj } (\text{zero-col } U \ ia)$
proof –
fix ia
assume $ia: ia \in \text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B$
hence $ia < \text{dim-row } B$ **using** $\text{diag-elem-indices-elem}[of \ ia - B]$ **by** simp
have $\text{Re } (\text{diag-idx-to-el } B \ i) = \text{Re } (B \ \$\$ \ (ia, \ ia))$
using $\text{diag-elem-indices-elem}[of \ ia - B]$ ia **by** simp
also have $\dots = B \ \$\$ \ (ia, \ ia)$ **using** Bii **using** $\langle ia < \text{dim-row } B \rangle \ \text{dim}B$
of-real-Re **by** blast
also have $\dots = \text{diag-mat } B \ ! \ ia$ **using** $\langle ia < \text{dim-row } B \rangle$ **unfolding**
diag-mat-def **by** simp
finally have $\text{Re } (\text{diag-idx-to-el } B \ i) = \text{diag-mat } B \ ! \ ia \ .$
thus $(\text{Re } (\text{diag-idx-to-el } B \ i)) \cdot_m \text{rank-1-proj } (\text{zero-col } U \ ia) =$
 $(\text{diag-mat } B \ ! \ ia) \cdot_m \text{rank-1-proj } (\text{zero-col } U \ ia)$ **by** simp
qed
qed
qed
qed
also have $\dots = \text{sum-mat}$
 $(\lambda i. (\text{diag-mat } B \ ! \ i) \cdot_m \text{rank-1-proj } (\text{zero-col } U \ i))$
 $(\bigcup_{j < \text{dist-el-card } B} \text{diag-elem-indices } (\text{diag-idx-to-el } B \ j) \ B)$
unfolding project-vecs-def sum-mat-def
proof (*rule disj-family-sum-with[symmetric]*)
show $\text{finite } \{.. < \text{dist-el-card } B\}$ **by** simp
show $\forall j. (\text{diag-mat } B \ ! \ j) \cdot_m \text{rank-1-proj } (\text{zero-col } U \ j) \in \text{fc-mats}$ **using** assms
fc-mats-carrier $\text{dim}P$
project-vecs-def project-vecs-dim zero-col-dim dim-eq
by (*metis* (*no-types*, *lifting*) $\text{rank-1-proj-carrier}$ cpx-sq-mat-smult)
show $\bigwedge i. i \in \{.. < \text{dist-el-card } B\} \implies \text{finite } (\text{diag-elem-indices } (\text{diag-idx-to-el } B \ i) \ B)$
by (*simp* *add: diag-elem-indices-finite*)
show $\text{disjoint-family-on } (\lambda n. \text{diag-elem-indices } (\text{diag-idx-to-el } B \ n) \ B)$
 $\{.. < \text{dist-el-card } B\}$
using $\text{diag-elem-indices-disjoint}[of \ B]$ $\text{dim}B$ dim-eq **by** simp
qed
also have $\dots = \text{sum-mat } (\lambda i. (\text{diag-mat } B \ ! \ i) \cdot_m \text{rank-1-proj } (\text{zero-col } U \ i)) \ \{.. <$
 $\text{dim}R\}$
using $\text{diag-elem-indices-union}[of \ B]$ $\text{dim}B$ dim-eq **by** simp
also have $\dots = \text{sum-mat } (\lambda i. (\text{diag-mat } B \ ! \ i) \cdot_m \text{rank-1-proj } (\text{Matrix.col } U \ i))$
 $\{.. < \text{dim}R\}$
proof (*rule sum-mat-cong*, *simp*)
show $\text{res: } \bigwedge i. i \in \{.. < \text{dim}R\} \implies (\text{diag-mat } B \ ! \ i) \cdot_m \text{rank-1-proj } (\text{zero-col } U \ i) \in \text{fc-mats}$
using assms fc-mats-carrier $\text{dim}P$ project-vecs-def project-vecs-dim zero-col-dim
 dim-eq
by (*metis* (*no-types*, *lifting*) $\text{rank-1-proj-carrier}$ cpx-sq-mat-smult)
show $\bigwedge i. i \in \{.. < \text{dim}R\} \implies (\text{diag-mat } B \ ! \ i) \cdot_m \text{rank-1-proj } (\text{Matrix.col } U \ i)$

```

∈ fc-mats
  using assms fc-mats-carrier dimP project-vecs-def project-vecs-dim zero-col-dim
  by (metis res lessThan-iff zero-col-col)
  show  $\bigwedge i. i \in \{..<dimR\} \implies (diag\text{-}mat\ B \ ! \ i) \cdot_m rank\text{-}1\text{-}proj\ (zero\text{-}col\ U\ i) =$ 
    (diag-mat B ! i)  $\cdot_m rank\text{-}1\text{-}proj\ (Matrix.col\ U\ i)$ 
  by (simp add: zero-col-col)
qed
also have ... = U * B * Complex-Matrix.adjoint U
proof (rule weighted-sum-rank-1-proj-unitary)
  show diagonal-mat B using dB .
  show Complex-Matrix.unitary U using unit .
  show U ∈ fc-mats using fc-mats-carrier dim-eq dimP by simp
  show B ∈ fc-mats using fc-mats-carrier dim-eq dimB by simp
qed
also have ... = A using A by simp
finally show ?thesis .
qed

lemma (in cpx-sq-mat) trace-hermitian-pos-real:
  fixes f::'a ⇒ real
  assumes hermitian A
  and Complex-Matrix.positive R
  and A ∈ fc-mats
  and R ∈ fc-mats
shows Complex-Matrix.trace (R * A) =
  Re (Complex-Matrix.trace (R * A))
proof -
  define prj-mems where prj-mems = make-pm A
  define p where p = proj-meas-size prj-mems
  define meas where meas = proj-meas-outcomes prj-mems
  have tre: Complex-Matrix.trace (R * A) =
    Complex-Matrix.trace (R *
      sum-mat (λi. (meas-outcome-val (meas i)) ·m meas-outcome-prj (meas i)) {..
  )
  using make-pm-sum assms meas-def p-def proj-meas-size-def proj-meas-outcomes-def
  prj-mems-def
    meas-outcome-val-def meas-outcome-prj-def by auto
  also have ... = Re (Complex-Matrix.trace (R *
    sum-mat (λi. (meas-outcome-val (meas i)) ·m meas-outcome-prj (meas i)) {..
  )
  proof (rule trace-sum-mat-proj-pos-real, (auto simp add: assms))
    fix i
    assume i < p
    thus projector (meas-outcome-prj (meas i)) using make-pm-projectors assms
    unfolding p-def meas-def prj-mems-def by simp
    show meas-outcome-prj (meas i) ∈ fc-mats using make-pm-square assms ⟨i <
  p⟩
  unfolding p-def meas-def prj-mems-def by simp
qed

```

also have ... = $Re (Complex-Matrix.trace (R * A))$ using *tre* by *simp*
 finally show *?thesis* .
 qed

lemma (in *cpx-sq-mat*) *hermitian-Re-spectrum*:

assumes *hermitian A*

and $A \in fc\text{-mats}$

and $\{Re\ x \mid x \in spectrum\ A\} = \{a, b\}$

shows $spectrum\ A = \{complex\text{-of-real}\ a, complex\text{-of-real}\ b\}$

proof

have *ar*: $\bigwedge (a::complex). a \in spectrum\ A \implies Re\ a = a$ using *hermitian-spectrum-real*
assms by *simp*

show $spectrum\ A \subseteq \{complex\text{-of-real}\ a, complex\text{-of-real}\ b\}$

proof

fix *x*

assume $x \in spectrum\ A$

hence $Re\ x = x$ using *ar* by *simp*

have $Re\ x \in \{a, b\}$ using $\langle x \in spectrum\ A \rangle$ *assms* by *blast*

thus $x \in \{complex\text{-of-real}\ a, complex\text{-of-real}\ b\}$ using $\langle Re\ x = x \rangle$ by *auto*

qed

show $\{complex\text{-of-real}\ a, complex\text{-of-real}\ b\} \subseteq spectrum\ A$

proof

fix *x*

assume $x \in \{complex\text{-of-real}\ a, complex\text{-of-real}\ b\}$

hence $x \in \{complex\text{-of-real}\ (Re\ x) \mid x \in spectrum\ A\}$ using *assms*

proof –

have $\bigwedge r. r \notin \{a, b\} \vee (\exists c. r = Re\ c \wedge c \in spectrum\ A)$

using $\langle \{Re\ x \mid x \in spectrum\ A\} = \{a, b\} \rangle$ by *blast*

then show *?thesis*

using $\langle x \in \{complex\text{-of-real}\ a, complex\text{-of-real}\ b\} \rangle$ by *blast*

qed

thus $x \in spectrum\ A$ using *ar* by *auto*

qed

qed

3.2.5 Similar properties for eigenvalues rather than spectrum indices

In this part we go the other way round: given an eigenvalue of *A*, *spectrum_to_pm_idx* permits to retrieve its index in the associated projective measurement

definition (in *cpx-sq-mat*) *spectrum-to-pm-idx*

where *spectrum-to-pm-idx A x = (let (B,U,-) = unitary-schur-decomposition A*

(eigvals A) in

diag-el-to-idx B x)

lemma (in *cpx-sq-mat*) *spectrum-to-pm-idx-bij*:

assumes *hermitian A*

and $A \in fc\text{-mats}$

shows *bij-betw (spectrum-to-pm-idx A) (spectrum A) {.. $card (spectrum A)$ }*

proof –

define p **where** $p = \text{proj-meas-size } (\text{make-pm } A)$

define M **where** $M = \text{proj-meas-outcomes } (\text{make-pm } A)$

have $es: \text{char-poly } A = (\prod (e :: \text{complex}) \leftarrow (\text{eigvals } A). [:- e, 1:])$

using $assms \text{fc-mats-carrier eigvals-poly-length dim-eq}$ **by** $auto$

obtain $B \ U \ Q$ **where** $us: \text{unitary-schur-decomposition } A (\text{eigvals } A) = (B, U, Q)$

by $(\text{cases } \text{unitary-schur-decomposition } A (\text{eigvals } A))$

hence $pr: \text{similar-mat-wit } A \ B \ U \ (\text{Complex-Matrix.adjoint } U) \wedge$
 $\text{diag-mat } B = (\text{eigvals } A)$

using $\text{hermitian-eigenvalue-real } assms \text{fc-mats-carrier } es \ \text{dim-eq}$ **by** $auto$

have $(p, M) = \text{make-pm } A$ **unfolding** $p\text{-def } M\text{-def}$ **using** make-pm-decomp **by**
 simp

hence $p = \text{dist-el-card } B$ **using** $assms \ us$ **unfolding** make-pm-def **by** simp

have $\text{dim}B: B \in \text{carrier-mat } \text{dim}R \ \text{dim}R$ **using** $\text{dim-eq } pr \ \text{assms}$
 fc-mats-carrier **unfolding** $\text{similar-mat-wit-def}$ **by** $auto$

have $Bvals: \text{diag-mat } B = \text{eigvals } A$ **using** $pr \ \text{hermitian-decomp-eigenvalues[of}$
 $A \ B \ U]$ **by** simp

have $\text{diag-elems } B = \text{spectrum } A$ **unfolding** spectrum-def **using** $\text{dim}B \ Bvals$
 $\text{diag-elems-set-diag-mat[of } B]$ **by** simp

moreover **have** $\text{dist-el-card } B = \text{card } (\text{spectrum } A)$ **using** $\text{spectrum-size[of } A \ p$
 $M]$ $assms$

$\langle p, M \rangle = \text{make-pm } A \ \langle p = \text{dist-el-card } B \rangle$ **by** simp

ultimately show $\text{bij-betw } (\text{spectrum-to-pm-idx } A) \ (\text{spectrum } A) \ \{.. < \text{card } (\text{spectrum}$
 $A)\}$

using $\text{diag-el-to-idx-bij } us$ **unfolding** $\text{spectrum-to-pm-idx-def } \text{Let-def}$
by $(\text{metis } (\text{mono-tags}, \text{lifting}) \text{bij-betw-cong case-prod-conv})$

qed

lemma **(in** cpx-sq-mat **)** $\text{spectrum-to-pm-idx-lt-card}$:

assumes $A \in \text{fc-mats}$

and $\text{hermitian } A$

and $a \in \text{spectrum } A$

shows $\text{spectrum-to-pm-idx } A \ a < \text{card } (\text{spectrum } A)$ **using** $\text{spectrum-to-pm-idx-bij[of}$
 $A]$ $assms$

by $(\text{meson } \text{bij-betw-apply lessThan-iff})$

lemma **(in** cpx-sq-mat **)** $\text{spectrum-to-pm-idx-inj}$:

assumes $\text{hermitian } A$

and $A \in \text{fc-mats}$

shows $\text{inj-on } (\text{spectrum-to-pm-idx } A) \ (\text{spectrum } A)$ **using** $assms \ \text{spectrum-to-pm-idx-bij}$
unfolding bij-betw-def **by** simp

lemma **(in** cpx-sq-mat **)** $\text{spectrum-meas-outcome-val-inv}$:

assumes $A \in \text{fc-mats}$

and $\text{hermitian } A$

and $\text{make-pm } A = (p, M)$

and $i < p$

shows $\text{spectrum-to-pm-idx } A \ (\text{meas-outcome-val } (M \ i)) = i$

proof –

have es : $char\text{-}poly\ A = (\prod (e :: complex) \leftarrow (eigvals\ A)). [:-\ e,\ 1:]$
using $assms\ fc\text{-}mats\text{-}carrier\ eigvals\text{-}poly\text{-}length\ dim\text{-}eq$ **by** $auto$
obtain $B\ U\ Q$ **where** us : $unitary\text{-}schur\text{-}decomposition\ A\ (eigvals\ A) = (B, U, Q)$

by $(cases\ unitary\text{-}schur\text{-}decomposition\ A\ (eigvals\ A))$
hence pr : $similar\text{-}mat\text{-}wit\ A\ B\ U\ (Complex\text{-}Matrix.\text{adjoint}\ U) \wedge$
 $diag\text{-}mat\ B = (eigvals\ A) \wedge (\forall i < dimR. B\$\$(i, i) \in Reals)$
using $hermitian\text{-}eigenvalue\text{-}real\ assms\ fc\text{-}mats\text{-}carrier\ es\ dim\text{-}eq$ **by** $auto$
have $dim\text{-}row\ B = dimR$ **using** $assms\ fc\text{-}mats\text{-}carrier\ dim\text{-}eq\ similar\text{-}mat\text{-}wit\text{-}dim\text{-}row$ [of
 A]
pr **by** $auto$
hence $(p, M) = (dist\text{-}el\text{-}card\ B,\ mk\text{-}meas\text{-}outcome\ B\ U)$ **using** $assms\ us$
unfolding $make\text{-}pm\text{-}def$ **by** $simp$
hence $M = mk\text{-}meas\text{-}outcome\ B\ U$ **by** $simp$
have $meas\text{-}outcome\text{-}val\ (M\ i) = Re\ (diag\text{-}idx\text{-}to\text{-}el\ B\ i)$
using $\langle M = mk\text{-}meas\text{-}outcome\ B\ U \rangle$ **unfolding** $mk\text{-}meas\text{-}outcome\text{-}def$
 $meas\text{-}outcome\text{-}val\text{-}def$ **by** $simp$
also **have** $\dots = diag\text{-}idx\text{-}to\text{-}el\ B\ i$ **using** pr
 $\langle (p, M) = (dist\text{-}el\text{-}card\ B,\ mk\text{-}meas\text{-}outcome\ B\ U) \rangle$ $\langle dim\text{-}row\ B = dimR \rangle$ $assms$

 $diag\text{-}idx\text{-}to\text{-}el\text{-}real$ **by** $auto$
finally **have** $meas\text{-}outcome\text{-}val\ (M\ i) = diag\text{-}idx\text{-}to\text{-}el\ B\ i$.
hence $spectrum\text{-}to\text{-}pm\text{-}idx\ A\ (meas\text{-}outcome\text{-}val\ (M\ i)) =$
 $spectrum\text{-}to\text{-}pm\text{-}idx\ A\ (diag\text{-}idx\text{-}to\text{-}el\ B\ i)$ **by** $simp$
also **have** $\dots = diag\text{-}el\text{-}to\text{-}idx\ B\ (diag\text{-}idx\text{-}to\text{-}el\ B\ i)$ **unfolding** $spectrum\text{-}to\text{-}pm\text{-}idx\text{-}def$

using us **by** $simp$
also **have** $\dots = i$ **using** $assms$ **unfolding** $diag\text{-}el\text{-}to\text{-}idx\text{-}def$
using $\langle (p, M) = (dist\text{-}el\text{-}card\ B,\ mk\text{-}meas\text{-}outcome\ B\ U) \rangle$ $bij\text{-}betw\text{-}inv\text{-}into\text{-}left$
 $diag\text{-}idx\text{-}to\text{-}el\text{-}bij$ **by** $fastforce$
finally **show** $?thesis$.
qed

lemma (in $cpx\text{-}sq\text{-}mat$) $meas\text{-}outcome\text{-}val\text{-}spectrum\text{-}inv$:

assumes $A \in fc\text{-}mats$
and $hermitian\ A$
and $x \in spectrum\ A$
and $make\text{-}pm\ A = (p, M)$
shows $meas\text{-}outcome\text{-}val\ (M\ (spectrum\text{-}to\text{-}pm\text{-}idx\ A\ x)) = x$
proof –
have es : $char\text{-}poly\ A = (\prod (e :: complex) \leftarrow (eigvals\ A)). [:-\ e,\ 1:]$
using $assms\ fc\text{-}mats\text{-}carrier\ eigvals\text{-}poly\text{-}length\ dim\text{-}eq$ **by** $auto$
obtain $B\ U\ Q$ **where** us : $unitary\text{-}schur\text{-}decomposition\ A\ (eigvals\ A) = (B, U, Q)$

by $(cases\ unitary\text{-}schur\text{-}decomposition\ A\ (eigvals\ A))$
hence pr : $similar\text{-}mat\text{-}wit\ A\ B\ U\ (Complex\text{-}Matrix.\text{adjoint}\ U) \wedge diagonal\text{-}mat$
 $B \wedge$
 $diag\text{-}mat\ B = (eigvals\ A) \wedge unitary\ U \wedge (\forall i < dimR. B\$\$(i, i) \in Reals)$

using *hermitian-eigenvalue-real assms fc-mats-carrier es dim-eq* **by** *auto*
have $\dim\text{-row } B = \dim R$ **using** *assms fc-mats-carrier dim-eq similar-mat-wit-dim-row*[of
A]
pr **by** *auto*
hence $(p, M) = (\text{dist-el-card } B, \text{mk-meas-outcome } B \ U)$ **using** *assms us*
unfolding *make-pm-def* **by** *simp*
hence $M = \text{mk-meas-outcome } B \ U$ **by** *simp*
have $\text{diag-mat } B = (\text{eigvals } A)$ **using** *pr* **by** *simp*
hence $x \in \text{set } (\text{diag-mat } B)$ **using** $\langle \text{diag-mat } B = \text{eigvals } A \rangle$ *assms* **unfolding**
spectrum-def **by** *simp*
hence $x \in \text{diag-elems } B$ **using** *diag-elems-set-diag-mat*[of *B*] **by** *simp*
hence $\text{diag-idx-to-el } B (\text{diag-el-to-idx } B \ x) = x$ **unfolding** *diag-el-to-idx-def*
by *(meson bij-betw-inv-into-right diag-idx-to-el-bij)*
moreover **have** $\text{spectrum-to-pm-idx } A \ x = \text{diag-el-to-idx } B \ x$ **unfolding** *spectrum-*
trum-to-pm-idx-def
using *us* **by** *simp*
moreover **have** $\text{meas-outcome-val } (M (\text{spectrum-to-pm-idx } A \ x)) =$
 $\text{Re } (\text{diag-idx-to-el } B (\text{diag-el-to-idx } B \ x))$ **using** $\langle M = \text{mk-meas-outcome } B \ U \rangle$
unfolding *mk-meas-outcome-def meas-outcome-val-def* **by** *(simp add: calcula-*
tion(2))
ultimately show *?thesis* **by** *simp*
qed

definition (in *cpx-sq-mat*) *eigen-projector* **where**
eigen-projector $A \ a =$
 $\text{meas-outcome-prj } ((\text{proj-meas-outcomes } (\text{make-pm } A)) (\text{spectrum-to-pm-idx } A \ a))$

lemma (in *cpx-sq-mat*) *eigen-projector-carrier*:
assumes $A \in \text{fc-mats}$
and $a \in \text{spectrum } A$
and *hermitian* A
shows *eigen-projector* $A \ a \in \text{fc-mats}$ **unfolding** *eigen-projector-def*
using *meas-outcome-prj-carrier*[of $A \ \text{spectrum-to-pm-idx } A \ a$]
spectrum-to-pm-idx-lt-card[of $A \ a$] *assms* **by** *simp*

We obtain the following result, which is similar to `make_pm_sum` but with
 a sum on the elements of the spectrum of Hermitian matrix A , which is a
 more standard statement of the spectral decomposition theorem.

lemma (in *cpx-sq-mat*) *make-pm-sum'*:
assumes $A \in \text{fc-mats}$
and *hermitian* A
shows $\text{sum-mat } (\lambda a. \ a \cdot_m (\text{eigen-projector } A \ a)) (\text{spectrum } A) = A$
proof –
define p **where** $p = \text{proj-meas-size } (\text{make-pm } A)$
define M **where** $M = \text{proj-meas-outcomes } (\text{make-pm } A)$
have $(p, M) = \text{make-pm } A$ **unfolding** *p-def M-def* **using** *make-pm-decomp* **by**
simp
define g **where**
 $g = (\lambda i. \ (\text{if } i < p$

```

    then complex-of-real (meas-outcome-val (M i)) ·m meas-outcome-prj (M i)
    else (0m dimR dimC)))
  have g: ∀ x. g x ∈ fc-mats
  proof
    fix x
    show g x ∈ fc-mats
    proof (cases x < p)
      case True
        hence (meas-outcome-val (M x)) ·m meas-outcome-prj (M x) ∈ fc-mats
          using meas-outcome-prj-carrier
          spectrum-size assms cpx-sq-mat-smult make-pm-proj-measurement proj-measurement-square

          ⟨(p,M) = make-pm A⟩ by metis
        then show ?thesis unfolding g-def using True by simp
      next
        case False
          then show ?thesis unfolding g-def using zero-mem by simp
    qed
  qed
  define h where h = (λa. (if a ∈ (spectrum A) then a ·m eigen-projector A a else
    (0m dimR dimC)))
  have h: ∀ x. h x ∈ fc-mats
  proof
    fix x
    show h x ∈ fc-mats
    proof (cases x ∈ spectrum A)
      case True
        then show ?thesis unfolding h-def using eigen-projector-carrier[of A x]
          assms True
          by (simp add: cpx-sq-mat-smult)
      next
        case False
          then show ?thesis unfolding h-def using zero-mem by simp
    qed
  qed
  have inj-on (spectrum-to-pm-idx A) (spectrum A) using assms spectrum-to-pm-idx-inj
  by simp
  moreover have {..<p} = spectrum-to-pm-idx A ‘ spectrum A using ⟨(p,M) =
  make-pm A⟩
    spectrum-to-pm-idx-bij spectrum-size unfolding bij-betw-def
    by (metis assms(1) assms(2))
  moreover have ∧x. x ∈ spectrum A ⇒ g (spectrum-to-pm-idx A x) = h x
  proof –
    fix a
    assume a ∈ spectrum A
    hence Re a = a using hermitian-spectrum-real assms by simp
    have spectrum-to-pm-idx A a < p using spectrum-to-pm-idx-lt-card[of A] spec-
    trum-size assms
      ⟨a ∈ spectrum A⟩ ⟨(p,M) = make-pm A⟩ by metis

```

have g (*spectrum-to-pm-idx* A a) =
 (*meas-outcome-val* (M (*spectrum-to-pm-idx* A a))) \cdot_m
meas-outcome-prj (M (*spectrum-to-pm-idx* A a))
using \langle *spectrum-to-pm-idx* A $a < p$ \rangle **unfolding** g -def **by** *simp*
also have $\dots = a \cdot_m$ *meas-outcome-prj* (M (*spectrum-to-pm-idx* A a))
using *meas-outcome-val-spectrum-inv*[of A *Re* a] \langle *Re* $a = a$ \rangle *assms* $\langle a \in$
spectrum $A \rangle$
 $\langle(p, M) =$ *make-pm* $A \rangle$ **by** *metis*
also have $\dots = h$ a **using** $\langle a \in$ *spectrum* $A \rangle$ **unfolding** *eigen-projector-def*
M-def h -def **by** *simp*
finally show g (*spectrum-to-pm-idx* A a) = h a .
qed
ultimately have *sum-mat* h (*spectrum* A) =
sum-mat g (*spectrum-to-pm-idx* A ‘*spectrum* A ’ **unfolding** *sum-mat-def*
using *sum-with-reindex-cong*[*symmetric*, of g h *spectrum-to-pm-idx* A *spectrum*
 A $\{.. < p\}$]
 g h **by** *simp*
also have $\dots =$ *sum-mat* g $\{.. < p\}$ **using** $\langle\{.. < p\} =$ *spectrum-to-pm-idx* A ‘
spectrum $A \rangle$ **by** *simp*
also have $\dots =$ *sum-mat* ($\lambda i.$ (*meas-outcome-val* (M i)) \cdot_m *meas-outcome-prj* (M
 i)) $\{.. < p\}$
proof (*rule* *sum-mat-cong*, *simp*)
fix i
assume $i \in \{.. < p\}$
hence $i < p$ **by** *simp*
show g $i \in$ *fc-mats* **using** g **by** *simp*
show g $i =$ (*meas-outcome-val* (M i)) \cdot_m *meas-outcome-prj* (M i) **unfolding**
 g -def
using $\langle i < p \rangle$ **by** *simp*
show (*meas-outcome-val* (M i)) \cdot_m *meas-outcome-prj* (M i) \in *fc-mats*
using *meas-outcome-prj-carrier* *spectrum-size* *assms* *cpx-sq-mat-smult*
make-pm-proj-measurement *proj-measurement-square* $\langle i < p \rangle$ $\langle(p, M) =$
make-pm $A \rangle$ **by** *metis*
qed
also have $\dots = A$ **using** *make-pm-sum* *assms* $\langle(p, M) =$ *make-pm* $A \rangle$ **unfolding**
 g -def **by** *simp*
finally have *sum-mat* h (*spectrum* A) = A .
moreover have *sum-mat* h (*spectrum* A) = *sum-mat* ($\lambda a.$ $a \cdot_m$ (*eigen-projector*
 A a)) (*spectrum* A)
proof (*rule* *sum-mat-cong*)
show *finite* (*spectrum* A) **using** *spectrum-finite*[of A] **by** *simp*
fix i
assume $i \in$ *spectrum* A
thus h $i = i \cdot_m$ *eigen-projector* A i **unfolding** h -def **by** *simp*
show h $i \in$ *fc-mats* **using** h **by** *simp*
show $i \cdot_m$ *eigen-projector* A $i \in$ *fc-mats* **using** *eigen-projector-carrier*[of A i]
assms
 $\langle i \in$ *spectrum* $A \rangle$ **by** (*metis* $\langle h$ $i = i \cdot_m$ *eigen-projector* A $i \rangle$ h)
qed

ultimately show *?thesis* by *simp*
 qed

end

theory *CHSH-Inequality* imports
Projective-Measurements

begin

4 The CHSH inequality

The local hidden variable assumption for quantum mechanics was developed to make the case that quantum theory was incomplete. In this part we formalize the CHSH inequality, which provides an upper-bound of a quantity involving expectations in a probability space, and exploit this inequality to show that the local hidden variable assumption does not hold.

4.1 Inequality statement

lemma *chsh-real*:

fixes $A0::real$

assumes $|A0 * B1| \leq 1$

and $|A0 * B0| \leq 1$

and $|A1 * B0| \leq 1$

and $|A1 * B1| \leq 1$

shows $|A0 * B1 - A0 * B0 + A1 * B0 + A1*B1| \leq 2$

proof –

have $A0 * B1 - A0 * B0 = A0 * B1 - A0 * B0 + A0 * B1 * A1 * B0 - A0 * B1 * A1 * B0$ by *simp*

also have $\dots = A0 * B1 * (1 + A1*B0) - A0 * B0 * (1 + A1* B1)$

by (*metis (no-types, opaque-lifting) add-diff-cancel-right calculation diff-add-eq group-cancel.sub1 mult.commute mult.right-neutral vector-space-over-itself.scale-left-commute vector-space-over-itself.scale-right-diff-distrib vector-space-over-itself.scale-right-distrib vector-space-over-itself.scale-scale*)

finally have $A0 * B1 - A0 * B0 = A0 * B1 * (1 + A1*B0) - A0 * B0 * (1 + A1* B1)$.

hence $|A0 * B1 - A0 * B0| \leq |A0 * B1 * (1 + A1*B0)| + |A0 * B0 * (1 + A1* B1)|$ by *simp*

also have $\dots = |A0 * B1| * |(1 + A1*B0)| + |A0 * B0| * |(1 + A1* B1)|$ by (*simp add: abs-mult*)

also have $\dots \leq |(1 + A1*B0)| + |(1 + A1* B1)|$

proof –
have $|A0 * B1| * |(1 + A1*B0)| \leq |(1 + A1*B0)|$
using *assms(1) mult-left-le-one-le[of |(1 + A1*B0)|]* **by** *simp*
moreover have $|A0 * B0| * |(1 + A1* B1)| \leq |(1 + A1* B1)|$
using *assms mult-left-le-one-le[of |(1 + A1*B1)|]* **by** *simp*
ultimately show *?thesis* **by** *simp*
qed
also have $\dots = 1 + A1*B0 + 1 + A1* B1$ **using** *assms* **by** (*simp add: abs-le-iff*)

also have $\dots = 2 + A1 * B0 + A1 * B1$ **by** *simp*
finally have *pls*: $|A0 * B1 - A0 * B0| \leq 2 + A1 * B0 + A1 * B1$.
have $A0 * B1 - A0 * B0 = A0 * B1 - A0 * B0 + A0 * B1 * A1 * B0 -$
 $A0 * B1 * A1 * B0$ **by** *simp*
also have $\dots = A0 * B1 * (1 - A1*B0) - A0 * B0 * (1 - A1* B1)$
proof –
have $A0 * (B1 - (B0 - A1 * (B1 * B0)) - A1 * (B1 * B0)) = A0 * (B1$
 $- B0)$
by *fastforce*
then show *?thesis*
by (*metis (no-types) add.commute calculation diff-diff-add mult.right-neutral*
vector-space-over-itself.scale-left-commute
vector-space-over-itself.scale-right-diff-distrib vector-space-over-itself.scale-scale)
qed
finally have $A0 * B1 - A0 * B0 = A0 * B1 * (1 - A1*B0) - A0 * B0 * (1$
 $- A1* B1)$.
hence $|A0 * B1 - A0 * B0| \leq |A0 * B1 * (1 - A1*B0)| + |A0 * B0 * (1 -$
 $A1* B1)|$ **by** *simp*
also have $\dots = |A0 * B1| * |(1 - A1*B0)| + |A0 * B0| * |(1 - A1* B1)|$ **by**
(*simp add: abs-mult*)
also have $\dots \leq |(1 - A1*B0)| + |(1 - A1* B1)|$
proof –
have $|A0 * B1| * |(1 - A1*B0)| \leq |(1 - A1*B0)|$
using *assms(1) mult-left-le-one-le[of |(1 - A1*B0)|]* **by** *simp*
moreover have $|A0 * B0| * |(1 - A1* B1)| \leq |(1 - A1* B1)|$
using *assms mult-left-le-one-le[of |(1 - A1*B1)|]* **by** *simp*
ultimately show *?thesis* **by** *simp*
qed
also have $\dots = 1 - A1*B0 + 1 - A1* B1$ **using** *assms* **by** (*simp add: abs-le-iff*)

also have $\dots = 2 - A1 * B0 - A1 * B1$ **by** *simp*
finally have *mns*: $|A0 * B1 - A0 * B0| \leq 2 - (A1 * B0 + A1 * B1)$ **by** *simp*
have *ls*: $|A0 * B1 - A0 * B0| \leq 2 - |A1 * B0 + A1 * B1|$
proof (*cases* $0 \leq A1 * B0 + A1 * B1$)
case *True*
then show *?thesis* **using** *mns* **by** *simp*
next
case *False*
then show *?thesis* **using** *pls* **by** *simp*
qed

have $|A0 * B1 - A0 * B0 + A1 * B0 + A1 * B1| \leq |A0 * B1 - A0 * B0|$
 $+ |A1 * B0 + A1 * B1|$
by *simp*
also have $\dots \leq 2$ **using** *ls* **by** *simp*
finally show *?thesis* .
qed

lemma (in *prob-space*) *chsh-expect*:

fixes $A0::'a \Rightarrow real$
assumes $AE\ w\ in\ M. |A0\ w| \leq 1$
and $AE\ w\ in\ M. |A1\ w| \leq 1$
and $AE\ w\ in\ M. |B0\ w| \leq 1$
and $AE\ w\ in\ M. |B1\ w| \leq 1$
and *integrable* $M\ (\lambda w. A0\ w * B1\ w)$
and *integrable* $M\ (\lambda w. A1\ w * B1\ w)$
and *integrable* $M\ (\lambda w. A1\ w * B0\ w)$
and *integrable* $M\ (\lambda w. A0\ w * B0\ w)$
shows $|expectation\ (\lambda w. A1\ w * B0\ w) + expectation\ (\lambda w. A0\ w * B1\ w) +$
 $expectation\ (\lambda w. A1\ w * B1\ w) - expectation\ (\lambda w. A0\ w * B0\ w)| \leq 2$
proof –
have *exeq*: $expectation\ (\lambda w. A1\ w * B0\ w) + expectation\ (\lambda w. A0\ w * B1\ w) +$
 $expectation\ (\lambda w. A1\ w * B1\ w) - expectation\ (\lambda w. A0\ w * B0\ w) =$
 $expectation\ (\lambda w. A0\ w * B1\ w - A0\ w * B0\ w + A1\ w * B0\ w + A1\ w * B1$
 $w)$
using *assms by auto*
have $|expectation\ (\lambda w. A0\ w * B1\ w - A0\ w * B0\ w + A1\ w * B0\ w + A1\ w * B1$
 $w)| \leq$
 $expectation\ (\lambda w. |A0\ w * B1\ w - A0\ w * B0\ w + A1\ w * B0\ w + A1\ w * B1$
 $w|)$
using *integral-abs-bound by blast*
also have $\dots \leq 2$
proof (*rule integral-le-const*)
show $AE\ w\ in\ M. |A0\ w * B1\ w - A0\ w * B0\ w + A1\ w * B0\ w + A1\ w * B1$
 $w| \leq (2::real)$
proof (*rule AE-mp*)
show $AE\ w\ in\ M. |A0\ w| \leq 1 \wedge |A1\ w| \leq 1 \wedge |B0\ w| \leq 1 \wedge |B1\ w| \leq 1$
using *assms by simp*
show $AE\ w\ in\ M. |A0\ w| \leq 1 \wedge |A1\ w| \leq 1 \wedge |B0\ w| \leq 1 \wedge |B1\ w| \leq 1 \longrightarrow$
 $|A0\ w * B1\ w - A0\ w * B0\ w + A1\ w * B0\ w + A1\ w * B1\ w| \leq 2$
proof
fix w
assume $w \in space\ M$
show $|A0\ w| \leq 1 \wedge |A1\ w| \leq 1 \wedge |B0\ w| \leq 1 \wedge |B1\ w| \leq 1 \longrightarrow$
 $|A0\ w * B1\ w - A0\ w * B0\ w + A1\ w * B0\ w + A1\ w * B1\ w| \leq 2$
proof
assume *ineq*: $|A0\ w| \leq 1 \wedge |A1\ w| \leq 1 \wedge |B0\ w| \leq 1 \wedge |B1\ w| \leq 1$
show $|A0\ w * B1\ w - A0\ w * B0\ w + A1\ w * B0\ w + A1\ w * B1\ w| \leq 2$
proof (*rule chsh-real*)
show $|A0\ w * B1\ w| \leq 1$ **using** *ineq by (simp add: abs-mult mult-le-one)*

```

    show |A0 w * B0 w| ≤ 1 using ineq by (simp add: abs-mult mult-le-one)
    show |A1 w * B1 w| ≤ 1 using ineq by (simp add: abs-mult mult-le-one)
    show |A1 w * B0 w| ≤ 1 using ineq by (simp add: abs-mult mult-le-one)
  qed
qed
qed
qed
show integrable M (λx. |A0 x * B1 x - A0 x * B0 x + A1 x * B0 x + A1 x
* B1 x|)
proof (rule Bochner-Integration.integrable-abs)
  show integrable M (λx. A0 x * B1 x - A0 x * B0 x + A1 x * B0 x + A1 x
* B1 x)
  using assms by auto
qed
qed
finally show ?thesis using exeq by simp
qed

```

The local hidden variable assumption states that separated quantum measurements are independent. It is standard for this assumption to be stated in a context where the hidden variable admits a density; it is stated here in a more general context involving expectations, with no assumption on the existence of a density.

definition *pos-rv*:: 'a measure ⇒ ('a ⇒ real) ⇒ bool **where**
pos-rv M Xr ≡ Xr ∈ borel-measurable M ∧ (AE w in M. (0::real) ≤ Xr w)

definition *prv-sum*:: 'a measure ⇒ complex Matrix.mat ⇒ (complex ⇒ 'a ⇒ real) ⇒ bool **where**
prv-sum M A Xr ≡ (AE w in M. (∑ a ∈ spectrum A. Xr a w) = 1)

definition (in *cpx-sq-mat*) *lhv* **where**
lhv M A B R XA XB ≡
 prob-space M ∧
 (∀ a ∈ spectrum A. *pos-rv* M (XA a)) ∧
 (*prv-sum* M A XA) ∧
 (∀ b ∈ spectrum B. *pos-rv* M (XB b)) ∧
 (*prv-sum* M B XB) ∧
 (∀ a ∈ spectrum A . ∀ b ∈ spectrum B.
 (integrable M (λw. XA a w * XB b w)) ∧
 integral^L M (λw. XA a w * XB b w) =
 Re (Complex-Matrix.trace(eigen-projector A a * (eigen-projector B b) * R)))

lemma (in *cpx-sq-mat*) *lhv-posl*:
assumes *lhv* M A B R XA XB
shows AE w in M. (∀ a ∈ spectrum A. 0 ≤ XA a w)
proof (rule AE-ball-countable[THEN iffD2])
show countable (spectrum A) **using** spectrum-finite[of A]

by (*simp add: countable-finite*)
 show $\forall a \in \text{spectrum } A. AE\ w\ \text{in } M. 0 \leq XA\ a\ w$ using *assms unfolding lhv-def pos-rv-def by simp*
 qed

lemma (in *cpx-sq-mat*) *lhv-lt1-l*:

assumes *lhv M A B R XA XB*

shows *AE w in M. ($\forall a \in \text{spectrum } A. XA\ a\ w \leq 1$)*

proof (*rule AE-mp*)

show *AE w in M. ($\forall a \in \text{spectrum } A. 0 \leq XA\ a\ w$) \wedge ($\sum a \in \text{spectrum } A. XA\ a\ w$) = 1*

using *lhv-posl assms unfolding lhv-def pos-rv-def prv-sum-def by simp*

show *AE w in M. ($\forall a \in \text{spectrum } A. 0 \leq XA\ a\ w$) \wedge ($\sum a \in \text{spectrum } A. XA\ a\ w$) = 1 \longrightarrow*

($\forall a \in \text{spectrum } A. XA\ a\ w \leq 1$)

proof

fix *w*

assume *w \in space M*

show *($\forall a \in \text{spectrum } A. 0 \leq XA\ a\ w$) \wedge ($\sum a \in \text{spectrum } A. XA\ a\ w$) = 1 \longrightarrow*
($\forall a \in \text{spectrum } A. XA\ a\ w \leq 1$)

proof (*rule impI*)

assume *pr: ($\forall a \in \text{spectrum } A. 0 \leq XA\ a\ w$) \wedge ($\sum a \in \text{spectrum } A. XA\ a\ w$) =*

1

show *$\forall a \in \text{spectrum } A. XA\ a\ w \leq 1$*

proof

fix *a*

assume *a \in spectrum A*

show *$XA\ a\ w \leq 1$*

proof (*rule pos-sum-1-le[*of spectrum A*]*)

show *finite (*spectrum A*) using *spectrum-finite[*of A*] by simp**

show *a \in spectrum A using $\langle a \in \text{spectrum } A \rangle$.*

show *$\forall i \in \text{spectrum } A. 0 \leq XA\ i\ w$ using *pr by simp**

show *($\sum i \in \text{spectrum } A. XA\ i\ w$) = 1 using *pr by simp**

qed

qed

qed

qed

qed

lemma (in *cpx-sq-mat*) *lhv-posr*:

assumes *lhv M A B R XA XB*

shows *AE w in M. ($\forall b \in \text{spectrum } B. 0 \leq XB\ b\ w$)*

proof (*rule AE-ball-countable[*THEN iffD2*]*)

show *countable (*spectrum B*) using *spectrum-finite[*of B*]**

by (*simp add: countable-finite*)

show *$\forall b \in \text{spectrum } B. AE\ w\ \text{in } M. 0 \leq XB\ b\ w$ using *assms unfolding lhv-def pos-rv-def by simp**

qed

lemma (in *cpx-sq-mat*) *lhv-lt1-r*:
assumes *lhv M A B R XA XB*
shows *AE w in M. ($\forall a \in \text{spectrum } B. XB a w \leq 1$)*
proof (rule *AE-mp*)
show *AE w in M. ($\forall a \in \text{spectrum } B. 0 \leq XB a w$) \wedge ($\sum_{a \in \text{spectrum } B} XB a w$) = 1*
using *lhv-posr assms unfolding lhv-def prv-sum-def pos-rv-def by simp*
show *AE w in M. ($\forall a \in \text{spectrum } B. 0 \leq XB a w$) \wedge ($\sum_{a \in \text{spectrum } B} XB a w$) = 1 \longrightarrow*
($\forall a \in \text{spectrum } B. XB a w \leq 1$)
proof
fix *w*
assume *w \in space M*
show *($\forall a \in \text{spectrum } B. 0 \leq XB a w$) \wedge ($\sum_{a \in \text{spectrum } B} XB a w$) = 1 \longrightarrow*
($\forall a \in \text{spectrum } B. XB a w \leq 1$)
proof (rule *impI*)
assume *pr: ($\forall a \in \text{spectrum } B. 0 \leq XB a w$) \wedge ($\sum_{a \in \text{spectrum } B} XB a w$) =*
1
show *$\forall a \in \text{spectrum } B. XB a w \leq 1$*
proof
fix *a*
assume *a \in spectrum B*
show *$XB a w \leq 1$*
proof (rule *pos-sum-1-le[*of spectrum B*]*)
show *finite (*spectrum B*) using *spectrum-finite[*of B*]* by *simp**
show *a \in spectrum B using $\langle a \in \text{spectrum } B \rangle$.*
show *$\forall i \in \text{spectrum } B. 0 \leq XB i w$ using *pr* by *simp**
show *($\sum_{i \in \text{spectrum } B} XB i w$) = 1 using *pr* by *simp**
qed
qed
qed
qed
qed

lemma (in *cpx-sq-mat*) *lhv-AE-propl*:
assumes *lhv M A B R XA XB*
shows *AE w in M. ($\forall a \in \text{spectrum } A. 0 \leq XA a w \wedge XA a w \leq 1$) \wedge ($\sum_{a \in \text{spectrum } A} XA a w$) = 1*
proof (rule *AE-conjI*)
show *AE w in M. $\forall a \in \text{spectrum } A. 0 \leq XA a w \wedge XA a w \leq 1$*
proof (rule *AE-mp*)
show *AE w in M. ($\forall a \in \text{spectrum } A. 0 \leq XA a w$) \wedge ($\forall a \in \text{spectrum } A. XA a w \leq 1$)*
using *assms lhv-posl[*of M A*] lhv-lt1-l[*of M A*] by simp*
show *AE w in M. ($\forall a \in \text{spectrum } A. 0 \leq XA a w$) \wedge ($\forall a \in \text{spectrum } A. XA a w \leq 1$) \longrightarrow*
*($\forall a \in \text{spectrum } A. 0 \leq XA a w \wedge XA a w \leq 1$) by *auto**
qed
show *AE w in M. ($\sum_{a \in \text{spectrum } A} XA a w$) = 1 using *assms unfolding**

lhv-def prv-sum-def
 by *simp*
 qed

lemma (in *cpx-sq-mat*) *lhv-AE-propr*:
 assumes *lhv M A B R XA XB*
 shows *AE w in M. ($\forall a \in \text{spectrum } B. 0 \leq XB \ a \ w \wedge XB \ a \ w \leq 1$) \wedge ($\sum_{a \in \text{spectrum } B. XB \ a \ w} = 1$)*
proof (rule *AE-conjI*)
 show *AE w in M. $\forall a \in \text{spectrum } B. 0 \leq XB \ a \ w \wedge XB \ a \ w \leq 1$*
proof (rule *AE-mp*)
 show *AE w in M. ($\forall a \in \text{spectrum } B. 0 \leq XB \ a \ w$) \wedge ($\forall a \in \text{spectrum } B. XB \ a \ w \leq 1$)*
 using *assms lhv-posr[of M - B] lhv-lt1-r[of M - B]* by *simp*
 show *AE w in M. ($\forall a \in \text{spectrum } B. 0 \leq XB \ a \ w$) \wedge ($\forall a \in \text{spectrum } B. XB \ a \ w \leq 1$) \longrightarrow*
 ($\forall a \in \text{spectrum } B. 0 \leq XB \ a \ w \wedge XB \ a \ w \leq 1$) by *auto*
 qed
 show *AE w in M. ($\sum_{a \in \text{spectrum } B. XB \ a \ w} = 1$)* using *assms unfolding lhv-def prv-sum-def*
 by *simp*
 qed

lemma (in *cpx-sq-mat*) *lhv-integral-eq*:
 fixes *c::real*
 assumes *lhv M A B R XA XB*
 and *a \in spectrum A*
 and *b \in spectrum B*
 shows *integral^L M ($\lambda w. XA \ a \ w * XB \ b \ w$) =*
 *Re (Complex-Matrix.trace(eigen-projector A a * (eigen-projector B b) * R))*
 using *assms unfolding lhv-def* by *simp*

lemma (in *cpx-sq-mat*) *lhv-integrable*:
 fixes *c::real*
 assumes *lhv M A B R XA XB*
 and *a \in spectrum A*
 and *b \in spectrum B*
 shows *integrable M ($\lambda w. XA \ a \ w * XB \ b \ w$)* using *assms unfolding lhv-def* by *simp*

lemma (in *cpx-sq-mat*) *lhv-scal-integrable*:
 fixes *c::real*
 assumes *lhv M A B R XA XB*
 and *a \in spectrum A*
 and *b \in spectrum B*
 shows *integrable M ($\lambda w. c * XA \ a \ w * d * XB \ b \ w$)*
proof –
 {
 fix *x*

```

    assume  $x \in \text{space } M$ 
    have  $c * d * (XA \ a \ x * XB \ b \ x) = c * XA \ a \ x * d * XB \ b \ x$  by simp
  } note eq = this
  have integrable  $M$  ( $\lambda w. XA \ a \ w * XB \ b \ w$ ) using assms unfolding lhv-def by
simp
  hence  $g$ : integrable  $M$  ( $\lambda w. c * d * (XA \ a \ w * XB \ b \ w)$ ) using integrable-real-mult-right
by simp
  show ?thesis
  proof (rule Bochner-Integration.integrable-cong[THEN iffD2], simp)
    show integrable  $M$  ( $\lambda w. c * d * (XA \ a \ w * XB \ b \ w)$ ) using  $g$  .
    show  $\bigwedge x. x \in \text{space } M \implies c * XA \ a \ x * d * XB \ b \ x = c * d * (XA \ a \ x * XB \ b \ x)$  using eq by simp
  qed
qed

```

lemma (in *cpx-sq-mat*) *lhv-lsum-scal-integrable*:

```

  assumes  $lhv \ M \ A \ B \ R \ XA \ XB$ 
  and  $a \in \text{spectrum } A$ 
  shows integrable  $M$  ( $\lambda x. \sum_{b \in \text{spectrum } B}. c * XA \ a \ x * (f \ b) * XB \ b \ x$ )
  proof (rule Bochner-Integration.integrable-sum)
    fix  $b$ 
    assume  $b \in \text{spectrum } B$ 
    thus integrable  $M$  ( $\lambda x. c * XA \ a \ x * f \ b * XB \ b \ x$ ) using  $\langle a \in \text{spectrum } A \rangle$  assms

    lhv-scal-integrable[of  $M$ ] by simp
  qed

```

lemma (in *cpx-sq-mat*) *lhv-sum-integrable*:

```

  assumes  $lhv \ M \ A \ B \ R \ XA \ XB$ 
  shows integrable  $M$  ( $\lambda w. (\sum_{a \in \text{spectrum } A}. (\sum_{b \in \text{spectrum } B}. f \ a * XA \ a \ w * g \ b * XB \ b \ w)))$ )
  proof (rule Bochner-Integration.integrable-sum)
    fix  $a$ 
    assume  $a \in \text{spectrum } A$ 
    thus integrable  $M$  ( $\lambda x. \sum_{b \in \text{spectrum } B}. f \ a * XA \ a \ x * g \ b * XB \ b \ x$ )
      using assms lhv-lsum-scal-integrable[of  $M$ ]
    by simp
  qed

```

lemma (in *cpx-sq-mat*) *spectrum-abs-1-weighted-suml*:

```

  assumes  $lhv \ M \ A \ B \ R \ Va \ Vb$ 
  and  $\{Re \ x \mid x \in \text{spectrum } A\} \neq \{\}$ 
  and  $\{Re \ x \mid x \in \text{spectrum } A\} \subseteq \{-1, 1\}$ 
  and hermitian  $A$ 
  and  $A \in \text{fc-mats}$ 
  shows AE  $w$  in  $M. |(\sum_{a \in \text{spectrum } A}. Re \ a * Va \ a \ w)| \leq 1$ 
  proof (rule AE-mp)
    show AE  $w$  in  $M. (\forall a \in \text{spectrum } A. 0 \leq Va \ a \ w \wedge Va \ a \ w \leq 1) \wedge (\sum_{a \in \text{spectrum } A}. Va \ a \ w) = 1$ 

```

using *assms lhw-AE-propl*[of $M A B - Va$] **by** *simp*
show $AE w$ in M . $(\forall a \in \text{spectrum } A. 0 \leq Va a w \wedge Va a w \leq 1) \wedge (\sum a \in \text{spectrum } A. Va a w) = 1 \longrightarrow$
 $|\sum a \in \text{spectrum } A. Re a * Va a w| \leq 1$
proof
fix w
assume $w \in \text{space } M$
show $(\forall a \in \text{spectrum } A. 0 \leq Va a w \wedge Va a w \leq 1) \wedge (\sum a \in \text{spectrum } A. Va a w) = 1 \longrightarrow$
 $|\sum a \in \text{spectrum } A. Re a * Va a w| \leq 1$
proof (*intro impI*)
assume *pr*: $(\forall a \in \text{spectrum } A. 0 \leq Va a w \wedge Va a w \leq 1) \wedge (\sum a \in \text{spectrum } A. Va a w) = 1$
show $|\sum a \in \text{spectrum } A. Re a * Va a w| \leq 1$
proof (*cases* $\{Re x \mid x. x \in \text{spectrum } A\} = \{-1, 1\}$)
case *True*
hence *sp*: $\text{spectrum } A = \{-1, 1\}$ **using** *hermitian-Re-spectrum*[of A] *assms* **by** *simp*
hence *scsum*: $(\sum a \in \text{spectrum } A. Re a * Va a w) = Va 1 w - Va (-1) w$
using *sum-2-elems* **by** *simp*
have *sum*: $(\sum a \in \text{spectrum } A. Va a w) = Va (-1) w + Va 1 w$
using *sp sum-2-elems* **by** *simp*
have $|Va 1 w - Va (-1) w| \leq 1$
proof (*rule fct-bound'*)
have $1 \in \text{spectrum } A$ **using** *sp* **by** *simp*
thus $0 \leq Va 1 w$ **using** *pr* **by** *simp*
have $-1 \in \text{spectrum } A$ **using** *sp* **by** *simp*
thus $0 \leq Va (-1) w$ **using** *pr* **by** *simp*
show $Va (-1) w + Va 1 w = 1$ **using** *pr sum* **by** *simp*
qed
thus *?thesis* **using** *scsum* **by** *simp*
next
case *False*
then **show** *?thesis*
proof (*cases* $\{Re x \mid x. x \in \text{spectrum } A\} = \{-1\}$)
case *True*
hence $\text{spectrum } A = \{-1\}$ **using** *hermitian-Re-spectrum*[of A] *assms* **by** *simp*
hence $(\sum a \in \text{spectrum } A. Re a * Va a w) = - Va (-1) w$ **by** *simp*
moreover **have** $-1 \in \text{spectrum } A$ **using** $\langle \text{spectrum } A = \{-1\} \rangle$ **by** *simp*
ultimately **show** *?thesis* **using** *pr* **by** *simp*
next
case *False*
hence $\{Re x \mid x. x \in \text{spectrum } A\} = \{1\}$ **using** *assms* $\langle \{Re x \mid x. x \in \text{spectrum } A\} \neq \{-1, 1\} \rangle$
last-subset[of $\{Re x \mid x. x \in \text{spectrum } A\}$] **by** *simp*
hence $\text{spectrum } A = \{1\}$ **using** *hermitian-Re-spectrum*[of A] *assms* **by** *simp*
hence $(\sum a \in \text{spectrum } A. Re a * Va a w) = Va 1 w$ **by** *simp*

moreover have $1 \in \text{spectrum } A$ **using** $\langle \text{spectrum } A = \{1\} \rangle$ **by simp**
ultimately show *?thesis* **using pr by simp**
qed
qed
qed
qed
qed

lemma (in *cpx-sq-mat*) *spectrum-abs-1-weighted-sumr*:
assumes $lhu\ M\ B\ A\ R\ Vb\ Va$
and $\{Re\ x\ |x.\ x \in \text{spectrum } A\} \neq \{\}$
and $\{Re\ x\ |x.\ x \in \text{spectrum } A\} \subseteq \{-1, 1\}$
and *hermitian* A
and $A \in \text{fc-mats}$
shows $AE\ w\ \text{in } M.\ |(\sum_{a \in \text{spectrum } A} Re\ a * Va\ a\ w)| \leq 1$
proof (*rule AE-mp*)
show $AE\ w\ \text{in } M.\ (\forall a \in \text{spectrum } A.\ 0 \leq Va\ a\ w \wedge Va\ a\ w \leq 1) \wedge (\sum_{a \in \text{spectrum } A} Va\ a\ w) = 1$
using *assms lhu-AE-propr* [of $M\ B\ A - Vb$] **by simp**
show $AE\ w\ \text{in } M.\ (\forall a \in \text{spectrum } A.\ 0 \leq Va\ a\ w \wedge Va\ a\ w \leq 1) \wedge (\sum_{a \in \text{spectrum } A} Va\ a\ w) = 1 \longrightarrow$
 $|\sum_{a \in \text{spectrum } A} Re\ a * Va\ a\ w| \leq 1$
proof
fix w
assume $w \in \text{space } M$
show $(\forall a \in \text{spectrum } A.\ 0 \leq Va\ a\ w \wedge Va\ a\ w \leq 1) \wedge (\sum_{a \in \text{spectrum } A} Va\ a\ w) = 1 \longrightarrow$
 $|\sum_{a \in \text{spectrum } A} Re\ a * Va\ a\ w| \leq 1$
proof (*intro impI*)
assume pr: $(\forall a \in \text{spectrum } A.\ 0 \leq Va\ a\ w \wedge Va\ a\ w \leq 1) \wedge (\sum_{a \in \text{spectrum } A} Va\ a\ w) = 1$
show $|\sum_{a \in \text{spectrum } A} Re\ a * Va\ a\ w| \leq 1$
proof (*cases* $\{Re\ x\ |x.\ x \in \text{spectrum } A\} = \{-1, 1\}$)
case *True*
hence sp: $\text{spectrum } A = \{-1, 1\}$ **using** *hermitian-Re-spectrum* [of A] *assms*
by simp
hence scsum: $(\sum_{a \in \text{spectrum } A} Re\ a * Va\ a\ w) = Va\ 1\ w - Va\ (-1)\ w$
using *sum-2-elems* **by simp**
have sum: $(\sum_{a \in \text{spectrum } A} Va\ a\ w) = Va\ (-1)\ w + Va\ 1\ w$
using *sp sum-2-elems* **by simp**
have $|Va\ 1\ w - Va\ (-1)\ w| \leq 1$
proof (*rule fct-bound'*)
have $1 \in \text{spectrum } A$ **using** *sp* **by simp**
thus $0 \leq Va\ 1\ w$ **using pr** **by simp**
have $-1 \in \text{spectrum } A$ **using** *sp* **by simp**
thus $0 \leq Va\ (-1)\ w$ **using pr** **by simp**
show $Va\ (-1)\ w + Va\ 1\ w = 1$ **using pr sum** **by simp**
qed
thus *?thesis* **using scsum** **by simp**

next
case *False*
then show *?thesis*
proof (*cases* $\{Re\ x \mid x. x \in spectrum\ A\} = \{-1\}$)
case *True*
hence $spectrum\ A = \{-1\}$ **using** *hermitian-Re-spectrum[of A] assms by simp*
hence $(\sum a \in spectrum\ A. Re\ a * Va\ a\ w) = -\ Va\ (-1)\ w$ **by** *simp*
moreover have $-1 \in spectrum\ A$ **using** $\langle spectrum\ A = \{-1\} \rangle$ **by** *simp*
ultimately show *?thesis using pr by simp*
next
case *False*
hence $\{Re\ x \mid x. x \in spectrum\ A\} = \{1\}$ **using** *assms $\langle \{Re\ x \mid x. x \in spectrum\ A\} \neq \{-1, 1\} \rangle$*
last-subset[of $\{Re\ x \mid x. x \in spectrum\ A\}$ **]** **by** *simp*
hence $spectrum\ A = \{1\}$ **using** *hermitian-Re-spectrum[of A] assms by simp*
hence $(\sum a \in spectrum\ A. Re\ a * Va\ a\ w) = Va\ 1\ w$ **by** *simp*
moreover have $1 \in spectrum\ A$ **using** $\langle spectrum\ A = \{1\} \rangle$ **by** *simp*
ultimately show *?thesis using pr by simp*
qed
qed
qed
qed
qed

definition *qt-expect where*
 $qt-expect\ A\ Va = (\lambda w. (\sum a \in spectrum\ A. Re\ a * Va\ a\ w))$

lemma (*in cpx-sq-mat*) *spectr-sum-integrable:*

assumes *lhw M A B R Va Vb*

shows *integrable M* $(\lambda w. qt-expect\ A\ Va\ w * qt-expect\ B\ Vb\ w)$

proof (*rule Bochner-Integration.integrable-cong[THEN iffD2]*)

show $M = M$ **by** *simp*

show $\bigwedge w. w \in space\ M \implies qt-expect\ A\ Va\ w * qt-expect\ B\ Vb\ w =$

$(\sum a \in spectrum\ A. (\sum b \in spectrum\ B. Re\ a * Va\ a\ w * Re\ b * Vb\ b\ w))$

proof –

fix *w*

assume $w \in space\ M$

show $qt-expect\ A\ Va\ w * qt-expect\ B\ Vb\ w =$

$(\sum a \in spectrum\ A. (\sum b \in spectrum\ B. Re\ a * Va\ a\ w * Re\ b * Vb\ b\ w))$

unfolding *qt-expect-def*

by (*metis (mono-tags, lifting) Finite-Cartesian-Product.sum-cong-aux sum-product*

vector-space-over-itself.scale-scale)

qed

show *integrable M* $(\lambda w. \sum a \in spectrum\ A. (\sum b \in spectrum\ B. Re\ a * Va\ a\ w * Re\ b * Vb\ b\ w))$

using *lhw-sum-integrable[of M] assms by simp*

qed

lemma (in *cpx-sq-mat*) *lhw-sum-integral'*:

assumes *lhw M A B R XA XB*

shows $\text{integral}^L M (\lambda w. (\sum a \in \text{spectrum } A. f a * XA a w) * (\sum b \in \text{spectrum } B. g b * XB b w)) =$

$(\sum a \in \text{spectrum } A. f a * (\sum b \in \text{spectrum } B. g b * \text{integral}^L M (\lambda w. XA a w * XB b w)))$

proof –

have $\text{integral}^L M (\lambda w. (\sum a \in \text{spectrum } A. f a * XA a w) * (\sum b \in \text{spectrum } B. g b * XB b w)) =$

$\text{integral}^L M (\lambda w. (\sum a \in \text{spectrum } A. (\sum b \in \text{spectrum } B. f a * XA a w * g b * XB b w)))$

proof (rule *Bochner-Integration.integral-cong, simp*)

fix *w*

assume $w \in \text{space } M$

show $(\sum a \in \text{spectrum } A. f a * XA a w) * (\sum b \in \text{spectrum } B. g b * XB b w) =$
 $(\sum a \in \text{spectrum } A. (\sum b \in \text{spectrum } B. f a * XA a w * (g b) * XB b w))$

by (*simp add: sum-product vector-space-over-itself.scale-scale*)

qed

also have $\dots = (\sum a \in \text{spectrum } A.$

$\text{integral}^L M (\lambda w. (\sum b \in \text{spectrum } B. f a * XA a w * g b * XB b w)))$

proof (rule *Bochner-Integration.integral-sum*)

fix *a*

assume $a \in \text{spectrum } A$

thus $\text{integrable } M (\lambda x. \sum b \in \text{spectrum } B. f a * XA a x * g b * XB b x)$

using *lhw-lsum-scal-integrable[of M] assms by simp*

qed

also have $\dots = (\sum a \in \text{spectrum } A. f a *$

$\text{integral}^L M (\lambda w. (\sum b \in \text{spectrum } B. XA a w * g b * XB b w)))$

proof –

have $\forall a \in \text{spectrum } A. \text{integral}^L M (\lambda w. (\sum b \in \text{spectrum } B. f a * XA a w * g b * XB b w)) =$
 $f a * \text{integral}^L M (\lambda w. (\sum b \in \text{spectrum } B. XA a w * g b * XB b w))$

proof

fix *a*

assume $a \in \text{spectrum } A$

have $(\text{LINT } w | M. (\sum b \in \text{spectrum } B. f a * XA a w * g b * XB b w)) =$
 $(\text{LINT } w | M. f a * (\sum b \in \text{spectrum } B. XA a w * g b * XB b w))$

proof (rule *Bochner-Integration.integral-cong, simp*)

fix *x*

assume $x \in \text{space } M$

show $(\sum b \in \text{spectrum } B. f a * XA a x * g b * XB b x) =$

$f a * (\sum b \in \text{spectrum } B. XA a x * g b * XB b x)$

by (*metis (no-types, lifting) Finite-Cartesian-Product.sum-cong-aux*

vector-space-over-itself.scale-scale vector-space-over-itself.scale-sum-right)

qed

also have $\dots = f a * (\text{LINT } w | M. (\sum b \in \text{spectrum } B. XA a w * g b * XB b w))$
by *simp*

finally show $(LINT\ w|M. (\sum_{b \in \text{spectrum } B}. f\ a * XA\ a\ w * g\ b * XB\ b\ w))$
 $=$
 $f\ a * (LINT\ w|M. (\sum_{b \in \text{spectrum } B}. XA\ a\ w * g\ b * XB\ b\ w)) .$
qed
thus *?thesis by simp*
qed
also have $\dots = (\sum_{a \in \text{spectrum } A}. f\ a * (\sum_{b \in \text{spectrum } B}. g\ b * \text{integral}^L\ M\ (\lambda w. XA\ a\ w * XB\ b\ w)))$
proof *(rule sum.cong, simp)*
fix a
assume $a \in \text{spectrum } A$
have $\text{integral}^L\ M\ (\lambda w. (\sum_{b \in \text{spectrum } B}. XA\ a\ w * g\ b * XB\ b\ w)) = (\sum_{b \in \text{spectrum } B}. \text{integral}^L\ M\ (\lambda w. XA\ a\ w * g\ b * XB\ b\ w))$
proof *(rule Bochner-Integration.integral-sum)*
show $\bigwedge b. b \in \text{spectrum } B \implies \text{integrable } M\ (\lambda x. XA\ a\ x * g\ b * XB\ b\ x)$
proof $-$
fix b
assume $b \in \text{spectrum } B$
thus $\text{integrable } M\ (\lambda x. XA\ a\ x * g\ b * XB\ b\ x)$
using *assms lhw-scal-integrable[of M - - - - a b 1] <a ∈ spectrum A> by*
simp
qed
qed
also have $\dots = (\sum_{b \in \text{spectrum } B}. g\ b * \text{integral}^L\ M\ (\lambda w. XA\ a\ w * XB\ b\ w))$
proof *(rule sum.cong, simp)*
fix x
assume $x \in \text{spectrum } B$
have $LINT\ w|M. XA\ a\ w * g\ x * XB\ x\ w = LINT\ w|M. g\ x * (XA\ a\ w * XB\ x\ w)$
by *(rule Bochner-Integration.integral-cong, auto)*
also have $\dots = g\ x * (LINT\ w|M. XA\ a\ w * XB\ x\ w)$
using *Bochner-Integration.integral-mult-right-zero[of M g x λw. XA a w * XB x w]*
by *simp*
finally show $LINT\ w|M. XA\ a\ w * g\ x * XB\ x\ w = g\ x * (LINT\ w|M. XA\ a\ w * XB\ x\ w) .$
qed
finally have $\text{integral}^L\ M\ (\lambda w. (\sum_{b \in \text{spectrum } B}. XA\ a\ w * g\ b * XB\ b\ w)) =$
 $(\sum_{b \in \text{spectrum } B}. g\ b * \text{integral}^L\ M\ (\lambda w. XA\ a\ w * XB\ b\ w)) .$
thus $f\ a * (LINT\ w|M. (\sum_{b \in \text{spectrum } B}. XA\ a\ w * g\ b * XB\ b\ w)) =$
 $f\ a * (\sum_{b \in \text{spectrum } B}. g\ b * \text{integral}^L\ M\ (\lambda w. XA\ a\ w * XB\ b\ w))$ **by**
simp
qed
finally show *?thesis .*
qed

lemma (in *cpx-sq-mat*) *sum-qt-expect-trace*:
assumes *lhv M A B R XA XB*
shows $(\sum a \in \text{spectrum } A. f a * (\sum b \in \text{spectrum } B. g b * \text{integral}^L M (\lambda w. XA a w * XB b w))) =$
 $(\sum a \in \text{spectrum } A. f a * (\sum b \in \text{spectrum } B. g b * \text{Re} (\text{Complex-Matrix.trace}(\text{eigen-projector } A a * (\text{eigen-projector } B b) * R))))$
proof (rule *sum.cong, simp*)
fix *a*
assume *a ∈ spectrum A*
have $(\sum b \in \text{spectrum } B. g b * (\text{LINT } w | M. XA a w * XB b w)) =$
 $(\sum b \in \text{spectrum } B. g b * \text{Re} (\text{Complex-Matrix.trace} (\text{eigen-projector } A a * \text{eigen-projector } B b * R)))$
proof (rule *sum.cong, simp*)
fix *b*
assume *b ∈ spectrum B*
show $g b * (\text{LINT } w | M. XA a w * XB b w) =$
 $g b * \text{Re} (\text{Complex-Matrix.trace} (\text{eigen-projector } A a * \text{eigen-projector } B b * R))$
R)
using *lhv-integral-eq[of M] assms ⟨a ∈ spectrum A⟩ ⟨b ∈ spectrum B⟩ by simp*
qed
thus $f a * (\sum b \in \text{spectrum } B. g b * (\text{LINT } w | M. XA a w * XB b w)) =$
 $f a * (\sum b \in \text{spectrum } B. g b * \text{Re} (\text{Complex-Matrix.trace} (\text{eigen-projector } A a * \text{eigen-projector } B b * R)))$
by simp
qed

lemma (in *cpx-sq-mat*) *sum-eigen-projector-trace-dist*:
assumes *hermitian B*
and *A ∈ fc-mats*
and *B ∈ fc-mats*
and *R ∈ fc-mats*
shows $(\sum b \in \text{spectrum } B. (b * \text{Complex-Matrix.trace}(A * (\text{eigen-projector } B b) * R))) = \text{Complex-Matrix.trace}(A * B * R)$
proof –
have $(\sum b \in \text{spectrum } B. b * \text{Complex-Matrix.trace} (A * \text{eigen-projector } B b * R)) =$
 $(\sum b \in \text{spectrum } B. \text{Complex-Matrix.trace} (A * (b \cdot_m (\text{eigen-projector } B b)) * R))$
proof (rule *sum.cong, simp*)
fix *b*
assume *b ∈ spectrum B*
have $b * \text{Complex-Matrix.trace} (A * \text{eigen-projector } B b * R) =$
 $\text{Complex-Matrix.trace} (b \cdot_m (A * \text{eigen-projector } B b * R))$
proof (rule *trace-smult[symmetric]*)
show $A * \text{eigen-projector } B b * R \in \text{carrier-mat } \text{dim}R \text{ dim}R$ **using** *eigen-projector-carrier*

assms fc-mats-carrier dim-eq ⟨b ∈ spectrum B⟩ cpx-sq-mat-mult by meson
qed

```

also have ... = Complex-Matrix.trace (A * (b ·m eigen-projector B b) * R)
proof –
  have b ·m (A * eigen-projector B b * R) = b ·m (A * (eigen-projector B b *
R))
  proof –
    have A * eigen-projector B b * R = A * (eigen-projector B b * R)
      by (metis ⟨b ∈ spectrum B⟩ assms(1) assms(2) assms(3) assms(4)
assoc-mult-mat dim-eq
      fc-mats-carrier eigen-projector-carrier)
    thus ?thesis by simp
  qed
also have ... = A * (b ·m (eigen-projector B b * R))
proof (rule mult-smult-distrib[symmetric])
  show A ∈ carrier-mat dimR dimR using eigen-projector-carrier assms
fc-mats-carrier dim-eq by simp
show eigen-projector B b * R ∈ carrier-mat dimR dimR using eigen-projector-carrier

  ⟨b ∈ spectrum B⟩ assms fc-mats-carrier dim-eq cpx-sq-mat-mult by blast
qed
also have ... = A * ((b ·m eigen-projector B b) * R)
proof –
  have b ·m (eigen-projector B b * R) = (b ·m eigen-projector B b) * R
  proof (rule mult-smult-assoc-mat[symmetric])
  show eigen-projector B b ∈ carrier-mat dimR dimR using eigen-projector-carrier

  ⟨b ∈ spectrum B⟩ assms fc-mats-carrier dim-eq by simp
  show R ∈ carrier-mat dimR dimR using assms fc-mats-carrier dim-eq by
simp
  qed
  thus ?thesis by simp
qed
also have ... = A * (b ·m eigen-projector B b) * R
  by (metis ⟨b ∈ spectrum B⟩ assms(1) assms(2) assms(3) assms(4) as-
soc-mult-mat
  cpx-sq-mat-smult dim-eq fc-mats-carrier eigen-projector-carrier)
  finally have b ·m (A * eigen-projector B b * R) = A * (b ·m eigen-projector
B b) * R .
  then show ?thesis by simp
qed
finally show b * Complex-Matrix.trace (A * eigen-projector B b * R) =
Complex-Matrix.trace (A * (b ·m eigen-projector B b) * R) .
qed
also have ... = Complex-Matrix.trace (A *
(sum-mat (λb. b ·m eigen-projector B b) (spectrum B)) * R)
proof (rule trace-sum-mat-mat-distrib, (auto simp add: assms))
  show finite (spectrum B) using spectrum-finite[of B] by simp
fix b
  assume b ∈ spectrum B
  show b ·m eigen-projector B b ∈ fc-mats

```

by (simp add: ⟨ $b \in \text{spectrum } B$ ⟩ *assms(1) assms(3) cpx-sq-mat-smult eigen-projector-carrier*)
 qed
 also have ... = *Complex-Matrix.trace* (A * B * R)
 proof –
 have *sum-mat* ($\lambda b. b \cdot_m \text{eigen-projector } B b$) (*spectrum B*) = B **using** *make-pm-sum'*
assms **by** *simp*
 thus ?thesis **by** *simp*
 qed
 finally show ?thesis .
 qed

lemma (in *cpx-sq-mat*) *sum-eigen-projector-trace-right*:

assumes *hermitian A*
 and $A \in \text{fc-mats}$
 and $B \in \text{fc-mats}$
 shows ($\sum a \in \text{spectrum } A. \text{Complex-Matrix.trace } (a \cdot_m \text{eigen-projector } A a * B)$)
 =
Complex-Matrix.trace (A * B)
 proof –
 have *sum-mat* ($\lambda a. a \cdot_m \text{eigen-projector } A a * B$) (*spectrum A*) =
sum-mat ($\lambda a. a \cdot_m \text{eigen-projector } A a$) (*spectrum A*) * B
 proof (rule *mult-sum-mat-distrib-right*)
 show *finite* (*spectrum A*) **using** *spectrum-finite[of A]* **by** *simp*
 show $\bigwedge a. a \in \text{spectrum } A \implies a \cdot_m \text{eigen-projector } A a \in \text{fc-mats}$
using *assms(1) assms(2) cpx-sq-mat-smult eigen-projector-carrier* **by** *blast*
 show $B \in \text{fc-mats}$ **using** *assms* **by** *simp*
 qed
 also have ... = A * B **using** *make-pm-sum'* *assms* **by** *simp*
 finally have *seq*: *sum-mat* ($\lambda a. a \cdot_m \text{eigen-projector } A a * B$) (*spectrum A*) =
 A * B .
 have ($\sum a \in \text{spectrum } A. \text{Complex-Matrix.trace } (a \cdot_m \text{eigen-projector } A a * B)$)
 =
Complex-Matrix.trace (*sum-mat* ($\lambda a. a \cdot_m \text{eigen-projector } A a * B$) (*spectrum*
 A))
 proof (rule *trace-sum-mat[symmetric]*)
 show *finite* (*spectrum A*) **using** *spectrum-finite[of A]* **by** *simp*
 show $\bigwedge a. a \in \text{spectrum } A \implies a \cdot_m \text{eigen-projector } A a * B \in \text{fc-mats}$
by (*simp* add: *assms(1) assms(2) assms(3) cpx-sq-mat-mult cpx-sq-mat-smult*
eigen-projector-carrier)
 qed
 also have ... = *Complex-Matrix.trace* (A * B) **using** *seq* **by** *simp*
 finally show ?thesis .
 qed

lemma (in *cpx-sq-mat*) *sum-eigen-projector-trace*:

assumes *hermitian A*
 and *hermitian B*
 and $A \in \text{fc-mats}$

and $B \in \text{fc-mats}$
and $R \in \text{fc-mats}$
shows $(\sum a \in \text{spectrum } A. a * (\sum b \in \text{spectrum } B. (b * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * (\text{eigen-projector } B b) * R)))) = \text{Complex-Matrix.trace}(A * B * R)$
proof –
have $(\sum a \in \text{spectrum } A. a * (\sum b \in \text{spectrum } B. (b * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * (\text{eigen-projector } B b) * R)))) = (\sum a \in \text{spectrum } A. \text{Complex-Matrix.trace}(a \cdot_m \text{eigen-projector } A a * (B * R)))$
proof (rule *sum.cong, simp*)
fix a
assume $a \in \text{spectrum } A$
hence $(\sum b \in \text{spectrum } B. b * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * \text{eigen-projector } B b * R)) = \text{Complex-Matrix.trace}(\text{eigen-projector } A a * B * R)$ **using** *sum-eigen-projector-trace-dist[of B eigen-projector A a R]* *assms eigen-projector-carrier*

by blast
hence $a * (\sum b \in \text{spectrum } B. (b * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * (\text{eigen-projector } B b) * R))) = a * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * B * R)$ **by simp**
also have $\dots = \text{Complex-Matrix.trace}(a \cdot_m (\text{eigen-projector } A a * B * R))$
using *trace-smult[symmetric, of eigen-projector A a * B * R dimR a]* *assms* $\langle a \in \text{spectrum } A \rangle$ *cpx-sq-mat-mult dim-eq fc-mats-carrier eigen-projector-carrier*
by force
also have $\dots = \text{Complex-Matrix.trace}(a \cdot_m \text{eigen-projector } A a * (B * R))$
proof –
have $a \cdot_m (\text{eigen-projector } A a * B * R) = a \cdot_m (\text{eigen-projector } A a * B) * R$
proof (rule *mult-smult-assoc-mat[symmetric]*)
show $R \in \text{carrier-mat dimR dimR}$ **using** *assms fc-mats-carrier dim-eq by simp*
show $\text{eigen-projector } A a * B \in \text{carrier-mat dimR dimR}$ **using** *assms eigen-projector-carrier*
cpx-sq-mat-mult fc-mats-carrier dim-eq $\langle a \in \text{spectrum } A \rangle$ **by blast**
qed
also have $\dots = a \cdot_m \text{eigen-projector } A a * B * R$
proof –
have $a \cdot_m (\text{eigen-projector } A a * B) = a \cdot_m \text{eigen-projector } A a * B$
using *mult-smult-assoc-mat[symmetric]*
proof –
show *?thesis*
by (metis $\langle \bigwedge nr nc n k B A. \llbracket A \in \text{carrier-mat nr } n; B \in \text{carrier-mat } n nc \rrbracket \implies k \cdot_m (A * B) = k \cdot_m A * B \rangle$ $\langle a \in \text{spectrum } A \rangle$ *assms(1) assms(3) assms(4) dim-eq fc-mats-carrier eigen-projector-carrier*)
qed

```

    thus ?thesis by simp
  qed
  also have ... = a ·m eigen-projector A a * (B * R)
    by (metis ⟨a ∈ spectrum A⟩ assms(1) assms(3) assms(4) assms(5) as-
soc-mult-mat
        cpx-sq-mat-smult dim-eq fc-mats-carrier eigen-projector-carrier)
  finally show ?thesis by simp
  qed
  finally show a * (∑ b ∈ spectrum B. (b *
    Complex-Matrix.trace(eigen-projector A a * (eigen-projector B b) * R))) =
    Complex-Matrix.trace (a ·m eigen-projector A a * (B * R)) .
  qed
  also have ... = Complex-Matrix.trace (A * (B * R))
  using sum-eigen-projector-trace-right[of A B * R] assms by (simp add: cpx-sq-mat-mult)

  also have ... = Complex-Matrix.trace (A * B * R)
  proof -
    have A * (B * R) = A * B * R
    proof (rule assoc-mult-mat[symmetric])
      show A ∈ carrier-mat dimR dimR using assms fc-mats-carrier dim-eq by
simp
      show B ∈ carrier-mat dimR dimR using assms fc-mats-carrier dim-eq by
simp
      show R ∈ carrier-mat dimR dimR using assms fc-mats-carrier dim-eq by
simp
    qed
    thus ?thesis by simp
  qed
  finally show ?thesis .
  qed

```

We obtain the main result of this part, which relates the quantum expectation value of a joint measurement with an expectation.

lemma (in *cpx-sq-mat*) *sum-qt-expect*:

```

  assumes lhv M A B R XA XB
  and A ∈ fc-mats
  and B ∈ fc-mats
  and R ∈ fc-mats
  and hermitian A
  and hermitian B
  shows integralL M (λw. (qt-expect A XA w) * (qt-expect B XB w)) =
    Re (Complex-Matrix.trace(A * B * R))
  proof -
    have br: ∀ b ∈ spectrum B. b ∈ Reals using assms hermitian-spectrum-real[of B]
  by auto
    have ar: ∀ a ∈ spectrum A. a ∈ Reals using hermitian-spectrum-real[of A] assms
  by auto
    have integralL M (λw. (∑ a ∈ spectrum A. Re a * XA a w) * (∑ b ∈ spectrum
  B. Re b * XB b w)) =

```

$(\sum a \in \text{spectrum } A. \text{Re } a * (\sum b \in \text{spectrum } B. \text{Re } b * \text{integral}^L M (\lambda w. XA a w * XB b w)))$
using *lhv-sum-integral'*[of M] *assms* **by** *simp*
also have ... = $(\sum a \in \text{spectrum } A. \text{Re } a * (\sum b \in \text{spectrum } B. \text{Re } b * \text{Re } (\text{Complex-Matrix.trace}(\text{eigen-projector } A a * (\text{eigen-projector } B b) * R))))$
using *assms sum-qt-expect-trace*[of M] **by** *simp*
also have ... = $(\sum a \in \text{spectrum } A. \text{Re } a * \text{Re } (\sum b \in \text{spectrum } B. (b * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * (\text{eigen-projector } B b) * R))))$
proof (*rule sum.cong, simp*)
fix a
assume $a \in \text{spectrum } A$
have $(\sum b \in \text{spectrum } B. \text{Re } b * \text{Re } (\text{Complex-Matrix.trace}(\text{eigen-projector } A a * \text{eigen-projector } B b * R))) =$
 $(\sum b \in \text{spectrum } B. \text{Re } (b * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * (\text{eigen-projector } B b) * R)))$
proof (*rule sum.cong, simp*)
fix b
assume $b \in \text{spectrum } B$
hence $b \in \text{Reals}$ **using** *hermitian-spectrum-real*[of B] *assms* **by** *simp*
hence $\text{Re } b = b$ **by** *simp*
thus $\text{Re } b * \text{Re } (\text{Complex-Matrix.trace}(\text{eigen-projector } A a * \text{eigen-projector } B b * R)) =$
 $\text{Re } (b * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * \text{eigen-projector } B b * R))$
using *hermitian-spectrum-real* **using** $\langle b \in \mathbb{R} \rangle$ *mult-real-cpx* **by** *auto*
qed
also have ... =
 $\text{Re } (\sum b \in \text{spectrum } B. b * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * (\text{eigen-projector } B b) * R))$ **by**
simp
finally have $(\sum b \in \text{spectrum } B. \text{Re } b * \text{Re } (\text{Complex-Matrix.trace}(\text{eigen-projector } A a * \text{eigen-projector } B b * R))) =$
 $\text{Re } (\sum b \in \text{spectrum } B. b * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * (\text{eigen-projector } B b) * R)) .$
thus $\text{Re } a * (\sum b \in \text{spectrum } B. \text{Re } b * \text{Re } (\text{Complex-Matrix.trace}(\text{eigen-projector } A a * \text{eigen-projector } B b * R))) =$
 $\text{Re } a * \text{Re } (\sum b \in \text{spectrum } B. (b * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * \text{eigen-projector } B b * R)))$
by *simp*
qed
also have ... = $(\sum a \in \text{spectrum } A. \text{Re } (a * (\sum b \in \text{spectrum } B. (b * \text{Complex-Matrix.trace}(\text{eigen-projector } A a * (\text{eigen-projector } B b) * R))))$
proof (*rule sum.cong, simp*)
fix x
assume $x \in \text{spectrum } A$
hence $\text{Re } x = x$ **using** *ar* **by** *simp*
thus $\text{Re } x * \text{Re } (\sum b \in \text{spectrum } B. b * \text{Complex-Matrix.trace}(\text{eigen-projector } A x * \text{eigen-projector } B b * R)) =$

$Re (x * (\sum_{b \in \text{spectrum } B} b * \text{Complex-Matrix.trace}(\text{eigen-projector } A \ x * \text{eigen-projector } B \ b * R)))$
using $\langle x \in \text{spectrum } A \rangle$ *ar mult-real-cpx* **by** *auto*
qed
also have $\dots = Re (\sum_{a \in \text{spectrum } A} a * (\sum_{b \in \text{spectrum } B} (b * \text{Complex-Matrix.trace}(\text{eigen-projector } A \ a * (\text{eigen-projector } B \ b * R))))$ **by**
simp
also have $\dots = Re (\text{Complex-Matrix.trace}(A * B * R))$ **using** *assms*
sum-eigen-projector-trace[of A B] **by** *simp*
finally show *?thesis unfolding qt-expect-def* .
qed

4.2 Properties of specific observables

In this part we consider a specific density operator and specific observables corresponding to joint bipartite measurements. We will compute the quantum expectation value of this system and prove that it violates the CHSH inequality, thus proving that the local hidden variable assumption cannot hold.

4.2.1 Ket 0, Ket 1 and the corresponding projectors

definition *ket-0::complex Matrix.vec* **where**
ket-0 = unit-vec 2 0

lemma *ket-0-dim:*
shows $\text{dim-vec } \text{ket-0} = 2$ **unfolding** *ket-0-def* **by** *simp*

lemma *ket-0-norm:*
shows $\|\text{ket-0}\| = 1$ **using** *unit-cpx-vec-length* **unfolding** *ket-0-def* **by** *simp*

lemma *ket-0-mat:*
shows $\text{ket-vec } \text{ket-0} = \text{Matrix.mat-of-cols-list } 2 \ [[1, 0]]$
by (*auto simp add: ket-vec-def Matrix.mat-of-cols-list-def ket-0-def*)

definition *ket-1::complex Matrix.vec* **where**
ket-1 = unit-vec 2 1

lemma *ket-1-dim:*
shows $\text{dim-vec } \text{ket-1} = 2$ **unfolding** *ket-1-def* **by** *simp*

lemma *ket-1-norm:*
shows $\|\text{ket-1}\| = 1$ **using** *unit-cpx-vec-length* **unfolding** *ket-1-def* **by** *simp*

definition *ket-01*
where $\text{ket-01} = \text{tensor-vec } \text{ket-0} \ \text{ket-1}$

lemma *ket-01-dim:*

shows $\dim\text{-vec } ket\text{-}01 = 4$ **unfolding** $ket\text{-}01\text{-def}$
by (*simp add: ket-0-dim ket-1-dim*)

definition $ket\text{-}10$
where $ket\text{-}10 = \text{tensor-vec } ket\text{-}1 \text{ } ket\text{-}0$

lemma $ket\text{-}10\text{-dim}$:
shows $\dim\text{-vec } ket\text{-}10 = 4$ **unfolding** $ket\text{-}10\text{-def}$ **by** (*simp add: ket-0-dim ket-1-dim*)

We define ket_psim , one of the Bell states (or EPR pair).

definition $ket\text{-psim}$ **where**
 $ket\text{-psim} = 1/(\text{sqrt } 2) \cdot_v (ket\text{-}01 - ket\text{-}10)$

lemma $ket\text{-psim-dim}$:
shows $\dim\text{-vec } ket\text{-psim} = 4$ **using** $ket\text{-}01\text{-dim } ket\text{-}10\text{-dim}$ **unfolding** $ket\text{-psim-def}$
by *simp*

lemma $ket\text{-psim-norm}$:
shows $\|ket\text{-psim}\| = 1$
proof –
have $\dim\text{-vec } ket\text{-psim} = 2^2$ **unfolding** $ket\text{-psim-def } ket\text{-}01\text{-def } ket\text{-}10\text{-def } ket\text{-}0\text{-def } ket\text{-}1\text{-def}$
by *simp*
moreover **have** $(\sum_{i < 4}. (\text{cmod } (\text{vec-index } ket\text{-psim } i))^2) = 1$
apply (*auto simp add: ket-psim-def ket-01-def ket-10-def ket-0-def ket-1-def*)
apply (*simp add: sum-4-elems*)
done
ultimately show *?thesis* **by** (*simp add: cpx-vec-length-def*)
qed

ρ_psim represents the density operator associated with the quantum state ket_psim .

definition $\rho\text{-psim}$ **where**
 $\rho\text{-psim} = \text{rank-1-proj } ket\text{-psim}$

lemma $\rho\text{-psim-carrier}$:
shows $\rho\text{-psim} \in \text{carrier-mat } 4 \ 4$ **using** $\text{rank-1-proj-carrier}[of \ ket\text{-psim}] \ ket\text{-psim-dim}$
 $\rho\text{-psim-def}$ **by** *simp*

lemma $\rho\text{-psim-dim-row}$:
shows $\dim\text{-row } \rho\text{-psim} = 4$ **using** $\rho\text{-psim-carrier}$ **by** *simp*

lemma $\rho\text{-psim-density}$:
shows $\text{density-operator } \rho\text{-psim}$ **unfolding** $\text{density-operator-def}$
proof
show $\text{Complex-Matrix.positive } \rho\text{-psim}$ **using** $\text{rank-1-proj-positive}[of \ ket\text{-psim}]$
 $ket\text{-psim-norm}$
 $\rho\text{-psim-def}$ **by** *simp*

show *Complex-Matrix.trace rho-psim = 1* **using** *rank-1-proj-trace[of ket-psim]*
ket-psim-norm
rho-psim-def **by** *simp*
qed

4.2.2 The X and Z matrices and two of their combinations

In this part we prove properties of two standard matrices in quantum theory, X and Z , as well as two of their combinations: $\frac{X+Z}{\sqrt{2}}$ and $\frac{Z-X}{\sqrt{2}}$. Note that all of these matrices are observables, they will be used to violate the CHSH inequality.

lemma *Z-carrier*: **shows** $Z \in \text{carrier-mat } 2 \ 2$ **unfolding** *Z-def* **by** *simp*

lemma *Z-hermitian*:

shows *hermitian Z* **using** *dagger-adjoint dagger-of-Z* **unfolding** *hermitian-def*
by *simp*

lemma *unitary-Z*:

shows *Complex-Matrix.unitary Z*

proof –

have *Complex-Matrix.adjoint Z * Z = (1_m 2)* **using** *dagger-adjoint[of Z]* **by**
simp

thus *?thesis* **unfolding** *unitary-def*

by (*metis* *Complex-Matrix.adjoint-adjoint* *Complex-Matrix.unitary-def* *Z-carrier*
adjoint-dim

carrier-matD(1) *inverts-mat-def* *unitary-adjoint*)

qed

lemma *X-carrier*: **shows** $X \in \text{carrier-mat } 2 \ 2$ **unfolding** *X-def* **by** *simp*

lemma *X-hermitian*:

shows *hermitian X* **using** *dagger-adjoint dagger-of-X* **unfolding** *hermitian-def*
by *simp*

lemma *unitary-X*:

shows *Complex-Matrix.unitary X*

proof –

have *Complex-Matrix.adjoint X * X = (1_m 2)* **using** *dagger-adjoint[of X]* **by**
simp

thus *?thesis* **unfolding** *unitary-def*

by (*metis* *Complex-Matrix.adjoint-adjoint* *Complex-Matrix.unitary-def* *X-carrier*
adjoint-dim

carrier-matD(1) *inverts-mat-def* *unitary-adjoint*)

qed

definition *XpZ*

where $XpZ = -1/\text{sqrt}(2) \cdot_m (X + Z)$

lemma *XpZ-carrier*:
shows $XpZ \in \text{carrier-mat } 2 \ 2$ **unfolding** *XpZ-def X-def Z-def* **by** *simp*

lemma *XpZ-hermitian*:
shows *hermitian XpZ*
proof –
have $X + Z \in \text{carrier-mat } 2 \ 2$ **using** *Z-carrier X-carrier* **by** *simp*
moreover **have** *hermitian (X + Z)* **using** *X-hermitian Z-hermitian hermitian-add Matrix.mat-carrier*
unfolding *X-def Z-def* **by** *blast*
ultimately show *?thesis* **using** *hermitian-smult[of X + Z 2 - 1 / sqrt 2]*
unfolding *XpZ-def*
by *auto*
qed

lemma *XpZ-inv*:
 $XpZ * XpZ = 1_m \ 2$ **unfolding** *XpZ-def X-def Z-def times-mat-def one-mat-def*
apply (*rule cong-mat, simp+*)
apply (*auto simp add: Matrix.scalar-prod-def*)
apply (*auto simp add: Gates.csqrt-2-sq*)
done

lemma *unitary-XpZ*:
shows *Complex-Matrix.unitary XpZ*
proof –
have *Complex-Matrix.adjoint XpZ * XpZ = (1_m 2)* **using** *XpZ-inv XpZ-hermitian*

unfolding *hermitian-def* **by** *simp*
thus *?thesis* **unfolding** *unitary-def*
by (*metis Complex-Matrix.adjoint-adjoint Complex-Matrix.unitary-def XpZ-carrier adjoint-dim carrier-matD(1) inverts-mat-def unitary-adjoint*)
qed

definition *ZmX*
where $ZmX = 1/\text{sqrt}(2) \cdot_m (Z - X)$

lemma *ZmX-carrier*:
shows $ZmX \in \text{carrier-mat } 2 \ 2$ **unfolding** *ZmX-def X-def Z-def*
by (*simp add: minus-carrier-mat*)

lemma *ZmX-hermitian*:
shows *hermitian ZmX*
proof –
have $Z - X \in \text{carrier-mat } 2 \ 2$ **unfolding** *X-def Z-def*
by (*simp add: minus-carrier-mat*)
moreover **have** *hermitian (Z - X)* **using** *X-hermitian Z-hermitian hermitian-minus Matrix.mat-carrier*
unfolding *X-def Z-def* **by** *blast*

ultimately show *?thesis* **using** *hermitian-smult[of Z - X 2 1 / sqrt 2]* **unfolding** *ZmX-def*
by *auto*
qed

lemma *ZmX-inv*:
 $ZmX * ZmX = 1_m$ **2 unfolding** *ZmX-def X-def Z-def times-mat-def one-mat-def*
apply (*rule cong-mat, simp+*)
apply (*auto simp add: Matrix.scalar-prod-def*)
apply (*auto simp add: Gates.csqrt-2-sq*)
done

lemma *unitary-ZmX*:
shows *Complex-Matrix.unitary ZmX*
proof –
have *Complex-Matrix.adjoint ZmX * ZmX = (1_m 2)* **using** *ZmX-inv ZmX-hermitian*

unfolding *hermitian-def* **by** *simp*
thus *?thesis* **unfolding** *unitary-def*
by (*metis Complex-Matrix.adjoint-adjoint Complex-Matrix.unitary-def ZmX-carrier*
adjoint-dim
carrier-matD(1) inverts-mat-def unitary-adjoint)
qed

definition *Z-XpZ*
where $Z-XpZ = \text{tensor-mat } Z \ XpZ$

lemma *Z-XpZ-carrier*:
shows $Z-XpZ \in \text{carrier-mat } 4 \ 4$ **unfolding** *Z-XpZ-def* **using** *tensor-mat-carrier*
XpZ-carrier
Z-carrier **by** *auto*

definition *X-XpZ*
where $X-XpZ = \text{tensor-mat } X \ XpZ$

lemma *X-XpZ-carrier*:
shows $X-XpZ \in \text{carrier-mat } 4 \ 4$ **using** *tensor-mat-carrier XpZ-carrier X-carrier*

unfolding *X-XpZ-def* **by** *auto*

definition *Z-ZmX*
where $Z-ZmX = \text{tensor-mat } Z \ ZmX$

lemma *Z-ZmX-carrier*:
shows $Z-ZmX \in \text{carrier-mat } 4 \ 4$ **using** *tensor-mat-carrier ZmX-carrier Z-carrier*

unfolding *Z-ZmX-def* **by** *auto*

definition *X-ZmX*

where $X\text{-Zm}X = \text{tensor-mat } X \text{ Zm}X$

lemma $X\text{-Zm}X\text{-carrier}$:

shows $X\text{-Zm}X \in \text{carrier-mat } 4 \ 4$ **using** $\text{tensor-mat-carrier } X\text{-carrier } \text{Zm}X\text{-carrier}$
unfolding $X\text{-Zm}X\text{-def}$ **by** auto

lemma $X\text{-Zm}X\text{-rho-psim}[simp]$:

shows $\text{Complex-Matrix.trace } (\text{rho-psim} * X\text{-Zm}X) = 1 / (\text{sqrt } 2)$
apply ($\text{auto simp add: ket-10-def ket-1-def } X\text{-Zm}X\text{-def } \text{Zm}X\text{-def } X\text{-def ket-01-def}$
 $Z\text{-def rho-psim-def ket-psim-def rank-1-proj-def outer-prod-def ket-0-def}$)
apply ($\text{auto simp add: Complex-Matrix.trace-def}$)
apply ($\text{simp add: sum-4-elems}$)
apply ($\text{simp add: csqrt-2-sq}$)
done

lemma $Z\text{-Zm}X\text{-rho-psim}[simp]$:

shows $\text{Complex-Matrix.trace } (\text{rho-psim} * Z\text{-Zm}X) = -1 / (\text{sqrt } 2)$
apply ($\text{auto simp add: rho-psim-def ket-psim-def ket-10-def}$)
apply ($\text{auto simp add: Z-Zm}X\text{-def } Z\text{-def } \text{Zm}X\text{-def } X\text{-def}$)
apply ($\text{auto simp add: rank-1-proj-def outer-prod-def ket-01-def ket-1-def ket-0-def}$)
apply ($\text{auto simp add: Complex-Matrix.trace-def sum-4-elems}$)
apply ($\text{simp add: csqrt-2-sq}$)
done

lemma $X\text{-Xp}Z\text{-rho-psim}[simp]$:

shows $\text{Complex-Matrix.trace } (\text{rho-psim} * X\text{-Xp}Z) = 1 / (\text{sqrt } 2)$
apply ($\text{auto simp add: rho-psim-def ket-psim-def ket-10-def}$)
apply ($\text{auto simp add: X-Xp}Z\text{-def } Z\text{-def } \text{Xp}Z\text{-def } X\text{-def}$)
apply ($\text{auto simp add: rank-1-proj-def outer-prod-def ket-01-def ket-1-def ket-0-def}$)
apply ($\text{auto simp add: Complex-Matrix.trace-def sum-4-elems}$)
apply ($\text{simp add: csqrt-2-sq}$)
done

lemma $Z\text{-Xp}Z\text{-rho-psim}[simp]$:

shows $\text{Complex-Matrix.trace } (\text{rho-psim} * Z\text{-Xp}Z) = 1 / (\text{sqrt } 2)$
apply ($\text{auto simp add: rho-psim-def ket-psim-def ket-10-def}$)
apply ($\text{auto simp add: Z-Xp}Z\text{-def } \text{Xp}Z\text{-def } X\text{-def } Z\text{-def}$)
apply ($\text{auto simp add: rank-1-proj-def outer-prod-def ket-01-def ket-1-def ket-0-def}$)
apply ($\text{auto simp add: Complex-Matrix.trace-def sum-4-elems}$)
apply ($\text{simp add: csqrt-2-sq}$)
done

definition $Z\text{-I}$ **where**

$Z\text{-I} = \text{tensor-mat } Z \ (1_m \ 2)$

lemma $Z\text{-I-carrier}$:

shows $Z\text{-I} \in \text{carrier-mat } 4 \ 4$ **using** $\text{tensor-mat-carrier } Z\text{-carrier}$ **unfolding**
 $Z\text{-I-def}$ **by** auto

lemma *Z-I-hermitian*:

shows *hermitian Z-I unfolding Z-I-def using tensor-mat-hermitian*[of $Z \ 2 \ 1_m \ 2 \ 2$]

by (*simp add: Z-carrier Z-hermitian hermitian-one*)

lemma *Z-I-unitary*:

shows *unitary Z-I unfolding Z-I-def using tensor-mat-unitary*[of $Z \ 1_m \ 2$]
Z-carrier unitary-Z

using *unitary-one by auto*

lemma *Z-I-spectrum*:

shows $\{Re\ x \mid x \in spectrum\ Z-I\} \subseteq \{-1, 1\}$ **using** *unitary-hermitian-Re-spectrum Z-I-hermitian*

Z-I-unitary Z-I-carrier by simp

definition *X-I where*

$X-I = tensor-mat\ X\ (1_m\ 2)$

lemma *X-I-carrier*:

shows $X-I \in carrier-mat\ 4\ 4$ **using** *tensor-mat-carrier X-carrier unfolding X-I-def by auto*

lemma *X-I-hermitian*:

shows *hermitian X-I unfolding X-I-def using tensor-mat-hermitian*[of $X \ 2 \ 1_m \ 2 \ 2$]

by (*simp add: X-carrier X-hermitian hermitian-one*)

lemma *X-I-unitary*:

shows *unitary X-I unfolding X-I-def using tensor-mat-unitary*[of $X \ 1_m \ 2$]
X-carrier unitary-X

using *unitary-one by auto*

lemma *X-I-spectrum*:

shows $\{Re\ x \mid x \in spectrum\ X-I\} \subseteq \{-1, 1\}$ **using** *unitary-hermitian-Re-spectrum X-I-hermitian*

X-I-unitary X-I-carrier by simp

definition *I-XpZ where*

$I-XpZ = tensor-mat\ (1_m\ 2)\ XpZ$

lemma *I-XpZ-carrier*:

shows $I-XpZ \in carrier-mat\ 4\ 4$ **using** *tensor-mat-carrier XpZ-carrier unfolding I-XpZ-def by auto*

lemma *I-XpZ-hermitian*:

shows *hermitian I-XpZ unfolding I-XpZ-def using tensor-mat-hermitian*[of $1_m \ 2 \ 2\ XpZ \ 2$]

by (*simp add: XpZ-carrier XpZ-hermitian hermitian-one*)

lemma *I-XpZ-unitary*:

shows *unitary I-XpZ unfolding I-XpZ-def using tensor-mat-unitary*[of 1_m 2 XpZ] *XpZ-carrier*
unitary-XpZ using unitary-one by auto

lemma *I-XpZ-spectrum*:

shows $\{Re\ x \mid x \in spectrum\ I-XpZ\} \subseteq \{-1, 1\}$ **using** *unitary-hermitian-Re-spectrum*
I-XpZ-hermitian I-XpZ-unitary I-XpZ-carrier by simp

definition *I-ZmX where*

I-ZmX = tensor-mat (1_m 2) *ZmX*

lemma *I-ZmX-carrier*:

shows *I-ZmX* \in *carrier-mat* 4 4 **using** *tensor-mat-carrier ZmX-carrier unfolding I-ZmX-def by auto*

lemma *I-ZmX-hermitian*:

shows *hermitian I-ZmX unfolding I-ZmX-def using tensor-mat-hermitian*[of 1_m 2 2 *ZmX* 2]
by (*simp add: ZmX-carrier ZmX-hermitian hermitian-one*)

lemma *I-ZmX-unitary*:

shows *unitary I-ZmX unfolding I-ZmX-def using tensor-mat-unitary*[of 1_m 2 *ZmX*] *ZmX-carrier*
unitary-ZmX using unitary-one by auto

lemma *I-ZmX-spectrum*:

shows $\{Re\ x \mid x \in spectrum\ I-ZmX\} \subseteq \{-1, 1\}$ **using** *unitary-hermitian-Re-spectrum*
I-ZmX-hermitian
I-ZmX-unitary I-ZmX-carrier by simp

lemma *X-I-ZmX-eq*:

shows $X-I * I-ZmX = X-ZmX$ **unfolding** *X-I-def I-ZmX-def X-ZmX-def using*
mult-distr-tensor
by (*metis* (*no-types*, *lifting*) *X-inv ZmX-inv index-mult-mat*(2) *index-mult-mat*(3)
index-one-mat(2)
index-one-mat(3) *left-mult-one-mat'* *pos2* *right-mult-one-mat'*)

lemma *X-I-XpZ-eq*:

shows $X-I * I-XpZ = X-XpZ$ **unfolding** *X-I-def I-XpZ-def X-XpZ-def using*
mult-distr-tensor
by (*metis* (*no-types*, *lifting*) *X-inv XpZ-inv index-mult-mat*(2) *index-mult-mat*(3)
index-one-mat(2)
index-one-mat(3) *left-mult-one-mat'* *pos2* *right-mult-one-mat'*)

lemma *Z-I-XpZ-eq*:

shows $Z-I * I-XpZ = Z-XpZ$ **unfolding** *Z-I-def I-XpZ-def Z-XpZ-def using*
mult-distr-tensor

by (*metis* (*no-types*, *lifting*) *Z-inv XpZ-inv index-mult-mat(2) index-mult-mat(3)* *index-one-mat(2)*)

index-one-mat(3) left-mult-one-mat' pos2 right-mult-one-mat')

lemma *Z-I-ZmX-eq*:

shows *Z-I * I-ZmX = Z-ZmX unfolding Z-I-def I-ZmX-def Z-ZmX-def using mult-distr-tensor*

by (*metis* (*no-types*, *lifting*) *Z-inv ZmX-inv index-mult-mat(2) index-mult-mat(3)* *index-one-mat(2)*)

index-one-mat(3) left-mult-one-mat' pos2 right-mult-one-mat')

4.2.3 No local hidden variable

We show that the local hidden variable hypothesis cannot hold by exhibiting a quantum expectation value that is greater than the upper-bound given by the CHSH inequality.

locale *bin-cpx = cpx-sq-mat +*

assumes *dim4: dimR = 4*

lemma (**in** *bin-cpx*) *X-I-XpZ-trace*:

assumes *lhv M X-I I-XpZ R Vx Vp*

and *R ∈ fc-mats*

shows *LINT w|M. (qt-expect X-I Vx w) * (qt-expect I-XpZ Vp w) = Re (Complex-Matrix.trace (R * X-XpZ))*

proof –

have *LINT w|M. (qt-expect X-I Vx w) * (qt-expect I-XpZ Vp w) = Re (Complex-Matrix.trace (X-I * I-XpZ * R))*

proof (*rule sum-qt-expect, (simp add: assms)*)

show *X-I ∈ fc-mats using X-I-carrier dim4 fc-mats-carrier dim-eq by simp*

show *R ∈ fc-mats using assms by simp*

show *I-XpZ ∈ fc-mats using I-XpZ-carrier dim4 fc-mats-carrier dim-eq by simp*

show *hermitian X-I using X-I-hermitian .*

show *hermitian I-XpZ using I-XpZ-hermitian .*

qed

also have *... = Re (Complex-Matrix.trace (X-XpZ * R)) using X-I-XpZ-eq by simp*

also have *... = Re (Complex-Matrix.trace (R * X-XpZ))*

proof –

have *Complex-Matrix.trace (X-XpZ * R) = Complex-Matrix.trace (R * X-XpZ)*

using *trace-comm[of X-XpZ 4 R] X-XpZ-carrier assms dim4 fc-mats-carrier dim-eq by simp*

thus *?thesis by simp*

qed

finally show *?thesis .*

qed

lemma (**in** *bin-cpx*) *X-I-XpZ-chsh*:

assumes $lhv\ M\ X-I\ I-XpZ\ rho-psim\ Vx\ Vp$
shows $LINT\ w|M. (qt-expect\ X-I\ Vx\ w) * (qt-expect\ I-XpZ\ Vp\ w) = 1/sqrt\ 2$
proof –
have $LINT\ w|M. (qt-expect\ X-I\ Vx\ w) * (qt-expect\ I-XpZ\ Vp\ w) = Re\ (Complex-Matrix.trace\ (rho-psim * X-XpZ))$
proof (*rule* $X-I-XpZ-trace$, (*simp* *add: assms*))
show $rho-psim \in fc-mats$ **using** $rho-psim-carrier\ fc-mats-carrier\ dim-eq\ dim4$
by *simp*
qed
also have $\dots = 1/sqrt\ 2$ **using** $X-XpZ-rho-psim$ **by** *simp*
finally show *?thesis* .
qed

lemma (*in bin-cpx*) $Z-I-XpZ-trace$:
assumes $lhv\ M\ Z-I\ I-XpZ\ R\ Vz\ Vp$
and $R \in fc-mats$
shows $LINT\ w|M. (qt-expect\ Z-I\ Vz\ w) * (qt-expect\ I-XpZ\ Vp\ w) = Re\ (Complex-Matrix.trace\ (R * Z-XpZ))$
proof –
have $LINT\ w|M. (qt-expect\ Z-I\ Vz\ w) * (qt-expect\ I-XpZ\ Vp\ w) = Re\ (Complex-Matrix.trace\ (Z-I * I-XpZ * R))$
proof (*rule* $sum-qt-expect$, (*simp* *add: assms*))
show $Z-I \in fc-mats$ **using** $Z-I-carrier\ dim4\ fc-mats-carrier\ dim-eq$ **by** *simp*
show $R \in fc-mats$ **using** *assms* **by** *simp*
show $I-XpZ \in fc-mats$ **using** $I-XpZ-carrier\ dim4\ fc-mats-carrier\ dim-eq$ **by** *simp*
show *hermitian* $Z-I$ **using** $Z-I-hermitian$.
show *hermitian* $I-XpZ$ **using** $I-XpZ-hermitian$.
qed
also have $\dots = Re\ (Complex-Matrix.trace\ (Z-XpZ * R))$ **using** $Z-I-XpZ-eq$ **by** *simp*
also have $\dots = Re\ (Complex-Matrix.trace\ (R * Z-XpZ))$
proof –
have $Complex-Matrix.trace\ (Z-XpZ * R) = Complex-Matrix.trace\ (R * Z-XpZ)$
using $trace-comm[of\ Z-XpZ\ 4\ R]\ Z-XpZ-carrier\ assms\ dim4\ fc-mats-carrier\ dim-eq$ **by** *simp*
thus *?thesis* **by** *simp*
qed
finally show *?thesis* .
qed

lemma (*in bin-cpx*) $Z-I-XpZ-chsh$:
assumes $lhv\ M\ Z-I\ I-XpZ\ rho-psim\ Vz\ Vp$
shows $LINT\ w|M. (qt-expect\ Z-I\ Vz\ w) * (qt-expect\ I-XpZ\ Vp\ w) = 1/sqrt\ 2$
proof –
have $LINT\ w|M. (qt-expect\ Z-I\ Vz\ w) * (qt-expect\ I-XpZ\ Vp\ w) =$

$Re (Complex-Matrix.trace (rho-psim * Z-XpZ))$
proof (rule $Z-I-XpZ$ -trace, (simp add: *assms*))
show $rho-psim \in fc-mats$ **using** $rho-psim-carrier fc-mats-carrier dim-eq dim4$
by *simp*
qed
also have $\dots = 1/sqrt\ 2$ **using** $Z-XpZ$ -rho-psim **by** *simp*
finally show *?thesis* **unfolding** $qt-expect-def$.
qed

lemma (in *bin-cpx*) $X-I-ZmX$ -trace:
assumes $lhv\ M\ X-I\ I-ZmX\ R\ Vx\ Vp$
and $R \in fc-mats$
shows $LINT\ w|M. (qt-expect\ X-I\ Vx\ w) * (qt-expect\ I-ZmX\ Vp\ w) =$
 $Re (Complex-Matrix.trace (R * X-ZmX))$
proof –
have $LINT\ w|M. (qt-expect\ X-I\ Vx\ w) * (qt-expect\ I-ZmX\ Vp\ w) =$
 $Re (Complex-Matrix.trace (X-I * I-ZmX * R))$
proof (rule *sum-qt-expect*, (simp add: *assms*))
show $X-I \in fc-mats$ **using** $X-I-carrier\ dim4\ fc-mats-carrier\ dim-eq$ **by** *simp*
show $R \in fc-mats$ **using** *assms* **by** *simp*
show $I-ZmX \in fc-mats$ **using** $I-ZmX-carrier\ dim4\ fc-mats-carrier\ dim-eq$ **by**
simp
show *hermitian* $X-I$ **using** $X-I-hermitian$.
show *hermitian* $I-ZmX$ **using** $I-ZmX-hermitian$.
qed
also have $\dots = Re (Complex-Matrix.trace (X-ZmX * R))$ **using** $X-I-ZmX-eq$ **by**
simp
also have $\dots = Re (Complex-Matrix.trace (R * X-ZmX))$
proof –
have $Complex-Matrix.trace (X-ZmX * R) = Complex-Matrix.trace (R * X-ZmX)$

using *trace-comm*[of $X-ZmX\ 4\ R$] $X-ZmX-carrier\ assms\ dim4\ fc-mats-carrier$
 $dim-eq$ **by** *simp*
thus *?thesis* **by** *simp*
qed
finally show *?thesis* .
qed

lemma (in *bin-cpx*) $X-I-ZmX$ -chsh:
assumes $lhv\ M\ X-I\ I-ZmX\ rho-psim\ Vx\ Vp$
shows $LINT\ w|M. (qt-expect\ X-I\ Vx\ w) * (qt-expect\ I-ZmX\ Vp\ w) =$
 $1/sqrt\ 2$
proof –
have $LINT\ w|M. (qt-expect\ X-I\ Vx\ w) * (qt-expect\ I-ZmX\ Vp\ w) =$
 $Re (Complex-Matrix.trace (rho-psim * X-ZmX))$
proof (rule $X-I-ZmX$ -trace, (simp add: *assms*))
show $rho-psim \in fc-mats$ **using** $rho-psim-carrier\ fc-mats-carrier\ dim-eq\ dim4$
by *simp*
qed

also have ... = $1/\text{sqrt } 2$ using $X\text{-ZmX-rho-psim}$ by simp
 finally show $?thesis$ unfolding $qt\text{-expect-def}$.
 qed

lemma (in bin-cpx) $Z\text{-I-ZmX-trace}$:

assumes $lhv M Z\text{-I } I\text{-ZmX } R Vz Vp$

and $R \in \text{fc-mats}$

shows $LINT w|M. (qt\text{-expect } Z\text{-I } Vz w) * (qt\text{-expect } I\text{-ZmX } Vp w) =$
 $Re (Complex\text{-Matrix.trace } (R * Z\text{-ZmX}))$

proof –

have $LINT w|M. (qt\text{-expect } Z\text{-I } Vz w) * (qt\text{-expect } I\text{-ZmX } Vp w) =$
 $Re (Complex\text{-Matrix.trace } (Z\text{-I} * I\text{-ZmX} * R))$

proof (rule sum-qt-expect , (simp add: assms))

show $Z\text{-I} \in \text{fc-mats}$ using $Z\text{-I-carrier dim}_4 \text{fc-mats-carrier dim-eq}$ by simp

show $R \in \text{fc-mats}$ using assms by simp

show $I\text{-ZmX} \in \text{fc-mats}$ using $I\text{-ZmX-carrier dim}_4 \text{fc-mats-carrier dim-eq}$ by
 simp

show $\text{hermitian } Z\text{-I}$ using $Z\text{-I-hermitian}$.

show $\text{hermitian } I\text{-ZmX}$ using $I\text{-ZmX-hermitian}$.

qed

also have ... = $Re (Complex\text{-Matrix.trace } (Z\text{-ZmX} * R))$ using $Z\text{-I-ZmX-eq}$ by
 simp

also have ... = $Re (Complex\text{-Matrix.trace } (R * Z\text{-ZmX}))$

proof –

have $Complex\text{-Matrix.trace } (Z\text{-ZmX} * R) = Complex\text{-Matrix.trace } (R * Z\text{-ZmX})$

using $\text{trace-comm}[of } Z\text{-ZmX } _ R]$ $Z\text{-ZmX-carrier assms dim}_4 \text{fc-mats-carrier}$
 dim-eq by simp

thus $?thesis$ by simp

qed

finally show $?thesis$.

qed

lemma (in bin-cpx) $Z\text{-I-ZmX-chsh}$:

assumes $lhv M Z\text{-I } I\text{-ZmX } rho\text{-psim } Vz Vp$

shows $LINT w|M. (qt\text{-expect } Z\text{-I } Vz w) * (qt\text{-expect } I\text{-ZmX } Vp w) =$
 $-1/\text{sqrt } 2$

proof –

have $LINT w|M. (qt\text{-expect } Z\text{-I } Vz w) * (qt\text{-expect } I\text{-ZmX } Vp w) =$
 $Re (Complex\text{-Matrix.trace } (rho\text{-psim} * Z\text{-ZmX}))$

proof (rule $Z\text{-I-ZmX-trace}$, (simp add: assms))

show $rho\text{-psim} \in \text{fc-mats}$ using $rho\text{-psim-carrier fc-mats-carrier dim-eq dim}_4$
 by simp

qed

also have ... = $-1/\text{sqrt } 2$ using $Z\text{-ZmX-rho-psim}$ by simp

finally show $?thesis$ unfolding $qt\text{-expect-def}$.

qed

lemma (in bin-cpx) chsh-upper-bound :

assumes *prob-space M*
and *lhv M X-I I-XpZ rho-psim Vx Vp*
and *lhv M Z-I I-XpZ rho-psim Vz Vp*
and *lhv M X-I I-ZmX rho-psim Vx Vm*
and *lhv M Z-I I-ZmX rho-psim Vz Vm*
shows $|(\text{LINT } w|M. \text{qt-expect } X-I \text{ Vx } w * \text{qt-expect } I-ZmX \text{ Vm } w) +$
 $(\text{LINT } w|M. \text{qt-expect } Z-I \text{ Vz } w * \text{qt-expect } I-XpZ \text{ Vp } w) +$
 $(\text{LINT } w|M. \text{qt-expect } X-I \text{ Vx } w * \text{qt-expect } I-XpZ \text{ Vp } w) -$
 $(\text{LINT } w|M. \text{qt-expect } Z-I \text{ Vz } w * \text{qt-expect } I-ZmX \text{ Vm } w)|$
 ≤ 2
proof (*rule prob-space.chsh-expect*)
show *AE w in M. |qt-expect X-I Vx w| ≤ 1* **unfolding** *qt-expect-def*
proof (*rule spectrum-abs-1-weighted-suml*)
show *X-I ∈ fc-mats* **using** *X-I-carrier fc-mats-carrier dim-eq dim4* **by** *simp*
show *hermitian X-I* **using** *X-I-hermitian* .
show *lhv M X-I I-XpZ rho-psim Vx Vp* **using** *assms* **by** *simp*
show $\{Re \ x \ |x. \ x \in \text{spectrum } X-I\} \subseteq \{-1, 1\}$ **using** *X-I-spectrum* **by** *simp*
show $\{Re \ x \ |x. \ x \in \text{spectrum } X-I\} \neq \{\}$ **using** *spectrum-ne X-I-hermitian ⟨X-I*
 $\in \text{fc-mats}\rangle$ **by** *auto*
qed
show *AE w in M. |qt-expect Z-I Vz w| ≤ 1* **unfolding** *qt-expect-def*
proof (*rule spectrum-abs-1-weighted-suml*)
show *Z-I ∈ fc-mats* **using** *Z-I-carrier fc-mats-carrier dim-eq dim4* **by** *simp*
show *hermitian Z-I* **using** *Z-I-hermitian* .
show *lhv M Z-I I-XpZ rho-psim Vz Vp* **using** *assms* **by** *simp*
show $\{Re \ x \ |x. \ x \in \text{spectrum } Z-I\} \subseteq \{-1, 1\}$ **using** *Z-I-spectrum* **by** *simp*
show $\{Re \ x \ |x. \ x \in \text{spectrum } Z-I\} \neq \{\}$ **using** *spectrum-ne Z-I-hermitian ⟨Z-I*
 $\in \text{fc-mats}\rangle$ **by** *auto*
qed
show *AE w in M. |qt-expect I-XpZ Vp w| ≤ 1* **unfolding** *qt-expect-def*
proof (*rule spectrum-abs-1-weighted-sumr*)
show *I-XpZ ∈ fc-mats* **using** *I-XpZ-carrier fc-mats-carrier dim-eq dim4* **by**
simp
show *hermitian I-XpZ* **using** *I-XpZ-hermitian* .
show *lhv M Z-I I-XpZ rho-psim Vz Vp* **using** *assms* **by** *simp*
show $\{Re \ x \ |x. \ x \in \text{spectrum } I-XpZ\} \subseteq \{-1, 1\}$ **using** *I-XpZ-spectrum* **by**
simp
show $\{Re \ x \ |x. \ x \in \text{spectrum } I-XpZ\} \neq \{\}$ **using** *spectrum-ne I-XpZ-hermitian*
 $\langle I-XpZ \in \text{fc-mats}\rangle$
by *auto*
qed
show *AE w in M. |qt-expect I-ZmX Vm w| ≤ 1* **unfolding** *qt-expect-def*
proof (*rule spectrum-abs-1-weighted-sumr*)
show *I-ZmX ∈ fc-mats* **using** *I-ZmX-carrier fc-mats-carrier dim-eq dim4* **by**
simp
show *hermitian I-ZmX* **using** *I-ZmX-hermitian* .
show *lhv M Z-I I-ZmX rho-psim Vz Vm* **using** *assms* **by** *simp*
show $\{Re \ x \ |x. \ x \in \text{spectrum } I-ZmX\} \subseteq \{-1, 1\}$ **using** *I-ZmX-spectrum* **by**
simp

show $\{Re\ x \mid x. x \in \text{spectrum } I\text{-ZmX}\} \neq \{\}$ **using** *spectrum-ne I-ZmX-hermitian*
(I-ZmX ∈ fc-mats)
by auto
qed
show *prob-space M using assms by simp*
show *integrable M (λw. qt-expect X-I Vx w * qt-expect I-ZmX Vm w)*
using *spectr-sum-integrable[of M] assms by simp*
show *integrable M (λw. qt-expect Z-I Vz w * qt-expect I-ZmX Vm w)*
using *spectr-sum-integrable[of M] assms by simp*
show *integrable M (λw. qt-expect X-I Vx w * qt-expect I-XpZ Vp w)*
using *spectr-sum-integrable[of M] assms by simp*
show *integrable M (λw. qt-expect Z-I Vz w * qt-expect I-XpZ Vp w)*
using *spectr-sum-integrable[of M] assms by simp*
qed

lemma (in *bin-cpx*) *quantum-value:*
assumes *lhv M X-I I-XpZ rho-psim Vx Vp*
and *lhv M Z-I I-XpZ rho-psim Vz Vp*
and *lhv M X-I I-ZmX rho-psim Vx Vm*
and *lhv M Z-I I-ZmX rho-psim Vz Vm*
shows $|(LINT\ w|M. qt-expect\ X-I\ Vx\ w * qt-expect\ I-ZmX\ Vm\ w) +$
 $(LINT\ w|M. qt-expect\ Z-I\ Vz\ w * qt-expect\ I-XpZ\ Vp\ w) +$
 $(LINT\ w|M. qt-expect\ X-I\ Vx\ w * qt-expect\ I-XpZ\ Vp\ w) -$
 $(LINT\ w|M. qt-expect\ Z-I\ Vz\ w * qt-expect\ I-ZmX\ Vm\ w)|$
 $= 2 * \text{sqrt } 2$
proof –
have *LINT w|M. qt-expect X-I Vx w * qt-expect I-ZmX Vm w = 1/sqrt 2*
using *X-I-ZmX-chsh[of M] assms unfolding qt-expect-def by simp*
moreover have *b: LINT w|M. qt-expect X-I Vx w * qt-expect I-XpZ Vp w =*
 $1/\text{sqrt } 2$
using *X-I-XpZ-chsh[of M] assms unfolding qt-expect-def by simp*
moreover have *c: LINT w|M. qt-expect Z-I Vz w * qt-expect I-XpZ Vp w =*
 $1/\text{sqrt } 2$
using *Z-I-XpZ-chsh[of M] assms unfolding qt-expect-def by simp*
moreover have *LINT w|M. qt-expect Z-I Vz w * qt-expect I-ZmX Vm w =*
 $-1/\text{sqrt } 2$
using *Z-I-ZmX-chsh[of M] assms unfolding qt-expect-def by simp*
ultimately have $(LINT\ w|M. qt-expect\ X-I\ Vx\ w * qt-expect\ I-ZmX\ Vm\ w) +$
 $(LINT\ w|M. qt-expect\ X-I\ Vx\ w * qt-expect\ I-XpZ\ Vp\ w) +$
 $(LINT\ w|M. qt-expect\ Z-I\ Vz\ w * qt-expect\ I-XpZ\ Vp\ w) -$
 $(LINT\ w|M. qt-expect\ Z-I\ Vz\ w * qt-expect\ I-ZmX\ Vm\ w) = 4/(\text{sqrt } 2)$ **by**
simp
also have $\dots = (4 * (\text{sqrt } 2))/(\text{sqrt } 2 * (\text{sqrt } 2))$
by (*metis mult-numeral-1-right real-divide-square-eq times-divide-eq-right*)
also have $\dots = (4 * (\text{sqrt } 2)) / 2$ **by simp**
also have $\dots = 2 * (\text{sqrt } 2)$ **by simp**
finally have $(LINT\ w|M. qt-expect\ X-I\ Vx\ w * qt-expect\ I-ZmX\ Vm\ w) +$
 $(LINT\ w|M. qt-expect\ X-I\ Vx\ w * qt-expect\ I-XpZ\ Vp\ w) +$
 $(LINT\ w|M. qt-expect\ Z-I\ Vz\ w * qt-expect\ I-XpZ\ Vp\ w) -$

```

      (LINT w|M. qt-expect Z-I Vz w * qt-expect I-ZmX Vm w) = 2 * sqrt 2 .
    thus ?thesis by (simp add: b c)
  qed

lemma (in bin-cpx) no-lhv:
  assumes lhv M X-I I-XpZ rho-psim Vx Vp
  and lhv M Z-I I-XpZ rho-psim Vz Vp
  and lhv M X-I I-ZmX rho-psim Vx Vm
  and lhv M Z-I I-ZmX rho-psim Vz Vm
shows False
proof -
  have prob-space M using assms unfolding lhv-def by simp
  have 2 * sqrt 2 = |(LINT w|M. qt-expect X-I Vx w * qt-expect I-ZmX Vm w) +
    (LINT w|M. qt-expect Z-I Vz w * qt-expect I-XpZ Vp w) +
    (LINT w|M. qt-expect X-I Vx w * qt-expect I-XpZ Vp w) -
    (LINT w|M. qt-expect Z-I Vz w * qt-expect I-ZmX Vm w)|
    using assms quantum-value[of M] by simp
  also have ... ≤ 2 using chsh-upper-bound[of M] assms ⟨prob-space M⟩ by simp
  finally have 2 * sqrt 2 ≤ 2 .
  thus False by simp
qed

end

```