

The CHSH inequality: Tsirelson's upper-bound and other results

Mnacho Echenim and Mehdi Mhalla and Coraline Mori

April 6, 2024

Abstract

The CHSH inequality, named after Clauser, Horne, Shimony and Holt, was used by Alain Aspect to prove experimentally that Einstein's hypothesis stating that quantum mechanics could be defined using local hidden variables was incorrect. The CHSH inequality is based on a setting in which an experiment consisting of two separate parties performing joint measurements is run several times, and a score is derived from these runs. If the local hidden variable hypothesis had been correct, this score would have been bounded by 2, but a suitable choice of observables in a quantum setting permits to violate this inequality when measuring the Bell state; this is the result that Aspect obtained experimentally. Tsirelson answered the question of how large this violation could be by proving that in the quantum setting, $2\sqrt{2}$ is the highest score that can be obtained when running this experiment. Along with elementary results on density matrices which represent quantum states in the finite dimensional setting, we formalize Tsirelson's result and summarize the main results on the CHSH score:

1. Under the local hidden variable hypothesis, this score admits 2 as an upper-bound.
2. When the density matrix under consideration is separable, the upper-bound cannot be violated.
3. When one of the parties in the experiment performs measures using commuting observables, this upper-bound remains valid.
4. Otherwise, the upper-bound of this score is $2\sqrt{2}$, regardless of the observables that are used and the quantum state that is measured, and
5. This upper-bound is reached for a suitable choice of observables when measuring the Bell state.

Contents

1 Basic algebraic results	2
----------------------------------	----------

2	Results in linear algebra	5
3	Results on tensor products	10
4	Preliminary results	18
4.1	Commutator and anticommutator	18
5	Maximum modulus in a spectrum	19
5.1	Definition and basic properties for Hermitian matrices	19
5.2	Eigenvector for the element with maximum modulus	23
6	The \mathcal{L}_2 operator norm	24
6.1	Definition and preliminary results	24
6.2	The \mathcal{L}_2 operator norm is equal to the maximum singular value	26
6.3	Consequences for the \mathcal{L}_2 operator norm	28
7	On density matrices	30
7.1	Density matrix characterization	30
7.2	Separable density matrices	31
7.3	Characterization of pure states	32
8	Quantum expectation values and traces	33
9	CHSH inequalities	34
9.1	Some intermediate results for particular observables	34
9.2	The CHSH operator and expectation	35
9.3	CHSH inequality for separable density matrices	40
9.4	CHSH inequality for commuting observables	41
9.5	Result summary on the CHSH inequalities	41

```
theory Tensor-Mat-Compl-Properties
imports
  Commuting-Hermitian.Spectral-Theory-Complements
  Projective-Measurements.Projective-Measurements
begin
```

1 Basic algebraic results

```
lemma pos-sum-gt-0:
  assumes finite I
  and ⋀ i. i ∈ I ⟹ (0::'a :: linordered-field) ≤ f i
  and 0 < sum f I
  shows ∃ j ∈ I. 0 < f j
  ⟨proof⟩
```

```

lemma pos-square-1-elem:
  assumes finite I
  and  $\bigwedge i. i \in I \implies (0::real) \leq f i$ 
  and sum f I = 1
  and sum ( $\lambda x. f x * f x$ ) I = 1
  shows  $\exists j \in I. f j = 1$ 
  ⟨proof⟩

lemma cpx-pos-square-1-elem:
  assumes finite I
  and  $\bigwedge i. i \in I \implies (0::complex) \leq f i$ 
  and sum f I = 1
  and sum ( $\lambda x. f x * f x$ ) I = 1
  shows  $\exists j \in I. f j = 1$ 
  ⟨proof⟩

lemma sum-eq-elmt:
  assumes finite I
  and  $\bigwedge i. i \in I \implies (0::'a :: linordered-field) \leq f i$ 
  and sum f I = c
  and  $\bigvee j \in I. f j = c$ 
  shows  $\forall k \in (I - \{j\}). f k = 0$ 
  ⟨proof⟩

lemma cpx-sum-eq-elmt:
  assumes finite I
  and  $\bigwedge i. i \in I \implies (0::complex) \leq f i$ 
  and sum f I = c
  and  $\bigvee j \in I. f j = c$ 
  shows  $\forall k \in (I - \{j\}). f k = 0$ 
  ⟨proof⟩

lemma sum-nat-div-mod:
  shows sum ( $\lambda i. sum (\lambda j. f i * g j) \{.. < (m::nat)\} \{.. < (n::nat)\}$ ) =
    sum ( $\lambda k. f (k \text{ div } m) * g (k \text{ mod } m)$ )  $\{.. < n*m\}$ 
  ⟨proof⟩

lemma abs-cmod-eq:
  fixes z::complex
  shows |z| = cmod z
  ⟨proof⟩

lemma real-cpx-abs-leq:
  fixes A::complex
  assumes A ∈ Reals
  and B ∈ Reals
  and |A * B| ≤ 1

```

```
shows |Re A * Re B| ≤ 1
⟨proof⟩
```

```
lemma cpx-real-abs-eq:
  fixes z::complex and r::real
  assumes z ∈ Reals
  and z = r
  shows |z| = |r|
⟨proof⟩
```

```
lemma cpx-real-abs-leq:
  fixes z::complex and r::real
  assumes z ∈ Reals
  and z = r
  and |r| ≤ k
  shows |z| ≤ (k::real)
⟨proof⟩
```

```
lemma cpx-abs-mult-le-1:
  fixes z::complex
  assumes |z| ≤ 1
  and |z'| ≤ 1
  shows |z*z'| ≤ 1
⟨proof⟩
```

```
lemma sum-abs-cpx:
  shows |sum K I| ≤ sum (λx. |(K x)::complex|) I
⟨proof⟩
```

```
lemma abs-mult-cpx:
  fixes z::complex
  assumes 0 ≤ (a::real)
  shows |a*z| = a * |z|
⟨proof⟩
```

```
lemma cpx-ge-0-real:
  fixes c::complex
  assumes 0 ≤ c
  and c ∈ Reals
  shows 0 ≤ Re c
⟨proof⟩
```

```
lemma cpx-of-real-ge-0:
  assumes 0 ≤ complex-of-real a
  shows 0 ≤ a
⟨proof⟩
```

```
lemma set-cst-list:
```

shows $(\bigwedge i. i < \text{length } l \implies \text{!}i = x) \implies 0 < \text{length } l \implies \text{set } l = \{x\}$
 $\langle \text{proof} \rangle$

lemma *pos-mult-Max*:
assumes *finite F*
and $F \neq \{\}$
and $0 \leq x$
and $\forall a \in F. 0 \leq (a::\text{real})$
shows $\text{Max.F} \{x * a | a \in F\} = x * \text{Max.F} F$
 $\langle \text{proof} \rangle$

lemma *square-Max*:
assumes *finite A*
and $A \neq \{\}$
and $\forall a \in A. 0 \leq ((f a)::\text{real})$
and $b = \text{Max.F} \{f a | a \in A\}$
shows $\text{Max.F} \{f a * f a | a \in A\} = b * b$
 $\langle \text{proof} \rangle$

lemma *ereal-Sup-switch*:
assumes $\forall m \in P. (b::\text{real}) \leq f m$
and $\forall m \in P. f m \leq (c::\text{real})$
and $P \neq \{\}$
shows $\text{ereal} (\text{Sup} (f ` P)) = (\bigsqcup m \in P. \text{ereal} (f m))$
 $\langle \text{proof} \rangle$

lemma *Sup-ge-real*:
assumes $a \in (A::\text{real set})$
and $\forall a \in A. a \leq c$
and $\forall a \in A. b \leq a$
shows $a \leq \text{Sup} A$
 $\langle \text{proof} \rangle$

lemma *Sup-real-le*:
assumes $\forall a \in (A::\text{real set}). a \leq c$
and $\forall a \in A. b \leq a$
and $A \neq \{\}$
shows $\text{Sup} A \leq c$
 $\langle \text{proof} \rangle$

2 Results in linear algebra

lemma *mat-add-eq-0-if*:
fixes $A::'a :: \text{group-add Matrix.mat}$
assumes $A \in \text{carrier-mat } n m$
and $B \in \text{carrier-mat } n m$
and $A + B = 0_m n m$
shows $B = -A$
 $\langle \text{proof} \rangle$

```

lemma trace-rank-1-proj:
  shows Complex-Matrix.trace (rank-1-proj v) =  $\|v\|^2$ 
  ⟨proof⟩

lemma trace-ch-expand:
  fixes A::'a::{minus,comm-ring} Matrix.mat
  assumes A ∈ carrier-mat n n
  and B ∈ carrier-mat n n
  and C ∈ carrier-mat n n
  and D ∈ carrier-mat n n
  shows Complex-Matrix.trace (A - B + C + D) =
    Complex-Matrix.trace A - Complex-Matrix.trace B +
    Complex-Matrix.trace C + Complex-Matrix.trace D
  ⟨proof⟩

lemma squared-A-trace:
  assumes A ∈ carrier-mat n n
  and unitarily-equiv A B U
  shows Complex-Matrix.trace (A*A) = Complex-Matrix.trace (B*B)
  ⟨proof⟩

lemma squared-A-trace':
  assumes A ∈ carrier-mat n n
  and unitary-diag A B U
  shows Complex-Matrix.trace (A*A) = ( $\sum_{i \in \{0..n\}} (B \$\$ (i,i)) * (B \$\$ (i,i))$ )
  ⟨proof⟩

lemma positive-square-trace:
  assumes A ∈ carrier-mat n n
  and Complex-Matrix.trace A = (1::real)
  and Complex-Matrix.trace (A*A) = 1
  and real-diag-decomp A B U
  and Complex-Matrix.positive A
  and 0 < n
  shows  $\exists j < n. B \$\$ (j,j) = 1 \wedge (\forall i < n. i \neq j \rightarrow B \$\$ (i,i) = 0)$ 
  ⟨proof⟩

lemma idty-square:
  shows ((1m n)::'a :: semiring-1 Matrix.mat) * (1m n) = 1m n
  ⟨proof⟩

lemma pos-hermitian-trace-reals:
  fixes A::complex Matrix.mat
  assumes A ∈ carrier-mat n n
  and B ∈ carrier-mat n n
  and 0 < n
  and Complex-Matrix.positive A

```

```

and hermitian B
shows Complex-Matrix.trace (B*A) ∈ Reals
⟨proof⟩

lemma pos-hermitian-trace-reals':
  fixes A::complex Matrix.mat
  assumes A ∈ carrier-mat n n
  and B ∈ carrier-mat n n
  and 0 < n
  and Complex-Matrix.positive A
  and hermitian B
  shows Complex-Matrix.trace (A*B) ∈ Reals
  ⟨proof⟩

lemma hermitian-commute:
  assumes hermitian A
  and hermitian B
  and A*B = B*A
  shows hermitian (A*B)
  ⟨proof⟩

lemma idty-unitary-diag:
  assumes unitary-diag (1m n) B U
  shows B = 1m n
  ⟨proof⟩

lemma diag-mat-idty:
  assumes 0 < n
  shows set (diag-mat ((1m n)::'a::{one,zero} Matrix.mat)) = {1}
    (is ?L = ?R)
  ⟨proof⟩

lemma idty-spectrum:
  assumes 0 < n
  shows spectrum ((1m n)::complex Matrix.mat) = {1}
  ⟨proof⟩

lemma spectrum-ne:
  fixes A::complex Matrix.mat
  assumes A ∈ carrier-mat n n
  and 0 < n
  shows spectrum A ≠ {} ⟨proof⟩

lemma unitary-diag-square-spectrum:
  fixes A::complex Matrix.mat
  assumes hermitian A
  and A ∈ carrier-mat n n
  and unitary-diag A B U

```

```

shows spectrum ( $A \otimes A$ ) = set (diag-mat ( $B \otimes B$ ))
⟨proof⟩

lemma diag-mat-square-eq:
  fixes  $B::'a::\{ring\}$  Matrix.mat
  assumes diagonal-mat  $B$ 
  and  $B \in carrier\text{-}mat n n$ 
  shows set (diag-mat ( $B \otimes B$ )) = { $b \otimes b | b. b \in set (diag\text{-}mat B)$ }
⟨proof⟩

lemma hermitian-square-spectrum-eq:
  fixes  $A::complex$  Matrix.mat
  assumes hermitian  $A$ 
  and  $A \in carrier\text{-}mat n n$ 
  and  $0 < n$ 
  shows spectrum ( $A \otimes A$ ) = { $a \otimes a | a. a \in spectrum A$ }
⟨proof⟩

lemma adjoint-uminus:
  shows Complex-Matrix.adjoint ( $-A$ ) = - (Complex-Matrix.adjoint  $A$ )
⟨proof⟩

lemma (in fixed-carrier-mat) sum-mat-zero:
  assumes finite  $I$ 
  and  $\bigwedge i. i \in I \implies A i \in fc\text{-mats}$ 
  and  $\bigwedge i. i \in I \implies f i = 0$ 
  shows sum-mat ( $\lambda i. (f i) \cdot_m (A i)$ )  $I = 0_m \dimR \dimC$  ⟨proof⟩

lemma (in fixed-carrier-mat) sum-mat-zero':
  fixes  $A::'b \Rightarrow 'a$  Matrix.mat
  assumes finite  $I$ 
  and  $\bigwedge i. i \in I \implies A i = 0_m \dimR \dimC$ 
  shows sum-mat  $A I = 0_m \dimR \dimC$  ⟨proof⟩

lemma (in fixed-carrier-mat) sum-mat-remove:
  assumes  $A \setminus I \subseteq fc\text{-mats}$ 
  and  $A: finite I$  and  $x: x \in I$ 
  shows sum-mat  $A I = A x + sum\text{-}mat A (I - \{x\})$  ⟨proof⟩

lemma (in fixed-carrier-mat) sum-mat-singleton:
  fixes  $A::'b \Rightarrow 'a$  Matrix.mat
  assumes finite  $I$ 
  and  $A \setminus I \subseteq fc\text{-mats}$ 
  and  $j \in I$ 
  and  $\forall i \in I. i \neq j \longrightarrow f i = 0$ 
  shows sum-mat ( $\lambda i. (f i) \cdot_m (A i)$ )  $I = f j \cdot_m (A j)$ 
⟨proof⟩

context fixed-carrier-mat

```

```

begin
lemma sum-mat-disj-union:
  assumes finite J
  and finite I
  and I ∩ J = {}
  and ∀ i ∈ I ∪ J. A i ∈ fc-mats
shows sum-mat A (I ∪ J) = sum-mat A I + sum-mat A J ⟨proof⟩

lemma sum-with-reindex-cong':
  fixes g :: 'c ⇒ 'a Matrix.mat
  assumes ∀ x. g x ∈ fc-mats
  and ∀ x. h x ∈ fc-mats
  and inj-on l B
  and ∀ x. x ∈ B ⇒ g (l x) = h x
shows sum-with (+) (0m dimR dimC) g (l ` B) =
sum-with (+) (0m dimR dimC) h B
⟨proof⟩

lemma sum-mat-cong':
shows finite I ⇒ (∀ i. i ∈ I ⇒ A i = B i) ⇒
  (∀ i. i ∈ I ⇒ A i ∈ fc-mats) ⇒
  (∀ i. i ∈ I ⇒ B i ∈ fc-mats) ⇒ I = J ⇒ sum-mat A I = sum-mat B J
⟨proof⟩

lemma sum-mat-reindex-cong:
  assumes finite B
  and ∀ x. x ∈ l` B ⇒ g x ∈ fc-mats
  and ∀ x. x ∈ B ⇒ h x ∈ fc-mats
  and inj-on l B
  and ∀ x. x ∈ B ⇒ g (l x) = h x
shows sum-mat g (l ` B) = sum-mat h B
⟨proof⟩

lemma sum-mat-mod-eq:
  fixes A :: nat ⇒ 'a Matrix.mat
  assumes ∀ x. x ∈ {..m} ⇒ A x ∈ fc-mats
shows sum-mat (λi. A (i mod m)) ((λi. n * m+i) ` {..< m}) = sum-mat A {..< m}
⟨proof⟩

lemma sum-mat-singleton':
  assumes A i ∈ fc-mats
shows sum-mat A {i} = A i
⟨proof⟩

end

context cpx-sq-mat
begin

```

```

lemma sum-mat-mod-div-ne-0:
  assumes  $\bigwedge k. k < (nC::nat) \implies A k \in \text{carrier-mat } n n$ 
  and  $\bigwedge j. j < (nD::nat) \implies B j \in \text{carrier-mat } m m$ 
  and  $0 < n$ 
  and  $0 < m$ 
  and  $\text{dimR} = n * m$ 
  and  $nD \neq 0$ 
shows sum-mat  $(\lambda i. \text{sum-mat} (\lambda j. f i * g j \cdot_m ((A i) \otimes (B j))) \{.. < nD\})$ 
 $\{.. < nC\} =$ 
 $\text{sum-mat} (\lambda i. (f (i \text{ div } nD) * g (i \text{ mod } nD)) \cdot_m$ 
 $((A (i \text{ div } nD)) \otimes (B (i \text{ mod } nD)))) \{.. < nC * nD\}$ 
(proof)

lemma sum-mat-mod-div-eq-0:
  assumes  $\bigwedge k. k < (nC::nat) \implies A k \in \text{carrier-mat } n n$ 
  and  $0 < n$ 
  and  $nD = 0$ 
  and  $\text{dimR} = n * m$ 
shows sum-mat  $(\lambda i. \text{sum-mat} (\lambda j. f i * g j \cdot_m ((A i) \otimes (B j))) \{.. < nD\})$ 
 $\{.. < nC\} =$ 
 $\text{sum-mat} (\lambda i. (f (i \text{ div } nD) * g (i \text{ mod } nD)) \cdot_m$ 
 $((A (i \text{ div } nD)) \otimes (B (i \text{ mod } nD)))) \{.. < nC * nD\}$ 
(proof)

lemma sum-mat-mod-div:
  assumes  $\bigwedge k. k < (nC::nat) \implies A k \in \text{carrier-mat } n n$ 
  and  $\bigwedge j. j < (nD::nat) \implies B j \in \text{carrier-mat } m m$ 
  and  $0 < n$ 
  and  $0 < m$ 
  and  $\text{dimR} = n * m$ 
shows sum-mat  $(\lambda i. \text{sum-mat} (\lambda j. f i * g j \cdot_m ((A i) \otimes (B j))) \{.. < nD\})$ 
 $\{.. < nC\} =$ 
 $\text{sum-mat} (\lambda i. (f (i \text{ div } nD) * g (i \text{ mod } nD)) \cdot_m$ 
 $((A (i \text{ div } nD)) \otimes (B (i \text{ mod } nD)))) \{.. < nC * nD\}$ 
(proof)

lemma sum-sum-mat-expand-ne-0:
  assumes  $\bigwedge k. k < (nC::nat) \implies A k \in \text{carrier-mat } n n$ 
  and  $\bigwedge j. j < (nD::nat) \implies B j \in \text{carrier-mat } m m$ 
  and  $R \in \text{carrier-mat } (n * m) (n * m)$ 
  and  $0 < n$ 
  and  $0 < m$ 
  and  $nD \neq 0$ 
  and  $\text{dimR} = n * m$ 
shows sum-mat  $(\lambda i. \text{sum-mat} (\lambda j. f i * g j \cdot_m ((A i) \otimes (B j))) * R) \{.. < nD\})$ 
 $\{.. < nC\} =$ 
 $\text{sum-mat} (\lambda i. (f (i \text{ div } nD) * g (i \text{ mod } nD)) \cdot_m$ 
 $((A (i \text{ div } nD)) \otimes (B (i \text{ mod } nD))) * R) \{.. < nC * nD\}$ 

```

$\langle proof \rangle$

```

lemma sum-sum-mat-expand-eq-0:
  assumes  $\bigwedge k. k < (nC::nat) \implies A k \in carrier\text{-}mat n n$ 
  and  $R \in carrier\text{-}mat (n*m) (n*m)$ 
  and  $0 < n$ 
  and  $0 < m$ 
  and  $nD = 0$ 
  and  $dimR = n * m$ 
  shows  $sum\text{-}mat (\lambda i. sum\text{-}mat (\lambda j. f i * g j \cdot_m ((A i) \otimes (B j)) * R) \{.. < nD\})$ 
   $\{.. < nC\} =$ 
   $sum\text{-}mat (\lambda i. (f (i \text{ div } nD) * g (i \text{ mod } nD)) \cdot_m$ 
   $((A (i \text{ div } nD)) \otimes (B (i \text{ mod } nD))) * R) \{.. < nC * nD\}$ 
   $\langle proof \rangle$ 

lemma sum-sum-mat-expand:
  assumes  $\bigwedge k. k < (nC::nat) \implies A k \in carrier\text{-}mat n n$ 
  and  $\bigwedge j. j < (nD::nat) \implies B j \in carrier\text{-}mat m m$ 
  and  $R \in carrier\text{-}mat (n*m) (n*m)$ 
  and  $0 < n$ 
  and  $0 < m$ 
  and  $dimR = n * m$ 
  shows  $sum\text{-}mat (\lambda i. sum\text{-}mat (\lambda j. f i * g j \cdot_m ((A i) \otimes (B j)) * R) \{.. < nD\})$ 
   $\{.. < nC\} =$ 
   $sum\text{-}mat (\lambda i. (f (i \text{ div } nD) * g (i \text{ mod } nD)) \cdot_m$ 
   $((A (i \text{ div } nD)) \otimes (B (i \text{ mod } nD))) * R) \{.. < nC * nD\}$ 
   $\langle proof \rangle$ 

end

```

3 Results on tensor products

```

lemma tensor-mat-trace:
  assumes  $A \in carrier\text{-}mat n n$ 
  and  $B \in carrier\text{-}mat m m$ 
  and  $0 < n$ 
  and  $0 < m$ 
  shows  $Complex\text{-}Matrix.trace (A \otimes B) = Complex\text{-}Matrix.trace A *$ 
   $Complex\text{-}Matrix.trace B$ 
   $\langle proof \rangle$ 

lemma tensor-vec-inner-prod:
  assumes  $u \in carrier\text{-}vec n$ 
  and  $v \in carrier\text{-}vec n$ 
  and  $a \in carrier\text{-}vec n$ 
  and  $b \in carrier\text{-}vec n$ 
  and  $0 < n$ 
  shows  $Complex\text{-}Matrix.inner\text{-}prod (tensor\text{-}vec u v) (tensor\text{-}vec a b) =$ 
   $Complex\text{-}Matrix.inner\text{-}prod u a * Complex\text{-}Matrix.inner\text{-}prod v b$ 

```

$\langle proof \rangle$

lemma *tensor-mat-positive*:
 assumes $A \in carrier\text{-mat } n \ n$
 and $B \in carrier\text{-mat } m \ m$
 and $0 < n$
 and $0 < m$
 and *Complex-Matrix.positive A*
 and *Complex-Matrix.positive B*
shows *Complex-Matrix.positive* ($A \otimes B$)
 $\langle proof \rangle$

lemma *tensor-mat-square-idty*:
 assumes $A * A = 1_m \ n$
 and $B * B = 1_m \ m$
 and $0 < n$
 and $0 < m$
shows $(A \otimes B) * (A \otimes B) = 1_m \ (n*m)$
 $\langle proof \rangle$

lemma *tensor-mat-commute*:
 assumes $A \in carrier\text{-mat } n \ n$
 and $B \in carrier\text{-mat } m \ m$
 and $C \in carrier\text{-mat } n \ n$
 and $D \in carrier\text{-mat } m \ m$
 and $0 < n$
 and $0 < m$
 and $A * C = C * A$
 and $B * D = D * B$
shows $(A \otimes B) * (C \otimes D) = (C \otimes D) * (A \otimes B)$
 $\langle proof \rangle$

lemma *tensor-mat-mult-id*:
 assumes $A \in carrier\text{-mat } n \ n$
 and $B \in carrier\text{-mat } m \ m$
 and $0 < n$
 and $0 < m$
shows $(A \otimes 1_m \ m) * (1_m \ n \otimes B) = A \otimes B$
 $\langle proof \rangle$

lemma *tensor-mat-trace-mult-distr*:
 assumes $A \in carrier\text{-mat } n \ n$
 and $B \in carrier\text{-mat } m \ m$
 and $C \in carrier\text{-mat } n \ n$
 and $D \in carrier\text{-mat } m \ m$
 and $0 < n$
 and $0 < m$
shows *Complex-Matrix.trace* ($((A \otimes B) * (C \otimes D)) =$

Complex-Matrix.trace ($A * C$) * (*Complex-Matrix.trace* ($B * D$))
(proof)

lemma *tensor-mat-diagonal*:
assumes $A \in \text{carrier-mat } n \ n$
and $B \in \text{carrier-mat } m \ m$
and $\text{diagonal-mat } A$
and $\text{diagonal-mat } B$
shows $\text{diagonal-mat} (A \otimes B)$ *(proof)*

lemma *tensor-mat-add-right*:
assumes $A \in \text{carrier-mat } n \ m$
and $B \in \text{carrier-mat } i \ j$
and $C \in \text{carrier-mat } i \ j$
and $0 < m$
and $0 < j$
shows $A \otimes (B + C) = (A \otimes B) + (A \otimes C)$
(proof)

lemma *tensor-mat-zero*:
assumes $B \in \text{carrier-mat } i \ j$
and $0 < j$
and $0 < m$
shows $0_m \ n \ m \otimes B = 0_m (n * i) (m * j)$
(proof)

lemma *tensor-mat-zero'*:
assumes $B \in \text{carrier-mat } i \ j$
and $0 < j$
and $0 < m$
shows $B \otimes 0_m \ n \ m = 0_m (i * n) (j * m)$
(proof)

lemma *tensor-mat-sum-right*:
fixes $A :: \text{complex Matrix.mat}$
assumes *finite I*
and $A \in \text{carrier-mat } n \ m$
and $\bigwedge k. k \in I \implies ((B k) :: \text{complex Matrix.mat}) \in \text{carrier-mat } i \ j$
and $0 < m$
and $0 < j$
and $\text{dimR} = n * i$
and $\text{dimC} = m * j$
shows $A \otimes (\text{fixed-carrier-mat.sum-mat } i \ j \ B \ I) =$
 $\text{fixed-carrier-mat.sum-mat} (n * i) (m * j) (\lambda i. A \otimes (B i)) \ I$
(proof)

lemma *tensor-mat-add-left*:
assumes $A \in \text{carrier-mat } n \ m$

```

and  $B \in carrier\text{-}mat n m$ 
and  $C \in carrier\text{-}mat i j$ 
and  $0 < m$ 
and  $0 < j$ 
shows  $(A + B) \otimes C = (A \otimes C) + (B \otimes C)$ 
⟨proof⟩

lemma tensor-mat-smult-left:
assumes  $A \in carrier\text{-}mat n m$ 
and  $B \in carrier\text{-}mat i j$ 
and  $0 < m$ 
and  $0 < j$ 
shows  $x \cdot_m A \otimes B = x \cdot_m (A \otimes B)$ 
⟨proof⟩

lemma tensor-mat-smult-right:
assumes  $A \in carrier\text{-}mat n m$ 
and  $B \in carrier\text{-}mat i j$ 
and  $0 < m$ 
and  $0 < j$ 
shows  $A \otimes (x \cdot_m B) = x \cdot_m (A \otimes B)$ 
⟨proof⟩

lemma tensor-mat-smult:
assumes  $A \in carrier\text{-}mat n m$ 
and  $B \in carrier\text{-}mat i j$ 
and  $0 < m$ 
and  $0 < j$ 
shows  $x \cdot_m A \otimes (y \cdot_m B) = x * y \cdot_m (A \otimes B)$ 
⟨proof⟩

lemma tensor-mat-singleton-right:
assumes  $0 < dim\text{-}col A$ 
and  $B \in carrier\text{-}mat 1 1$ 
shows  $A \otimes B = B \$$(0,0) \cdot_m A$ 
⟨proof⟩

lemma tensor-mat-singleton-left:
assumes  $0 < dim\text{-}col A$ 
and  $B \in carrier\text{-}mat 1 1$ 
shows  $B \otimes A = B \$$(0,0) \cdot_m A$ 
⟨proof⟩

lemma tensor-mat-sum-left:
assumes finite  $I$ 
and  $B \in carrier\text{-}mat i j$ 
and  $\bigwedge k. k \in I \implies A k \in carrier\text{-}mat n m$ 
and  $0 < m$ 
and  $0 < j$ 

```

```

and dimR = n *i
and dimC = m*j
shows (fixed-carrier-mat.sum-mat n m A I)  $\otimes$  B =
fixed-carrier-mat.sum-mat (n*i) (m*j) ( $\lambda i.$  (A i)  $\otimes$  B) I
⟨proof⟩

lemma tensor-mat-diag-elem:
assumes A ∈ carrier-mat n n
and B ∈ carrier-mat m m
and i < n * m
and 0 < n*m
shows (A  $\otimes$  B) $$ (i, i) = A $$ (i div m, i div m) *
B $$ (i mod m, i mod m)
⟨proof⟩

context cpx-sq-mat
begin

lemma tensor-mat-sum-mat-right:
assumes finite I
and A ∈ carrier-mat n n
and  $\bigwedge k.$  k ∈ I  $\implies$  B k ∈ carrier-mat i i
and 0 < n
and 0 < i
and dimR = n *i
shows A  $\otimes$  (fixed-carrier-mat.sum-mat i i B I) = sum-mat ( $\lambda i.$  A  $\otimes$  (B i)) I
⟨proof⟩

lemma tensor-mat-sum-mat-left:
assumes finite I
and B ∈ carrier-mat i i
and  $\bigwedge k.$  k ∈ I  $\implies$  A k ∈ carrier-mat n n
and 0 < n
and 0 < i
and dimR = n *i
shows (fixed-carrier-mat.sum-mat n n A I)  $\otimes$  B = sum-mat ( $\lambda i.$  (A i)  $\otimes$  B) I
⟨proof⟩

lemma tensor-mat-sum-nat-mod-div-ne-0:
assumes  $\bigwedge k.$  k < (nC::nat)  $\implies$  A k ∈ carrier-mat n n
and  $\bigwedge j.$  j < (nD::nat)  $\implies$  B j ∈ carrier-mat m m
and fixed-carrier-mat.sum-mat n n ( $\lambda i.$  f i  $\cdot_m$  (A i)) {.. < nC} = C
and fixed-carrier-mat.sum-mat m m ( $\lambda j.$  g j  $\cdot_m$  (B j)) {.. < nD} = D
and 0 < n
and 0 < m
and nD ≠ 0
and dimR = n *m
shows sum-mat ( $\lambda i.$  (f (i div nD) * g (i mod nD))  $\cdot_m$ 
((A (i div nD))  $\otimes$  (B (i mod nD))))
```

```

{..< nC*nD} = C⊗ D ⟨proof⟩

lemma tensor-mat-sum-nat-mod-div-eq-0:
  assumes  $\bigwedge k. k < (nC::nat) \implies A k \in \text{carrier-mat } n n$ 
  and  $\text{fixed-carrier-mat.sum-mat } n n (\lambda i. f i \cdot_m (A i)) \{.. < nC\} = C$ 
  and  $\text{fixed-carrier-mat.sum-mat } m m (\lambda j. g j \cdot_m (B j)) \{.. < nD\} = D$ 
  and  $0 < n$ 
  and  $0 < m$ 
  and  $nD = 0$ 
  and  $\text{dimR} = n *m$ 
  shows  $\text{sum-mat } (\lambda i. (f (i \text{ div } nD) * g (i \text{ mod } nD)) \cdot_m ((A (i \text{ div } nD)) \otimes (B (i \text{ mod } nD))))$ 
  {..< nC*nD} = C⊗ D
  ⟨proof⟩

lemma tensor-mat-sum-nat-mod-div:
  assumes  $\bigwedge k. k < (nC::nat) \implies A k \in \text{carrier-mat } n n$ 
  and  $\bigwedge j. j < (nD::nat) \implies B j \in \text{carrier-mat } m m$ 
  and  $\text{fixed-carrier-mat.sum-mat } n n (\lambda i. f i \cdot_m (A i)) \{.. < nC\} = C$ 
  and  $\text{fixed-carrier-mat.sum-mat } m m (\lambda j. g j \cdot_m (B j)) \{.. < nD\} = D$ 
  and  $0 < n$ 
  and  $0 < m$ 
  and  $\text{dimR} = n *m$ 
  shows  $\text{sum-mat } (\lambda i. (f (i \text{ div } nD) * g (i \text{ mod } nD)) \cdot_m ((A (i \text{ div } nD)) \otimes (B (i \text{ mod } nD))))$ 
  {..< nC*nD} = C⊗ D
  ⟨proof⟩

end

lemma tensor-mat-sum-mult-trace-expand-ne-0:
  assumes  $\bigwedge k. k < (nC::nat) \implies A k \in \text{carrier-mat } n n$ 
  and  $\bigwedge j. j < (nD::nat) \implies B j \in \text{carrier-mat } m m$ 
  and  $R \in \text{carrier-mat } (n*m) (n*m)$ 
  and  $\text{fixed-carrier-mat.sum-mat } n n (\lambda i. f i \cdot_m (A i)) \{.. < nC\} = C$ 
  and  $\text{fixed-carrier-mat.sum-mat } m m (\lambda j. g j \cdot_m (B j)) \{.. < nD\} = D$ 
  and  $0 < n$ 
  and  $0 < m$ 
  and  $nD \neq 0$ 
  shows  $\text{sum } (\lambda i. \text{Complex-Matrix.trace } ((f (i \text{ div } nD) * g (i \text{ mod } nD)) \cdot_m ((A (i \text{ div } nD)) \otimes (B (i \text{ mod } nD))) * R)) \{.. < nC * nD\} =$ 
   $\text{Complex-Matrix.trace } ((C \otimes D) * R)$ 
  ⟨proof⟩

lemma tensor-mat-sum-mult-trace-expand-eq-0:
  assumes  $\bigwedge k. k < (nC::nat) \implies A k \in \text{carrier-mat } n n$ 
  and  $R \in \text{carrier-mat } (n*m) (n*m)$ 
  and  $\text{fixed-carrier-mat.sum-mat } n n (\lambda i. f i \cdot_m (A i)) \{.. < nC\} = C$ 
  and  $\text{fixed-carrier-mat.sum-mat } m m (\lambda j. g j \cdot_m (B j)) \{.. < nD\} = D$ 
```

```

and 0 < n
and 0 < m
and nD = 0
shows sum ( $\lambda i.$  Complex-Matrix.trace (( $f (i \text{ div } nD) * g (i \text{ mod } nD)$ ) $\cdot_m$ 
((A (i div nD))  $\otimes$  (B (i mod nD))) * R)) {.. $nC * nD\}$  =
Complex-Matrix.trace ((C  $\otimes$  D) * R)
⟨proof⟩

lemma tensor-mat-sum-mult-trace-expand:
assumes  $\bigwedge k. k < (nC::nat) \implies A k \in \text{carrier-mat } n n$ 
and  $\bigwedge j. j < (nD::nat) \implies B j \in \text{carrier-mat } m m$ 
and R ∈ carrier-mat (n*m) (n*m)
and fixed-carrier-mat.sum-mat n n ( $\lambda i.$  f i  $\cdot_m$  (A i)) {.. $nC\} = C$ 
and fixed-carrier-mat.sum-mat m m ( $\lambda j.$  g j  $\cdot_m$  (B j)) {.. $nD\} = D$ 
and 0 < n
and 0 < m
shows sum ( $\lambda i.$  Complex-Matrix.trace (( $f (i \text{ div } nD) * g (i \text{ mod } nD)$ ) $\cdot_m$ 
((A (i div nD))  $\otimes$  (B (i mod nD))) * R)) {.. $nC * nD\}$  =
Complex-Matrix.trace ((C  $\otimes$  D) * R)
⟨proof⟩

lemma tensor-mat-sum-mult-trace-ne-0:
assumes  $\bigwedge k. k < (nC::nat) \implies A k \in \text{carrier-mat } n n$ 
and  $\bigwedge j. j < (nD::nat) \implies B j \in \text{carrier-mat } m m$ 
and R ∈ carrier-mat (n*m) (n*m)
and fixed-carrier-mat.sum-mat n n ( $\lambda i.$  f i  $\cdot_m$  (A i)) {.. $nC\} = C$ 
and fixed-carrier-mat.sum-mat m m ( $\lambda j.$  g j  $\cdot_m$  (B j)) {.. $nD\} = D$ 
and 0 < n
and 0 < m
and 0 ≠ nD
shows sum ( $\lambda i.$  (sum ( $\lambda j.$  Complex-Matrix.trace ((f i * g j) $\cdot_m$ 
((A i)  $\otimes$  (B j)) * R)) {.. $nD\})) {.. $nC\} =$ 
Complex-Matrix.trace ((C  $\otimes$  D) * R)
⟨proof⟩

lemma tensor-mat-sum-mult-trace-eq-0:
assumes  $\bigwedge k. k < (nC::nat) \implies A k \in \text{carrier-mat } n n$ 
and R ∈ carrier-mat (n*m) (n*m)
and fixed-carrier-mat.sum-mat n n ( $\lambda i.$  f i  $\cdot_m$  (A i)) {.. $nC\} = C$ 
and fixed-carrier-mat.sum-mat m m ( $\lambda j.$  g j  $\cdot_m$  (B j)) {.. $nD\} = D$ 
and 0 < n
and 0 < m
and 0 = (nD::nat)
shows sum ( $\lambda i.$  (sum ( $\lambda j.$  Complex-Matrix.trace ((f i * g j) $\cdot_m$ 
((A i)  $\otimes$  (B j)) * R)) {.. $nD\})) {.. $nC\} =$ 
Complex-Matrix.trace ((C  $\otimes$  D) * R)
⟨proof⟩

lemma tensor-mat-sum-mult-trace:$$ 
```

```

assumes  $\bigwedge k. k < (nC::nat) \implies A[k] \in carrier\text{-}mat[n][n]$ 
and  $\bigwedge j. j < (nD::nat) \implies B[j] \in carrier\text{-}mat[m][m]$ 
and  $R \in carrier\text{-}mat[(n*m)][(n*m)]$ 
and  $fixed\text{-}carrier\text{-}mat.sum\text{-}mat[n][n](\lambda i. f[i] \cdot_m (A[i]))\{.. < nC\} = C$ 
and  $fixed\text{-}carrier\text{-}mat.sum\text{-}mat[m][m](\lambda j. g[j] \cdot_m (B[j]))\{.. < nD\} = D$ 
and  $0 < n$ 
and  $0 < m$ 
shows  $sum(\lambda i. (sum(\lambda j. Complex\text{-}Matrix.trace((f[i]*g[j]) \cdot_m ((A[i]) \otimes (B[j])) * R))\{.. < nD\})\{.. < nC\} =$ 
 $Complex\text{-}Matrix.trace((C \otimes D) * R)$ 
⟨proof⟩

```

```

lemma tensor\text{-}mat\text{-}make\text{-}pm\text{-}mult\text{-}trace:
assumes  $A \in carrier\text{-}mat[n][n]$ 
and  $hermitian A$ 
and  $B \in carrier\text{-}mat[m][m]$ 
and  $hermitian B$ 
and  $R \in carrier\text{-}mat[(n*m)][(n*m)]$ 
and  $(nA, M) = cpx\text{-}sq\text{-}mat.make\text{-}pm[n][n]A$ 
and  $(nB, N) = cpx\text{-}sq\text{-}mat.make\text{-}pm[m][m]B$ 
and  $0 < n$ 
and  $0 < m$ 
shows  $sum(\lambda i. (sum(\lambda j. Complex\text{-}Matrix.trace((complex\text{-}of\text{-}real(meas\text{-}outcome\text{-}val(M[i])) * complex\text{-}of\text{-}real(meas\text{-}outcome\text{-}val(N[j]))) \cdot_m ((meas\text{-}outcome\text{-}prj(M[i])) \otimes (meas\text{-}outcome\text{-}prj(N[j]))) * R))\{.. < nB\}))\{.. < nA\} =$ 
 $Complex\text{-}Matrix.trace((A \otimes B) * R)$ 
⟨proof⟩

```

```

lemma tensor\text{-}mat\text{-}mat\text{-}conj:
assumes  $A \in carrier\text{-}mat[n][n]$ 
and  $B \in carrier\text{-}mat[n][n]$ 
and  $U \in carrier\text{-}mat[n][n]$ 
and  $C \in carrier\text{-}mat[m][m]$ 
and  $D \in carrier\text{-}mat[m][m]$ 
and  $V \in carrier\text{-}mat[m][m]$ 
and  $0 < n$ 
and  $0 < m$ 
and  $A = mat\text{-}conj U B$ 
and  $C = mat\text{-}conj V D$ 
shows  $A \otimes C = mat\text{-}conj(U \otimes V)(B \otimes D)$ 
⟨proof⟩

```

```

lemma unitarily-equiv\text{-}mat\text{-}conj[simp]:
assumes  $unitarily\text{-}equiv A B U$ 
shows  $A = mat\text{-}conj U B$  ⟨proof⟩

```

```

lemma hermitian\text{-}tensor\text{-}mat\text{-}decomp:

```

```

assumes  $A \in carrier\text{-}mat n n$ 
and  $C \in carrier\text{-}mat m m$ 
and  $\text{unitary-diag } A B U$ 
and  $\text{unitary-diag } C D V$ 
and  $0 < n$ 
and  $0 < m$ 
shows  $\text{unitary-diag } (A \otimes C) (B \otimes D) (U \otimes V)$ 
 $\langle proof \rangle$ 

end

```

```

theory Matrix-L2-Operator-Norm
imports
    Tensor-Mat-Compl-Properties
begin

```

We formalize the \mathcal{L}_2 operator norm on matrices on nonempty vector spaces. This norm can be defined on a matrix A by $\|A\|_2 = \sup\{\|A \cdot v\|_2 \mid \|v\|_2 = 1\}$, and it is equal to the maximum singular value of A .

4 Preliminary results

4.1 Commutator and anticommutator

We define the notions of commutator and anticommutator of two matrices. When these matrices commute, their commutator is the zero matrix.

```

definition commutator :: complex Matrix.mat  $\Rightarrow$  complex Matrix.mat  $\Rightarrow$ 
    complex Matrix.mat where
    commutator  $A B = A * B - B * A$ 

```

```

definition anticommutator where
    anticommutator  $A B = A * B + B * A$ 

```

```

lemma commutator-dim:
assumes  $A \in carrier\text{-}mat n n$ 
and  $B \in carrier\text{-}mat n n$ 
shows commutator  $A B \in carrier\text{-}mat n n$   $\langle proof \rangle$ 

```

```

lemma anticommutator-dim:
assumes  $A \in carrier\text{-}mat n n$ 
and  $B \in carrier\text{-}mat n n$ 
shows anticommutator  $A B \in carrier\text{-}mat n n$   $\langle proof \rangle$ 

```

```

lemma commutator-zero-iff:
assumes  $A \in carrier\text{-}mat n n$ 
and  $B \in carrier\text{-}mat n n$ 
shows commutator  $A B = 0_m n n \longleftrightarrow A * B = B * A$ 

```

$\langle proof \rangle$

```

lemma anticommutator-zero-iff:
  fixes A::'a ::ring Matrix.mat
  assumes A ∈ carrier-mat n n
  and B ∈ carrier-mat n n
  shows anticommutator A B = 0m n n  $\longleftrightarrow$  B*A = -(A*B)
  ⟨proof⟩

lemma commutator-mult-expand:
  assumes A ∈ carrier-mat n n
  and B ∈ carrier-mat n n
  and C ∈ carrier-mat n n
  and D ∈ carrier-mat n n
  shows commutator A B * commutator C D =
    A * B * (C * D) - A * B * (D * C) - B * A * (C * D) + B * A * (D * C)
  ⟨proof⟩

```

5 Maximum modulus in a spectrum

We prove some basic results on the maximum modulus of elements in a matrix A , and focus on the case where A is a Hermitian matrix.

5.1 Definition and basic properties for Hermitian matrices

```

definition spmax:: complex Matrix.mat ⇒ real where
  spmax A = Max.F {cmod a | a. a ∈ spectrum A}

```

```

lemma spmax-mem:
  assumes A ∈ carrier-mat n n
  and 0 < n
  shows spmax A ∈ {cmod a | a. a ∈ spectrum A}
  ⟨proof⟩

```

```

lemma spmax-geq-0:
  assumes A ∈ carrier-mat n n
  and 0 < n
  shows 0 ≤ spmax A
  ⟨proof⟩

```

```

lemma Re-inner-mult-diag-le:
  fixes B::complex Matrix.mat
  assumes diagonal-mat B
  and B ∈ carrier-mat n n
  and 0 < n
  and M = Max.F {Re (conjugate a) | a. a ∈ diag-elems B}
  shows ∀ v ∈ carrier-vec n. Re (inner-prod (B *v v) v) ≤
    M * Re ((inner-prod v v))

```

$\langle proof \rangle$

```
lemma Re-inner-mult-diag-le':
  fixes B::complex Matrix.mat
  assumes diagonal-mat B
  and B ∈ carrier-mat n n
  and 0 < n
  and (M::real) = Max.F {cmod a|a. a ∈ diag-elems B}
  and v ∈ carrier-vec n
shows cmod (inner-prod v (B *v v)) ≤ M * inner-prod v v
⟨proof⟩
```

```
lemma hermitian-mult-inner-prod-le:
  fixes A::complex Matrix.mat
  assumes A ∈ carrier-mat n n
  and 0 < n
  and hermitian A
  and v ∈ carrier-vec n
shows cmod (inner-prod v (A *v v)) ≤ (spmax A) * (inner-prod v v)
⟨proof⟩
```

```
lemma hermitian-trace-rank-le:
assumes A ∈ carrier-mat n n
and hermitian A
and v ∈ carrier-vec n
and 0 < n
shows cmod (Complex-Matrix.trace (A * (rank-1-proj v))) ≤
(spmax A) * (inner-prod v v)
⟨proof⟩
```

```
lemma hermitian-pos-decomp-cmod-le:
assumes A ∈ carrier-mat n n
and C ∈ carrier-mat n n
and 0 < n
and hermitian C
and Complex-Matrix.positive A
shows cmod (Complex-Matrix.trace (C * A)) ≤
Re (Complex-Matrix.trace A) * (spmax C)
⟨proof⟩
```

```
lemma hermitian-density-cmod-le:
  fixes R::complex Matrix.mat
  assumes R ∈ carrier-mat n n
  and A ∈ carrier-mat n n
  and 0 < n
  and hermitian A
  and density-operator R
shows cmod (Complex-Matrix.trace (A * R)) ≤ (spmax A)
⟨proof⟩
```

```

lemma tensor-mat-hermitian-positive-le:
  assumes A ∈ carrier-mat n n
  and B ∈ carrier-mat m m
  and C ∈ carrier-mat n n
  and D ∈ carrier-mat m m
  and 0 < n
  and 0 < m
  and hermitian A
  and hermitian B
  and Complex-Matrix.positive C
  and Complex-Matrix.positive D
  shows cmod (Complex-Matrix.trace ((A ⊗ B)*(C ⊗ D))) ≤
    Re (Complex-Matrix.trace C) * Re (Complex-Matrix.trace D) *
    spmax A * spmax B
  ⟨proof⟩

```

```

lemma tensor-mat-hermitian-density-le:
  assumes A ∈ carrier-mat n n
  and B ∈ carrier-mat m m
  and C ∈ carrier-mat n n
  and D ∈ carrier-mat m m
  and 0 < n
  and 0 < m
  and hermitian A
  and hermitian B
  and density-operator C
  and density-operator D
  shows cmod (Complex-Matrix.trace ((A ⊗ B)*(C ⊗ D))) ≤
    spmax A * spmax B
  ⟨proof⟩

```

```

lemma idty-spmmax:
  assumes 0 < n
  shows spmax (1m n) = 1 ⟨proof⟩

```

```

lemma spmax-uminus:
  fixes A::complex Matrix.mat
  assumes hermitian A
  and A ∈ carrier-mat n n
  and 0 < n
  shows spmax (-A) = spmax A
  ⟨proof⟩

```

```

lemma spmax-smult:
  fixes A::complex Matrix.mat

```

```

assumes hermitian A
and A ∈ carrier-mat n n
and 0 < n
shows spmax (x ·m A) = cmod x * spmax A
⟨proof⟩

lemma spmax-smult-pos:
fixes A::complex Matrix.mat
assumes hermitian A
and A ∈ carrier-mat n n
and 0 < n
and 0 ≤ x
shows spmax (x ·m A) = x * spmax A
⟨proof⟩

lemma hermitian-square-spmax:
fixes A::complex Matrix.mat
assumes hermitian A
and A ∈ carrier-mat n n
and 0 < n
shows spmax (A*A) = spmax A * spmax A
⟨proof⟩

lemma hermitian-square-idty-spmax:
assumes 0 < n
and A ∈ carrier-mat n n
and hermitian A
and A*A = 1m n
shows spmax A = 1
⟨proof⟩

lemma hermitian-mult-density-trace:
assumes A ∈ carrier-mat n n
and R ∈ carrier-mat n n
and 0 < n
and hermitian A
and A * A = 1m n
and density-operator R
shows |Complex-Matrix.trace (A*R)| ≤ 1
⟨proof⟩

lemma tensor-mat-hermitian-density-spmax-le:
assumes A ∈ carrier-mat n n
and B ∈ carrier-mat m m
and C ∈ carrier-mat n n
and D ∈ carrier-mat m m
and 0 < n
and 0 < m
and hermitian A

```

```

and hermitian  $B$ 
and  $A * A = 1_m \ n$ 
and  $B * B = 1_m \ m$ 
and density-operator  $C$ 
and density-operator  $D$ 
shows  $cmod (\text{Complex-Matrix.trace} ((A \otimes B) * (C \otimes D))) \leq 1$ 
⟨proof⟩

```

5.2 Eigenvector for the element with maximum modulus

```

definition spmax-wit where
spmax-wit  $A = (\text{SOME } k. \ \text{eigenvalue } A \ k \wedge \text{spmax } A = cmod \ k)$ 

```

```

lemma spmax-wit-eigenvalue:
assumes  $A \in \text{carrier-mat} \ n \ n$ 
and  $0 < n$ 
shows eigenvalue  $A$  (spmax-wit  $A$ )  $\wedge$  spmax  $A = cmod (\text{spmax-wit } A)$ 
⟨proof⟩

```

```

lemma find-eigen-spmax-neq-0:
assumes  $A \in \text{carrier-mat} \ n \ n$ 
and  $0 < n$ 
shows find-eigenvector  $A$  (spmax-wit  $A$ )  $\neq 0_v \ n$  ⟨proof⟩

```

```

lemma find-eigen-spmax-dim:
assumes  $A \in \text{carrier-mat} \ n \ n$ 
and  $0 < n$ 
shows dim-vec (vec-normalize (find-eigenvector  $A$  (spmax-wit  $A$ )))  $= n$ 
⟨proof⟩

```

```

lemma nrm-spmax-eigenvector-eq:
assumes  $v = \text{vec-normalize} (\text{find-eigenvector } A \ (\text{spmax-wit } A))$ 
and  $A \in \text{carrier-mat} \ n \ n$ 
and  $0 < n$ 
shows  $cmod (\text{inner-prod } v \ (A *_v v)) = \text{spmax } A$ 
⟨proof⟩

```

6 The \mathcal{L}_2 operator norm

6.1 Definition and preliminary results

```

definition rvec-norm where
rvec-norm  $v = \text{Re} (\text{vec-norm } v)$ 

```

```

definition L2-op-nrm where
L2-op-nrm  $A =$ 
 $\text{Sup} \ \{rvec-norm (A *_v v) \mid v. \ \text{dim-vec } v = \text{dim-col } A \wedge \text{rvec-norm } v = 1\}$ 

```

```

lemma mat-mult-inner-prod-le:
  fixes A::complex Matrix.mat
  assumes 0 < dim-col A
  and v ∈ carrier-vec (dim-col A)
  shows cmod (inner-prod (A *v v) (A *v v)) ≤
    spmax ((Complex-Matrix.adjoint A) * A) * (inner-prod v v)
  ⟨proof⟩

lemma normalized-rvec-norm:
  assumes v ≠ 0v (dim-vec v)
  shows rvec-norm (vec-normalize v) = 1
  ⟨proof⟩

lemma vec-norm-smult:
  shows vec-norm (c ·v v) = (cmod c) * (vec-norm v)
  ⟨proof⟩

lemma rvec-norm-smult:
  shows rvec-norm (c ·v v) = (cmod c) * (rvec-norm v)
  ⟨proof⟩

lemma mult-mat-zero-vec:
  assumes A ∈ carrier-mat n m
  and v = 0v m
  shows A *v v = 0v n
  ⟨proof⟩

lemma mat-mult-vec-normalize:
  assumes dim-col A = dim-vec v
  shows A *v v = vec-norm v ·v (A *v (vec-normalize v))
  ⟨proof⟩

lemma vec-norm-real:
  shows vec-norm v ∈ Reals
  ⟨proof⟩

lemma rvec-norm-geq-0:
  shows 0 ≤ rvec-norm v ⟨proof⟩

lemma rvec-norm-triangle:
  assumes dim-vec u = dim-vec v
  shows rvec-norm (u + v) ≤ rvec-norm u + rvec-norm v
  ⟨proof⟩

lemma cmod-vec-norm:
  shows cmod (vec-norm v) = vec-norm v
  ⟨proof⟩

lemma cmod-rvec-norm:

```

```

shows cmod (rvec-norm v) = rvec-norm v
⟨proof⟩

lemma inner-prod-rvec-norm-pow2:
shows (rvec-norm v)2 = v ·c v
⟨proof⟩

lemma rvec-norm-mat-mult-le:
assumes v ∈ carrier-vec (dim-col A)
and 0 < dim-col A
shows cmod (inner-prod (A *v v) (A *v v))
    ≤ spmax (Complex-Matrix.adjoint A * A) * (rvec-norm v)2
⟨proof⟩

lemma square-leq:
assumes a2 ≤ b * c2
and 0 ≤ c
shows a ≤ (sqrt b) * c
⟨proof⟩

lemma rvec-set-ne:
assumes 0 < dim-col A
shows {rvec-norm (A *v v)|v. dim-vec v = dim-col A ∧ rvec-norm v = 1} ≠ {}
⟨proof⟩

lemma unitary-col-vec-norm:
assumes U ∈ carrier-mat n n
and unitary U
and i < n
shows vec-norm (Matrix.col U i) = 1 ⟨proof⟩

lemma unitary-col-rvec-norm:
assumes U ∈ carrier-mat n n
and unitary U
and i < n
shows rvec-norm (Matrix.col U i) = 1 ⟨proof⟩

lemma Cauchy-Schwarz-complex-rvec-norm:
assumes dim-vec x = dim-vec y
shows cmod (inner-prod x y) ≤ rvec-norm x * rvec-norm y
⟨proof⟩

```

6.2 The \mathcal{L}_2 operator norm is equal to the maximum singular value

definition max-sgval **where**
 $\text{max-sgval } A = \sqrt(\text{spmax} (\text{Complex-Matrix.adjoint } A * A))$

lemma max-sgval-geq-0:

```

assumes  $A \in \text{carrier-mat } n \ n$ 
and  $0 < n$ 
shows  $0 \leq \max\text{-sgval } A$ 
⟨proof⟩

lemma max-sgval-uminus:
shows  $\max\text{-sgval } (-A) = \max\text{-sgval } A$ 
⟨proof⟩

lemma rvec-leq-sg-spmax:
assumes  $0 < \dim\text{-col } A$ 
and  $v \in \text{carrier-vec } (\dim\text{-col } A)$ 
shows  $\text{rvec-norm } (A *_v v) \leq (\max\text{-sgval } A) * \text{rvec-norm } v$ 
⟨proof⟩

lemma max-sgval-smult:
assumes  $A \in \text{carrier-mat } n \ n$ 
and  $0 < n$ 
shows  $\max\text{-sgval } (a \cdot_m A) = \text{cmod } a * \max\text{-sgval } A$ 
⟨proof⟩

lemma L2-op-nrm-le-max-sgval:
assumes  $0 < \dim\text{-col } A$ 
shows  $L2\text{-op-nrm } A \leq \max\text{-sgval } A$  ⟨proof⟩

lemma max-sgval-eigen:
assumes  $A \in \text{carrier-mat } n \ n$ 
and  $0 < n$ 
and  $C = \text{Complex-Matrix.adjoint } A * A$ 
and  $v = \text{vec-normalize } (\text{find-eigenvector } C \ (\text{spmax-wit } C))$ 
shows  $\text{rvec-norm } (A *_v v) = \max\text{-sgval } A$ 
⟨proof⟩

lemma rvec-normalize-leq-L2-op-nrm:
assumes  $\text{rvec-norm } v = 1$ 
and  $\dim\text{-col } A = \dim\text{-vec } v$ 
and  $0 < \dim\text{-col } A$ 
shows  $\text{rvec-norm } (A *_v v) \leq L2\text{-op-nrm } A$ 
⟨proof⟩

lemma max-sgval-le-L2-op-nrm:
assumes  $A \in \text{carrier-mat } n \ n$ 
and  $0 < n$ 
shows  $\max\text{-sgval } A \leq L2\text{-op-nrm } A$ 
⟨proof⟩

```

```

lemma vec-norm-leq-L2-op-nrm:
  assumes A ∈ carrier-mat n n
  and v ∈ carrier-vec n
  and 0 < n
  and vec-norm v = 1
shows vec-norm (A *v v) ≤ L2-op-nrm A
⟨proof⟩

lemma rvec-norm-leq-L2-op-nrm:
  assumes A ∈ carrier-mat n n
  and v ∈ carrier-vec n
  and 0 < n
  and rvec-norm v = 1
shows rvec-norm (A *v v) ≤ L2-op-nrm A ⟨proof⟩

lemma cmod-trace-rank-le-L2-op-nrm:
  assumes A ∈ carrier-mat n n
  and v ∈ carrier-vec n
  and 0 < n
  and rvec-norm v = 1
shows cmod (Complex-Matrix.trace (A * rank-1-proj v)) ≤ L2-op-nrm A
⟨proof⟩

lemma expect-val-L2-op-nrm:
  fixes A::complex Matrix.mat
  assumes A ∈ carrier-mat n n
  and R ∈ carrier-mat n n
  and 0 < n
  and density-operator R
shows cmod (Complex-Matrix.trace (A * R)) ≤ L2-op-nrm A
⟨proof⟩

```

6.3 Consequences for the \mathcal{L}_2 operator norm

```

lemma L2-op-nrm-geq-0:
  assumes A ∈ carrier-mat n n
  and 0 < n
shows 0 ≤ L2-op-nrm A
⟨proof⟩

lemma L2-op-nrm-max-sgval-eq:
  assumes A ∈ carrier-mat n n
  and 0 < n
shows L2-op-nrm A = max-sgval A
⟨proof⟩

```

```

lemma rvec-leq-L2-op-nrm:
  assumes A ∈ carrier-mat n n
  and 0 < n
  and v ∈ carrier-vec n
  shows rvec-norm (A *v v) ≤ (L2-op-nrm A) * rvec-norm v
  ⟨proof⟩

lemma L2-op-nrm-mult-le:
  assumes A ∈ carrier-mat n n
  and B ∈ carrier-mat n n
  and 0 < n
  shows L2-op-nrm (A*B) ≤ L2-op-nrm A * L2-op-nrm B
  ⟨proof⟩

lemma L2-op-nrm-smult:
  assumes A ∈ carrier-mat n n
  and 0 < n
  shows L2-op-nrm (c ·m A) = cmod c * L2-op-nrm A
  ⟨proof⟩

lemma L2-op-nrm-uminus:
  assumes A ∈ carrier-mat n n
  and 0 < n
  shows L2-op-nrm (-A) = L2-op-nrm A
  ⟨proof⟩

lemma L2-op-nrm-triangle:
  assumes A ∈ carrier-mat n n
  and B ∈ carrier-mat n n
  and 0 < n
  shows L2-op-nrm (A+B) ≤ L2-op-nrm A + L2-op-nrm B
  ⟨proof⟩

lemma L2-op-nrm-triangle':
  assumes A ∈ carrier-mat n n
  and B ∈ carrier-mat n n
  and 0 < n
  shows L2-op-nrm (A-B) ≤ L2-op-nrm A + L2-op-nrm B
  ⟨proof⟩

lemma hermitian-max-sgval-eq:
  fixes A::complex Matrix.mat
  assumes hermitian A
  and 0 < dim-row A
  shows max-sgval A = spmax A
  ⟨proof⟩

lemma hermitian-L2-op-nrm-spmax-eq:
  fixes A::complex Matrix.mat

```

```

assumes hermitian A
and 0 < dim-row A
shows L2-op-nrm A = spmax A
⟨proof⟩

lemma hermitian-L2-op-nrm-sqrt:
fixes A::complex Matrix.mat
assumes hermitian A
and 0 < dim-row A
shows L2-op-nrm A = sqrt (L2-op-nrm (A*A))
⟨proof⟩

lemma idty-L2-op-nrm:
assumes 0 < n
shows L2-op-nrm (1m n) = 1
⟨proof⟩

lemma commutator-L2-op-nrm-le:
assumes A ∈ carrier-mat n n
and B ∈ carrier-mat n n
and 0 < n
shows L2-op-nrm (commutator A B) ≤ 2 * L2-op-nrm A * L2-op-nrm B
⟨proof⟩

lemma herm-sq-id-L2-op-nrm:
assumes 0 < n
and A ∈ carrier-mat n n
and hermitian A
and A*A = 1m n
shows L2-op-nrm A = 1
⟨proof⟩

lemma comm-L2-op-nrm-le:
assumes A ∈ carrier-mat n n
and B ∈ carrier-mat n n
and 0 < n
and A*A = 1m n
and B*B = 1m n
and hermitian A
and hermitian B
shows L2-op-nrm (commutator A B) ≤ 2
⟨proof⟩

lemma idty-smult-nat-L2-op-nrm:
assumes 0 < n
shows L2-op-nrm ((m::nat) ·m (1m n)) = m
⟨proof⟩
end

```

```

theory Density-Matrix-Basics
imports
  Matrix-L2-Operator-Norm
begin

```

7 On density matrices

7.1 Density matrix characterization

Density matrices are defined as positive operators with trace 1, we prove in this section that they are exactly the convex combinations of pure states.

```

lemma (in cpx-sq-mat) mixed-state-density-operator:
  assumes ⋀ i. i ∈ {.. $n$ ::nat} ⇒ 0 ≤ p i
  and sum p {.. $n$ } = 1
  and ⋀ i. i ∈ {.. $n$ } ⇒ dim-vec (v i) = dimR
  and ⋀ i. i ∈ {.. $n$ } ⇒ ‖v i‖ = 1
  shows density-operator (sum-mat (λ i. (p i) ·m (rank-1-proj (v i))) {.. $n$ })
    ⟨proof⟩

```

```

lemma (in cpx-sq-mat) density-operator-mixed-state:
  assumes R ∈ fc-mats
  and density-operator R
  shows ∃ p v (n::nat). (∀ i ∈ {.. $n$ }. 0 ≤ p i) ∧
    (∀ i ∈ {.. $n$ }. dim-vec (v i) = dimR) ∧
    (∀ i ∈ {.. $n$ }. ‖v i‖ = 1) ∧ (sum p {.. $n$ } = 1) ∧
    (R = sum-mat (λ i. (p i) ·m (rank-1-proj (v i))) {.. $n$ })
    ⟨proof⟩

```

```

lemma (in cpx-sq-mat) density-operator-iff-mixed-state:
  assumes R ∈ fc-mats
  shows density-operator R ↔
    (∃ p v (n::nat). (∀ i ∈ {.. $n$ }. 0 ≤ p i) ∧
      (∀ i ∈ {.. $n$ }. dim-vec (v i) = dimR) ∧
      (∀ i ∈ {.. $n$ }. ‖v i‖ = 1) ∧ (sum p {.. $n$ } = 1) ∧
      (R = sum-mat (λ i. (p i) ·m (rank-1-proj (v i))) {.. $n$ })) (is ?L ↔ ?R)
    ⟨proof⟩

```

7.2 Separable density matrices

We define the notion of a separable density matrix: this is a matrix of the form $\sum_{i=1}^n p_i \rho_A^i \otimes \rho_B^i$, where the p_i s are positive and sum up to 1.

```

definition separately-decomposes where
separately-decomposes R (n::nat) nA nB K F S ≡
  (∀ a < n. (0::complex) ≤ (complex-of-real (K a)) ∧
    F a ∈ carrier-mat nA nA ∧ S a ∈ carrier-mat nB nB ∧
    density-operator (F a) ∧ density-operator (S a)) ∧ 0 < nA * nB ∧

```

sum $K \{.. < n\} = 1 \wedge R = \text{fixed-carrier-mat.sum-mat } (nA * nB) (nA * nB)$
 $(\lambda a. K a \cdot_m ((F a) \otimes (S a))) \{.. < n\}$

definition *separable-density* **where**
separable-density $nA \ nB \ R \equiv$
 $\exists (n::nat) \ K \ F \ S. \text{separately-decomposes } R \ n \ nA \ nB \ K \ F \ S$

lemma *separately-decomposes-carrier*:
assumes *separately-decomposes* $R (n::nat) \ nA \ nB \ K \ F \ S$
and $0 < nA$
and $0 < nB$
shows $R \in \text{carrier-mat } (nA * nB) (nA * nB)$
 $\langle \text{proof} \rangle$

lemma *separately-decomposes-carrier-pos*:
assumes *separately-decomposes* $R \ n \ nA \ nB \ K \ F \ S$
shows $0 < nA \ 0 < nB$
 $\langle \text{proof} \rangle$

lemma *separable-density-carrier*:
assumes *separable-density* $nA \ nB \ R$
and $0 < nA$
and $0 < nB$
shows $R \in \text{carrier-mat } (nA * nB) (nA * nB)$
 $\langle \text{proof} \rangle$

lemma *separately-decomposes-trace*:
assumes *separately-decomposes* $R \ n \ nA \ nB \ K \ F \ S$
shows *Complex-Matrix.trace* $R = 1$
 $\langle \text{proof} \rangle$

lemma *separately-decomposes-positive*:
assumes *separately-decomposes* $R \ n \ nA \ nB \ K \ F \ S$
and $0 < nA$
and $0 < nB$
shows *Complex-Matrix.positive* R
 $\langle \text{proof} \rangle$

A separable density matrix is indeed a density matrix:

lemma *separable-density-operator*:
assumes *separable-density* $nA \ nB \ R$
and $0 < nA$
and $0 < nB$
shows *density-operator* $R \ \langle \text{proof} \rangle$

7.3 Characterization of pure states

A density matrix represents a pure state if it is the rank 1 projection of a single vector. These can be characterized either as the density matrices with

a square of trace 1, or as the density matrices that are projectors.

definition *pure-density-operator* **where**
pure-density-operator $R \equiv (\exists v. R = \text{rank-1-proj } v)$

lemma *density-pure-single-diag*:
assumes $A \in \text{carrier-mat } n \ n$
and $\text{Complex-Matrix.trace } A = (1::\text{real})$
and $\text{Complex-Matrix.trace } (A*A) = (1::\text{real})$
and $\text{unitary-diag } A \ B \ U$
and $I = \{0 .. < n\}$
and $\forall i \in I. A \$\$ (i,i) \geq 0$
and $\forall i \in I. B \$\$ (i,i) \geq 0$
shows $\exists j \in I. B \$\$ (j,j) = 1 \wedge (\forall i \in I - \{j\}. B \$\$ (i,i) = 0)$
(proof)

lemma *rank-1-proj-square-trace*:
fixes $v :: \text{complex Matrix.vec}$
assumes $A = \text{rank-1-proj } v$
shows $\text{Complex-Matrix.trace } (A*A) = \|v\|^2 * \text{Complex-Matrix.trace } A$
(proof)

lemma *rank-1-proj-trace'*:
assumes $\text{Complex-Matrix.trace } (\text{rank-1-proj } v) = 1$
shows $\|v\| = 1$
(proof)

lemma *density-square-pure*:
assumes $A \in \text{carrier-mat } n \ n$
and $0 < n$
and $\text{density-operator } A$
and $\text{Complex-Matrix.trace } (A*A) = 1$
shows *pure-density-operator* A
(proof)

lemma *density-square-pure'*:
assumes *density-operator* A
and $A = \text{rank-1-proj } v$
shows $\text{Complex-Matrix.trace } (A*A) = 1$
(proof)

lemma
assumes $A \in \text{carrier-mat } n \ n$
and $0 < n$
and *density-operator* A
shows *pure-density-charact*:
 $(\text{pure-density-operator } A) \longleftrightarrow (\text{Complex-Matrix.trace } (A*A) = 1)$
and *pure-density-charact'*:
 $(\text{pure-density-operator } A) \longleftrightarrow (A*A = A)$
(proof)

8 Quantum expectation values and traces

The expectation value of a projective measurement is the average outcome value of the measurement, where each outcome value is weighted by the probability that it occurs. We show that the expectation value of a density matrix ρ for an observable represented by the Hermitian matrix A is $\text{Tr}(A \cdot \rho)$.

```

definition (in cpx-sq-mat) expect-value where
expect-value R p M =
  sum (λi. meas-outcome-prob R M i * (meas-outcome-val (M i))) {..< p}

definition (in cpx-sq-mat) obs-expect-value where
obs-expect-value R A =
  expect-value R (proj-meas-size (make-pm A)) (proj-meas-outcomes (make-pm A))

lemma (in cpx-sq-mat) expect-value-trace:
assumes proj-measurement p M
and R ∈ fc-mats
shows expect-value R p M =
  Complex-Matrix.trace (sum-mat
    (λi. meas-outcome-val (M i) ·m (meas-outcome-prj (M i))) {..< p} * R)
  ⟨proof⟩

lemma (in cpx-sq-mat) expect-value-hermitian:
assumes A ∈ fc-mats
and hermitian A
and make-pm A = (p, M)
and R ∈ fc-mats
shows expect-value R p M = Complex-Matrix.trace (A * R)
  ⟨proof⟩

lemma obs-expect-value:
assumes A ∈ carrier-mat n n
and hermitian A
and R ∈ carrier-mat n n
and 0 < n
shows cpx-sq-mat.obs-expect-value n n R A = Complex-Matrix.trace (A * R)
  ⟨proof⟩

end

theory Tsirelson
imports
  Projective-Measurements.CHSH-Inequality
  Matrix-L2-Operator-Norm Density-Matrix-Basics

begin

```

This part contains a formalization of the CHSH operator and the CHSH

quantum expectation, along with Tsirelson’s proof that this quantum expectation cannot be greater than $2\sqrt{2}$. The development of this proof permits to extract the additional result that when one of the parties involved in the CHSH experiment makes measurements on commuting observables, the quantum expectation cannot be greater than 2. This is the same upper-bound as in the case where a local hidden variable hypothesis is made.

9 CHSH inequalities

The CHSH operator is used to represent the experiment in which two parties each perform measurements using two observables, respectively A_1, A_2 and B_1, B_2 . Given the resource R , in general a density matrix representing an entangled state, the CHSH expectation represents the quantum expectation of performing simultaneous measurements on R . The CHSH setting also assumes that along with being Hermitian matrices, all the squared observables are equal to the identity and commute with the observables of the other party.

9.1 Some intermediate results for particular observables

lemma *chsh-complex*:

```
fixes A0::complex
assumes A0 ∈ Reals
and B0 ∈ Reals
and A1 ∈ Reals
and B1 ∈ Reals
and |A0 * B1| ≤ 1
and |A0 * B0| ≤ 1
and |A1 * B0| ≤ 1
and |A1 * B1| ≤ 1
```

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

lemma (in bin-cpx) *Z-XpZ-rho-trace*:

shows $\text{Complex-Matrix.trace}(Z-I * I-XpZ * \rho) = 1/\sqrt{2}$
 $\langle proof \rangle$

lemma (in bin-cpx) *X-XpZ-rho-trace*:

shows $\text{Complex-Matrix.trace}(X-I * I-XpZ * \rho) = 1/\sqrt{2}$
 $\langle proof \rangle$

lemma (in bin-cpx) *X-ZmX-rho-trace*:

shows $\text{Complex-Matrix.trace}(X-I * I-ZmX * \rho) = 1/\sqrt{2}$
 $\langle proof \rangle$

lemma (in bin-cpx) Z-ZmX-rho-trace:
shows Complex-Matrix.trace (Z-I * I-ZmX * rho-psim) = -1/sqrt 2
 $\langle proof \rangle$

9.2 The CHSH operator and expectation

definition CHSH-op :: 'a::conjugatable-field Matrix.mat \Rightarrow 'a Matrix.mat \Rightarrow
' a Matrix.mat \Rightarrow ' a Matrix.mat \Rightarrow ' a Matrix.mat
where

CHSH-op A0 A1 B0 B1 = A0 * B1 - A0 * B0 + A1 * B0 + A1 * B1

definition CHSH-expect :: 'a::conjugatable-field Matrix.mat \Rightarrow 'a Matrix.mat \Rightarrow
' a Matrix.mat \Rightarrow ' a Matrix.mat \Rightarrow ' a Matrix.mat \Rightarrow 'a
where

CHSH-expect A0 A1 B0 B1 R = Complex-Matrix.trace ((CHSH-op A0 A1 B0 B1)
* R)

definition CHSH-cond :: nat \Rightarrow 'a::conjugatable-field Matrix.mat \Rightarrow
' a::conjugatable-field Matrix.mat \Rightarrow
' a::conjugatable-field Matrix.mat \Rightarrow 'a::conjugatable-field Matrix.mat \Rightarrow bool
where

CHSH-cond n A0 A1 B0 B1 =
(A0 \in carrier-mat n n \wedge
A0 * A0 = 1_m n \wedge
A1 \in carrier-mat n n \wedge
A1 * A1 = 1_m n \wedge
B0 \in carrier-mat n n \wedge
B0 * B0 = 1_m n \wedge
B1 \in carrier-mat n n \wedge
B1 * B1 = 1_m n \wedge
A0 * B1 = B1 * A0 \wedge
A0 * B0 = B0 * A0 \wedge
A1 * B0 = B0 * A1 \wedge
A1 * B1 = B1 * A1)

definition CHSH-cond-hermit where
CHSH-cond-hermit n A0 A1 B0 B1 \longleftrightarrow CHSH-cond n A0 A1 B0 B1 \wedge hermitian
A0 \wedge
hermitian A1 \wedge hermitian B0 \wedge hermitian B1

lemma CHSH-op-dim:
assumes A0 \in carrier-mat n m
and A1 \in carrier-mat n m
and B0 \in carrier-mat m p
and B1 \in carrier-mat m p
shows CHSH-op A0 A1 B0 B1 \in carrier-mat n p $\langle proof \rangle$

lemma CHSH-op-hermitian:
assumes hermitian A0

```

and hermitian B0
and hermitian A1
and hermitian B1
and A0 * B0 = B0 * A0
and A1 * B0 = B0 * A1
and A0 * B1 = B1 * A0
and A1 * B1 = B1 * A1
shows hermitian (CHSH-op A0 A1 B0 B1)
⟨proof⟩

lemma CHSH-cond-hermit-expect-eq:
assumes CHSH-cond-hermit n A0 A1 B0 B1
and R ∈ carrier-mat n n
and 0 < n
shows CHSH-expect A0 A1 B0 B1 R =
    cpx-sq-mat.obs-expect-value n n R (CHSH-op A0 A1 B0 B1)
⟨proof⟩

lemma CHSH-op-expand-right:
fixes A0:'a::conjugatable-field Matrix.mat
assumes A0 ∈ carrier-mat n m
and A1 ∈ carrier-mat n m
and B0 ∈ carrier-mat m p
and B1 ∈ carrier-mat m p
and R ∈ carrier-mat p p'
shows (CHSH-op A0 A1 B0 B1) * R =
    A0 * B1 * R - A0 * B0 * R + A1 * B0 * R + A1 * B1 * R
⟨proof⟩

lemma CHSH-op-expand-left:
fixes A0:'a::conjugatable-field Matrix.mat
assumes A0 ∈ carrier-mat n m
and A1 ∈ carrier-mat n m
and B0 ∈ carrier-mat m p
and B1 ∈ carrier-mat m p
and R ∈ carrier-mat p n
shows R * (CHSH-op A0 A1 B0 B1) =
    R * (A0 * B1) - R * (A0 * B0) + R * (A1 * B0) + R * (A1 * B1)
⟨proof⟩

lemma CHSH-expect-expand:
assumes A0 ∈ carrier-mat n m
and A1 ∈ carrier-mat n m
and B0 ∈ carrier-mat m p
and B1 ∈ carrier-mat m p
and R ∈ carrier-mat p n
shows CHSH-expect A0 A1 B0 B1 R =
    Complex-Matrix.trace (A0 * B1 * R) -
    Complex-Matrix.trace (A0 * B0 * R) +

```

```

Complex-Matrix.trace (A1 * B0 * R) +
Complex-Matrix.trace (A1 * B1 * R)
⟨proof⟩

```

lemma CHSH-condD:

```

assumes CHSH-cond n A0 A1 B0 B1
shows A0 ∈ carrier-mat n n
A0 * A0 = 1m n
A1 ∈ carrier-mat n n
A1 * A1 = 1m n
B0 ∈ carrier-mat n n
B0 * B0 = 1m n
B1 ∈ carrier-mat n n
B1 * B1 = 1m n
A0 * B1 = B1 * A0
A0 * B0 = B0 * A0
A1 * B0 = B0 * A1
A1 * B1 = B1 * A1 ⟨proof⟩

```

lemma CHSH-cond-simps[simp]:

```

assumes CHSH-cond n A0 A1 B0 B1
shows A1 * B1 * (A0 * B1) = A1 * A0
A1 * B1 * (A1 * B0) = B1 * B0
A1 * B1 * (A1 * B1) = 1m n
A1 * B1 * (A0 * B0) = A1 * A0 * (B1 * B0)
A1 * B0 * (A0 * B1) = A1 * A0 * (B0 * B1)
A1 * B0 * (A0 * B0) = A1 * A0
A1 * B0 * (A1 * B0) = 1m n
A1 * B0 * (A1 * B1) = B0 * B1
A0 * B0 * (A0 * B1) = B0 * B1
A0 * B0 * (A0 * B0) = 1m n
A0 * B0 * (A1 * B0) = A0 * A1
A0 * B0 * (A1 * B1) = A0 * A1 * (B0 * B1)
A0 * B1 * (A0 * B1) = 1m n
A0 * B1 * (A0 * B0) = B1 * B0
A0 * B1 * (A1 * B0) = A0 * A1 * (B1 * B0)
A0 * B1 * (A1 * B1) = A0 * A1
⟨proof⟩

```

lemma CHSH-op-square:

```

assumes CHSH-cond n A0 A1 B0 B1
shows (CHSH-op A0 A1 B0 B1) * (CHSH-op A0 A1 B0 B1) =
(4::nat) ·m (1m n) - (commutator A0 A1) * (commutator B0 B1)
⟨proof⟩

```

lemma CHSH-cond-hermitD:

```

assumes CHSH-cond-hermit n A0 A1 B0 B1
shows CHSH-cond n A0 A1 B0 B1
hermitian A0

```

hermitian A1
hermitian B0
hermitian B1
 $\langle proof \rangle$

lemma *CHSH-cond-hermit-unitary*:
assumes *CHSH-cond-hermit n A0 A1 B0 B1*
shows *unitary A0 unitary A1 unitary B0 unitary B1*
 $\langle proof \rangle$

lemma *CHSH-expect-add*:
assumes *A0 ∈ carrier-mat n n*
and *A1 ∈ carrier-mat n n*
and *B0 ∈ carrier-mat n n*
and *B1 ∈ carrier-mat n n*
and *R0 ∈ carrier-mat n n*
and *R1 ∈ carrier-mat n n*
shows *CHSH-expect A0 A1 B0 B1 (R0 + R1) =*
CHSH-expect A0 A1 B0 B1 R0 +
CHSH-expect A0 A1 B0 B1 R1
 $\langle proof \rangle$

lemma *CHSH-expect-zero*:
assumes *A0 ∈ carrier-mat n n*
and *A1 ∈ carrier-mat n n*
and *B0 ∈ carrier-mat n n*
and *B1 ∈ carrier-mat n n*
shows *CHSH-expect A0 A1 B0 B1 (0_m n n) = 0*
 $\langle proof \rangle$

lemma (in cpx-sq-mat) *CHSH-expect-sum*:
assumes *finite S*
and *A0 ∈ fc-mats*
and *A1 ∈ fc-mats*
and *B0 ∈ fc-mats*
and *B1 ∈ fc-mats*
and *$\bigwedge i. i \in S \implies R i \in fc-mats$*
shows *CHSH-expect A0 A1 B0 B1 (sum-mat R S) =*
sum ($\lambda i. CHSH-expect A0 A1 B0 B1 (R i)$) S *$\langle proof \rangle$*

lemma *CHSH-expect-smult*:
assumes *A0 ∈ carrier-mat n n*
and *A1 ∈ carrier-mat n n*
and *B0 ∈ carrier-mat n n*
and *B1 ∈ carrier-mat n n*
and *R0 ∈ carrier-mat n n*
shows *CHSH-expect A0 A1 B0 B1 (a ·_m R0) =*
*a * CHSH-expect A0 A1 B0 B1 R0*

$\langle proof \rangle$

```
lemma CHSH-expect-real:
assumes 0 < n
and CHSH-cond-hermit n A0 A1 B0 B1
and R ∈ carrier-mat n n
and Complex-Matrix.positive R
shows CHSH-expect A0 A1 B0 B1 R ∈ Reals
⟨proof⟩

lemma CHSH-op-square-L2-op-nrm-le:
assumes CHSH-cond-hermit n A0 A1 B0 B1
and 0 < n
shows L2-op-nrm ((CHSH-op A0 A1 B0 B1) * (CHSH-op A0 A1 B0 B1)) ≤ 8
⟨proof⟩

lemma CHSH-op-square-spmmax-le:
assumes CHSH-cond-hermit n A0 A1 B0 B1
and 0 < n
shows spmax ((CHSH-op A0 A1 B0 B1) * (CHSH-op A0 A1 B0 B1)) ≤ 8
⟨proof⟩

lemma CHSH-op-L2-op-nrm-le:
assumes CHSH-cond-hermit n A0 A1 B0 B1
and 0 < n
shows L2-op-nrm (CHSH-op A0 A1 B0 B1) ≤ 2 * sqrt 2
⟨proof⟩

lemma (in cpx-sq-mat) CHSH-cond-hermit-lhv-upper:
assumes CHSH-cond-hermit dimR A0 A1 B0 B1
and lhv M A0 B1 R U0 V1
and lhv M A0 B0 R U0 V0
and lhv M A1 B0 R U1 V0
and lhv M A1 B1 R U1 V1
and 0 < n
shows |(LINT w|M. qt-expect A0 U0 w * qt-expect B1 V1 w) −
(LINT w|M. qt-expect A0 U0 w * qt-expect B0 V0 w) +
(LINT w|M. qt-expect A1 U1 w * qt-expect B0 V0 w) +
(LINT w|M. qt-expect A1 U1 w * qt-expect B1 V1 w)| ≤ 2
⟨proof⟩

lemma (in cpx-sq-mat) CHSH-expect-lhv-lint-eq:
assumes R ∈ fc-mats
and Complex-Matrix.positive R
and CHSH-cond-hermit dimR A0 A1 B0 B1
and lhv M A0 B1 R U0 V1
and lhv M A0 B0 R U0 V0
and lhv M A1 B0 R U1 V0
```

and $\text{lhv } M \ A1 \ B1 \ R \ U1 \ V1$
shows $(\text{LINT } w|M. \ \text{qt-expect } A0 \ U0 \ w * \text{qt-expect } B1 \ V1 \ w) -$
 $(\text{LINT } w|M. \ \text{qt-expect } A0 \ U0 \ w * \text{qt-expect } B0 \ V0 \ w) +$
 $(\text{LINT } w|M. \ \text{qt-expect } A1 \ U1 \ w * \text{qt-expect } B0 \ V0 \ w) +$
 $(\text{LINT } w|M. \ \text{qt-expect } A1 \ U1 \ w * \text{qt-expect } B1 \ V1 \ w) =$
 $\text{CHSH-expect } A0 \ A1 \ B0 \ B1 \ R \ (\text{is } ?L = ?R)$
 $\langle \text{proof} \rangle$

9.3 CHSH inequality for separable density matrices

definition CHSH-cond-local **where**

$\text{CHSH-cond-local } n \ m \ A0 \ A1 \ B0 \ B1 \equiv$
 $A0 \in \text{carrier-mat } n \ n \wedge A1 \in \text{carrier-mat } n \ n \wedge$
 $B0 \in \text{carrier-mat } m \ m \wedge B1 \in \text{carrier-mat } m \ m \wedge$
 $\text{hermitian } A0 \wedge \text{hermitian } A1 \wedge \text{hermitian } B0 \wedge \text{hermitian } B1 \wedge$
 $A0 * A0 = 1_m \ n \wedge A1 * A1 = 1_m \ n \wedge B0 * B0 = 1_m \ m \wedge B1 * B1 = 1_m \ m$

lemma $\text{CHSH-cond-local-imp-cond-hermit:}$

assumes $\text{CHSH-cond-local } n \ m \ A0 \ A1 \ B0 \ B1$
and $0 < n$
and $0 < m$
shows $\text{CHSH-cond-hermit } (n*m) \ (A0 \otimes 1_m \ m) \ (A1 \otimes 1_m \ m)$
 $(1_m \ n \otimes B0) \ (1_m \ n \otimes B1)$
 $\langle \text{proof} \rangle$

lemma limit-CHSH-cond:

shows $\text{CHSH-cond-hermit } 4 \ Z\text{-}I \ X\text{-}I \ I\text{-}ZmX \ I\text{-}XpZ$
 $\langle \text{proof} \rangle$

lemma $\text{CHSH-expect-separable-expand:}$

assumes $\text{separately-decomposes } R \ n \ nA \ nB \ K \ F \ S$
and $A0 \in \text{carrier-mat } nA \ nA$
and $A1 \in \text{carrier-mat } nA \ nA$
and $B0 \in \text{carrier-mat } nB \ nB$
and $B1 \in \text{carrier-mat } nB \ nB$
shows $\text{CHSH-expect } (A0 \otimes 1_m \ nB) \ (A1 \otimes 1_m \ nB) \ (1_m \ nA \otimes B0) \ (1_m \ nA \otimes B1)$
 $R =$
 $\text{sum } (\lambda a. \ K \ a * \text{CHSH-expect } (A0 \otimes 1_m \ nB) \ (A1 \otimes 1_m \ nB) \ (1_m \ nA \otimes B0) \ (1_m \ nA \otimes B1))$
 $((F \ a) \otimes (S \ a))) \ \{.. < n\}$
 $\langle \text{proof} \rangle$

lemma $\text{CHSH-expect-tensor-leq:}$

assumes $\text{CHSH-cond-local } nA \ nB \ A0 \ A1 \ B0 \ B1$
and $RA \in \text{carrier-mat } nA \ nA$
and $\text{density-operator } RA$
and $RB \in \text{carrier-mat } nB \ nB$
and $\text{density-operator } RB$
and $0 < nA$

and $0 < nB$
shows $|CHSH\text{-expect}(A0 \otimes 1_m nB)(A1 \otimes 1_m nB)(1_m nA \otimes B0)(1_m nA \otimes B1)(RA \otimes RB)| \leq 2$
 $\langle proof \rangle$

9.4 CHSH inequality for commuting observables

lemma *CHSH-op-square-commute-L2-op-nrm-eq*:
assumes *CHSH-cond-hermit n A0 A1 B0 B1*
and $0 < n$
and *commutator A0 A1 = 0_m n n* \vee *commutator B0 B1 = 0_m n n*
shows *L2-op-nrm ((CHSH-op A0 A1 B0 B1) * (CHSH-op A0 A1 B0 B1)) = 4*
 $\langle proof \rangle$

lemma *CHSH-op-square-commute-spmax-eq*:
assumes *CHSH-cond-hermit n A0 A1 B0 B1*
and $0 < n$
and *commutator A0 A1 = 0_m n n* \vee *commutator B0 B1 = 0_m n n*
shows *spmax ((CHSH-op A0 A1 B0 B1) * (CHSH-op A0 A1 B0 B1)) = 4*
 $\langle proof \rangle$

lemma *CHSH-op-commute-L2-op-nrm-eq*:
assumes *CHSH-cond-hermit n A0 A1 B0 B1*
and $0 < n$
and *commutator A0 A1 = 0_m n n* \vee *commutator B0 B1 = 0_m n n*
shows *L2-op-nrm (CHSH-op A0 A1 B0 B1) = 2*
 $\langle proof \rangle$

9.5 Result summary on the CHSH inequalities

Under the local hidden variable hypothesis, this value is bounded by 2.

lemma *CHSH-expect-lhv-leq*:
assumes *R ∈ carrier-mat n n*
and $0 < n$
and *Complex-Matrix.positive R*
and *CHSH-cond-hermit n A0 A1 B0 B1*
and *cpx-sq-mat.lhv n n M A0 B1 R U0 V1*
and *cpx-sq-mat.lhv n n M A0 B0 R U0 V0*
and *cpx-sq-mat.lhv n n M A1 B0 R U1 V0*
and *cpx-sq-mat.lhv n n M A1 B1 R U1 V1*
shows $|CHSH\text{-expect}(A0 A1 B0 B1 R)| \leq 2$
 $\langle proof \rangle$

When the considered density operator is separable, this value is still bounded by 2.

lemma *CHSH-expect-separable-leq*:
assumes *CHSH-cond-local nA nB A0 A1 B0 B1*
and *separable-density nA nB R*
and *A0 ∈ carrier-mat nA nA*

```

and A1 ∈ carrier-mat nA nA
and B0 ∈ carrier-mat nB nB
and B1 ∈ carrier-mat nB nB
shows |CHSH-expect (A0⊗1m nB) (A1⊗1m nB) (1m nA⊗B0) (1m nA⊗B1)
R|
≤ 2
⟨proof⟩

```

When any of the pairs of observables used in the measurements commutes, this value remains bounded by 2.

```

lemma CHSH-expect-commute-leq:
assumes CHSH-cond-hermit n A0 A1 B0 B1
and R ∈ carrier-mat n n
and density-operator R
and 0 < n
and commutator A0 A1 = 0m n n ∨ commutator B0 B1 = 0m n n
shows |CHSH-expect A0 A1 B0 B1 R| ≤ 2
⟨proof⟩

```

In the general case, this value is bounded by $2 \cdot \sqrt{2}$.

```

lemma CHSH-expect-gen-leq:
assumes CHSH-cond-hermit n A0 A1 B0 B1
and R ∈ carrier-mat n n
and density-operator R
and 0 < n
shows |CHSH-expect A0 A1 B0 B1 R| ≤ (2 * sqrt 2)
⟨proof⟩

```

The bound $2 \cdot \sqrt{2}$ can be reached by a suitable choice of observables, when the Bell state is measured.

```

lemma CHSH-expect-limit:
shows |CHSH-expect Z-I X-I I-ZmX I-XpZ rho-psim| = 2 * sqrt 2
⟨proof⟩

```

end