Classifying the specified normal-part equivalence with the selected bounded lifts

`MathlibAnnex.Analysis.CStarAlgebra.exists_normalPart_selected_classification_eq`

theorem

Classifies the given normal-part homeomorphism using its fixed factors and an operator image equality.

Statement

Let H, K be complex Hilbert spaces, and let $A \subseteq \mathcal{B}(H)$ and $B \subseteq \mathcal{B}(K)$ be nonzero norm-closed complex $*$-subalgebras acting irreducibly on H and K, respectively. Neither algebra is assumed unital. For $x, y \in H \setminus \{0\}$ and $z, w \in K \setminus \{0\}$, define the bounded complex-linear functionals

$$\begin{aligned} f^A_{x,y} &: A \to \mathbb{C}, \qquad f^A_{x,y}(a) = \langle y, a(x) \rangle, \\ f^B_{z,w} &: B \to \mathbb{C}, \qquad f^B_{z,w}(b) = \langle w, b(z) \rangle. \end{aligned}$$

Their normal parts are

$$\begin{aligned} N(A) &= \{\ker f^A_{x,y} : x, y \in H \setminus \{0\}\}, \\ N(B) &= \{\ker f^B_{z,w} : z, w \in K \setminus \{0\}\}. \end{aligned} \tag{1}$$

The inner product is conjugate-linear in its first variable. For $\mathcal{D} \subseteq N(A)$, the relative geometric closure is $c_A(\mathcal{D}) = \{J \in N(A) : \bigcap_{I \in \mathcal{D}} I \subseteq J\}$; define c_B analogously. Let $e : N(A) \to N(B)$ be a finite-homeomorphism: a bijection satisfying $e(c_A(\mathcal{D})) = c_B(e(\mathcal{D}))$ for every finite $\mathcal{D} \subseteq N(A)$. Assume H is infinite-dimensional and, additionally, both e and e^{-1} are fully continuous:

$$\begin{aligned} e(c_A(\mathcal{D})) &\subseteq c_B(e(\mathcal{D})) \qquad (\mathcal{D} \subseteq N(A)), \\ e^{-1}(c_B(\mathcal{E})) &\subseteq c_A(e^{-1}(\mathcal{E})) \qquad (\mathcal{E} \subseteq N(B)). \end{aligned} \tag{2}$$

Let $U, V : H \to K$ and σ_1, σ_2 be the semilinear lifts and scalar automorphisms selected in the algebraic realization of this e, using the dimension bound guaranteed by infinite-dimensionality. Thus $U(cx) = \sigma_1(c)U(x)$ and $V(cx) = \sigma_2(c)V(x)$ for $c \in \mathbb{C}$ and $x \in H$. Then $\sigma_1 = \sigma_2$. There exist either two continuous complex-linear equivalences $U_c, V_c : H \to K$, or two continuous conjugate-linear equivalences $U_c, V_c : H \to K$, such that

$$\begin{aligned} U_c(x) &= U(x), & V_c(x) &= V(x) & (x \in H), \\ U_c^{-1}(z) &= U^{-1}(z), & V_c^{-1}(z) &= V^{-1}(z) & (z \in K). \end{aligned} \tag{3}$$

For either common scalar type, one of the following **global** orientation branches holds. Write F_{U_c,V_c} for the normal-part equivalence given by $F_{U_c,V_c}(\ker f^A_{x,y}) = \ker f^B_{U_c x, V_c y}$. Let s_B be the closure-space homeomorphism induced by the involution, so that $s_B(\ker f^B_{z,w}) = \ker f^B_{w,z}$.

Direct branch. The specified equivalence is $e = F_{U_c,V_c}$. For every $x, y \in H \setminus \{0\}$ and $z, w \in K \setminus \{0\}$,

$$\begin{aligned} e(\ker f^A_{x,y}) &= \ker f^B_{U_c x, V_c y}, \\ e^{-1}(\ker f^B_{z,w}) &= \ker f^A_{U_c^{-1} z, V_c^{-1} w}. \end{aligned} \tag{4}$$

Swapped branch. The specified equivalence is $e = s_B \circ F_{U_c,V_c}$. For every $x, y \in H \setminus \{0\}$ and $z, w \in K \setminus \{0\}$,

$$\begin{aligned} e(\ker f^A_{x,y}) &= \ker f^B_{V_c y, U_c x}, \\ e^{-1}(\ker f^B_{z,w}) &= \ker f^A_{U_c^{-1} w, V_c^{-1} z}. \end{aligned} \tag{5}$$

The kernels on each side are subsets of the corresponding algebra. The forward and inverse formulas use the same branch and the same maps in (3). In either orientation, the common scalar type determines the operator-set equality

$$\begin{cases} V_c^* B U_c = A, & U_c, V_c \text{ complex-linear}, \\ V_c^\dagger B U_c = A, & U_c, V_c \text{ conjugate-linear}. \end{cases} \tag{6}$$

Here $V_c^* B U_c = \{V_c^* b U_c : b \in B\}$, and similarly for $V_c^\dagger B U_c$. In the second case, the conjugate adjoint $V_c^\dagger : K \to H$ is characterized by $\langle V_c^\dagger p, q\rangle = \langle V_c q, p\rangle$ for $p \in K$ and $q \in H$. For each $b \in B$, the corresponding product $V_c^* b U_c$ or $V_c^\dagger b U_c$ is a bounded complex-linear operator on H.

Assumptions

The two nonzero concrete $*$-algebras are norm closed and act irreducibly on their respective Hilbert spaces. Neither algebra is assumed unital. The source space H is infinite dimensional.

Both full-continuity conditions (2) are required, in addition to finite continuity in both directions. Continuity of the semilinear factors, equality of their scalar types, and the four identities (3) are conclusions.

Conclusion

The given e has one common scalar type and one global direct or swapped orientation. The same maps U_c, V_c and their inverses satisfy all point and kernel formulas. Their underlying functions are precisely those fixed by the algebraic realization of e.

In either scalar case the corresponding operator-set equality (6) holds. This is equality of concrete operator sets; it does not assert that $U_c = V_c$, that either map is unitary, or that the transformation is a multiplicative $*$-isomorphism.

Main citations

The selected classification theorem gives the common scalar type, the four function identities, both orientations, all coefficient-kernel formulas and the operator image equality. The algebraic realization fixes the underlying maps. The linear and conjugate-linear operator criteria specify the

two transformations.

Lean source signature (exact)

The preceding `variable` declarations introduce H, K, the algebras `S`, `T`, the irreducibility hypotheses and the finite-homeomorphism e. Their occurrence in the arguments and conclusion makes them theorem parameters. The three arguments displayed below are infinite-dimensionality and the two directions of full continuity. In the conclusion type, the three `let` bindings name the rank bound and the selected maps U, V; they are not proof steps or a second choice of maps.

The following contiguous literal signature ends immediately before the final proof assignment. The full source retains the proof, and the surrounding declarations are quoted separately below.

```
theorem exists_normalPart_selected_classification_eq
    (hH : ¬ FiniteDimensional ℂ H)
    (hcont : ((NormedSpace.geometricStructureClosure ℂ S).subspace (normalPart S)).Continuous
      ((NormedSpace.geometricStructureClosure ℂ T).subspace (normalPart T)) e.toEquiv)
    (hcontInv : ((NormedSpace.geometricStructureClosure ℂ T).subspace (normalPart
T)).Continuous
      ((NormedSpace.geometricStructureClosure ℂ S).subspace (normalPart S)) e.toEquiv.symm) :
    let hd := (three_le_rank_of_normalPart_finiteHomeomorph S T hS hT e hH).1
    let U := normalPartFirstSemilinearEquiv S T hS hT e hd
    let V := normalPartSecondSemilinearEquiv S T hS hT e hd
    normalPartFirstScalarEquiv S T hS hT e hd =
      normalPartSecondScalarEquiv S T hS hT e hd ∧
    (
    (∃ Uc Vc : H ≃L[ℂ] K,
      (∀ x, Uc x = U x) ∧ (∀ x, Vc x = V x) ∧
      (∀ z, Uc.symm z = U.symm z) ∧ (∀ z, Vc.symm z = V.symm z) ∧
      (((e.toEquiv = normalPartSemilinearEquiv S T hS hT (fun _ => rfl)
            Uc.toLinearEquiv Vc.toLinearEquiv) ∧
        (∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),
          e.toEquiv (normalPartPoint S hS x y hx hy) =
            normalPartPoint T hT (Uc x) (Vc y)
              (by simpa only [map_zero] using Uc.injective.ne hx)
              (by simpa only [map_zero] using Vc.injective.ne hy)) ∧
        (∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),
          e.toEquiv.symm (normalPartPoint T hT z w hz hw) =
            normalPartPoint S hS (Uc.symm z) (Vc.symm w)
              (by simpa only [map_zero] using Uc.symm.injective.ne hz)
              (by simpa only [map_zero] using Vc.symm.injective.ne hw)) ∧
        (∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),
          (e.toEquiv (normalPartPoint S hS x y hx hy)).val.val =
            {b : T | inner ℂ (Vc y) ((b : K →L[ℂ] K) (Uc x)) = 0}) ∧
        (∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),
          (e.toEquiv.symm (normalPartPoint T hT z w hz hw)).val.val =
            {a : S | inner ℂ (Vc.symm w) ((a : H →L[ℂ] H) (Uc.symm z)) = 0})) ∨
       ((e.toEquiv = (normalPartSemilinearEquiv S T hS hT (fun _ => rfl)
            Uc.toLinearEquiv Vc.toLinearEquiv).trans (normalPartStarHomeomorph T hT).toEquiv) ∧
        (∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),
          e.toEquiv (normalPartPoint S hS x y hx hy) =
            normalPartPoint T hT (Vc y) (Uc x)
              (by simpa only [map_zero] using Vc.injective.ne hy)
              (by simpa only [map_zero] using Uc.injective.ne hx)) ∧
        (∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),
```

```
          e.toEquiv.symm (normalPartPoint T hT z w hz hw) =
            normalPartPoint S hS (Uc.symm w) (Vc.symm z)
              (by simpa only [map_zero] using Uc.symm.injective.ne hw)
              (by simpa only [map_zero] using Vc.symm.injective.ne hz)) ∧
        (∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),
          (e.toEquiv (normalPartPoint S hS x y hx hy)).val.val =
            {b : T | inner ℂ (Uc x) ((b : K →L[ℂ] K) (Vc y)) = 0}) ∧
        (∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),
          (e.toEquiv.symm (normalPartPoint T hT z w hz hw)).val.val =
            {a : S | inner ℂ (Vc.symm z) ((a : H →L[ℂ] H) (Uc.symm w)) = 0}))) ∧
      InnerProductSpace.adjointSandwich Uc.toContinuousLinearMap Vc.toContinuousLinearMap ''
        (T : Set (K →L[ℂ] K)) = (S : Set (H →L[ℂ] H))) ∨
    (∃ Uc Vc : H ≃SL[starRingEnd ℂ] K,
      (∀ x, Uc x = U x) ∧ (∀ x, Vc x = V x) ∧
      (∀ z, Uc.symm z = U.symm z) ∧ (∀ z, Vc.symm z = V.symm z) ∧
      (((e.toEquiv = normalPartSemilinearEquiv S T hS hT (fun _ => rfl)
            Uc.toLinearEquiv Vc.toLinearEquiv) ∧
        (∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),
          e.toEquiv (normalPartPoint S hS x y hx hy) =
            normalPartPoint T hT (Uc x) (Vc y)
              (by simpa only [map_zero] using Uc.injective.ne hx)
              (by simpa only [map_zero] using Vc.injective.ne hy)) ∧
        (∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),
          e.toEquiv.symm (normalPartPoint T hT z w hz hw) =
            normalPartPoint S hS (Uc.symm z) (Vc.symm w)
              (by simpa only [map_zero] using Uc.symm.injective.ne hz)
              (by simpa only [map_zero] using Vc.symm.injective.ne hw)) ∧
        (∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),
          (e.toEquiv (normalPartPoint S hS x y hx hy)).val.val =
            {b : T | inner ℂ (Vc y) ((b : K →L[ℂ] K) (Uc x)) = 0}) ∧
        (∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),
          (e.toEquiv.symm (normalPartPoint T hT z w hz hw)).val.val =
            {a : S | inner ℂ (Vc.symm w) ((a : H →L[ℂ] H) (Uc.symm z)) = 0})) ∨
       ((e.toEquiv = (normalPartSemilinearEquiv S T hS hT (fun _ => rfl)
            Uc.toLinearEquiv Vc.toLinearEquiv).trans (normalPartStarHomeomorph T hT).toEquiv) ∧
        (∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),
          e.toEquiv (normalPartPoint S hS x y hx hy) =
            normalPartPoint T hT (Vc y) (Uc x)
              (by simpa only [map_zero] using Vc.injective.ne hy)
              (by simpa only [map_zero] using Uc.injective.ne hx)) ∧
        (∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),
          e.toEquiv.symm (normalPartPoint T hT z w hz hw) =
            normalPartPoint S hS (Uc.symm w) (Vc.symm z)
              (by simpa only [map_zero] using Uc.symm.injective.ne hw)
              (by simpa only [map_zero] using Vc.symm.injective.ne hz)) ∧
        (∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),
          (e.toEquiv (normalPartPoint S hS x y hx hy)).val.val =
            {b : T | inner ℂ (Uc x) ((b : K →L[ℂ] K) (Vc y)) = 0}) ∧
        (∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),
          (e.toEquiv.symm (normalPartPoint T hT z w hz hw)).val.val =
            {a : S | inner ℂ (Vc.symm z) ((a : H →L[ℂ] H) (Uc.symm w)) = 0}))) ∧
      InnerProductSpace.conjugateAdjointSandwich Uc.toContinuousLinearMap
Vc.toContinuousLinearMap ''
        (T : Set (K →L[ℂ] K)) = (S : Set (H →L[ℂ] H))))
```

In the source, `S` and `T` denote the algebras A and B used here, respectively.

In the source	Mathematical meaning

Ambient spaces and algebras
`variable {H : Type u} [NormedAddCommGroup H] [InnerProductSpace ℂ H] [CompleteSpace H]`
`variable {K : Type v} [NormedAddCommGroup K] [InnerProductSpace ℂ K] [CompleteSpace K]`
`variable (S : NonUnitalStarSubalgebra ℂ (H →L[ℂ] H))`
`variable (T : NonUnitalStarSubalgebra ℂ (K →L[ℂ] K))`
`variable [IsClosed (S : Set (H →L[ℂ] H))] [IsClosed (T : Set (K →L[ℂ] K))]`
`variable (hS : NonUnitalCStarRepresentation.IsIrreducible (NonUnitalStarSubalgebraClass.subtype S))`
`variable (hT : NonUnitalCStarRepresentation.IsIrreducible (NonUnitalStarSubalgebraClass.subtype T))`
	The Hilbert spaces H, K and the nonzero norm-closed irreducible complex $*$-subalgebras A, B, denoted S, T in the source.

Given finite homeomorphism
`variable (e : ClosureSpace.FiniteHomeomorph`
`  ((NormedSpace.geometricStructureClosure ℂ S).subspace (normalPart S))`
`  ((NormedSpace.geometricStructureClosure ℂ T).subspace (normalPart T)))`
	The same bijection $e : N(A) \to N(B)$ and its actual inverse, with finite continuity for the relative closures c_A, c_B in both directions.

Full continuity hypotheses
`(hH : ¬ FiniteDimensional ℂ H)`
`(hcont : ((NormedSpace.geometricStructureClosure ℂ S).subspace (normalPart S)).Continuous`
`  ((NormedSpace.geometricStructureClosure ℂ T).subspace (normalPart T)) e.toEquiv)`
`(hcontInv : ((NormedSpace.geometricStructureClosure ℂ T).subspace (normalPart T)).Continuous`
`  ((NormedSpace.geometricStructureClosure ℂ S).subspace (normalPart S)) e.toEquiv.symm) :`
	The infinite-dimensionality of H, and both full-continuity inclusions in (2), for arbitrary subsets $\mathcal{D} \subseteq N(A)$ and $\mathcal{E} \subseteq N(B)$. The functions supplied are the given e and its inverse.

Derived rank and selected maps
`let hd := (three_le_rank_of_normalPart_finiteHomeomorph S T hS hT e hH).1`
`let U := normalPartFirstSemilinearEquiv S T hS hT e hd`
`let V := normalPartSecondSemilinearEquiv S T hS hT e hd`
	The rank theorem supplies $3 \leq \dim_{\mathbb{C}}^{\mathrm{alg}} H$ under the stated hypotheses. It is not an additional assumption. The same e and this bound select the same $U, V : H \to K$ used in the Statement.

Common scalar type
`normalPartFirstScalarEquiv S T hS hT e hd =`
`  normalPartSecondScalarEquiv S T hS hT e hd ∧`
	The two scalar automorphisms of the selected lifts agree:
	$$\sigma_1 = \sigma_2.$$
	The alternatives that follow classify their common type; they do not allow a mixed linear/conjugate-linear pair.

In the source	Mathematical meaning

Complex-linear realization
`(∃ Uc Vc : H ≃L[ℂ] K,`
`  (∀ x, Uc x = U x) ∧ (∀ x, Vc x = V x) ∧`
`  (∀ z, Uc.symm z = U.symm z) ∧ (∀ z, Vc.symm z = V.symm z) ∧`
	Continuous complex-linear isomorphisms $U_c, V_c : H \to K$ have the same functions and the same inverses as the selected lifts:
	$$U_c(x) = U(x), \qquad V_c(x) = V(x) \quad (x \in H),$$
	$$U_c^{-1}(z) = U^{-1}(z), \qquad V_c^{-1}(z) = V^{-1}(z) \quad (z \in K).$$

Complex-linear direct map
`(((e.toEquiv = normalPartSemilinearEquiv S T hS hT (fun _ => rfl)`
`      Uc.toLinearEquiv Vc.toLinearEquiv) ∧`
	The direct branch identifies the given map with the transport by these same U_c, V_c:
	$$e = F_{U_c,V_c}.$$
	Taking underlying linear or semilinear maps changes no function.

Complex-linear direct forward point
`(∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),`
`  e.toEquiv (normalPartPoint S hS x y hx hy) =`
`    normalPartPoint T hT (Uc x) (Vc y)`
`      (by simpa only [map_zero] using Uc.injective.ne hx)`
`      (by simpa only [map_zero] using Vc.injective.ne hy)) ∧`
	For every $x, y \in H \setminus \{0\}$, the first equation of (4):
	$$e(\ker f^A_{x,y}) = \ker f^B_{U_c x, V_c y}.$$

Complex-linear direct inverse point
`(∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),`
`  e.toEquiv.symm (normalPartPoint T hT z w hz hw) =`
`    normalPartPoint S hS (Uc.symm z) (Vc.symm w)`
`      (by simpa only [map_zero] using Uc.symm.injective.ne hz)`
`      (by simpa only [map_zero] using Vc.symm.injective.ne hw)) ∧`
	For every $z, w \in K \setminus \{0\}$, the second equation of (4):
	$$e^{-1}(\ker f^B_{z,w}) = \ker f^A_{U_c^{-1} z, V_c^{-1} w}.$$

Complex-linear direct underlying sets
`(∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),`
`  (e.toEquiv (normalPartPoint S hS x y hx hy)).val.val =`
`    {b : T | inner ℂ (Vc y) ((b : K →L[ℂ] K) (Uc x)) = 0}) ∧`
`(∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),`
`  (e.toEquiv.symm (normalPartPoint T hT z w hz hw)).val.val =`
`    {a : S | inner ℂ (Vc.symm w) ((a : H →L[ℂ] H) (Uc.symm z)) = 0})) v`
	The same two equalities are also equalities of subsets of B and A, respectively, with the same nonzero-vector quantifiers:
	$$e(\ker f^A_{x,y}) = \ker f^B_{U_c x, V_c y},$$
	$$e^{-1}(\ker f^B_{z,w}) = \ker f^A_{U_c^{-1} z, V_c^{-1} w}.$$
	The projections expose the underlying kernel sets, not different points.

Complex-linear swapped map
`((e.toEquiv = (normalPartSemilinearEquiv S T hS hT (fun _ => rfl)`
`     Uc.toLinearEquiv Vc.toLinearEquiv).trans (normalPartStarHomeomorph T hT).toEquiv) ∧`
	The other orientation composes with the target star map:
	$$e = s_B \circ F_{U_c,V_c}.$$
	Here $s_B(\ker f^B_{z,w}) = \ker f^B_{w,z}$. The disjunction selects one orientation for all vectors.

In the source	Mathematical meaning

Complex-linear swapped forward point
`(∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),`
`  e.toEquiv (normalPartPoint S hS x y hx hy) =`
`  normalPartPoint T hT (Vc y) (Uc x)`
`  (by simpa only [map_zero] using Vc.injective.ne hy)`
`  (by simpa only [map_zero] using Uc.injective.ne hx)) ∧`
	For every $x, y \in H \setminus \{0\}$, the first equation of (5):
	$$e(\ker f^A_{x,y}) = \ker f^B_{V_c y, U_c x}.$$

Complex-linear swapped inverse point
`(∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),`
`  e.toEquiv.symm (normalPartPoint T hT z w hz hw) =`
`  normalPartPoint S hS (Uc.symm w) (Vc.symm z)`
`  (by simpa only [map_zero] using Uc.symm.injective.ne hw)`
`  (by simpa only [map_zero] using Vc.symm.injective.ne hz)) ∧`
	For every $z, w \in K \setminus \{0\}$, the second equation of (5):
	$$e^{-1}(\ker f^B_{z,w}) = \ker f^A_{U_c^{-1} w, V_c^{-1} z}.$$

Complex-linear swapped underlying sets
`(∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),`
`  (e.toEquiv (normalPartPoint S hS x y hx hy)).val.val =`
`  {b : T | inner ℂ (Uc x) ((b : K →L[ℂ] K) (Vc y)) = 0}) ∧`
`(∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),`
`  (e.toEquiv.symm (normalPartPoint T hT z w hz hw)).val.val =`
`  {a : S | inner ℂ (Vc.symm z) ((a : H →L[ℂ] H) (Uc.symm w)) = 0})) ∧`
	The underlying subsets satisfy the same two equations in (5), for all the same nonzero pairs:
	$$e(\ker f^A_{x,y}) = \ker f^B_{V_c y, U_c x},$$
	$$e^{-1}(\ker f^B_{z,w}) = \ker f^A_{U_c^{-1} w, V_c^{-1} z}.$$
	The orientation and actual inverse are unchanged.

Complex-linear operator image
`InnerProductSpace.adjointSandwich Uc.toContinuousLinearMap Vc.toContinuousLinearMap ''`
`  (T : Set (K →L[ℂ] K)) = (S : Set (H →L[ℂ] H))) v`
	In either orientation of the complex-linear case,
	$$V_c^* B U_c = \{V_c^* b U_c : b \in B\} = A \subseteq \mathcal{B}(H).$$
	This is equality, not just inclusion, and uses the bounded Hilbert-space adjoint.

Conjugate-linear realization
`(∃ Uc Vc : H ≃SL[starRingEnd ℂ] K,`
`  (∀ x, Uc x = U x) ∧ (∀ x, Vc x = V x) ∧`
`  (∀ z, Uc.symm z = U.symm z) ∧ (∀ z, Vc.symm z = V.symm z) ∧`
	Continuous conjugate-linear isomorphisms $U_c, V_c : H \to K$ instead realize the selected lifts, again with the same inverse functions:
	$$U_c(x) = U(x), \qquad V_c(x) = V(x) \quad (x \in H),$$
	$$U_c^{-1}(z) = U^{-1}(z), \qquad V_c^{-1}(z) = V^{-1}(z) \quad (z \in K).$$
	This is the other scalar-type alternative, not a second choice within the complex-linear case.

Conjugate-linear direct map
`(((e.toEquiv = normalPartSemilinearEquiv S T hS hT (fun _ => rfl)`
`  Uc.toLinearEquiv Vc.toLinearEquiv) ∧`
	The direct branch identifies the given map with the transport by these same U_c, V_c:
	$$e = F_{U_c, V_c}.$$
	Taking underlying linear or semilinear maps changes no function.

In the source	Mathematical meaning
Conjugate-linear direct forward point `(∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0), e.toEquiv (normalPartPoint S hS x y hx hy) = normalPartPoint T hT (Uc x) (Vc y) (by simpa only [map_zero] using Uc.injective.ne hx) (by simpa only [map_zero] using Vc.injective.ne hy)) ∧`	For every $x, y \in H \setminus \{0\}$, the first equation of (4): $$e(\ker f^A_{x,y}) = \ker f^B_{U_c x, V_c y}.$$
Conjugate-linear direct inverse point `(∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0), e.toEquiv.symm (normalPartPoint T hT z w hz hw) = normalPartPoint S hS (Uc.symm z) (Vc.symm w) (by simpa only [map_zero] using Uc.symm.injective.ne hz) (by simpa only [map_zero] using Vc.symm.injective.ne hw)) ∧`	For every $z, w \in K \setminus \{0\}$, the second equation of (4): $$e^{-1}(\ker f^B_{z,w}) = \ker f^A_{U_c^{-1} z, V_c^{-1} w}.$$
Conjugate-linear direct underlying sets `(∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0), (e.toEquiv (normalPartPoint S hS x y hx hy)).val.val = {b : T | inner ℂ (Vc y) ((b : K →L[ℂ] K) (Uc x)) = 0}) ∧ (∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0), (e.toEquiv.symm (normalPartPoint T hT z w hz hw)).val.val = {a : S | inner ℂ (Vc.symm w) ((a : H →L[ℂ] H) (Uc.symm z)) = 0})) ∨`	The same two equalities are also equalities of subsets of B and A, respectively, with the same nonzero-vector quantifiers: $$e(\ker f^A_{x,y}) = \ker f^B_{U_c x, V_c y},$$ $$e^{-1}(\ker f^B_{z,w}) = \ker f^A_{U_c^{-1} z, V_c^{-1} w}.$$ The projections expose the underlying kernel sets, not different points.
Conjugate-linear swapped map `((e.toEquiv = (normalPartSemilinearEquiv S T hS hT (fun _ => rfl) Uc.toLinearEquiv Vc.toLinearEquiv).trans (normalPartStarHomeomorph T hT).toEquiv) ∧`	The other orientation composes with the target star map: $$e = s_B \circ F_{U_c, V_c}.$$ Here $s_B(\ker f^B_{z,w}) = \ker f^B_{w,z}$. The disjunction selects one orientation for all vectors.
Conjugate-linear swapped forward point `(∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0), e.toEquiv (normalPartPoint S hS x y hx hy) = normalPartPoint T hT (Vc y) (Uc x) (by simpa only [map_zero] using Vc.injective.ne hy) (by simpa only [map_zero] using Uc.injective.ne hx)) ∧`	For every $x, y \in H \setminus \{0\}$, the first equation of (5): $$e(\ker f^A_{x,y}) = \ker f^B_{V_c y, U_c x}.$$
Conjugate-linear swapped inverse point `(∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0), e.toEquiv.symm (normalPartPoint T hT z w hz hw) = normalPartPoint S hS (Uc.symm w) (Vc.symm z) (by simpa only [map_zero] using Uc.symm.injective.ne hw) (by simpa only [map_zero] using Vc.symm.injective.ne hz)) ∧`	For every $z, w \in K \setminus \{0\}$, the second equation of (5): $$e^{-1}(\ker f^B_{z,w}) = \ker f^A_{U_c^{-1} w, V_c^{-1} z}.$$

In the source	Mathematical meaning

Conjugate-linear swapped underlying sets
`(∀ (x y : H) (hx : x ≠ 0) (hy : y ≠ 0),`
`  (e.toEquiv (normalPartPoint S hS x y hx hy)).val.val =`
`  {b : T | inner ℂ (Uc x) ((b : K →L[ℂ] K) (Vc y)) = 0}) ∧`
`(∀ (z w : K) (hz : z ≠ 0) (hw : w ≠ 0),`
`  (e.toEquiv.symm (normalPartPoint T hT z w hz hw)).val.val =`
`  {a : S | inner ℂ (Vc.symm z) ((a : H →L[ℂ] H) (Uc.symm w)) = 0})) ∧`
	The underlying subsets satisfy the same two equations in (5), for all the same nonzero pairs:
	$$e(\ker f^A_{x,y}) = \ker f^B_{V_c y, U_c x},$$
	$$e^{-1}(\ker f^B_{z,w}) = \ker f^A_{U_c^{-1} w, V_c^{-1} z}.$$
	The orientation and actual inverse are unchanged.

Conjugate-linear operator image
`InnerProductSpace.conjugateAdjointSandwich Uc.toContinuousLinearMap Vc.toContinuousLinearMap ''`
`  (T : Set (K →L[ℂ] K)) = (S : Set (H →L[ℂ] H))) := by`
	In either orientation of the conjugate-linear case,
	$$V_c^\dagger B U_c = \{V_c^\dagger b U_c : b \in B\} = A \subseteq \mathcal{B}(H).$$
	The dagger denotes the conjugate adjoint. Each displayed composite is a bounded complex-linear operator.

Adjoint composite
`def adjointSandwich (U V : H →L[𝕜] K) (b : K →L[𝕜] K) : H →L[𝕜] H :=`
`  V.adjoint.comp (b.comp U)`
	The composite $V_c^* \circ b \circ U_c$ in the complex-linear case.

Conjugate-adjoint composite
`def conjugateAdjointSandwich (U V : H →L⋆[𝕜] K) (b : K →L[𝕜] K) : H →L[𝕜] H :=`
`  (conjugateAdjoint V).comp (b.comp U)`
	The composite $V_c^\dagger \circ b \circ U_c$ in the conjugate-linear case, with the same order of factors.

Exact surrounding source context (separate excerpts)

The following are separate exact ambient excerpts, not lines spliced into the signature. The local inverse-pair instances express genuine inverse scalar maps, not an assumption that every field automorphism is involutive.

```
local instance (σ : ℂ ≃+* ℂ) : RingHomInvPair σ.toRingHom σ.symm.toRingHom :=
  RingHomInvPair.of_ringEquiv σ
local instance (σ : ℂ ≃+* ℂ) : RingHomInvPair σ.symm.toRingHom σ.toRingHom :=
  RingHomInvPair.of_ringEquiv_symm σ
```

The ambient declarations are active here. The local scalar variables of the earlier `Evaluation` and `SameFunctions` sections have gone out of scope before the selected theorem; they are not additional inputs.

```
universe u v
variable {H : Type u} [NormedAddCommGroup H] [InnerProductSpace ℂ H] [CompleteSpace H]
variable {K : Type v} [NormedAddCommGroup K] [InnerProductSpace ℂ K] [CompleteSpace K]
variable (S : NonUnitalStarSubalgebra ℂ (H →L[ℂ] H))
variable (T : NonUnitalStarSubalgebra ℂ (K →L[ℂ] K))
variable [IsClosed (S : Set (H →L[ℂ] H))] [IsClosed (T : Set (K →L[ℂ] K))]
variable (hS : NonUnitalCStarRepresentation.IsIrreducible (NonUnitalStarSubalgebraClass.subtype
S))
variable (hT : NonUnitalCStarRepresentation.IsIrreducible (NonUnitalStarSubalgebraClass.subtype
T))
variable (e : ClosureSpace.FiniteHomeomorph
  ((NormedSpace.geometricStructureClosure ℂ S).subspace (normalPart S))
  ((NormedSpace.geometricStructureClosure ℂ T).subspace (normalPart T)))
```

Related exact source for the reading guide

Conjugate-linear operator image · Fixed source

```
      InnerProductSpace.conjugateAdjointSandwich Uc.toContinuousLinearMap
Vc.toContinuousLinearMap ''
        (T : Set (K →L[ℂ] K)) = (S : Set (H →L[ℂ] H))) := by
```

Conjugate-adjoint composite · Fixed source

```
def conjugateAdjointSandwich (U V : H →L⋆[𝕜] K) (b : K →L[𝕜] K) : H →L[𝕜] H :=
  (conjugateAdjoint V).comp (b.comp U)
```

Adjoint composite · Fixed source

```
def adjointSandwich (U V : H →L[𝕜] K) (b : K →L[𝕜] K) : H →L[𝕜] H :=
  V.adjoint.comp (b.comp U)
```