The Gelfand–Naimark–Segal Construction

Richard Schmoetten and Jacques D. Fleuriot

June 25, 2026

Abstract

This entry formalises complete normed algebras equipped with an involution, so-called C^*-algebras. We provide both a class definition, and a locale for C^*-algebras on carrier sets in the spirit of existing developments of linear algebra and smooth manifolds. Bounded operators on a complex Hilbert space, with the operator norm and adjoints, form such an algebra. The main theorem of this entry is a result in the converse direction: the Gelfand–Naimark–Segal (GNS) construction, which starts with a single suitable functional on a C^*-algebra in order to obtain both a Hilbert space and a representation of the algebra in terms of bounded operators on that space. This is implemented as a type construction in Isabelle/HOL, taking advantage of existing mechanisms for quotient types, and integrating with existing type classes for Hilbert spaces and Cauchy completions.

Contents

1 C^*-algebras

theory *Cstar-Algebra-On*
imports
Lie-Groups.Algebra-On
Types-To-Sets-Extension.SML-Topological-Space
Complex-Bounded-Operators.Complex-Vector-Spaces
HOL−Analysis.Abstract-Metric-Spaces
begin

bundle *scaleC-syntax* **begin**
notation *Complex-Vector-Spaces0.scaleC-class.scaleC* (**infixr** ‹$*_C$› *75*)
end

unbundle *no scaleC-syntax*

1.1 Additional lemmas for *Lie-Groups.Algebra-On*

lemma (**in** *algebra-on*)
shows *distR'*: $[\![x{\in}S;\ y{\in}S;\ z{\in}S]\!] \Longrightarrow (x-y) \bullet z = x \bullet z - y \bullet z$
and *distL'*: $[\![x{\in}S;\ y{\in}S;\ z{\in}S]\!] \Longrightarrow z \bullet (x-y) = z \bullet x - z \bullet y$
⟨*proof*⟩

1.2 Involutive rings and algebras

locale *involutive-semigroup* = *semigroup-add-on-with* +
fixes *involution* :: *'a⇒'a* (‹-†› [*99*] *100*)

assumes *involution*[*simp*]: $x{\in}S \Longrightarrow (x^\dagger)^\dagger = x$
and *antiautomorphism*: $[\![x{\in}S;\ y{\in}S]\!] \Longrightarrow (pls\ x\ y)^\dagger = pls\ (y^\dagger)\ (x^\dagger)$
and *involution-mem*[*simp*]: $x{\in}S \Longrightarrow x^\dagger \in S$

— Bundle for class constant notation (e.g. for (+)): disable inside Ballarin's set-based locale for rings, re-enable once we involve type classes for e.g. base fields of algebras.
bundle *class-ring-notation* **begin**
notation *plus* (**infixl** + *65*)
notation *minus* (**infixl** ‹−› *65*)
notation *uminus* (‹(‹*open-block notation*=‹*prefix* −›› − -)› [*81*] *80*)
end

unbundle *no class-ring-notation*
locale *involutive-ring* =
Ring-Theory.ring R addition multiplication zero unit +
involutive-semigroup R multiplication involution
for R :: $'a$ *set* **and** *addition* :: $'a{\Rightarrow}'a{\Rightarrow}'a$ (**infixl** + *65*)
and *multiplication* (**infixl** · *70*) **and** *zero* (**0**) **and** *unit* (**1**)
and *involution* :: $'a{\Rightarrow}'a$ (‹-†› [*99*] *100*) +
assumes *ivl-ring*: $[\![x{\in}R;\ y{\in}R]\!] \Longrightarrow (x{+}y)^\dagger = x^\dagger + y^\dagger$
begin

lemma *unit-self-adjoint*: $\mathbf{1}^\dagger = \mathbf{1}$
⟨*proof*⟩

lemma *ivl-uminus*: $x{\in}R \Longrightarrow$ *involution* $(-x) = -$ *involution* x
⟨*proof*⟩

lemma *ivl-minus*: $[\![x{\in}R;\ y{\in}R]\!] \Longrightarrow$ *involution* $(x{-}y) =$ *involution* $x -$ *involution* y
⟨*proof*⟩

end
unbundle *class-ring-notation*

A general definition of *-algebra might look as follows.

Definition 1 (*-algebra)**.** Let A be a ring with involution *, and R any commutative ring with involution $'$. A is a *-algebra if it is an associative algebra over R, such that:

$$\forall r \in R. \forall a \in A. (ra)^* = r'a^*$$

A *-*homomorphism* $f\colon A \to B$ is an algebra homomorphism that respects involution:

$$\forall a \in A. f(a)^{*_B} = f(a^{*_A})$$

Since nearly all formalisation seems to use classes for the vector addition and base field, and some texts (e.g. [2]) define *-algebras using the complex

numbers as a base ring directly, we do so as well, without worrying about the involution of the base ring being specified only implicitly, on a type-level. Even category theory is, most of the time, only interested in categories of spaces over a shared base field/ring. Note also our involutive complex algebras are unital (and associative).

locale *involutive-complex-algebra* =
assoc-algebra-1-on S scale multiplication unit +
involutive-ring S (+) *multiplication 0 unit involution*
for *S* :: *'a*::*ab-group-add set*
and *scale* :: *complex* ⇒ *'a* ⇒ *'a* (**infixr** ‹$*_C$› *75*)
and *multiplication* :: *'a* ⇒ *'a* ⇒ *'a* (**infixr** ‹•› *74*)
and *unit* :: *'a* (‹**1**›)
and *involution* :: *'a*⇒*'a* (‹-†› [*99*] *100*) +
assumes *ivl-scale*: $s{\in}S \Longrightarrow (c *_C s)^\dagger = cnj\ c *_C s^\dagger$

lemma *involutive-complex-algebraI*:
fixes *S* :: *'a*::*ab-group-add set*
and *scl* :: *complex* ⇒ *'a* ⇒ *'a*
and *mlt* :: *'a* ⇒ *'a* ⇒ *'a*
and *e* :: *'a*
and *involution* :: *'a*⇒*'a*
assumes *assoc-algebra-1-on S scl mlt e*
and *involution*: $\forall x.\ x{\in}S \longrightarrow involution\ (involution\ x) = x$
and *antiautomorphism*: $\forall x\ y.\ x{\in}S \longrightarrow y{\in}S \longrightarrow involution\ (mlt\ x\ y) = mlt\ (involution\ y)\ (involution\ x)$
and *ivl-mem*: $\forall x.\ x{\in}S \longrightarrow involution\ x \in S$
and *ivl-ring*: $\forall x\ y.\ x{\in}S \longrightarrow y{\in}S \longrightarrow involution\ (x{+}y) = (involution\ x) + (involution\ y)$
and *ivl-scale*: $\forall x\ c.\ x{\in}S \longrightarrow involution\ (scl\ c\ x) = scl\ (cnj\ c)\ (involution\ x)$
shows *involutive-complex-algebra S scl mlt e involution*
⟨*proof*⟩

lemma (**in** *module-on*) *sub-module*:
assumes *subspace A* $A{\subseteq}S$
shows *module-on A scale*
⟨*proof*⟩

locale *involutive-subalgebra* =
subalg: *involutive-complex-algebra A* + *involutive-complex-algebra B* **for** *A B* +
assumes $A \subseteq B$

lemma (**in** *involutive-complex-algebra*) *subalgebraI*:
fixes *A*
assumes *unital-subspace*: *m1.subspace A* $A \subseteq S$ $\mathbf{1} \in A$
and *mult-closed*: $\forall x\ y.\ x \in A \longrightarrow y \in A \longrightarrow x \bullet y \in A$
and *involution-closed*: $\forall x.\ x \in A \longrightarrow involution\ x \in A$
shows *involutive-subalgebra* ($*_C$) (•) **1** *involution A S*

⟨*proof*⟩

1.3 Preliminaries and experiments

context *Metric-space* **begin**

sublocale *topological-space-ow M mopen*
⟨*proof*⟩

end

1.4 Subfields of a *field* on a type universe

Compare to *subspace* in *HOL−Types-To-Sets.Linear-Algebra-On*.

abbreviation *closed-under-un K op* ≡ ∀ *x*∈*K*. *op x* ∈ *K*
abbreviation *closed-under-bin K op* ≡ ∀ *x*∈*K*. ∀ *y*∈*K*. *op x y* ∈ *K*

locale *subfield* =
fixes *K* :: *'a*::*field set*
assumes *subfield-1*[*simp*]: *1* ∈ *K*
and *subfield-closed*[*simp*]:
x∈*K* ⟹ − *x* ∈ *K*
x∈*K* ⟹ *inverse x* ∈ *K*
⟦*x*∈*K*; *y*∈*K*⟧ ⟹ *x* + *y* ∈ *K*
⟦*x*∈*K*; *y*∈*K*⟧ ⟹ *x* ∗ *y* ∈ *K*
begin

— The zero element of a field coincides with the zero element of any of its subfields. Since *0* is a class constant, it is always the zero element of the field on the type universe.
lemma *subfield-0* [*simp*]: *0* ∈ *K*
⟨*proof*⟩

end

— Any field is a subfield of itself.
lemma *subfield-UNIV*: *subfield UNIV*
⟨*proof*⟩
lemma *real-subfield-complex*: *subfield* {*c*. *Im c* = *0*}
⟨*proof*⟩

1.4.1 ... as a transfer relation?

The price of being able to talk about a subfield of a different type is representational definiteness. The subfield of type 'a is only a subfield "up to isomorphism". Quite categorical in flavour, this should make no practical difference. See also skeletal categories: https://ncatlab.org/nlab/show/skeleton

context includes *lifting-syntax* **begin**

definition *subfield-rel* (R :: *'a*::*field* ⇒ *'b*::*field* ⇒ *bool*) ≡
left-total R ∧ *bi-unique R* ∧ ~~*right-unique R* ∧ *right-total R*~~ ∧
R 1 1 ∧
(*R* ===> *R* ===> *R*) (+) (+) ∧
(*R* ===> *R* ===> *R*) (∗) (∗) ∧
(*R* ===> *R*) *uminus uminus* ∧
(*R* ===> *R*) *inverse inverse*

lemma *subfield-rel-0*: *R 0 0* **if** *subfield-rel R*
⟨*proof*⟩

lemma *subfield-rel-UNIV*: *subfield-rel* (=)
⟨*proof*⟩

lemma *real-subfield-rel-complex*: *subfield-rel* (λ*r c*. *complex-of-real r* = *c*)
⟨*proof*⟩

end

1.5 Topological Fields and Vector Spaces

The topology will be defined, as usual for locales, on a carrier set only. Here, that carrier is implicit as the union of all open sets, like for manifolds. This means e.g. addition is defined on the type universe, but is continuous only on the subset of that universe covered by the topology.

abbreviation *continuous-binary-op t f* ≡ *continuous-map* (*prod-topology t t*) *t* (λ(*a*,*b*). *f a b*)
locale *topological-field-ow* = *topological-space-ow F* τ
for *F* :: *'at*::*field set*
and τ :: *'at set* ⇒ *bool* +
assumes *continuous-plus*: *continuous-binary-op* (*topology* τ) (+)
and *continuous-times*: *continuous-binary-op* (*topology* τ) (∗)
and *continuous-uminus*: *continuous-map* (*topology* τ) (*topology* τ) *uminus*
and *continuous-inverse*: *continuous-map* (*topology* τ) (*topology* τ) *inverse*
begin

A somewhat Frankensteinian construction. The locale *topological-space-ow* is taken from the ETTS-powered *Types-To-Sets-Extension.SML-Topological-Space*, and transferred from *topological-space*. By contrast, *continuous-map* stems from *HOL−Analysis.Abstract-Topology*. The field operations are still class constants, similarly to *vector-space-on*, with topological properties on the carrier *F* only.

end

locale *topological-vector-space-on* =

vector-space-on V scale +
topological-space-ow V τ_V +
top-F: *topological-field-ow F* τ_F
for F :: $'a$::*field set* **and** τ_F :: $'a$ *set* ⇒ *bool*
and V :: $'b$::*ab-group-add set*
and *scale* :: $'a$ ⇒ $'b$ ⇒ $'b$ (**infixr** ∗*s* *75*)
and τ_V :: $'b$ *set* ⇒ *bool* +
assumes *continuous-add*: *continuous-binary-op* (*topology* τ_V) (+)
and *continuous-scale*: *continuous-map* (*prod-topology* (*topology* τ_F) (*topology* τ_V)) (*topology* τ_V) (λ(*a*,*v*). *scale a v*)

1.6 Normed Fields and Vector Spaces

locale *normed-field* = *subfield F* **for** F :: $'a$::*field set* +
fixes *norm* :: $'a$⇒*real* (||-|| *100*)
assumes *normed-field*: $x \in F \Longrightarrow ||x|| \geq 0$
$x \in F \Longrightarrow ||x|| = 0 \longleftrightarrow x = 0$
$x \in F \Longrightarrow ||-x|| = ||x||$
$[\![x \in F;\ y \in F]\!] \Longrightarrow ||x+y|| \leq ||x||+||y||$
$[\![x \in F;\ y \in F]\!] \Longrightarrow ||x*y|| \leq ||x||*||y||$

locale *valued-field* = *normed-field* +
assumes *multiplicative-norm*: $[\![x \in F;\ y \in F]\!] \Longrightarrow ||x*y|| = ||x||*||y||$
begin

lemma *norm-1* [*simp*]: $||1|| = 1$
⟨*proof*⟩

end

locale *seminormed-vector-space-on* =
vector-space-on V scale +
F: *valued-field F field-abs*
for F :: $'f$::*field set* **and** *field-abs* :: $'f$⇒*real* (||-|| [*80*])
and V :: $'v$::*ab-group-add set* **and** *scale* :: $'f$⇒$'v$⇒$'v$ +
fixes *vector-norm* :: $'v$::*ab-group-add* ⇒ *real* (||-|| [*65*]) — At priority 65, you can write sums inside the norm.
assumes *triangle-add*: $[\![x \in V;\ y \in V]\!] \Longrightarrow$
vector-norm (*x*+*y*) ≤ *vector-norm x* + *vector-norm y*
and *absolute-homogeneous*: $[\![a \in F;\ x \in V]\!] \Longrightarrow$
vector-norm (*scale a x*) = (*field-abs a*) ∗ (*vector-norm x*)
begin

lemma *norm-0*: *vector-norm 0* = *0*
⟨*proof*⟩

lemma *norm-uminus*: $||-x|| = ||x||$ **if** *x*: $x \in V$
⟨*proof*⟩

lemma *norm-nonneg* [*simp*]: $x \in V \Longrightarrow ||x|| \geq 0$
⟨*proof*⟩

end

locale *normed-vector-space-on* = *seminormed-vector-space-on* +
assumes *positive-definite*: $[\![x \in V;\ \textit{vector-norm}\ x = 0]\!] \Longrightarrow x = 0$
begin

lemma *norm-0-iff*: *vector-norm* $x = 0 \longleftrightarrow x = 0$ **if** $x \in V$
⟨*proof*⟩

abbreviation (*input*) *induced-metric*
where *induced-metric* $x\ y \equiv ||y - x||$

The locale *Metric-space*, which we would like to instantiate, requires things like $0 \leq ||y - x||$. This is not true in general: $?x \in V \Longrightarrow 0 \leq ||?x||$ only guarantees this behaviour inside the carrier set. Thus we have to "totalise" the induced metric for use in the existing *Metric-space*, by defining behaviour outside the carrier set to satisfy the axioms of the locale.

definition *totalized-induced-metric* (d)

where d $x\ y \equiv$ *if* $x \in V \wedge y \in V$ *then induced-metric* $x\ y$ *else* 0

lemma *induced-metric-simps* [*simp*]:
$[\![x \in V;\ y \in V]\!] \Longrightarrow$ d $x\ y =$ *induced-metric* $x\ y$
$x \notin V \Longrightarrow$ d $x\ y = 0$
$y \notin V \Longrightarrow$ d $x\ y = 0$
⟨*proof*⟩

lemma *metric-translation-invariant*: $[\![x \in V;\ y \in V;\ z \in V]\!] \Longrightarrow$ d $x\ y =$ d $(x+z)$ $(y+z)$
⟨*proof*⟩
lemma *metric-translation-invariant'*: $[\![x \in V;\ y \in V]\!] \Longrightarrow$ d $x\ y =$ d $(x-y)\ 0$
⟨*proof*⟩

sublocale *Metric-space* V d
⟨*proof*⟩

end

lemma *totalized-induced-metric-eq-dist*:
normed-vector-space-on.totalized-induced-metric UNIV norm = *dist*
⟨*proof*⟩

term ‹x::$'a$::*real-normed-vector*›

locale *banach-space-on* = *normed-vector-space-on* +
  assumes *mcomplete*

thm *class.cbanach-def*
lemma *cbanach-is-banach-space-on*:
  shows *banach-space-on UNIV cmod* (*UNIV*::$'cb$::*cbanach set*) *scaleC norm*
  ⟨*proof*⟩

1.7 C* algebras

locale *Cstar-algebra* =
    banach-space-on F cmod V scale vector-norm +

    involutive-complex-algebra V scale multiplication unit involution
  for *F* :: *complex set*
    and *V* :: $'v$::*ab-group-add set*
    and *scale* :: *complex* ⇒ $'v$ ⇒ $'v$ (**infixr** ‹$*_C$› *75*)
    and *multiplication* :: $'v$ ⇒ $'v$ ⇒ $'v$ (**infixr** ‹•› *74*)
    and *vector-norm* :: $'v$ ⇒ *real* (‹||-||› [*65*])
    and *unit* :: $'v$ (‹**1**›)
    and *involution* :: $'v$⇒$'v$ (‹-†› [*99*] *100*) +
  assumes *normed-algebra* : ⋀*A B*. ⟦*A*∈*V*; *B*∈*V*⟧ ⟹ ||*A*•*B*|| ≤ ||*A*|| ∗ ||*B*||
  assumes *normed-unital*: ||**1**|| = *1*
  assumes *involution-isometric*: ⋀*A*. *A*∈*V* ⟹ ||$A^\dagger$|| = ||*A*||
  assumes *Cstar*: ⋀*A*. *A*∈*V* ⟹ ||($A^\dagger$) • *A*|| = ||$A^\dagger$|| ∗ ||*A*||
begin

lemma *Bstar*: ||($A^\dagger$) • *A*|| = ||*A*||^2 **if** *A*∈*V* **for** *A*
  ⟨*proof*⟩

end

lemma *Bstar-implies-isometric*:
  assumes *seminormed-vector-space-on F cmod V scale vector-norm*
    involutive-complex-algebra V scale multiplication unit involution
    and ⋀*A B*. ⟦*A*∈*V*; *B*∈*V*⟧ ⟹ *vector-norm* (*multiplication A B*) ≤ *vector-norm A* ∗ *vector-norm B vector-norm unit* = *1*
    and ⋀*A*. *A*∈*V* ⟹ *vector-norm* (*multiplication* (*involution A*) *A*) = (*vector-norm A*)^2
    and *A*∈*V*
  shows *vector-norm* (*involution A*) = *vector-norm A*
⟨*proof*⟩

locale *Cstar-subalgebra* =
subalg: *Cstar-algebra* - *A* + *Cstar-algebra* - *B* **for** *A B* +
assumes $A \subseteq B$

lemma (**in** *banach-space-on*) *closed-iff-complete*:
assumes $A \subseteq V$
shows *closedin mtopology A* $\longleftrightarrow$ *Metric-space.mcomplete A* d
⟨*proof*⟩

lemma (**in** *normed-vector-space-on*) *from-subspace*:
assumes *subspace A* $A \subseteq V$
shows *normed-vector-space-on F field-abs A scale vector-norm*
⟨*proof*⟩

Not particularly elegantly or generally written: it doesn't matter which (induced) metric we use, because all cases outside of the subspace carrier A don't matter. Used for C*-subalgebra intro lemma.

lemma (**in** *normed-vector-space-on*) *mcomplete-submetric-equal*:
assumes $A \subseteq V$ *subspace A*
shows *Metric-space.mcomplete A* (*normed-vector-space-on.totalized-induced-metric A vector-norm*) = *Metric-space.mcomplete A* d
⟨*proof*⟩

lemma (**in** *banach-space-on*) *closed-iff-complete′*:
defines *complete-in-induced-submetric* ≡
$\lambda A\ n$. *Metric-space.mcomplete A* (*normed-vector-space-on.totalized-induced-metric A n*)
assumes $A \subseteq V$ *subspace A*
shows *closedin mtopology A* $\longleftrightarrow$ *complete-in-induced-submetric A vector-norm*
⟨*proof*⟩

lemma (**in** *Cstar-algebra*) *subalgebraI*:
fixes *A*
assumes *unital-subalgebra*: *subspace A* $A \subseteq V$ $\mathbf{1} \in A$
and *mult-closed*: $\forall x\ y.\ x \in A \longrightarrow y \in A \longrightarrow x \bullet y \in A$
and *involution-closed*: $\forall x.\ x \in A \longrightarrow$ *involution* $x \in A$
and *closed-subset*: *closedin mtopology A*
shows *Cstar-subalgebra F scale* (•) *vector-norm* **1** *involution A V*
⟨*proof*⟩

lemma (**in** *Cstar-algebra*) *subalgebraI′*:
fixes *A*
assumes *unital-subalgebra*: *subspace A* $A \subseteq V$ $\mathbf{1} \in A$
and *mult-closed*: $\forall x\ y.\ x \in A \longrightarrow y \in A \longrightarrow x \bullet y \in A$
and *involution-closed*: $\forall x.\ x \in A \longrightarrow$ *involution* $x \in A$
and *complete-subset*: *Metric-space.mcomplete A* d
shows *Cstar-subalgebra F scale* (•) *vector-norm* **1** *involution A V*
⟨*proof*⟩

1.8 Cauchy-Schwarz Inequality for functionals on involutive algebras

context *involutive-complex-algebra* **begin**

Disable the notation from `Jacobson_Basic_Algebra` that conflicts with underlying class constants used for our algebras.

no-notation *additive.inverse* (− - [66] 65) — conflicts with *uminus*
no-notation *subtraction* (**infixl** − 65) — conflicts with (−)

lemma *additive-inverse-uminus*[*simp*]:
additive.inverse A = − *A* **if** *A*∈*S*
⟨*proof*⟩

lemma *subtraction-minus*[*simp*]:
subtraction A B = *A* − *B* **if** *A*∈*S B*∈*S*
⟨*proof*⟩

declare *ivl-uminus* [[*simp del*]]
lemma *involution-uminus'*[*simp*]: *involution* (− *B*) = − (*involution B*) **if** *B*∈*S* **for** *B*
⟨*proof*⟩

declare *ivl-minus* [[*simp del*]]
lemma *ivl-minus'*[*simp*]: *involution* (*A* − *B*) = *involution A* − (*involution B*)
if *A*∈*S B*∈*S* **for** *B*
⟨*proof*⟩

end

The nomenclature of "functional" is taken from operator algebras, where operators are often functions. Here "positive" is not strict: the corresponding axiom is named after non-negativity.

locale *positive-normalised-linear-functional* =
involutive-complex-algebra S scale multiplication unit involution +
linear-on S UNIV scale (∗) *ω*
for *S* :: *'a*::*ab-group-add set*
and *scale* :: *complex* ⇒ *'a* ⇒ *'a* (**infixr** ‹∗$_C$› 75)
and *multiplication* :: *'a* ⇒ *'a* ⇒ *'a* (**infixr** ‹•› 74)
and *unit* :: *'a* (‹**1**›)
and *involution* :: *'a*⇒*'a* (‹-†› 73)
and *ω* :: *'a*⇒*complex* +
assumes *nonneg*: *A*∈*S* ⟹ *ω* ((*A*†) • *A*) ≥ 0
and *one* [*simp*]: *ω* **1** = 1
begin

notation *cmod* (‖-‖) — *abs* has omitted priority

lemma *nonneg'*:
assumes $A \in S$
shows *Re* $(\omega\ (\mathit{involution}\ A \bullet A)) \geq 0$ *Re* $(\omega\ (A \bullet \mathit{involution}\ A)) \geq 0$ *Im* $(\omega\ (A \bullet \mathit{involution}\ A)) = 0$ *Im* $(\omega\ (\mathit{involution}\ A \bullet A)) = 0$
⟨*proof*⟩

lemma *involution-conjugate*: $\omega\ (\mathit{involution}\ A) = \mathit{cnj}\ (\omega\ A)$ **if** $A \in S$
⟨*proof*⟩

lemma *cnj-ivl*: *cnj* $(\omega\ (\mathit{involution}\ A)) = \omega\ (A)$ **if** $A \in S$
⟨*proof*⟩

Any such functional can be used to define a sesquilinear form that acts as a generalised inner product. This interpretation is justified below by providing a Cauchy-Schwarz inequality.

definition *functional-inner* :: $'a \Rightarrow 'a \Rightarrow \mathit{complex}$ (⟨-|-⟩ *76*)
where *functional-inner* $A\ B \equiv \omega\ ((A^\dagger) \bullet B)$

named-theorems *finner-simps*

lemma *finner-nonneg* [*finner-simps*]:
assumes $A \in S$
shows $\langle A|A\rangle \geq 0$
⟨*proof*⟩

lemma *finner-scale* [*finner-simps*]:
$\langle c *_C A|B\rangle = (\mathit{cnj}\ c) * \langle A|B\rangle$
$\langle A|c *_C B\rangle = c * \langle A|B\rangle$
if $A{\in}S\ B{\in}S$ **for** c **and** $A\ B$
⟨*proof*⟩

lemma *finner-plus* [*finner-simps*]: $\langle A{+}B|C\rangle = \langle A|C\rangle + \langle B|C\rangle$ $\langle A|B{+}C\rangle = \langle A|B\rangle + \langle A|C\rangle$
if $A{\in}S\ B{\in}S\ C{\in}S$ **for** $A\ B\ C$
⟨*proof*⟩

lemma *finner-uminus* [*finner-simps*]: $\langle \mathit{uminus}\ A|B\rangle = \mathit{uminus}\ (\langle A|B\rangle)$ $\langle A|\mathit{uminus}\ B\rangle = \mathit{uminus}\ (\langle A|B\rangle)$
if $A{\in}S\ B{\in}S$ **for** c **and** $A\ B$
⟨*proof*⟩

lemma *finner-minus* [*finner-simps*]: $\langle A{-}B|C\rangle = \langle A|C\rangle - \langle B|C\rangle$ $\langle A|B{-}C\rangle = \langle A|B\rangle - \langle A|C\rangle$
if $A{\in}S\ B{\in}S\ C{\in}S$ **for** c **and** $A\ B\ C$
⟨*proof*⟩

lemma *finner-cnj*:$\langle A|B\rangle = \mathit{cnj}\ (\langle B|A\rangle)$

if $A{\in}S$ $B{\in}S$ **for** A B
⟨*proof*⟩

lemma *finner-real-commute*:
assumes $A{\in}S$ $B{\in}S$ $\langle A|B\rangle \in \mathbb{R}$
shows $\langle A|B\rangle = \langle B|A\rangle$
⟨*proof*⟩

lemma *finner-zero-commute*:
assumes $A{\in}S$ $B{\in}S$ $\langle A|B\rangle = 0$
shows $\langle B|A\rangle = 0$
⟨*proof*⟩

lemma *finner-self-0-if*: $\langle A|A\rangle = 0$ **if** $A = 0$
⟨*proof*⟩

lemma *pythagorean-theorem*:
assumes $A{\in}S$ $B{\in}S$ $\langle A|B\rangle = 0$
shows $\langle A{+}B|A{+}B\rangle = \langle A|A\rangle + \langle B|B\rangle$
⟨*proof*⟩

lemma *finner-csqrt-nonneg*: $csqrt\ (\langle X|X\rangle) \geq 0$ **if** $X{\in}S$
⟨*proof*⟩

lemma *finner-cmod-eq*: $\|\langle B|B\rangle\| = \langle B|B\rangle$ **if** $B{\in}S$
⟨*proof*⟩

definition *functional-norm* :: $'a \Rightarrow real$ $(||\text{-}||_\omega)$
where $functional\text{-}norm\ X \equiv Re\ (csqrt\ (\langle X|X\rangle))$

lemma *fnorm-finner-squared*: $\langle B|B\rangle = ||B||_\omega{}^2$ $\|\langle B|B\rangle\| = ||B||_\omega{}^2$ **if** $B{\in}S$
⟨*proof*⟩

lemma *fnorm-nonneg*: $||X||_\omega \geq 0$
⟨*proof*⟩

lemma *finner-lindep*:
fixes c::*complex*
assumes $B{\in}S$
shows $\|\langle c *_C B|B\rangle\| = \|c\| * ||B||_\omega{}^2$
⟨*proof*⟩

lemma *fnorm-scale*: $||c *_C X||_\omega = \|c\| * ||X||_\omega$ **if** $X{\in}S$ **for** c::*complex*
⟨*proof*⟩

lemma *finner-lindep-cong*:
assumes $B{\in}S$ $A = c *_C B$
shows $\|\langle A|B\rangle\| = ||A||_\omega * ||B||_\omega$
⟨*proof*⟩

context — Controlling notation for partitions/division. **begin**
no-notation *equivalence.Partition* (**infixl** ‹'/› *75*)

— Stolen from *Polygonal-Number-Theorem.Polygonal-Number-Theorem-Lemmas.qua-disc* on the AFP. The import brought in too many dependencies to be worth it.
lemma *qua-disc*:
fixes *a b c* :: *real*
assumes *a>0*
assumes ∀ *x*::*real*. *a*∗*x*^2+*b*∗*x* + *c* ≥ *0*
shows *b*^2 − *4* ∗ *a* ∗ *c* ≤ *0*
⟨*proof*⟩

lemma *sign-of-discriminant*:
fixes *a b c* :: *real*
assumes ∀ *x*::*real*. *a*∗*x*^2 + *b*∗*x* + *c* ≥*0*
shows *a* ≥ *0* ⟹ *b*^2 − *4* ∗ *a* ∗ *c* ≤ *0* *a* < *0* ⟹ *b*^2 − *4* ∗ *a* ∗ *c* ≥ *0*
⟨*proof*⟩

end

lemma *cauchy-schwarz-square*:
assumes *A*∈*S* *B*∈*S*
shows |⟨*A*|*B*⟩|^2 ≤ ⟨*A*|*A*⟩ ∗ ⟨*B*|*B*⟩
⟨*proof*⟩

lemma *cauchy-schwarz*:
assumes *A*∈*S* *B*∈*S*
shows ‖⟨*A*|*B*⟩‖ ≤ ||*A*||$_\omega$ ∗ ||*B*||$_\omega$
⟨*proof*⟩

1.9 Identities for states on unital C^*-algebras

For now, see [1, page 49, Prop. 2.3.11]. We already have *?A* ∈ *S* ⟹ *ω* (*?A*†) = *cnj* (*ω* *?A*).

lemma *state-norm-leq-inner*: ‖*ω* *A*‖^2 ≤ ⟨*A*|*A*⟩ **if** *A*∈*S*
⟨*proof*⟩

lemma *state-zero-if-inner-zero*: *ω* *A* = *0* **if** *ω* (*involution A* • *A*) = *0* *A*∈*S*
⟨*proof*⟩

end

locale *Cstar-state* =
Cstar-algebra F S scale multiplication vector-norm unit involution +
positive-normalised-linear-functional S scale multiplication unit involution ω

for F :: *complex set*
and S :: $'a$::*ab-group-add set*
and *scale* :: *complex* ⇒ $'a$ ⇒ $'a$ (**infixr** ‹$*_C$› *75*)
and *multiplication* :: $'a$ ⇒ $'a$ ⇒ $'a$ (**infixr** ‹•› *74*)
and *vector-norm* :: $'a$ ⇒ *real* (‹||-||› [*65*])
and *unit* :: $'a$ (‹**1**›)
and *involution* :: $'a$⇒$'a$ (-† [*99*] *100*)
and ω :: $'a$⇒*complex*
begin

lemma *mult-inner-norm-bound*:
assumes $\bigwedge A\ B.\ [\![A{\in}S;\ B{\in}S]\!] \Longrightarrow \langle B \bullet A|B \bullet A\rangle \leq ||B||^2 * \omega\ (A^\dagger \bullet A)$
— This assumption is, in fact, a theorem. It can be proven using spectral theory, by defining positive operators to enable the following ordering relation:

$$A^\dagger B^\dagger B A \leq \|B\|^2 A^\dagger A$$

See Bratteli and Robinson's textbook, p. 51 for proof of the lemma, and p. 36 for ordering on positive operators [1]. The definition of an operator as positive depends on the operator's spectrum (i.e. it is in the positive half-line, p. 32). Spectral theory is, for now, out of scope for this development.
and $A{\in}S$ $B{\in}S$
shows $|\langle B \bullet A|A\rangle| \leq \langle A|A\rangle * ||B||$
⟨*proof*⟩

end

1.10 Morphisms

locale *involutive-complex-algebras* =
A1: *involutive-complex-algebra S1 scale1 multiplication1 unit1 involution1* +
A2: *involutive-complex-algebra S2 scale2 multiplication2 unit2 involution2*
for *S1* :: $'a$::*ab-group-add set*
and *scale1* :: *complex* ⇒ $'a$ ⇒ $'a$ (**infixr** ‹$*_{C1}$› *75*)
and *multiplication1* :: $'a$ ⇒ $'a$ ⇒ $'a$ (**infixr** ‹•$_1$› *74*)
and *unit1* :: $'a$ (‹**1**$_1$›)
and *involution1* :: $'a$⇒$'a$
and *S2* :: $'b$::*ab-group-add set*
and *scale2* :: *complex* ⇒ $'b$ ⇒ $'b$ (**infixr** ‹$*_{C2}$› *75*)
and *multiplication2* :: $'b$ ⇒ $'b$ ⇒ $'b$ (**infixr** ‹•$_2$› *74*)
and *unit2* :: $'b$ (‹**1**$_2$›)
and *involution2* :: $'b$⇒$'b$

We define *-homomorphisms to be unital always. I believe a minority of authors consider non-unital homomorphisms of rings, but then you need to consider degenerate cases (e.g. $\lambda x.\ 0$). For some discussion, see e.g. https://mathoverflow.net/questions/34332/consequences-of-not-requiring-ring-homomorphisms-to-be-unital .

locale *involutive-complex-homomorphism* = *involutive-complex-algebras* +

fixes $f :: 'a \Rightarrow 'b$
assumes *f-codomain*: $f`S1 \subseteq S2$
and *f-unit*: $f\ \mathbf{1}_1 = \mathbf{1}_2$
and *f-scale*: $\bigwedge r\ x.\ [\![x \in S1]\!] \Longrightarrow f\ (r *_{C1} x) = r *_{C2} (f\ x)$
and *f-plus*: $\bigwedge x\ y.\ [\![x \in S1;\ y \in S1]\!] \Longrightarrow f\ (x + y) = (f\ x) + (f\ y)$
and *f-multiplication*: $\bigwedge x\ y.\ [\![x \in S1;\ y \in S1]\!] \Longrightarrow$
$f\ (x \bullet_1 y) = (f\ x) \bullet_2 (f\ y)$
and *f-involution*: $\bigwedge x.\ [\![x \in S1]\!] \Longrightarrow$
$f\ (involution1\ x) = involution2\ (f\ x)$
begin

lemma *linear-f*: *linear-on S1 S2 scale1 scale2 f*
⟨*proof*⟩

interpretation *linear-f*: *linear-on S1 S2 scale1 scale2 f*
⟨*proof*⟩

end

abbreviation (**in** *involutive-complex-algebras*) *homomorphism*
where *homomorphism f* ≡ *involutive-complex-homomorphism*
S1 scale1 multiplication1 unit1 involution1
S2 scale2 multiplication2 unit2 involution2
f

locale *involutive-complex-isomorphism* = *involutive-complex-homomorphism* +
assumes *bij-f*: *bij-betw f S1 S2*

abbreviation (**in** *involutive-complex-algebras*) *isomorphism*
where *isomorphism f* ≡ *involutive-complex-isomorphism*
S1 scale1 multiplication1 unit1 involution1
S2 scale2 multiplication2 unit2 involution2
f

locale *involutive-complex-automorphism* =
involutive-complex-isomorphism S scale multiplication unit involution
S scale multiplication unit involution f
for *S scale multiplication unit involution f*

abbreviation (**in** *involutive-complex-algebra*) *automorphism*
where *automorphism f* ≡ *involutive-complex-automorphism S scale multiplication unit involution f*

locale *Cstar-algebras* =
A1: *Cstar-algebra F1 S1 scale1 multiplication1 vnorm1 unit1 involution1* +
A2: *Cstar-algebra F2 S2 scale2 multiplication2 vnorm2 unit2 involution2*

for *F1* :: *complex set*
and *S1* :: *'a::ab-group-add set*
and *scale1* :: *complex* ⇒ *'a* ⇒ *'a* (**infixr** ‹$*_{C1}$› *75*)
and *multiplication1* :: *'a* ⇒ *'a* ⇒ *'a* (**infixr** ‹$\bullet_1$› *74*)
and *vnorm1* :: *'a* ⇒ *real* (‹||-||$_1$› [*65*])
and *unit1* :: *'a* (‹$\mathbf{1}_1$›)
and *involution1* :: *'a⇒'a*
and *F2* :: *complex set*
and *S2* :: *'b::ab-group-add set*
and *scale2* :: *complex* ⇒ *'b* ⇒ *'b* (**infixr** ‹$*_{C2}$› *75*)
and *multiplication2* :: *'b* ⇒ *'b* ⇒ *'b* (**infixr** ‹$\bullet_2$› *74*)
and *vnorm2* :: *'b* ⇒ *real* (‹||-||$_2$› [*65*])
and *unit2* :: *'b* (‹$\mathbf{1}_2$›)
and *involution2* :: *'b⇒'b*
begin

sublocale *involutive-complex-algebras*
⟨*proof*⟩

end

A C^*-homomorphism is a homomorphism of *-algebras between C^*-algebras.

locale *Cstar-homomorphism* = *Cstar-algebras* + *involutive-complex-homomorphism*

lemma (**in** *Cstar-algebras*) *homomorphismI*:
assumes *homomorphism f*
shows *Cstar-homomorphism F1 S1 scale1 multiplication1 vnorm1 unit1 involution1*
F2 S2 scale2 multiplication2 vnorm2 unit2 involution2 f
⟨*proof*⟩

locale *Cstar-isomorphism* = *Cstar-algebras* + *involutive-complex-isomorphism*

locale *Cstar-automorphism* =
Cstar-isomorphism F V scale multiplication vector-norm unit involution
F V scale multiplication vector-norm unit involution f
for *F* :: *complex set*
and *V* :: *'v::ab-group-add set*
and *scale* :: *complex* ⇒ *'v* ⇒ *'v* (**infixr** ‹$*_C$› *75*)
and *multiplication* :: *'v* ⇒ *'v* ⇒ *'v* (**infixr** ‹$\bullet$› *74*)
and *vector-norm* :: *'v* ⇒ *real* (‹||-||› [*65*])
and *unit* :: *'v* (‹$\mathbf{1}$›)
and *involution* :: *'v⇒'v* (‹-†› *73*)
and *f*
begin

lemma *is-star-aut*: *involutive-complex-automorphism V* $(*_C)$ $(\bullet)$ **1** *involution f*
⟨*proof*⟩

sublocale *as-star-aut*: *involutive-complex-automorphism V*
⟨*proof*⟩

end

lemma (**in** *Cstar-algebra*) *automorphismI*:
assumes *automorphism f*
shows *Cstar-automorphism F V scale multiplication vector-norm unit involution f*
⟨*proof*⟩

end

2 Completions of Normed Vector Spaces

theory *Inner-Completion*
imports
Complex-Bounded-Operators.Complex-Inner-Product
Gromov-Hyperbolicity.Metric-Completion
begin

2.1 Helper lemmas

lemma *Cauchy-norm-bounded*:
assumes *Cauchy X*
shows $\exists B.\ \forall n.\ norm\ (X\ n) < B$
⟨*proof*⟩

lemma *convergent-Cauchy-norm*:
fixes *u v*::*nat* ⇒ (*'a*::*real-normed-vector*)
assumes *Cauchy u*
shows *convergent* $(\lambda n.\ norm\ (u\ n))$
⟨*proof*⟩

lemma *Cauchy-norm-if-Cauchy*:
assumes *Cauchy u*
shows *Cauchy* $(\lambda n.\ norm\ (u\ n))$
⟨*proof*⟩

lemma *lim-ge*: *convergent f* $\Longrightarrow$ $(\bigwedge n.\ f\ n \geq x)$ $\Longrightarrow$ *lim f* $\geq x$
for *x* :: *'a*::*linorder-topology*
⟨*proof*⟩

lemma

assumes *convergent u*
shows *lim-Re*: *lim* (λ*n*. *Re* (*u n*)) = *Re* (*lim* (λ*n*. *u n*))
and *lim-Im*: *lim* (λ*n*. *Im* (*u n*)) = *Im* (*lim* (λ*n*. *u n*))
⟨*proof*⟩

lemma *Cauchy-sum*:
fixes *x1 x2* :: *nat*⇒'*a*::*real-normed-vector*
assumes *Cauchy x1 Cauchy x2*
shows *Cauchy* (*x1*+*x2*)
⟨*proof*⟩

2.2 The Cauchy completion of a normed vector space is a Banach space.

instantiation *metric-completion* :: (*real-normed-vector*) *scaleR*
begin
lift-definition *scaleR-metric-completion* :: *real* ⇒ '*a metric-completion* ⇒ '*a metric-completion*
is λ*r x*. λ*n*. (*scaleR r*) (*x n*)
⟨*proof*⟩
instance ⟨*proof*⟩
end

instantiation *metric-completion* :: (*real-normed-vector*) *real-vector*
begin

lift-definition *uminus-metric-completion* :: '*a metric-completion* ⇒ '*a metric-completion*
is *uminus*
⟨*proof*⟩

lift-definition *zero-metric-completion* :: '*a metric-completion*
is λ*n*. *0*
⟨*proof*⟩

lift-definition *plus-metric-completion* ::
'*a metric-completion* ⇒ '*a metric-completion* ⇒ '*a metric-completion*
is *plus*
⟨*proof*⟩

definition *minus-metric-completion* ::
'*a metric-completion* ⇒ '*a metric-completion* ⇒ '*a metric-completion*
where *minus-metric-completion a b* ≡ *a* + (−*b*)

instance
⟨*proof*⟩
end

instantiation *metric-completion* :: (*complex-normed-vector*) *scaleC*

begin
lift-definition *scaleC-metric-completion* :: *complex* ⇒ *'a metric-completion* ⇒ *'a metric-completion*
is *λc x. λn. (scaleC c) (x n)*
⟨*proof*⟩
instance
⟨*proof*⟩
end

instance *metric-completion* :: (*complex-normed-vector*) *complex-vector*
⟨*proof*⟩

instantiation *metric-completion* :: (*real-normed-vector*) *banach*
begin

lift-definition *norm-metric-completion* :: *'a metric-completion* ⇒ *real*
is *λX. lim (λn. norm (X n))*
⟨*proof*⟩

definition *sgn-metric-completion* :: *'a metric-completion* ⇒ *'a metric-completion*
where *sgn-metric-completion x = x /*$_R$ *norm x*

lemma *metric-completion-triangle-ineq*:
fixes *x y* :: *'a metric-completion*
shows *norm (x + y)* ≤ *norm x + norm y*
⟨*proof*⟩

instance
⟨*proof*⟩
end

instance *metric-completion* :: (*complex-normed-vector*) *cbanach*
⟨*proof*⟩

2.3 The Cauchy completion of an inner product space is a Hilbert space.

instantiation *metric-completion* :: (*complex-inner*) *chilbert-space*
begin

lift-definition *cinner-metric-completion* ::
'a metric-completion ⇒ *'a metric-completion* ⇒ *complex*
is *λx y. lim (λn. cinner (x n) (y n))*
⟨*proof*⟩

instance
⟨*proof*⟩

end

2.4 The embedding *to-metric-completion*

And now we get all sorts of metric lemmas for inner products, e.g. an isometry (see *isometry-on UNIV to-metric-completion*) into a dense subset (see *closure* (*range to-metric-completion*) = *UNIV*) of a *chilbert-space*. This isometry is (complex-)linear, and preserves norms and inner products.

lemma *to-metric-completion-linear*: *linear to-metric-completion* (**is** ‹*linear ?g*›)
  and *to-metric-completion-clinear*: *clinear to-metric-completion* (**is** ‹*clinear ?f*›)
⟨*proof*⟩

lemma *to-metric-completion-norm*:
  shows *norm* (*to-metric-completion c*) = *norm c*
  ⟨*proof*⟩

lemma *to-metric-completion-bounded-clinear*:
  shows *bounded-clinear to-metric-completion*
  ⟨*proof*⟩

lemma *to-metric-completion-cinner*:
    (*to-metric-completion c*) $\cdot_C$ (*to-metric-completion d*) = *c* $\cdot_C$ *d*
  ⟨*proof*⟩

end

3 The Gelfand–Naimark–Segal Construction

theory *Gelfand-Naimark-Segal*
imports
  Inner-Completion
  Cstar-Algebra-On
  Complex-Bounded-Operators.Complex-Bounded-Linear-Function
  Smooth-Manifolds.Analysis-More
begin

unbundle *lifting-syntax*

3.1 Strengthen lifting lemmas

3.1.1 Lifting to the metric completion

lemma *inj-to-metric-completion*: *inj to-metric-completion*
  ⟨*proof*⟩

definition *from-metric-completion* ≡ *the-inv to-metric-completion*

lemma *lift-to-metric-completion2*:

fixes f::($'a$::*metric-space*) $\Rightarrow$ ($'b$::*complete-space*)
assumes *uniformly-continuous-on UNIV f*
shows $\exists\, g.$ (*uniformly-continuous-on UNIV g*)
$\wedge$ (f = g *o to-metric-completion*)
$\wedge$ ($\forall\, x \in$ *range to-metric-completion.* $g\ x = f$ (*from-metric-completion* x))
⟨*proof*⟩

lemma *lift-to-metric-completion-isometry2*:
fixes f::($'a$::*metric-space*) $\Rightarrow$ ($'b$::*complete-space*)
assumes *isometry-on UNIV f*
shows $\exists\, g.$ *isometry-on UNIV g*
$\wedge$ *range* g = *closure*(*range* f)
$\wedge$ f = g *o to-metric-completion*
$\wedge$ ($\forall\, x \in$ *range to-metric-completion.* $g\ x = f$ (*from-metric-completion* x))
⟨*proof*⟩

lemma *dense-imp-conv-seq*:
assumes *closure* S = U $x{\in}U$
shows $\exists\, s.$ $(s\ n) \in S \wedge s \longrightarrow x$
⟨*proof*⟩

lemma *to-metric-completion-sequence-ex*:
shows $\exists\, s.$ (*to-metric-completion o* s) $\longrightarrow x$
⟨*proof*⟩

lemma *cinner-extensionality-dense*:
assumes *dense-S*: *closure* S = *UNIV*
and *inner-on-S*: $\forall\, z{\in}S.$ *cinner* $z\ x$ = *cinner* $z\ y$
shows $x = y$
⟨*proof*⟩

lemma *cinner-extensionality-metric-completion*:
assumes $\forall\, z.$ *cinner* (*to-metric-completion* z) x = *cinner* (*to-metric-completion* z) y
shows $x = y$
⟨*proof*⟩

lemma *completion-ext-simp1*:
fixes f::($'m$::*metric-space*) $\Rightarrow$ ($'c$::*complete-space*) **and** g
defines $D \equiv$ *range to-metric-completion*
shows (f = g *o to-metric-completion*) $\longleftrightarrow$
($\forall\, x \in D.$ $g\ x = f$ (*from-metric-completion* x))
⟨*proof*⟩

lemma *lift-to-metric-completion4*:
fixes f::($'m$::*metric-space*) $\Rightarrow$ ($'c$::*complete-space*)
assumes *uniformly-continuous-on UNIV f*
shows $\exists !g.$ (*uniformly-continuous-on UNIV g*)
$\wedge$ (f = $g \circ$ *to-metric-completion*)

⟨*proof*⟩

lemma *lift-to-metric-completion3*:
fixes *f*::('*m*::*metric-space*) ⇒ ('*c*::*complete-space*)
assumes *uniformly-continuous-on UNIV f*
shows ∃!*g*. (*uniformly-continuous-on UNIV g*)
∧ (*f* = *g o to-metric-completion*)
∧ (∀ *x* ∈ *range to-metric-completion*. *g x* = *f* (*from-metric-completion x*))
⟨*proof*⟩

3.1.2 Lifting a bounded operator from an inner product space to a Hilbert space

lemma *to-metric-completion-cblinfun-exists*:
fixes *f* :: ‹'*a*::*complex-normed-vector* ⇒ '*b*::*cbanach*›
assumes *f-add*: ‹⋀*x y*. *f* (*x* + *y*) = *f x* + *f y*›
assumes *f-scale*: ‹⋀*c x y*. *f* (*c* ∗*C* *x*) = *c* ∗*C* *f x*›
assumes *bounded*: ‹⋀*x*. *norm* (*f x*) ≤ *B* ∗ *norm x*›
shows *cblinfun-extension-exists* (*range to-metric-completion*)
(*map-fun* (*from-metric-completion*) *id f*)
(**is** ‹*cblinfun-extension-exists ?D ?f*›)
⟨*proof*⟩

lemma *to-metric-completion-cblinfun-exists'*:
fixes *f* :: ‹('*a*::*complex-normed-vector*, '*b*::*cbanach*) *cblinfun*›
shows *cblinfun-extension-exists* (*range to-metric-completion*)
(*map-fun* (*from-metric-completion*) *id f*)
⟨*proof*⟩

3.2 The C^*-algebra of bounded operators on a Hilbert space

The assumption on the next lemma is a nondegeneracy condition found as the defining assumption of *assoc-algebra-1-on*. It's similar also to the class *zero-neq-one*, which enters for example into *ring-1*, *field*, and *one-dim*.

lemma *Cstar-cblinfun*:
includes *cblinfun-syntax*
assumes (*id-cblinfun* :: ('*b* ⇒*CL* ('*b*::*chilbert-space*))) ≠ *0*
shows *Cstar-algebra UNIV* (*UNIV* :: ('*b* ⇒*CL* ('*b*::*chilbert-space*)) *set*) (∗*C*) (*oCL*) *norm id-cblinfun adj*
(**is** *Cstar-algebra ?C ?B - - - - -*)
⟨*proof*⟩

The above can be used e.g. as *Cstar-algebra UNIV UNIV* (∗*C*) *cblinfun-compose norm id-cblinfun adj*, or using *zero-neq-one*.

3.3 Type class of C^*-algebras

Make a bundle to (de)activate dagger notation for adjoints of functions between types in the class of complex inner product spaces.

bundle *cadjoint-syntax* **begin**
notation *Complex-Inner-Product.cadjoint* (-† [*99*] *100*)
end
unbundle *no cadjoint-syntax*

class *cstar* = *complex-normed-algebra-1* + *cbanach* +
fixes *involution*::'*a*⇒'*a* (-† [*99*] *100*)
assumes *ivl-scaleC*: (*c* $*_C$ *s*)† = *cnj c* $*_C$ *s*† — ⟦*involutive-complex-algebra ?S ?scale ?multiplication ?unit ?involution*; *?s* ∈ *?S*⟧ ⟹ *?involution* (*?scale ?c ?s*) = *?scale* (*cnj ?c*) (*?involution ?s*)
and *ivl-plus* [*simp*]: (*x*+*y*)† = *x*† + *y*† — ⟦*involutive-ring ?R ?addition ?multiplication ?zero ?unit ?involution*; *?x* ∈ *?R*; *?y* ∈ *?R*⟧ ⟹ *?involution* (*?addition ?x ?y*) = *?addition* (*?involution ?x*) (*?involution ?y*)
and *involution*[*simp*]: (*x*†)† = *x* — ⟦*involutive-semigroup ?S ?pls ?involution*; *?x* ∈ *?S*⟧ ⟹ *?involution* (*?involution ?x*) = *?x*
and *ivl-times* [*simp*]: (*x* ∗ *y*)† = *y*† ∗ *x*† — ⟦*involutive-semigroup ?S ?pls ?involution*; *?x* ∈ *?S*; *?y* ∈ *?S*⟧ ⟹ *?involution* (*?pls ?x ?y*) = *?pls* (*?involution ?y*) (*?involution ?x*)
and *ivl-isometric*: *norm* (*A*†) = *norm A* — ⟦*Cstar-algebra ?F ?V ?scale ?multiplication ?vector-norm ?unit ?involution*; *?A* ∈ *?V*⟧ ⟹ *?vector-norm* (*?involution ?A*) = *?vector-norm ?A*
and *Cstar*: *norm* (*A*† ∗ *A*) = (*norm* (*A*†)) ∗ (*norm A*) — ⟦*Cstar-algebra ?F ?V ?scale ?multiplication ?vector-norm ?unit ?involution*; *?A* ∈ *?V*⟧ ⟹ *?vector-norm* (*?multiplication* (*?involution ?A*) *?A*) = *?vector-norm* (*?involution ?A*) ∗ *?vector-norm ?A*
begin

A class analogue of *Cstar-algebra*, reproducing the axioms *Cstar-algebra-axioms ?V ?multiplication ?vector-norm ?unit ?involution* ≡ ((∀ *A B*. *A* ∈ *?V* ⟶ *B* ∈ *?V* ⟶ *?vector-norm* (*?multiplication A B*) ≤ *?vector-norm A* ∗ *?vector-norm B*) ∧ *?vector-norm ?unit* = *1*) ∧ (∀ *A*. *A* ∈ *?V* ⟶ *?vector-norm* (*?involution A*) = *?vector-norm A*) ∧ (∀ *A*. *A* ∈ *?V* ⟶ *?vector-norm* (*?multiplication* (*?involution A*) *A*) = *?vector-norm* (*?involution A*) ∗ *?vector-norm A*) one-by-one. Some assumptions of *Cstar-algebra* are in *real-normed-algebra*, compare:

- ⟦*Cstar-algebra ?F ?V ?scale ?multiplication ?vector-norm ?unit ?involution*; *?A* ∈ *?V*; *?B* ∈ *?V*⟧ ⟹ *?vector-norm* (*?multiplication ?A ?B*) ≤ *?vector-norm ?A* ∗ *?vector-norm ?B* with *norm* (*?x* ∗ *?y*) ≤ *norm ?x* ∗ *norm ?y*,
- *Cstar-algebra ?F ?V ?scale ?multiplication ?vector-norm ?unit ?involution* ⟹ *?vector-norm ?unit* = *1* with *norm 1* = *1*.

end

bundle *cstar-syntax* **begin**
notation *involution* (-† [*99*] *100*)
end

unbundle *cstar-syntax*

lemma *cstar-locale*: *Cstar-algebra UNIV* (*UNIV*::*'a*::*cstar set*)
scaleC times norm 1 involution
⟨*proof*⟩

We *have to* interpret *cstar* as a *Cstar-algebra* in stages, and name the interpretation of the *involutive-complex-algebra*, otherwise there is a naming conflict between ⟦*algebra-on ?S ?scale ?amult*; ∀ *x*∈*?S*. *?amult x x* = *0*; ∀ *x*∈*?S*. ∀ *y*∈*?S*. ∀ *z*∈*?S*. *0* = *?amult x* (*?amult y z*) + *?amult y* (*?amult z x*) + *?amult z* (*?amult x y*)⟧ ⟹ *lie-algebra ?S ?scale ?amult* and ⟦*vector-space-on ?S ?scale*; ⋀*x y z*. ⟦*x* ∈ *?S*; *y* ∈ *?S*; *z* ∈ *?S*⟧ ⟹ *?lie-bracket* (*x* + *y*) *z* = *?lie-bracket x z* + *?lie-bracket y z* ∧ *?lie-bracket z* (*x* + *y*) = *?lie-bracket z x* + *?lie-bracket z y*; ⋀*a x y*. ⟦*x* ∈ *?S*; *y* ∈ *?S*⟧ ⟹ *?lie-bracket* (*?scale a x*) *y* = *?scale a* (*?lie-bracket x y*) ∧ *?lie-bracket x* (*?scale a y*) = *?scale a* (*?lie-bracket x y*); ⋀*x y*. ⟦*x* ∈ *?S*; *y* ∈ *?S*⟧ ⟹ *?lie-bracket x y* ∈ *?S*; ∀ *x*∈*?S*. *?lie-bracket x x* = *0*; ∀ *x*∈*?S*. ∀ *y*∈*?S*. ∀ *z*∈*?S*. *0* = *?lie-bracket x* (*?lie-bracket y z*) + *?lie-bracket y* (*?lie-bracket z x*) + *?lie-bracket z* (*?lie-bracket x y*)⟧ ⟹ *lie-algebra ?S ?scale ?lie-bracket*.

interpretation *cstar1*: *involutive-complex-algebra*
UNIV::*'a*::*cstar set scaleC times 1 involution*
⟨*proof*⟩

interpretation *Cstar-algebra*
UNIV UNIV::*'a*::*cstar set scaleC times norm 1 involution*
⟨*proof*⟩

Some simp set adjustments.

declare *cstar1.involution-uminus'*[*simp del*]
declare *cstar1.ivl-minus'*[*simp del*]

context *cstar* **begin**
named-theorems *cstar-simps*
lemmas *cross3-simps*(*10*,*11*,*23*,*24*,*27*,*28*)[*cstar-simps*]
lemmas
cstar1.involution-uminus'[*OF UNIV-I*, *cstar-simps*]
cstar1.ivl-minus'[*OF UNIV-I UNIV-I*, *cstar-simps*]
end

3.4 GNS construction

context
fixes $\omega :: 'a::cstar \Rightarrow complex$
assumes *omega-nonneg*: $\bigwedge A::'a.\ \omega\ (A^\dagger * A) \geq 0$
and *one* [*simp*]: $\omega\ 1 = 1$
and *clinear* ω
begin

3.4.1 Interpretation of existing locales

thm *linear-def clinear-def linear-on-def*

interpretation ω: *clinear* ω
⟨*proof*⟩

lemma *cstar-state*: *Cstar-state UNIV UNIV scaleC times norm 1 involution* ω
⟨*proof*⟩

interpretation *Cstar-state UNIV UNIV scaleC times norm 1 involution* ω
⟨*proof*⟩

3.4.2 Some identities for *cstar* and its left ideal.

Show the kernel of our intended inner product is a (left) ideal.

definition *zero-inner-set*: $\mathfrak{I} = \{A::'a.\ \omega\ (A^\dagger * A) = 0\}$

lemma *is-zero-inner*: $\omega\ (A^\dagger * A) = 0$ **if** $A \in \mathfrak{I}$
⟨*proof*⟩

lemma *zero-inner-zero*: $\omega\ i = 0$ **if** $i \in \mathfrak{I}$
⟨*proof*⟩

lemma *zero-inner-times*: $a*i \in \mathfrak{I}$ **if** $i \in \mathfrak{I}$ **for** $a\ i :: 'a$
⟨*proof*⟩

lemma *zero-inner-zero2*: $\omega\ (a*i) = 0$ **if** $i \in \mathfrak{I}$
⟨*proof*⟩

end

class *cstar-state* = *cstar* +
fixes *state* :: $'a \Rightarrow complex$ ($\langle\omega\rangle$)
assumes *omega-nonneg*: $\bigwedge A::'a.\ \omega\ (A^\dagger * A) \geq 0$
and *omega-one* [*simp*]: $\omega\ 1 = 1$
and *omega-linear*[*simp*]: $\omega\ (u + v) = \omega\ u + \omega\ v$
$\omega\ (c *_C u) = c * \omega\ u$
and *extra-assm*[*no-atp*]:
$\langle\omega\ ((B*A)^\dagger * B*A) \leq (norm\ B)^2 * \omega\ (A^\dagger * A)\rangle$

lemma *omega-clinear*: *clinear* ω
⟨*proof*⟩

interpretation *omega-linear*: *Vector-Spaces.linear scaleC times* ω
⟨*proof*⟩
interpretation *cstar-state*: *Cstar-state UNIV UNIV scaleC times norm 1 involution* ω
⟨*proof*⟩

lemma *zero-inner-zero'*: $\omega\ A = 0$ **if** $\omega\ (A^\dagger * A) = 0$
⟨*proof*⟩

lemma *zero-inner-zero2'*: $\omega\ (B*A) = 0$ **if** $\omega\ (A^\dagger * A) = 0$
⟨*proof*⟩

Some identities, restated for *cstar-state*.

lemma *state-ivl-cnj*: $\omega\ (A^\dagger) = cnj\ (\omega\ A)$
⟨*proof*⟩

lemma *state-cnj-ivl* [*cstar-simps*]: $cnj\ (\omega\ (A^\dagger)) = \omega\ A$
⟨*proof*⟩

lemma *state-norm-leq-inner*: $|\omega\ A|^2 \leq \omega\ (A^\dagger * A)$
⟨*proof*⟩

lemma *zero-inner-equivp*: *equivp* $(\lambda u\ v.\ \exists A.\ \omega\ (A^\dagger * A) = 0 \wedge u = v + A)$
⟨*proof*⟩

3.4.3 Quotient construction for the inner product

quotient-type (**overloaded**) *'a gns-precomplete* =
'a::*cstar-state* / $\lambda u\ v.\ \exists A.\ (\omega\ (A^\dagger * A) = 0 \wedge u = v + A)$
⟨*proof*⟩

instantiation *gns-precomplete* :: (*cstar-state*) *scaleC*
begin

lift-definition *scaleC-gns-precomplete* ::
complex ⇒ *'a gns-precomplete* ⇒ *'a gns-precomplete*
is *scaleC*
⟨*proof*⟩

lift-definition *scaleR-gns-precomplete* ::
real ⇒ *'a gns-precomplete* ⇒ *'a gns-precomplete*
is *scaleR*
⟨*proof*⟩

instance
⟨*proof*⟩

end

The lifting of the vector space structure is almost entirely a transfer+sledgehammer affair: identities proven earlier are enough to let sledgehammer find respectfulness proofs.

instantiation *gns-precomplete* :: (*cstar-state*) *complex-vector*
begin

lift-definition *zero-gns-precomplete* :: *'a gns-precomplete*
is *0* ⟨*proof*⟩

lemma $(a+b)* \ c = a* \ c + b* \ (c::'a)$
⟨*proof*⟩

lift-definition *minus-gns-precomplete* ::
'a gns-precomplete ⇒ *'a gns-precomplete* ⇒ *'a gns-precomplete*
is *minus*
⟨*proof*⟩

lift-definition *uminus-gns-precomplete* ::
'a gns-precomplete ⇒ *'a gns-precomplete*
is *uminus* ⟨*proof*⟩

lift-definition *plus-gns-precomplete* ::
'a gns-precomplete ⇒ *'a gns-precomplete* ⇒ *'a gns-precomplete*
is *plus*
⟨*proof*⟩

instance
⟨*proof*⟩

end

instantiation *gns-precomplete* :: (*cstar-state*) *complex-inner*
begin

The interesting definition is for the inner product, which implies that of the norm.

lift-definition *cinner-gns-precomplete* ::
'a gns-precomplete ⇒ *'a gns-precomplete* ⇒ *complex*
is $\lambda u \ v. \ \omega \ (u^\dagger * v)$
⟨*proof*⟩
lift-definition *norm-gns-precomplete* :: *'a gns-precomplete* ⇒ *real*
is $\lambda u. \ Re \ (csqrt \ (\omega \ (u^\dagger * u)))$

⟨*proof*⟩

The distance is given by the norm (and vector subtraction), as is standard.

definition *dist-gns-precomplete* ::
  'a gns-precomplete ⇒ *'a gns-precomplete* ⇒ *real*
  where *dist-gns-precomplete x y* = *norm* (*y*−*x*)

The uniform and topological structures are prescribed, in turn, by the metric structure.

definition *uniformity-gns-precomplete*::((*'a gns-precomplete*) × (*'a gns-precomplete*)) *filter*
  where *uniformity-gns-precomplete* = (*INF e*∈{*0* <..}. *principal* {(*x*, *y*). *dist x y* < *e*})

definition *open-gns-precomplete* :: *'a gns-precomplete set* ⇒ *bool*
  where *open-gns-precomplete U* = (∀ *x*∈*U*. *eventually* (λ(*x'*, *y*). *x'* = *x* ⟶ *y* ∈ *U*) *uniformity*)

definition *sgn-gns-precomplete* ::
  'a gns-precomplete ⇒ *'a gns-precomplete*
  where *sgn-gns-precomplete x* ≡ *x* /$_R$ *norm x*

lemma *norm-sym-gns-precomplete*:
  fixes *x y* :: *'a gns-precomplete*
  shows *norm* (*y* − *x*) = *norm* (*x* − *y*)
⟨*proof*⟩

instance
  ⟨*proof*⟩

end

type-synonym (*'a*) *gns* = *'a gns-precomplete metric-completion*

3.4.4 Representation of *'a* on the Hilbert space *'a gns*.

The action on the dense subspace *range to-gns* is given by lifted multiplication. Notice this isn't quite the dense subspace – this is the action on the precomplete quotient, a type, not a set on *'a gns*.

lift-definition *action-gns-precomplete* ::
  'a::*cstar-state* ⇒ *'a gns-precomplete* ⇒ *'a gns-precomplete*
  is (∗)
  ⟨*proof*⟩

lemma *action-gns-precomplete-alt*: *action-gns-precomplete u* ≡
  λ*v*. *abs-gns-precomplete* (*u* ∗ (*rep-gns-precomplete v*))

⟨*proof*⟩

definition *to-gns* :: *'a::cstar-state* ⇒ *'a gns*
where *to-gns* ≡ *to-metric-completion* ∘ *abs-gns-precomplete*

definition *from-gns* :: *'a::cstar-state gns* ⇒ *'a*
where *from-gns* ≡ *rep-gns-precomplete* ∘ *from-metric-completion*

definition *action-gns-precomplete'* ::
'a::cstar-state ⇒ *'a gns-precomplete* ⇒ *'a gns*
where *action-gns-precomplete' u* ≡
to-metric-completion ∘ (*action-gns-precomplete u*)

lemma *action-gns-precomplete'-alt*: *action-gns-precomplete'* ≡
(*id* −−−> *id* −−−> *to-metric-completion*) *action-gns-precomplete*
⟨*proof*⟩

lemma *action-gns-precomplete'-alt2*: *action-gns-precomplete'* ≡
(*id* −−−> *rep-gns-precomplete* −−−> (*to-gns*)) (∗)
⟨*proof*⟩

3.4.5 The action is bounded.

This is where ω ((*?B* ∗ *?A*)† ∗ *?B* ∗ *?A*) ≤ *complex-of-real* ((*norm ?B*)²) ∗ ω (*?A*† ∗ *?A*) is first used.

lemma *state-norm-sandwich-leq*: ‹|ω (*A*† ∗ *B* ∗ *A*)| ≤ ω (*A*† ∗ *A*) ∗ *norm B*›
⟨*proof*⟩

lemma *action-gns-bounded-norm-square*:
(*norm* ((*action-gns-precomplete u*) *v*))² ≤ (*norm u*)² ∗ (*norm v*)²
⟨*proof*⟩

lemma *action-gns-bounded-norm-square'*:
shows (*norm* ((*action-gns-precomplete' u*) *v*))² ≤ (*norm u*)² ∗ (*norm v*)²
⟨*proof*⟩

lemma *action-gns-bounded-norm*:
shows (*norm* ((*action-gns-precomplete u*) *v*)) ≤ (*norm u*) ∗ (*norm v*)
⟨*proof*⟩

lemma *action-gns-bounded-norm'*:
shows (*norm* ((*action-gns-precomplete' u*) *v*)) ≤ (*norm u*) ∗ (*norm v*)
⟨*proof*⟩

lemma *action-gns-blin*: *bounded-linear* (*action-gns-precomplete u*)
⟨*proof*⟩

lemma *action-gns-blin'*: *bounded-linear* (*action-gns-precomplete' u*)
⟨*proof*⟩

lemma *action-gns-unif-cts'*: *uniformly-continuous-on UNIV* (*action-gns-precomplete'* *u*)
⟨*proof*⟩
definition *action-gns-closure*:: *'a*::*cstar-state* ⇒ *'a gns*⇒*'a gns*
where *action-gns-closure u* ≡ *THE g*.
(*uniformly-continuous-on UNIV g*) ∧
(*action-gns-precomplete' u* = *g* ∘ *to-metric-completion*)

lemma *action-gns-closure*:
uniformly-continuous-on UNIV (*action-gns-closure u*)
action-gns-precomplete' u =
(*action-gns-closure u*) ∘ *to-metric-completion*
⋀*x*. *x*∈*range to-metric-completion* ⟹
(*action-gns-closure u*) *x* = *action-gns-precomplete' u* (*from-metric-completion* *x*)
⟨*proof*⟩

lemma *action-gns-cblin*: *bounded-clinear* (*action-gns-precomplete u*)
⟨*proof*⟩

lemma *action-gns-cblin'*: *bounded-clinear* (*action-gns-precomplete' u*)
⟨*proof*⟩
lemma *linear-continuous-imp-bounded*:
assumes *linear f continuous-on UNIV f*
shows *bounded-linear f*
⟨*proof*⟩

lemma *linear-continuous-iff-bounded*:
assumes *linear f*
shows *continuous-on UNIV f* ⟷ *bounded-linear f*
⟨*proof*⟩

lemma *clinear-continuous-iff-bounded*:
assumes *clinear f*
shows *continuous-on UNIV f* ⟷ *bounded-clinear f*
⟨*proof*⟩

lemma *linear-UCont-iff-bounded*:
assumes *linear f*
shows *uniformly-continuous-on UNIV f* ⟷ *bounded-linear f*
⟨*proof*⟩

lemma *clinear-UCont-iff-bounded*:
assumes *clinear f*
shows *uniformly-continuous-on UNIV f* ⟷ *bounded-clinear f*
⟨*proof*⟩

definition *action-cblinfun* ::
  'a::cstar-state ⇒ (*'a gns*, *'a gns*) *cblinfun* (‹π_ω›)
 where *action-cblinfun u* ≡ *cblinfun-extension*
  (*range to-metric-completion*)
  (*map-fun from-metric-completion id* (*action-gns-precomplete' u*))

lemma *action-cblinfun*: *cblinfun-extension-exists*
  (*range to-metric-completion*)
  (*map-fun from-metric-completion id* (*action-gns-precomplete' u*))
 for *u* :: *'a::cstar-state*
⟨*proof*⟩

lemma *action-cblinfun-apply*:
 shows *action-cblinfun u* (*to-metric-completion x*) = *action-gns-precomplete' u x*
 ⟨*proof*⟩

lemma *action-cblinfun-cong*:
 assumes *cspan* (*range to-metric-completion*) = *cspan* (*to-metric-completion ' B'*)
  ⋀*x*. *x* ∈ *B'* ⟹ *action-cblinfun u* (*to-metric-completion x*) = *g* (*to-metric-completion x*)
 shows *cblinfun-extension* (*to-metric-completion ' B'*) *g* = *action-cblinfun u*
 ⟨*proof*⟩

lemmas *action-cblinfun-apply'* = *cblinfun-extension-apply*[*OF action-cblinfun*, *folded action-cblinfun-def*]
  and *action-cblinfun-cong'* = *cblinfun-extension-cong*[*OF - - - action-cblinfun*, *folded action-cblinfun-def*]
 and *action-cblinfun-restrict'* = *cblinfun-extension-exists-restrict*[*OF - - action-cblinfun*, *folded action-cblinfun-def*]

lemma *action-UCont*: *isUCont* (*action-cblinfun u*)
 ⟨*proof*⟩

lemma *action-closure-cblinfun*:
  action-gns-closure u = (*action-cblinfun u*)
⟨*proof*⟩

The cyclic vector is the abstraction of the unit

definition *gns-1* (‹Ω›)
 where *gns-1* ≡ *to-gns 1*

lemma *gns-1'*: Ω ·$_C$ (*action-gns-closure u* Ω) = *ω u*
⟨*proof*⟩
lemma *cyclic-set*:
 shows {π_ω *u* Ω | *u*. *True*} = {*to-gns u* | *u*. *True*}
  and {*action-gns-closure u* Ω | *u*. *True*} = {*to-gns u* | *u*. *True*}
⟨*proof*⟩

lemma *gns-cyclic'*: *Elementary-Topology.closure* {*action-gns-closure u Ω* | *u* :: *'a*::*cstar-state*. *True*} = *UNIV*
⟨*proof*⟩

3.4.6 The action is a homomorphism

hide-const (**open**) *closure*

Following *Smooth-Manifolds.Analysis-More* to instantiate the vector space on functions.

instantiation *fun* :: (*type*,*scaleC*) *scaleC*
begin
definition *scaleC-fun* :: *complex* ⇒ (*'a* ⇒ *'b*) ⇒ *'a* ⇒ *'b*
where *scaleC-fun r f* = (λ*x*. *r* ∗$_C$ *f x*)
instance
⟨*proof*⟩
end

instance *fun* :: (*type*, *complex-vector*) *complex-vector*
⟨*proof*⟩
lemma *action-injective-on-quotient*: ∃ *I*. *ω* (*I*† ∗ *I*) = *0* ∧ *a* = *b* + *I*
if ∀ *x*. *action-gns-precomplete a x* = *action-gns-precomplete b x*
for *a b* :: *'a*::*cstar-state* **and** *x* :: *'a gns-precomplete*
⟨*proof*⟩
lemma *action-precomplete-morphism*:
shows *action-gns-precomplete* (*a*+*b*) = *action-gns-precomplete a* + *action-gns-precomplete b*
and *action-gns-precomplete* (*a*∗*b*) = *action-gns-precomplete a* ∘ *action-gns-precomplete b*
and *action-gns-precomplete* (*c*∗$_C$ *a*) = *c* ∗$_C$ (*action-gns-precomplete a*)
and *action-gns-precomplete 1* = *id*
and *action-gns-precomplete* (*involution a*) = *cadjoint* (*action-gns-precomplete a*)
⟨*proof*⟩

lemma *action-precomplete-morphism'*:
shows *action-gns-precomplete'* (*a*+*b*) = *action-gns-precomplete' a* + *action-gns-precomplete' b*
and *action-gns-precomplete'* (*c*∗$_C$ *a*) = *c* ∗$_C$ (*action-gns-precomplete' a*)
and *action-gns-precomplete' 1* = *to-metric-completion*
and *action-gns-precomplete'* (*a*∗*b*) = *action-gns-precomplete' a* ∘ *from-metric-completion* ∘ *action-gns-precomplete' b*
and *action-gns-precomplete'* (*a*†) = *to-metric-completion* ∘ *cadjoint* (*from-metric-completion* ∘ *action-gns-precomplete' a*)
⟨*proof*⟩

method *action-uniq-eqI* =
(*unfold action-gns-closure-def*,
intro the1-equality[*OF lift-to-metric-completion4*[*OF action-gns-unif-cts'*]] *conjI*,

fold action-gns-closure-def)

lemma
shows *action-closure-morphism*:
action-gns-closure (*a*+*b*) = *action-gns-closure a* + *action-gns-closure b*
action-gns-closure (*a*∗*b*) = *comp* (*action-gns-closure a*) (*action-gns-closure b*)
action-gns-closure (*c*∗$_C$*a*) = *c* ∗$_C$ (*action-gns-closure a*)
action-gns-closure 1 = *id-cblinfun*
action-gns-closure (*involution a*) = *cadjoint* (*action-gns-closure a*)
and *action-cblinfun-morphism*:
π_ω (*a*+*b*) = π_ω *a* + π_ω *b*
π_ω (*a*∗*b*) = *cblinfun-compose* (π_ω *a*) (π_ω *b*)
π_ω (*c*∗$_C$*a*) = *c* ∗$_C$ (π_ω *a*)
π_ω *1* = *id-cblinfun*
π_ω (*involution a*) = *adj* (π_ω *a*)
⟨*proof*⟩

3.5 GNS theorem for π_ω

It's a little sparser than in textbooks: the existence of the Hilbert space and a representation in terms of bounded operators is implicit in the construction of *'a gns* and π_ω.

theorem *GNS-construction*:
shows *gns-1*: ∀ *u*. Ω ∙$_C$ (π_ω *u* Ω) = ω *u*
and *gns-cyclic*: *closure* {π_ω *u* Ω | *u*. *True*} = *UNIV*
and *gns-hom*: π_ω (*a*+*b*) = π_ω *a* + π_ω *b*
π_ω (*a*∗*b*) = *cblinfun-compose* (π_ω *a*) (π_ω *b*)
π_ω (*c*∗$_C$*a*) = *c* ∗$_C$ (π_ω *a*)
π_ω *1* = *id-cblinfun*
π_ω (*involution a*) = *adj* (π_ω *a*)
⟨*proof*⟩

unbundle *cblinfun-syntax*
no-notation *gns-1* (‹Ω›)
no-notation *action-cblinfun* (‹π_ω›)

For presentation only, we give the existential form of the GNS construction. There's no disadvantage to using explicit constructions and ∀ *u*. *gns-1* ∙$_C$ (*action-cblinfun u* ∗$_V$ *gns-1*) = ω *u*
closure {*action-cblinfun u* ∗$_V$ *gns-1* |*u*. *True*} = *UNIV*
action-cblinfun (*?a* + *?b*) = *action-cblinfun ?a* + *action-cblinfun ?b*
action-cblinfun (*?a* ∗ *?b*) = *action-cblinfun ?a* o_{CL} *action-cblinfun ?b*
action-cblinfun (*?c* ∗$_C$ *?a*) = *?c* ∗$_C$ *action-cblinfun ?a*
action-cblinfun 1 = *id-cblinfun*
action-cblinfun (*?a*†) = *action-cblinfun ?a*∗.

theorem *GNS-construction'*:
obtains π_ω :: *'a*::*cstar-state* ⇒ (*'a gns* ⇒$_{CL}$ *'a gns*)

and $\Omega :: 'a\ gns$
where — properties of Ω
$\forall u.\ \Omega \cdot_C (\pi_\omega\ u\ \Omega) = \omega\ u$
and $closure\ \{\pi_\omega\ u\ \Omega \mid u.\ True\} = UNIV$
— π is a *-homomorphism
and $\pi_\omega\ (a+b) = \pi_\omega\ a + \pi_\omega\ b$
and $\pi_\omega\ (a*b) = cblinfun\text{-}compose\ (\pi_\omega\ a)\ (\pi_\omega\ b)$
and $\pi_\omega\ (c*_C a) = c *_C (\pi_\omega\ a)$
and $\pi_\omega\ 1 = id\text{-}cblinfun$
and $\pi_\omega\ (involution\ a) = adj\ (\pi_\omega\ a)$
⟨*proof*⟩
unbundle *no cblinfun-syntax*
notation *gns-1* (‹Ω›)
notation *action-cblinfun* (‹π_ω›)

unbundle *no lifting-syntax*
unbundle *no cstar-syntax*
unbundle *scaleC-syntax*

end

References

[1] O. Bratteli and D. W. Robinson. *Operator algebras and quantum statistical mechanics.* Texts and monographs in physics. Springer-Verlag, New York, 1979.

[2] C. J. Fewster and K. Rejzner. Algebraic Quantum Field Theory – an introduction.