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

Mnacho Echenim and Mehdi Mhalla and Coraline Mori

September 13, 2023

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

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 $\bigwedge i.\ i \in I \Longrightarrow (0 :: 'a :: linordered\text{-}field) \leq f\ i$
and $0 < sum\ f\ I$
shows $\exists j \in I.\ 0 < f\ j$
⟨*proof*⟩

lemma *pos-square-1-elem*:
assumes *finite I*
and $\bigwedge i.\ i \in I \Longrightarrow (0::real) \leq f\ i$
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 $\bigwedge i.\ i \in I \Longrightarrow (0::complex) \leq f\ i$
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 $\bigwedge i.\ i \in I \Longrightarrow (0::'a :: linordered\text{-}field) \leq f\ i$
and $sum\ f\ I = c$
and $j \in I$
and $f\ j = c$
shows $\forall k \in (I - \{j\}).\ f\ k = 0$
⟨*proof*⟩

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

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

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

lemma *real-cpx-abs-leq*:
fixes $A::complex$
assumes $A \in Reals$
and $B \in Reals$
and $|A * B| \leq 1$

shows $|Re\ A * Re\ B| \leq 1$
⟨*proof*⟩

lemma *cpx-real-abs-eq*:
fixes z::*complex* **and** r::*real*
assumes $z \in Reals$
and $z = r$
shows $|z| = |r|$
⟨*proof*⟩

lemma *cpx-real-abs-leq*:
fixes z::*complex* **and** r::*real*
assumes $z \in Reals$
and $z = r$
and $|r| \leq k$
shows $|z| \leq (k$::*real*$)$
⟨*proof*⟩

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

lemma *sum-abs-cpx*:
shows $|sum\ K\ I| \leq sum\ (\lambda x.\ |(K\ x)$::*complex*$|)\ I$
⟨*proof*⟩

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

lemma *cpx-ge-0-real*:
fixes c::*complex*
assumes $0 \leq c$
and $c \in Reals$
shows $0 \leq Re\ c$
⟨*proof*⟩

lemma *cpx-of-real-ge-0*:
assumes $0 \leq complex\text{-}of\text{-}real\ a$
shows $0 \leq a$
⟨*proof*⟩

lemma *set-cst-list*:

shows $(\bigwedge i.\ i < \mathit{length}\ l \Longrightarrow l!i = x) \Longrightarrow 0 < \mathit{length}\ l \Longrightarrow \mathit{set}\ l = \{x\}$
⟨*proof*⟩

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

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

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

lemma *Sup-ge-real*:
assumes $a{\in}\ (A{::}\mathit{real\ set})$
and $\forall\, a \in A.\ a \leq c$
and $\forall\, a \in A.\ b \leq a$
shows $a \leq \mathit{Sup}\ A$
⟨*proof*⟩

lemma *Sup-real-le*:
assumes $\forall\, a \in (A{::}\mathit{real\ set}).\ a \leq c$
and $\forall\, a \in A.\ b \leq a$
and $A{\neq}\ \{\}$
shows $\mathit{Sup}\ A \leq c$
⟨*proof*⟩

2 Results in linear algebra

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

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* ∈ {*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 ∃*j*<*n*. *B* $$ (*j*,*j*) = *1* ∧ (∀ *i*<*n*. *i*≠*j* ⟶ *B* $$ (*i*,*i*) = *0*)
⟨*proof*⟩

lemma *idty-square*:
shows ((1_m *n*):: '*a* :: *semiring-1 Matrix.mat*) ∗ (1_m *n*) = 1_m *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* $(B*A) \in$ *Reals*
⟨*proof*⟩

lemma *pos-hermitian-trace-reals'*:
fixes *A*::*complex Matrix.mat*
assumes $A \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
and $0 < n$
and *Complex-Matrix.positive A*
and *hermitian B*
shows *Complex-Matrix.trace* $(A*B) \in$ *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* $(1_m\ n)$ *B U*
shows $B = 1_m\ n$
⟨*proof*⟩

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

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

lemma *spectrum-ne*:
fixes *A*::*complex Matrix.mat*
assumes $A \in$ *carrier-mat n n*
and $0 < n$
shows *spectrum A* $\neq \{\}$ ⟨*proof*⟩

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

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

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

lemma *hermitian-square-spectrum-eq*:
fixes A::*complex Matrix.mat*
assumes *hermitian* A
and $A\in$ *carrier-mat* n n
and $0 < n$
shows *spectrum* $(A*A)$ = $\{a*a \mid a.\ a\in$ *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 $\bigwedge i.\ i \in I \Longrightarrow A\ i \in$ *fc-mats*
and $\bigwedge i.\ i\in I \Longrightarrow f\ i = 0$
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 $\bigwedge i.\ i \in I \Longrightarrow A\ i = 0_m$ *dimR dimC*
shows *sum-mat* $A\ I = 0_m$ *dimR dimC* ⟨*proof*⟩

lemma (**in** *fixed-carrier-mat*) *sum-mat-remove*:
assumes A ' $I \subseteq$ *fc-mats*
and A: *finite* I **and** x: $x \in I$
shows *sum-mat* $A\ I = A\ x$ + *sum-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 ' $I \subseteq$ *fc-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 $I \cap J = \{\}$
and $\forall\ i \in I \cup J.\ A\ i \in$ *fc-mats*
shows *sum-mat A* $(I \cup J)$ = *sum-mat A I* + *sum-mat A J* ⟨*proof*⟩

lemma *sum-with-reindex-cong'*:
fixes $g :: 'c \Rightarrow 'a$ *Matrix.mat*
assumes $\forall x.\ g\ x \in$ *fc-mats*
and $\forall x.\ h\ x \in$ *fc-mats*
and *inj-on l B*
and $\bigwedge x.\ x \in B \Longrightarrow g\ (l\ x) = h\ x$
shows *sum-with* (+) $(0_m$ *dimR dimC*) *g* (*l* ‘ *B*) =
sum-with (+) $(0_m$ *dimR dimC*) *h B*
⟨*proof*⟩

lemma *sum-mat-cong'*:
shows *finite I* $\Longrightarrow$ $(\bigwedge i.\ i \in I \Longrightarrow A\ i = B\ i) \Longrightarrow$
$(\bigwedge i.\ i \in I \Longrightarrow A\ i \in$ *fc-mats*$) \Longrightarrow$
$(\bigwedge i.\ i \in I \Longrightarrow B\ i \in$ *fc-mats*$) \Longrightarrow I = J \Longrightarrow$ *sum-mat A I* = *sum-mat B J*
⟨*proof*⟩

lemma *sum-mat-reindex-cong*:
assumes *finite B*
and $\bigwedge x.\ x \in l\text{‘ } B \Longrightarrow g\ x \in$ *fc-mats*
and $\bigwedge x.\ x \in B \Longrightarrow h\ x \in$ *fc-mats*
and *inj-on l B*
and $\bigwedge x.\ x \in B \Longrightarrow g\ (l\ x) = h\ x$
shows *sum-mat g* (*l* ‘ *B*) = *sum-mat h B*
⟨*proof*⟩

lemma *sum-mat-mod-eq*:
fixes *A* :: *nat* ⇒ ′*a Matrix.mat*
assumes $\bigwedge x.\ x \in \{..<m\} \Longrightarrow A\ x \in$ *fc-mats*
shows *sum-mat* $(\lambda i.\ A\ (i\ mod\ m))$ $((\lambda i.\ n * m+i)\text{‘}\{..<m\}) =$ *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 $\bigwedge k.\ k < (nC::nat) \Longrightarrow A\ k \in carrier\text{-}mat\ n\ n$
and $\bigwedge j.\ j < (nD::nat) \Longrightarrow B\ j \in carrier\text{-}mat\ m\ m$
and $0 < n$
and $0 < m$
and $dimR = n * m$
and $nD \neq 0$
shows $sum\text{-}mat\ (\lambda i.\ sum\text{-}mat\ (\lambda j.\ f\ i * g\ j \cdot_m ((A\ i) \bigotimes (B\ j)))\ \{..< nD\})$
$\{..< nC\} =$
$sum\text{-}mat\ (\lambda i.\ (f\ (i\ div\ nD) * g\ (i\ mod\ nD)) \cdot_m$
$((A\ (i\ div\ nD)) \bigotimes (B\ (i\ mod\ nD))))\ \{..< nC * nD\}$
⟨*proof*⟩

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

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

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

⟨*proof*⟩

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

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

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 $Complex\text{-}Matrix.trace\ (A \bigotimes B) = Complex\text{-}Matrix.trace\ A *$
$Complex\text{-}Matrix.trace\ B$
⟨*proof*⟩

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 $Complex\text{-}Matrix.inner\text{-}prod\ (tensor\text{-}vec\ u\ v)\ (tensor\text{-}vec\ a\ b) =$
$Complex\text{-}Matrix.inner\text{-}prod\ u\ a * Complex\text{-}Matrix.inner\text{-}prod\ v\ b$

⟨*proof*⟩

lemma *tensor-mat-positive*:
assumes $A \in$ *carrier-mat n n*
and $B \in$ *carrier-mat m m*
and $0 < n$
and $0 < m$
and *Complex-Matrix.positive A*
and *Complex-Matrix.positive B*
shows *Complex-Matrix.positive* $(A \bigotimes B)$
⟨*proof*⟩

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

lemma *tensor-mat-commute*:
assumes $A \in$ *carrier-mat n n*
and $B \in$ *carrier-mat m m*
and $C \in$ *carrier-mat n n*
and $D \in$ *carrier-mat m m*
and $0 < n$
and $0 < m$
and $A * C = C * A$
and $B * D = D * B$
shows $(A \bigotimes B) * (C \bigotimes D) = (C \bigotimes D) * (A \bigotimes B)$
⟨*proof*⟩

lemma *tensor-mat-mult-id*:
assumes $A \in$ *carrier-mat n n*
and $B \in$ *carrier-mat m m*
and $0 < n$
and $0 < m$
shows $(A \bigotimes 1_m\ m) * (1_m\ n \bigotimes B) = A \bigotimes B$
⟨*proof*⟩

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

Complex-Matrix.trace $(A * C) * (\textit{Complex-Matrix.trace}\ (B * D))$
⟨*proof*⟩

lemma *tensor-mat-diagonal*:
assumes $A \in$ *carrier-mat n n*
and $B \in$ *carrier-mat m m*
and *diagonal-mat A*
and *diagonal-mat B*
shows *diagonal-mat* $(A \bigotimes B)$ ⟨*proof*⟩

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

lemma *tensor-mat-zero*:
assumes $B \in$ *carrier-mat i j*
and $0 < j$
and $0 < m$
shows $0_m\ n\ m \bigotimes B = 0_m\ (n * i)\ (m * j)$
⟨*proof*⟩

lemma *tensor-mat-zero'*:
assumes $B \in$ *carrier-mat i j*
and $0 < j$
and $0 < m$
shows $B \bigotimes 0_m\ n\ m = 0_m\ (i * n)\ (j*m)$
⟨*proof*⟩

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

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

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

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

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

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

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

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

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

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

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

context *cpx-sq-mat*
begin

lemma *tensor-mat-sum-mat-right*:
assumes *finite I*
and $A \in \textit{carrier-mat}\ n\ n$
and $\bigwedge k.\ k \in I \Longrightarrow B\ k \in \textit{carrier-mat}\ i\ i$
and $0 < n$
and $0 < i$
and $dimR = n * i$
shows $A \bigotimes (\textit{fixed-carrier-mat.sum-mat}\ i\ i\ B\ I) = \textit{sum-mat}\ (\lambda i.\ A \bigotimes (B\ i))\ I$
⟨*proof*⟩

lemma *tensor-mat-sum-mat-left*:
assumes *finite I*
and $B \in \textit{carrier-mat}\ i\ i$
and $\bigwedge k.\ k \in I \Longrightarrow A\ k \in \textit{carrier-mat}\ n\ n$
and $0 < n$
and $0 < i$
and $dimR = n * i$
shows $(\textit{fixed-carrier-mat.sum-mat}\ n\ n\ A\ I) \bigotimes B = \textit{sum-mat}\ (\lambda i.\ (A\ i) \bigotimes B)\ I$
⟨*proof*⟩

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

$\{..< nC*nD\} = C \bigotimes D$ ⟨*proof*⟩

lemma *tensor-mat-sum-nat-mod-div-eq-0*:
assumes $\bigwedge k.\ k < (nC::nat) \Longrightarrow A\ k \in carrier\text{-}mat\ n\ n$
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* $(\lambda i.\ (f\ (i\ div\ nD) * g\ (i\ mod\ nD)) \cdot_m$
$((A\ (i\ div\ nD)) \bigotimes (B\ (i\ mod\ nD))))$
$\{..< nC*nD\} = C \bigotimes D$
⟨*proof*⟩

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

end

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

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

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

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

lemma *tensor-mat-make-pm-mult-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 $sum\ (\lambda i.\ (sum\ (\lambda j.\ Complex\text{-}Matrix.trace$
$((complex\text{-}of\text{-}real\ (meas\text{-}outcome\text{-}val\ (M\ i)) *$
$complex\text{-}of\text{-}real\ (meas\text{-}outcome\text{-}val\ (N\ j))) \cdot_m$
$((meas\text{-}outcome\text{-}prj\ (M\ i)) \bigotimes (meas\text{-}outcome\text{-}prj\ (N\ j))) * R))\ \{..< nB\}))$
$\{..< nA\} =$
$Complex\text{-}Matrix.trace\ ((A \bigotimes B) * R)$
⟨*proof*⟩

lemma *tensor-mat-mat-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 $A \bigotimes C = mat\text{-}conj\ (U \bigotimes V)\ (B \bigotimes D)$
⟨*proof*⟩

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

lemma *hermitian-tensor-mat-decomp*:

assumes $A \in$ *carrier-mat n n*
and $C \in$ *carrier-mat m m*
and *unitary-diag A B U*
and *unitary-diag C D V*
and $0 < n$
and $0 < m$
shows *unitary-diag* $(A \otimes C)$ $(B \otimes D)$ $(U \otimes V)$
⟨*proof*⟩

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* ⇒ *complex Matrix.mat* ⇒
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-mat n n*
and $B \in$ *carrier-mat n n*
shows *commutator A B* ∈ *carrier-mat n n* ⟨*proof*⟩

lemma *anticommutator-dim*:
assumes $A \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
shows *anticommutator A B* ∈ *carrier-mat n n* ⟨*proof*⟩

lemma *commutator-zero-iff*:
assumes $A \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
shows *commutator A B* = 0_m *n n* ⟷ *A*∗*B* = *B* ∗ *A*

⟨*proof*⟩

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* = 0_m *n n* ⟷ *B*∗*A* = −(*A*∗*B*)
⟨*proof*⟩

lemma *commutator-mult-expand*:
assumes $A \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
and $C \in$ *carrier-mat n n*
and $D \in$ *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 $A \in$ *carrier-mat n n*
and $0 < n$
shows *spmax A* ∈ {*cmod a*|*a*. *a* ∈ *spectrum A*}
⟨*proof*⟩

lemma *spmax-geq-0*:
assumes $A \in$ *carrier-mat n n*
and $0 < n$
shows $0 \leq$ *spmax A*
⟨*proof*⟩

lemma *Re-inner-mult-diag-le*:
fixes *B*::*complex Matrix.mat*
assumes *diagonal-mat B*
and $B \in$ *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*))

⟨*proof*⟩

lemma *Re-inner-mult-diag-le'*:
fixes *B*::*complex Matrix.mat*
assumes *diagonal-mat B*
and $B \in$ *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 $A \in$ *carrier-mat n n*
and $0 < n$
and *hermitian A*
and $v \in$ *carrier-vec n*
shows *cmod* (*inner-prod v* (*A* $*_v$ *v*)) ≤ (*spmax A*) ∗ (*inner-prod v v*)
⟨*proof*⟩

lemma *hermitian-trace-rank-le*:
assumes $A \in$ *carrier-mat n n*
and *hermitian A*
and $v \in$ *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 $A\in$ *carrier-mat n n*
and $B\in$ *carrier-mat m m*
and $C\in$ *carrier-mat n n*
and $D\in$ *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* $((A \bigotimes B)*(C \bigotimes D))) \leq$
Re (*Complex-Matrix.trace C*) $*$ *Re* (*Complex-Matrix.trace D*) $*$
spmax A $*$ *spmax B*
⟨*proof*⟩

lemma *tensor-mat-hermitian-density-le*:
assumes $A\in$ *carrier-mat n n*
and $B\in$ *carrier-mat m m*
and $C\in$ *carrier-mat n n*
and $D\in$ *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* $((A \bigotimes B)*(C \bigotimes D))) \leq$
spmax A $*$ *spmax B*
⟨*proof*⟩

lemma *idty-spmax*:
assumes $0 < n$
shows *spmax* $(1_m\ n) = 1$ ⟨*proof*⟩

lemma *spmax-uminus*:
fixes *A*::*complex Matrix.mat*
assumes *hermitian A*
and $A\in$ *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* $(x \cdot_m A) = cmod\ x * spmax\ A$
⟨*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* $(x \cdot_m A) = x * spmax\ A$
⟨*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 $A*A = 1_m\ n$
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 $A * A = 1_m\ n$
and *density-operator R*
shows $|Complex\text{-}Matrix.trace\ (A*R)| \leq 1$
⟨*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* (*Complex-Matrix.trace* (($A \bigotimes B$)*($C \bigotimes D$))) ≤ 1
⟨*proof*⟩

5.2 Eigenvector for the element with maximum modulus

definition *spmax-wit* **where**
spmax-wit A = (*SOME k. eigenvalue A k* ∧ *spmax A* = *cmod k*)

lemma *spmax-wit-eigenvalue*:
assumes *A*∈ *carrier-mat n n*
and *0* < *n*
shows *eigenvalue A* (*spmax-wit A*) ∧ *spmax A* = *cmod* (*spmax-wit A*)
⟨*proof*⟩

lemma *find-eigen-spmax-neq-0*:
assumes *A*∈ *carrier-mat n n*
and *0* < *n*
shows *find-eigenvector A* (*spmax-wit A*) ≠ $0_v\ n$ ⟨*proof*⟩

lemma *find-eigen-spmax-dim*:
assumes *A*∈ *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* = *vec-normalize* (*find-eigenvector A* (*spmax-wit A*))
and *A*∈ *carrier-mat n n*
and *0* < *n*
shows *cmod* (*inner-prod v* (*A* $*_v$ *v*)) = *spmax A*
⟨*proof*⟩

6 The $\mathcal{L}_2$ operator norm

6.1 Definition and preliminary results

definition *rvec-norm* **where**
rvec-norm v = *Re* (*vec-norm v*)

definition *L2-op-nrm* **where**
L2-op-nrm A =
Sup {*rvec-norm* (*A* $*_v$ *v*) |*v. dim-vec v* = *dim-col A* ∧ *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 ≠ 0$_v$ (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 = 0$_v$ m*
shows *A *$_v$ v = 0$_v$ 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 $a^2 \leq b * c^2$
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**
max-sgval A = *sqrt* (*spmax* (*Complex-Matrix.adjoint A* ∗ *A*))

lemma *max-sgval-geq-0*:

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

lemma *max-sgval-uminus*:
shows *max-sgval* $(-A) =$ *max-sgval A*
⟨*proof*⟩

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

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

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

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

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

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

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

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

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

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

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

lemma *L2-op-nrm-geq-0*:
assumes $A \in$ *carrier-mat n n*
and $0 < n$
shows $0 \leq$ *L2-op-nrm A*
⟨*proof*⟩

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

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

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

lemma *L2-op-nrm-smult*:
assumes $A \in$ *carrier-mat n n*
and $0 < n$
shows *L2-op-nrm* $(c \cdot_m A) =$ *cmod c* $*$ *L2-op-nrm A*
⟨*proof*⟩

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

lemma *L2-op-nrm-triangle*:
assumes $A \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
and $0 < n$
shows *L2-op-nrm* $(A+B) \leq$ *L2-op-nrm A* $+$ *L2-op-nrm B*
⟨*proof*⟩

lemma *L2-op-nrm-triangle'*:
assumes $A \in$ *carrier-mat n n*
and $B \in$ *carrier-mat n n*
and $0 < n$
shows *L2-op-nrm* $(A-B) \leq$ *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 (1*$_m$ *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 = 1*$_m$ *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 = 1*$_m$ *n*
and *B∗B = 1*$_m$ *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) ·*$_m$ *(1*$_m$ *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 $\bigwedge i.\ i \in \{..< (n::nat)\} \Longrightarrow 0 \leq p\ i$
and $sum\ p\ \{..< n\} = 1$
and $\bigwedge i.\ i \in \{..< n\} \Longrightarrow dim\text{-}vec\ (v\ i) = dimR$
and $\bigwedge i.\ i \in \{..< n\} \Longrightarrow \|v\ i\| = 1$
shows $density\text{-}operator\ (sum\text{-}mat\ (\lambda\ i.\ (p\ i)\ \cdot_m\ (rank\text{-}1\text{-}proj\ (v\ i)))\ \{..< n\})$
⟨*proof*⟩

lemma (**in** *cpx-sq-mat*) *density-operator-mixed-state*:
assumes $R \in$ *fc-mats*
and *density-operator R*
shows $\exists\ p\ v\ (n::nat).\ (\forall\, i \in \{..< n\}.\ 0 \leq p\ i) \wedge$
$(\forall\, i \in \{..< n\}.\ dim\text{-}vec\ (v\ i) = dimR) \wedge$
$(\forall\, i \in \{..< n\}.\ \|v\ i\| = 1) \wedge (sum\ p\ \{..< n\} = 1) \wedge$
$(R = sum\text{-}mat\ (\lambda\ i.\ (p\ i)\ \cdot_m\ (rank\text{-}1\text{-}proj\ (v\ i)))\ \{..< n\})$
⟨*proof*⟩

lemma (**in** *cpx-sq-mat*) *density-operator-iff-mixed-state*:
assumes $R \in$ *fc-mats*
shows *density-operator R* $\longleftrightarrow$
$(\exists\ p\ v\ (n::nat).\ (\forall\, i \in \{..< n\}.\ 0 \leq p\ i) \wedge$
$(\forall\, i \in \{..< n\}.\ dim\text{-}vec\ (v\ i) = dimR) \wedge$
$(\forall\, i \in \{..< n\}.\ \|v\ i\| = 1) \wedge (sum\ p\ \{..< n\} = 1) \wedge$
$(R = sum\text{-}mat\ (\lambda\ i.\ (p\ i)\ \cdot_m\ (rank\text{-}1\text{-}proj\ (v\ i)))\ \{..< n\}))$ (**is** *?L* $\longleftrightarrow$ *?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_is are positive and sum up to 1.

definition *separately-decomposes* **where**
$separately\text{-}decomposes\ R\ (n::nat)\ nA\ nB\ K\ F\ S \equiv$
$(\forall\, a < n.\ (0::complex) \leq (complex\text{-}of\text{-}real\ (K\ a)) \wedge$
$F\ a \in carrier\text{-}mat\ nA\ nA \wedge S\ a \in carrier\text{-}mat\ nB\ nB \wedge$
$density\text{-}operator\ (F\ a) \wedge density\text{-}operator\ (S\ a)) \wedge 0 < nA * nB \wedge$

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

definition *separable-density* **where**
separable-density nA nB R ≡
$\exists\ (n::nat)\ K\ F\ S.$ *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 $R \in$ *carrier-mat* $(nA*nB)$ $(nA*nB)$
⟨*proof*⟩

lemma *separately-decomposes-carrier-pos*:
assumes *separately-decomposes R n nA nB K F S*
shows $0 < nA$ $0 < nB$
⟨*proof*⟩

lemma *separable-density-carrier*:
assumes *separable-density nA nB R*
and $0 < nA$
and $0 < nB$
shows $R \in$ *carrier-mat* $(nA*nB)$ $(nA*nB)$
⟨*proof*⟩

lemma *separately-decomposes-trace*:
assumes *separately-decomposes R n nA nB K F S*
shows *Complex-Matrix.trace R* = *1*
⟨*proof*⟩

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*
⟨*proof*⟩

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* ⟨*proof*⟩

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 ≡ (∃ *v*. *R* = *rank-1-proj v*)

lemma *density-pure-single-diag*:
assumes *A* ∈ *carrier-mat n n*
and *Complex-Matrix.trace A* = (*1*::*real*)
and *Complex-Matrix.trace* (*A*∗*A*) = (*1*::*real*)
and *unitary-diag A B U*
and *I* = {*0* ..< *n*}
and ∀ *i*∈*I*. *A* $$ (*i*,*i*) ≥ *0*
and ∀ *i*∈*I*. *B* $$ (*i*,*i*) ≥ *0*
shows ∃*j*∈*I*. *B* $$ (*j*,*j*) = *1* ∧ (∀ *i* ∈ *I*−{*j*}. *B* $$ (*i*,*i*) = *0*)
⟨*proof*⟩

lemma *rank-1-proj-square-trace*:
fixes *v*::*complex Matrix.vec*
assumes *A* = *rank-1-proj v*
shows *Complex-Matrix.trace* (*A*∗*A*) = ‖*v*‖^2 ∗ *Complex-Matrix.trace A*
⟨*proof*⟩

lemma *rank-1-proj-trace′*:
assumes *Complex-Matrix.trace* (*rank-1-proj v*) = *1*
shows ‖*v*‖ = *1*
⟨*proof*⟩

lemma *density-square-pure*:
assumes *A* ∈ *carrier-mat n n*
and *0* < *n*
and *density-operator A*
and *Complex-Matrix.trace* (*A*∗*A*) = *1*
shows *pure-density-operator A*
⟨*proof*⟩

lemma *density-square-pure′*:
assumes *density-operator A*
and *A* = *rank-1-proj v*
shows *Complex-Matrix.trace* (*A*∗*A*) = *1*
⟨*proof*⟩

lemma
assumes *A*∈ *carrier-mat n n*
and *0* < *n*
and *density-operator A*
shows *pure-density-charact*:
(*pure-density-operator A*) ⟷ (*Complex-Matrix.trace* (*A*∗*A*) = *1*)
and *pure-density-charact′*:
(*pure-density-operator A*) ⟷ (*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 $\mathrm{Tr}(A{\cdot}\rho)$.

definition (**in** *cpx-sq-mat*) *expect-value* **where**
expect-value R p M =
  sum (λ*i*. *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*
    (λ*i*. *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 $2 \cdot \sqrt{2}$. 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 matrics, 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 $|A0 * B1| \leq 1$
  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*⟩

lemma (**in** *bin-cpx*) *Z-XpZ-rho-trace*:
  shows *Complex-Matrix.trace* (*Z-I* ∗ *I-XpZ* ∗ *rho-psim*) = *1*/*sqrt 2*
⟨*proof*⟩

lemma (**in** *bin-cpx*) *X-XpZ-rho-trace*:
  shows *Complex-Matrix.trace* (*X-I* ∗ *I-XpZ* ∗ *rho-psim*) = *1*/*sqrt 2*
⟨*proof*⟩

lemma (**in** *bin-cpx*) *X-ZmX-rho-trace*:
  shows *Complex-Matrix.trace* (*X-I* ∗ *I-ZmX* ∗ *rho-psim*) = *1*/*sqrt 2*
⟨*proof*⟩

lemma (**in** *bin-cpx*) *Z-ZmX-rho-trace*:
shows *Complex-Matrix.trace* (*Z-I* ∗ *I-ZmX* ∗ *rho-psim*) = −*1*/*sqrt 2*
⟨*proof*⟩

9.2 The CHSH operator and expectation

definition *CHSH-op* :: *'a*::*conjugatable-field Matrix.mat* ⇒ *'a Matrix.mat* ⇒
'a Matrix.mat ⇒ *'a Matrix.mat* ⇒ *'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* ⇒ *'a Matrix.mat* ⇒
'a Matrix.mat ⇒ *'a Matrix.mat* ⇒ *'a Matrix.mat* ⇒ *'a*
where
CHSH-expect A0 A1 B0 B1 R = *Complex-Matrix.trace* ((*CHSH-op A0 A1 B0 B1*)
∗ *R*)

definition *CHSH-cond* :: *nat* ⇒ *'a*::*conjugatable-field Matrix.mat* ⇒
'a::*conjugatable-field Matrix.mat* ⇒
'a::*conjugatable-field Matrix.mat* ⇒ *'a*::*conjugatable-field Matrix.mat* ⇒ *bool*
where
CHSH-cond n A0 A1 B0 B1 =
(*A0* ∈ *carrier-mat n n* ∧
A0 ∗ *A0* = 1_m *n* ∧
A1 ∈ *carrier-mat n n* ∧
A1 ∗ *A1* = 1_m *n* ∧
B0 ∈ *carrier-mat n n* ∧
B0 ∗ *B0* = 1_m *n* ∧
B1 ∈ *carrier-mat n n* ∧
B1 ∗ *B1* = 1_m *n* ∧
A0 ∗ *B1* = *B1* ∗ *A0* ∧
A0 ∗ *B0* = *B0* ∗ *A0* ∧
A1 ∗ *B0* = *B0* ∗ *A1* ∧
A1 ∗ *B1* = *B1* ∗ *A1*)

definition *CHSH-cond-hermit* **where**
CHSH-cond-hermit n A0 A1 B0 B1 ⟷ *CHSH-cond n A0 A1 B0 B1* ∧ *hermitian*
A0 ∧
hermitian A1 ∧ *hermitian B0* ∧ *hermitian B1*

lemma *CHSH-op-dim*:
assumes *A0* ∈ *carrier-mat n m*
and *A1* ∈ *carrier-mat n m*
and *B0* ∈ *carrier-mat m p*
and *B1* ∈ *carrier-mat m p*
shows *CHSH-op A0 A1 B0 B1* ∈ *carrier-mat n p* ⟨*proof*⟩

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 $R \in$ *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 $A0 \in$ *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*
and $R \in$ *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 $A0 \in$ *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*
and $R \in$ *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 $A0 \in$ *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*
and $R \in$ *carrier-mat p n*
shows *CHSH-expect A0 A1 B0 B1 R* =
Complex-Matrix.trace $(A0 * B1 * R) -$
Complex-Matrix.trace $(A0 * B0 * R) +$

$Complex\text{-}Matrix.trace\ (A1 * B0 * R) +$
$Complex\text{-}Matrix.trace\ (A1 * B1 * R)$
⟨*proof*⟩

lemma *CHSH-condD*:
assumes *CHSH-cond n A0 A1 B0 B1*
shows $A0 \in carrier\text{-}mat\ n\ n$
$A0 * A0 = 1_m\ n$
$A1 \in carrier\text{-}mat\ n\ n$
$A1 * A1 = 1_m\ n$
$B0 \in carrier\text{-}mat\ n\ n$
$B0 * B0 = 1_m\ n$
$B1 \in carrier\text{-}mat\ n\ n$
$B1 * B1 = 1_m\ n$
$A0 * B1 = B1 * A0$
$A0 * B0 = B0 * A0$
$A1 * B0 = B0 * A1$
$A1 * B1 = B1 * A1$ ⟨*proof*⟩

lemma *CHSH-cond-simps*[*simp*]:
assumes *CHSH-cond n A0 A1 B0 B1*
shows $A1 * B1 * (A0 * B1) = A1 * A0$
$A1 * B1 * (A1 * B0) = B1 * B0$
$A1 * B1 * (A1 * B1) = 1_m\ n$
$A1 * B1 * (A0 * B0) = A1 * A0 * (B1 * B0)$
$A1 * B0 * (A0 * B1) = A1 * A0 * (B0 * B1)$
$A1 * B0 * (A0 * B0) = A1 * A0$
$A1 * B0 * (A1 * B0) = 1_m\ n$
$A1 * B0 * (A1 * B1) = B0 * B1$
$A0 * B0 * (A0 * B1) = B0 * B1$
$A0 * B0 * (A0 * B0) = 1_m\ n$
$A0 * B0 * (A1 * B0) = A0 * A1$
$A0 * B0 * (A1 * B1) = A0 * A1 * (B0 * B1)$
$A0 * B1 * (A0 * B1) = 1_m\ n$
$A0 * B1 * (A0 * B0) = B1 * B0$
$A0 * B1 * (A1 * B0) = A0 * A1 * (B1 * B0)$
$A0 * B1 * (A1 * B1) = A0 * A1$
⟨*proof*⟩

lemma *CHSH-op-square*:
assumes *CHSH-cond n A0 A1 B0 B1*
shows $(CHSH\text{-}op\ A0\ A1\ B0\ B1) * (CHSH\text{-}op\ A0\ A1\ B0\ B1) =$
$(4::nat) \cdot_m (1_m\ n) - (commutator\ A0\ A1) * (commutator\ B0\ B1)$
⟨*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
⟨*proof*⟩

lemma *CHSH-cond-hermit-unitary*:
assumes *CHSH-cond-hermit n A0 A1 B0 B1*
shows *unitary A0 unitary A1 unitary B0 unitary B1*
⟨*proof*⟩

lemma *CHSH-expect-add*:
assumes $A0 \in$ *carrier-mat n n*
and $A1 \in$ *carrier-mat n n*
and $B0 \in$ *carrier-mat n n*
and $B1 \in$ *carrier-mat n n*
and $R0 \in$ *carrier-mat n n*
and $R1 \in$ *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
⟨*proof*⟩

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

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

lemma *CHSH-expect-smult*:
assumes $A0 \in$ *carrier-mat n n*
and $A1 \in$ *carrier-mat n n*
and $B0 \in$ *carrier-mat n n*
and $B1 \in$ *carrier-mat n n*
and $R0 \in$ *carrier-mat n n*
shows *CHSH-expect A0 A1 B0 B1* $(a\ \cdot_m\ R0) =$
$a *$ *CHSH-expect A0 A1 B0 B1 R0*

⟨*proof*⟩

lemma *CHSH-expect-real*:
assumes $0 < n$
and *CHSH-cond-hermit n A0 A1 B0 B1*
and $R \in$ *carrier-mat n n*
and *Complex-Matrix.positive R*
shows *CHSH-expect A0 A1 B0 B1 R* $\in$ *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-spmax-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*) $\leq 2 *$ *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 $R \in$ *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 *lhv M A1 B1 R U1 V1*
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*) =
CHSH-expect A0 A1 B0 B1 R (**is** *?L = ?R*)
⟨*proof*⟩

9.3 CHSH inequality for separable density matrices

definition *CHSH-cond-local* **where**
CHSH-cond-local n m A0 A1 B0 B1 ≡
A0 ∈ *carrier-mat n n* ∧ *A1* ∈ *carrier-mat n n* ∧
B0 ∈ *carrier-mat m m* ∧ *B1* ∈ *carrier-mat m m* ∧
hermitian A0 ∧ *hermitian A1* ∧ *hermitian B0* ∧ *hermitian B1* ∧
*A0 * A0* = 1_m *n* ∧ *A1 * A1* = 1_m *n* ∧ *B0 * B0* = 1_m *m* ∧ *B1 * B1* = 1_m *m*

lemma *CHSH-cond-local-imp-cond-hermit*:
assumes *CHSH-cond-local n m A0 A1 B0 B1*
and *0 < n*
and *0 < m*
shows *CHSH-cond-hermit* (*n*m*) (*A0* ⊗ 1_m *m*) (*A1* ⊗ 1_m *m*)
(*1*$_m$ *n* ⊗ *B0*) (*1*$_m$ *n* ⊗ *B1*)
⟨*proof*⟩

lemma *limit-CHSH-cond*:
shows *CHSH-cond-hermit 4 Z-I X-I I-ZmX I-XpZ*
⟨*proof*⟩

lemma *CHSH-expect-separable-expand*:
assumes *separately-decomposes R n nA nB K F S*
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*⊗ 1_m *nB*) (*A1*⊗ 1_m *nB*) (*1*$_m$ *nA*⊗ *B0*) (*1*$_m$ *nA*⊗ *B1*)
R =
sum (λ*a*. *K a * CHSH-expect* (*A0*⊗ 1_m *nB*) (*A1*⊗ 1_m *nB*) (*1*$_m$ *nA*⊗ *B0*) (*1*$_m$ *nA*⊗ *B1*)
((*F a*) ⊗ (*S a*))) {..< *n*}
⟨*proof*⟩

lemma *CHSH-expect-tensor-leq*:
assumes *CHSH-cond-local nA nB A0 A1 B0 B1*
and *RA* ∈ *carrier-mat nA nA*
and *density-operator RA*
and *RB* ∈ *carrier-mat nB nB*
and *density-operator RB*
and *0 < nA*

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

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* $((CHSH\text{-}op\ A0\ A1\ B0\ B1) * (CHSH\text{-}op\ A0\ A1\ B0\ B1)) = 4$
⟨*proof*⟩

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* $((CHSH\text{-}op\ A0\ A1\ B0\ B1) * (CHSH\text{-}op\ A0\ A1\ B0\ B1)) = 4$
⟨*proof*⟩

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* $(CHSH\text{-}op\ A0\ A1\ B0\ B1) = 2$
⟨*proof*⟩

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 $R \in$ *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 $|CHSH\text{-}expect\ A0\ A1\ B0\ B1\ R| \leq 2$
⟨*proof*⟩

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 $A0 \in$ *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* ⨂ 1_m *nB*) (*A1* ⨂ 1_m *nB*) (1_m *nA* ⨂ *B0*) (1_m *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* = 0_m *n n* ∨ *commutator B0 B1* = 0_m *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