

# The Stone-Čech Compactification

Mike Stannett

February 6, 2026

## Contents

<b>1</b>	<b><math>C^*</math>-embedding</b>	<b>4</b>
<b>2</b>	<b>Weak topologies</b>	<b>9</b>
2.1	Tychonov spaces carry the weak topology induced by $C^*(X)$	14
2.2	A topology is a weak topology if it admits a continuous function set that separates points from closed sets . . . . .	19
2.3	A product topology is the weak topology induced by its projections if the projections separate points from closed sets. . .	21
2.4	Evaluation is an embedding for weak topologies . . . . .	23
<b>3</b>	<b>Compactification</b>	<b>26</b>
3.1	Definition . . . . .	26
3.2	Example: The Alexandroff compactification of a non-compact locally-compact Hausdorff space . . . . .	26
3.3	Example: The closure of a subset of a compact space . . . . .	27
3.4	Example: A compact space is a compactification of itself . . .	27
3.5	Example: A closed non-trivial real interval is a compactification of its interior . . . . .	27
<b>4</b>	<b>The Stone-Čech compactification of a Tychonov space</b>	<b>28</b>
4.1	Definition of $\beta X$ . . . . .	33
4.2	$\beta X$ is a compactification of $X$ . . . . .	34
4.3	Evaluation is a $C^*$ -embedding of $X$ into $\beta X$ . . . . .	35
4.4	The Stone-Čech Extension Property: Any continuous map from $X$ to a compact Hausdorff space $K$ extends uniquely to a continuous map from $\beta X$ to $K$ . . . . .	38

Building on parts of HOL-Analysis, we provide mathematical components for work on the Stone-Čech compactification. The main concepts covered are:  $C^*$ -embedding, weak topologies and compactification, focusing in particular on the Stone-Čech compactification of an arbitrary Tychonov space  $X$ . Many of the proofs given here derive from those of Willard (*General*

*Topology*, 1970, Addison-Wesley) and Walker (*The Stone-Čech Compactification*, 1974, Springer-Verlag).

Using traditional topological proof strategies we define the evaluation and projection functions for product spaces, and show that product spaces carry the weak topology induced by their projections whenever those projections separate points both from each other and from closed sets.

In particular, we show that the evaluation map from an arbitrary Tychonov space  $X$  into  $\beta X$  is a dense  $C^*$ -embedding, and then verify the Stone-Čech Extension Property: any continuous map from  $X$  to a compact Hausdorff space  $K$  extends uniquely to a continuous map from  $\beta X$  to  $K$ .

**theory** *Stone-Cech*

**imports** *HOL.Topological-Spaces*

*HOL.Set*

*HOL-Analysis.Urysohn*

**begin**

Concrete definitions of finite intersections and arbitrary unions, and their relationship to the Analysis.Abstract\_Topology versions.

**definition** *finite-intersections-of* :: 'a set set  $\Rightarrow$  'a set set

**where** *finite-intersections-of*  $S = \{ (\bigcap F) \mid F . F \subseteq S \wedge \text{finite}' F \}$

**definition** *arbitrary-unions-of* :: 'a set set  $\Rightarrow$  'a set set

**where** *arbitrary-unions-of*  $S = \{ (\bigcup F) \mid F . F \subseteq S \}$

**lemma** *generator-imp-arbitrary-union*:

**shows**  $S \subseteq \text{arbitrary-unions-of } S$

**unfolding** *arbitrary-unions-of-def* **by** *blast*

**lemma** *finite-intersections-container*:

**shows**  $\forall s \in \text{finite-intersections-of } S . \bigcup S \cap s = s$

**unfolding** *finite-intersections-of-def* **by** *blast*

**lemma** *generator-imp-finite-intersection*:

**shows**  $S \subseteq \text{finite-intersections-of } S$

**unfolding** *finite-intersections-of-def* **by** *blast*

**lemma** *finite-intersections-equiv*:

**shows**  $(\text{finite}' \text{ intersection-of } (\lambda x . x \in S)) U \longleftrightarrow U \in \text{finite-intersections-of } S$

**unfolding** *finite-intersections-of-def intersection-of-def*

**by** *auto*

**lemma** *arbitrary-unions-equiv*:

**shows**  $(\text{arbitrary union-of } (\lambda x . x \in S)) U \longleftrightarrow U \in \text{arbitrary-unions-of } S$

**unfolding** *arbitrary-unions-of-def union-of-def arbitrary-def*

by *auto*

Supplementary information about topological bases and the topologies they generate

**definition** *base-generated-on-by* :: 'a set  $\Rightarrow$  'a set set  $\Rightarrow$  'a set set  
where *base-generated-on-by*  $X S = \{ X \cap s \mid s . s \in \text{finite-intersections-of } S \}$

**definition** *opens-generated-on-by* :: 'a set  $\Rightarrow$  'a set set  $\Rightarrow$  'a set set  
where *opens-generated-on-by*  $X S = \text{arbitrary-unions-of } (\text{base-generated-on-by } X S)$

**definition** *base-generated-by* :: 'a set set  $\Rightarrow$  'a set set  
where *base-generated-by*  $S = \text{finite-intersections-of } S$

**definition** *opens-generated-by* :: 'a set set  $\Rightarrow$  'a set set  
where *opens-generated-by*  $S = \text{arbitrary-unions-of } (\text{base-generated-by } S)$

**lemma** *generators-are-basic*:  
shows  $S \subseteq \text{base-generated-by } S$   
unfolding *base-generated-by-def* *finite-intersections-of-def*  
by *blast*

**lemma** *basics-are-open*:  
shows *base-generated-by*  $S \subseteq \text{opens-generated-by } S$   
unfolding *opens-generated-by-def* *arbitrary-unions-of-def*  
by *blast*

**lemma** *generators-are-open*:  
shows  $S \subseteq \text{opens-generated-by } S$   
using *generators-are-basic* *basics-are-open*  
by *blast*

**lemma** *generated-topospace*:  
assumes  $T = \text{topology-generated-by } S$   
shows *topospace*  $T = \bigcup S$   
using *assms* by *simp*

**lemma** *base-generated-by-alt*:  
shows *base-generated-by*  $S = \text{base-generated-on-by } (\bigcup S) S$   
unfolding *base-generated-by-def* *base-generated-on-by-def*  
using *finite-intersections-container[of S]*  
by *auto*

**lemma** *opens-generated-by-alt*:  
shows *opens-generated-by*  $S = \text{arbitrary-unions-of } (\text{finite-intersections-of } S)$   
unfolding *opens-generated-by-def* *base-generated-by-def*  
by *simp*

**lemma** *opens-generated-unfolded*:

**shows** *opens-generated-by*  $S = \{\bigcup A \mid A . A \subseteq \{\bigcap B \mid B . \text{finite}' B \wedge B \subseteq S\}\}$   
**apply** (*simp add: opens-generated-by-alt*)  
**unfolding** *finite-intersections-of-def arbitrary-unions-of-def*  
**by** *blast*

**lemma** *opens-eq-generated-topology:*

**shows** *openin (topology-generated-by S) U*  $\longleftrightarrow$   $U \in \text{opens-generated-by } S$

**proof** –

**have** *openin (topology-generated-by S) = arbitrary union-of finite' intersection-of*  
 $(\lambda x. x \in S)$

**by** (*metis generate-topology-on-eq istopology-generate-topology-on topology-inverse'*)

**also have**  $\dots = \text{arbitrary union-of } (\lambda U . U \in \text{finite-intersections-of } S)$

**using** *finite-intersections-equiv[of S]* **by** *presburger*

**also have**  $\dots = (\lambda U . U \in \text{arbitrary-unions-of } (\text{finite-intersections-of } S))$

**using** *arbitrary-unions-equiv[of finite-intersections-of S]* **by** *presburger*

**finally show** *?thesis*

**using** *opens-generated-by-alt* **by** *auto*

**qed**

## 1 $C^*$ -embedding

**abbreviation** *continuous-from-to*

$:: 'a \text{ topology} \Rightarrow 'b \text{ topology} \Rightarrow ('a \Rightarrow 'b) \text{ set } (\langle \text{cts}[-, -] \rangle)$

**where** *continuous-from-to X Y*  $\equiv \{ f . \text{continuous-map } X Y f \}$

**abbreviation** *continuous-from-to-extensional*

$:: 'a \text{ topology} \Rightarrow 'b \text{ topology} \Rightarrow ('a \Rightarrow 'b) \text{ set } (\langle \text{cts}_E[-, -] \rangle)$

**where** *continuous-from-to-extensional X Y*  $\equiv (\text{topspace } X \rightarrow_E \text{topspace } Y) \cap \text{cts}[X, Y]$

**abbreviation** *continuous-maps-from-to-shared-where*  $::$

$'a \text{ topology} \Rightarrow ('b \text{ topology} \Rightarrow \text{bool}) \Rightarrow ('a \Rightarrow 'b) \text{ set} \Rightarrow \text{bool} (\langle \text{cts}'\text{-on - to}'\text{-shared -} \rangle)$

**where** *continuous-maps-from-to-shared-where X P*  
 $\equiv (\lambda fs . (\exists Y . P Y \wedge fs \subseteq \text{cts}[X, Y]))$

**definition** *dense-in*  $:: 'a \text{ topology} \Rightarrow 'a \text{ set} \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$

**where** *dense-in T A B*  $\equiv T \text{ closure-of } A = B$

**lemma** *dense-in-closure:*

**assumes** *dense-in T A B*

**shows** *dense-in (subtopology T B) A B*

**by** (*metis Int-UNIV-right Int-absorb Int-commute assms closure-of-UNIV closure-of-restrict*

*closure-of-subtopology dense-in-def topspace-subtopology*)

**abbreviation** *dense-embedding*  $:: 'a \text{ topology} \Rightarrow 'b \text{ topology} \Rightarrow ('a \Rightarrow 'b) \Rightarrow \text{bool}$

**where** *dense-embedding small big f*  $\equiv (\text{embedding-map small big } f)$

$\wedge$  dense-in big (f'topospace small) (topospace big)

**lemma** *continuous-maps-on-dense-subset*:

**assumes** (cts-on X to-shared Hausdorff-space) {f,g}

**and** dense-in X D (topospace X)

**and**  $\forall x \in D . f x = g x$

**shows**  $\forall x \in \text{topospace } X . f x = g x$

**proof** –

**obtain** Y **where** continuous-map X Y f  $\wedge$  continuous-map X Y g  $\wedge$  Hausdorff-space Y

**using** *assms(1)* **by** *auto*

**thus** ?thesis **using** *assms* dense-in-def forall-in-closure-of-eq **by** *fastforce*  
**qed**

**lemma** *continuous-map-on-dense-embedding*:

**assumes** (cts-on X to-shared Hausdorff-space) {f,g}

**and** dense-embedding D X e

**and**  $\forall d \in \text{topospace } D . (f \circ e) d = (g \circ e) d$

**shows**  $\forall x \in \text{topospace } X . f x = g x$

**using** *assms* continuous-maps-on-dense-subset[of f g X e 'topospace D]

**unfolding** dense-in-def **by** *fastforce*

**definition** *range'* :: 'a topology  $\Rightarrow$  ('a  $\Rightarrow$  real)  $\Rightarrow$  real set

**where** *range'* X f = euclideanreal closure-of (f 'topospace X)

**abbreviation** *fbounded-below* :: ('a  $\Rightarrow$  real)  $\Rightarrow$  'a topology  $\Rightarrow$  bool

**where** *fbounded-below* f X  $\equiv$  ( $\exists m . \forall y \in \text{topospace } X . f y \geq m$ )

**abbreviation** *fbounded-above* :: ('a  $\Rightarrow$  real)  $\Rightarrow$  'a topology  $\Rightarrow$  bool

**where** *fbounded-above* f X  $\equiv$  ( $\exists M . \forall y \in \text{topospace } X . f y \leq M$ )

**abbreviation** *fbounded* :: ('a  $\Rightarrow$  real)  $\Rightarrow$  'a topology  $\Rightarrow$  bool

**where** *fbounded* f X  $\equiv$  ( $\exists m M . \forall y \in \text{topospace } X . m \leq f y \wedge f y \leq M$ )

**lemma** *fbounded-iff*:

**shows** *fbounded* f X  $\longleftrightarrow$  *fbounded-below* f X  $\wedge$  *fbounded-above* f X

**by** *auto*

**abbreviation** *c-of* :: 'a topology  $\Rightarrow$  ('a  $\Rightarrow$  real) set ( $\langle C(-) \rangle$ )

**where** *C(X)*  $\equiv$  { f . continuous-map X euclideanreal f }

**abbreviation** *cstar-of* :: 'a topology  $\Rightarrow$  ('a  $\Rightarrow$  real) set ( $\langle C*(-) \rangle$ )

**where** *C\* X*  $\equiv$  { f | f . f  $\in$  *c-of* X  $\wedge$  *fbounded* f X }

**definition** *cstar-id* :: 'a topology  $\Rightarrow$  ('a  $\Rightarrow$  real)  $\Rightarrow$  'a  $\Rightarrow$  real

**where** *cstar-id* X = ( $\lambda f \in C* X . f$ )

**abbreviation** *c-embedding* :: 'a topology  $\Rightarrow$  'b topology  $\Rightarrow$  ('a  $\Rightarrow$  'b)  $\Rightarrow$  bool  
**where** *c-embedding*  $S X e \equiv$  *embedding-map*  $S X e \wedge$   
 $(\forall fS \in C(S) . \exists fX \in C(X) . \forall x \in \text{topspace } S . fS x =$   
 $fX (e x))$

**abbreviation** *cstar-embedding* :: 'a topology  $\Rightarrow$  'b topology  $\Rightarrow$  ('a  $\Rightarrow$  'b)  $\Rightarrow$  bool  
**where** *cstar-embedding*  $S X e \equiv$  *embedding-map*  $S X e \wedge$   
 $(\forall fS \in C^*(S) . \exists fX \in C^*(X) . \forall x \in \text{topspace } S . fS x =$   
 $fX (e x))$

**definition** *c-embedded* :: 'a topology  $\Rightarrow$  'b topology  $\Rightarrow$  bool  
**where** *c-embedded*  $S X \equiv$   $(\exists e . \text{c-embedding } S X e)$

**definition** *cstar-embedded* :: 'a topology  $\Rightarrow$  'b topology  $\Rightarrow$  bool  
**where** *cstar-embedded*  $S X \equiv$   $(\exists e . \text{cstar-embedding } S X e)$

**lemma** *bounded-range-iff-fbounded*:

**assumes**  $f \in C X$   
**shows**  $\text{bounded } (f \text{ 'topspace } X) \longleftrightarrow \text{fbounded } f X$   
**(is ?lhs  $\longleftrightarrow$  ?rhs)**

**proof**

**assume** ?lhs  
**then obtain**  $x e$  **where**  $\forall y \in f \text{ 'topspace } X . \text{dist } x y \leq e$   
**using** *bounded-def*[of  $f \text{ 'topspace } X$ ] **by** *auto*  
**hence**  $\forall y \in f \text{ 'topspace } X . y \in \{ (x-e) .. (x+e) \}$   
**using** *dist-real-def* **by** *auto*  
**thus** ?rhs **by** *auto*

**next**

**assume** ?rhs  
**then obtain**  $m M$  **where**  $\forall y \in f \text{ 'topspace } X . y \in \{ m..M \}$  **by** *auto*  
**thus** ?lhs **using** *bounded-closed-interval*[of  $m M$ ] *subsetI* *bounded-subset*  
**by** *meson*

**qed**

Combinations of functions in  $C(X)$  and  $C^*(X)$

**abbreviation** *fconst* ::  $real \Rightarrow 'a \Rightarrow real$   
**where** *fconst*  $v \equiv (\lambda x . v)$

**definition** *fmin* :: ('a  $\Rightarrow real$ )  $\Rightarrow$  ('a  $\Rightarrow real$ )  $\Rightarrow$  ('a  $\Rightarrow real$ )  
**where** *fmin*  $f g = (\lambda x . \text{min } (f x) (g x))$

**definition** *fmax* :: ('a  $\Rightarrow real$ )  $\Rightarrow$  ('a  $\Rightarrow real$ )  $\Rightarrow$  ('a  $\Rightarrow real$ )  
**where** *fmax*  $f g = (\lambda x . \text{max } (f x) (g x))$

**definition** *fmid* :: ('a  $\Rightarrow real$ )  $\Rightarrow$  ('a  $\Rightarrow real$ )  $\Rightarrow$  ('a  $\Rightarrow real$ )  $\Rightarrow$  'a  $\Rightarrow real$   
**where** *fmid*  $f m M = \text{fmax } m (\text{fmin } f M)$

**definition**  $fbound :: ('a \Rightarrow real) \Rightarrow real \Rightarrow real \Rightarrow 'a \Rightarrow real$   
**where**  $fbound\ f\ m\ M = fmid\ f\ (fconst\ m)\ (fconst\ M)$

**lemma**  $fmin-cts$ :  
**assumes**  $(f \in C\ X) \wedge (g \in C\ X)$   
**shows**  $fmin\ f\ g \in C\ X$   
**using**  $assms\ continuous-map-real-min[of\ X\ f\ g]\ fmin-def[of\ f\ g]$  **by**  $auto$

**lemma**  $fmax-cts$ :  
**assumes**  $(f \in C\ X) \wedge (g \in C\ X)$   
**shows**  $fmax\ f\ g \in C\ X$   
**using**  $assms\ continuous-map-real-max[of\ X\ f\ g]\ fmax-def[of\ f\ g]$  **by**  $auto$

**lemma**  $fmid-cts$ :  
**assumes**  $(f \in C\ X) \wedge (m \in C\ X) \wedge (M \in C\ X)$   
**shows**  $fmid\ f\ m\ M \in C\ X$   
**unfolding**  $fmid-def$  **using**  $assms\ fmin-cts[of\ f\ X\ M]\ fmax-cts[of\ m\ X\ (fmin\ f\ M)]$   
**by**  $auto$

**lemma**  $fconst-cts$ :  
**shows**  $fconst\ v \in C\ X$   
**by**  $simp$

**lemma**  $fbound-cts$ :  
**assumes**  $f \in C\ X$   
**shows**  $fbound\ f\ m\ M \in C\ X$   
**unfolding**  $fbound-def$   
**using**  $assms\ fmid-cts[of\ f\ X\ fconst\ m\ fconst\ M]\ fconst-cts[of\ m\ X]\ fconst-cts[of\ M\ X]$   
**by**  $auto$

Bounded and bounding functions

**lemma**  $fconst-bounded$ :  
**shows**  $fbounded\ (fconst\ v)\ X$   
**by**  $auto$

**lemma**  $fmin-bounded-below$ :  
**assumes**  $fbounded-below\ f\ X \wedge fbounded-below\ g\ X$   
**shows**  $fbounded-below\ (fmin\ f\ g)\ X$   
**proof** –  
**obtain**  $mf\ mg$  **where**  $\forall\ y \in\ topspace\ X . f\ y \geq\ mf \wedge g\ y \geq\ mg$  **using**  $assms$  **by**  $auto$   
**hence**  $\forall\ y \in\ topspace\ X . fmin\ f\ g\ y \geq\ min\ mf\ mg$  **unfolding**  $fmin-def\ min-def$   
**by**  $auto$   
**thus**  $?thesis$  **by**  $auto$

**qed**

**lemma** *fmax-bounded-above*:

**assumes** *fbounded-above*  $f X \wedge$  *fbounded-above*  $g X$

**shows** *fbounded-above*  $(fmax\ f\ g)\ X$

**proof** –

**obtain**  $mf\ mg$  **where**  $\forall y \in topspace\ X . f\ y \leq mf \wedge g\ y \leq mg$  **using** *assms* **by**  
*auto*

**hence**  $\forall y \in topspace\ X . fmax\ f\ g\ y \leq max\ mf\ mg$  **unfolding** *fmax-def* *max-def*  
**by** *auto*

**thus** *?thesis* **by** *auto*

**qed**

**lemma** *fmid-bounded*:

**assumes** *fbounded*  $m X \wedge$  *fbounded*  $M X$

**shows** *fbounded*  $(fmid\ f\ m\ M)\ X$

**proof** –

**obtain**  $mmin\ mmax\ Mmin\ Mmax$

**where**  $\forall y \in topspace\ X . mmin \leq m\ y \wedge m\ y \leq mmax \wedge Mmin \leq M\ y \wedge M$   
 $y \leq Mmax$

**using** *assms* **by** *blast*

**hence**  $\forall y \in topspace\ X . min\ mmin\ Mmin \leq (fmid\ f\ m\ M\ y) \wedge (fmid\ f\ m\ M\ y)$   
 $\leq max\ mmax\ Mmax$

**unfolding** *fmid-def* *fmax-def* *fmin-def* *max-def* *min-def* **by** *auto*

**thus** *?thesis* **by** *auto*

**qed**

**lemma** *fbound-bounded*:

**shows** *fbounded*  $(fbound\ f\ m\ M)\ X$

**using** *fmid-bounded*[*of*  $X\ fconst\ m\ fconst\ M$ ] *fconst-bounded*[*of*  $X\ m$ ] *fconst-bounded*[*of*  
 $X\ M$ ]

**unfolding** *fbound-def* **by** *simp*

Members of  $C^*(X)$

**lemma** *fconst-cstar*:

**shows** *fconst*  $v \in C^*\ X$

**using** *fconst-cts*[*of*  $v\ X$ ] *fconst-bounded*[*of*  $X\ v$ ]

**by** *auto*

**lemma** *fbound-cstar*:

**assumes**  $f \in C\ X$

**shows** *fbound*  $f\ m\ M \in C^*\ X$

**using** *assms* *fbound-cts*[*of*  $f\ X\ m\ M$ ] *fbound-bounded*[*of*  $X\ f\ m\ M$ ]

**by** *auto*

**lemma** *cstar-nonempty*:

**shows**  $\{\} \neq C^*\ X$

**using** *fconst-cstar* **by** *blast*

## 2 Weak topologies

**definition** *funcset-types* :: 'a set  $\Rightarrow$  ('b  $\Rightarrow$  'a  $\Rightarrow$  'c)  $\Rightarrow$  ('b  $\Rightarrow$  'c topology)  $\Rightarrow$  'b set  $\Rightarrow$  bool  
**where** *funcset-types* S F T I = ( $\forall$  i  $\in$  I . F i  $\in$  S  $\rightarrow$  *topspace* (T i))

**lemma** *cstar-types*:

**shows** *funcset-types* (*topspace* X) (*cstar-id* X) ( $\lambda$ f  $\in$  C\* X . *euclideanreal*) (C\* X)

**unfolding** *funcset-types-def*  
**by** *simp*

**lemma** *cstar-types-restricted*:

**shows** *funcset-types* (*topspace* X) (*cstar-id* X)  
( $\lambda$ f  $\in$  C\* X . (*subtopology euclideanreal* (*range'* X f))) (C\* X)

**proof** –

**have**  $\forall$  f  $\in$  C\* X . f ' *topspace* X  $\subseteq$  *range'* X f **using** *range'-def*[of X]  
**by** (*metis closedin-subtopology-refl closedin-topospace closure-of-subset*  
*topospace-euclidean-subtopology*)

**thus** ?thesis **unfolding** *funcset-types-def*  
**by** (*simp add: image-subset-iff cstar-id-def*)

**qed**

**definition** *inverse'* :: ('a  $\Rightarrow$  'b)  $\Rightarrow$  'a set  $\Rightarrow$  'b set  $\Rightarrow$  'a set  
**where** *inverse'* f source target = { x  $\in$  source . f x  $\in$  target }

**lemma** *inverse'-alt*:

**shows** *inverse'* f s t = (f - ' t)  $\cap$  s  
**using** *inverse'-def*[of f s t] **by** *auto*

**definition** *open-sets-induced-by-func* :: ('a  $\Rightarrow$  'b)  $\Rightarrow$  'a set  $\Rightarrow$  'b topology  $\Rightarrow$  'a set set

**where** *open-sets-induced-by-func* f source T  
= { (*inverse'* f source V) | V . *openin* T V  $\wedge$  f  $\in$  source  $\rightarrow$  *topspace* T }

**definition** *weak-generators* :: 'a set  $\Rightarrow$  ('b  $\Rightarrow$  'a  $\Rightarrow$  'c)  $\Rightarrow$  ('b  $\Rightarrow$  'c topology)  $\Rightarrow$  'b set  $\Rightarrow$  'a set set

**where** *weak-generators* source funcs tops index  
=  $\bigcup$  { *open-sets-induced-by-func* (funcs i) source (tops i) | i . i  $\in$  index }

**definition** *weak-base* :: 'a set  $\Rightarrow$  ('b  $\Rightarrow$  'a  $\Rightarrow$  'c)  $\Rightarrow$  ('b  $\Rightarrow$  'c topology)  $\Rightarrow$  'b set  $\Rightarrow$  'a set set

**where** *weak-base* source funcs tops index = *base-generated-by* (*weak-generators* source funcs tops index)

**definition** *weak-opens* :: 'a set  $\Rightarrow$  ('b  $\Rightarrow$  'a  $\Rightarrow$  'c)  $\Rightarrow$  ('b  $\Rightarrow$  'c topology)  $\Rightarrow$  'b set  $\Rightarrow$  'a set set

**where** *weak-opens source funcs tops index* = *opens-generated-by* (*weak-generators source funcs tops index*)

**definition** *weak-topology* :: 'a set  $\Rightarrow$  ('b  $\Rightarrow$  'a  $\Rightarrow$  'c)  $\Rightarrow$  ('b  $\Rightarrow$  'c topology)  $\Rightarrow$  'b set  $\Rightarrow$  'a topology

**where** *weak-topology source funcs tops index*  
= *topology-generated-by* (*weak-generators source funcs tops index*)

**lemma** *weak-topology-alt*:

**shows** *openin* (*weak-topology* *S F T I*) *U*  $\longleftrightarrow$  *U*  $\in$  *weak-opens* *S F T I*

**using** *weak-topology-def*[*of S F T I*] *weak-opens-def*[*of S F T I*]

*opens-eq-generated-topology*[*of weak-generators S F T I U*]

**by** *auto*

**lemma** *weak-generators-exist-for-each-point-and-axis*:

**assumes** *x*  $\in$  *S*

**and** *funcset-types* *S F T I*

**and** *i*  $\in$  *I*

**and** *b* = *inverse'* (*F i*) *S* (*topspace* (*T i*))

**and** *F i*  $\in$  *S*  $\rightarrow$  *topspace* (*T i*)

**shows** *x*  $\in$  *b*  $\wedge$  *b*  $\in$  *weak-generators* *S F T I*

**proof** –

**have** *xprops*: *x*  $\in$  {*r*  $\in$  *S* . *F i r*  $\in$  *topspace* (*T i*)}

**using** *assms*(2) *funcset-types-def*[*of S F T I*] *assms*(3) *assms*(1)

**by** *blast*

**hence** *part1*: *x*  $\in$  *b* **using** *assms*(4) *inverse'-def*[*of F i S topspace* (*T i*)]

**by** *auto*

**have** *openin* (*T i*) (*topspace* (*T i*)) **by** *simp*

**hence** *b*  $\in$  *open-sets-induced-by-func* (*F i*) *S* (*T i*)

**using** *open-sets-induced-by-func-def*[*of F i S T i*] *assms*(4) *assms*(5)

*inverse'-def*[*of F i S topspace* (*T i*)] *xprops*

**by** *auto*

**thus** *?thesis* **using** *part1* *weak-generators-def*[*of S F T I*] *assms*(3) **by** *auto*

**qed**

**lemma** *weak-generators-topspace*:

**assumes** *W* = *weak-topology* *S F T I*

**shows** *topspace* *W* =  $\bigcup$  (*weak-generators* *S F T I*)

**using** *weak-topology-def*[*of S F T I*] *assms* **by** *simp*

**lemma** *weak-topology-topspace*:

**assumes** *W* = *weak-topology* *S F T I*

**and** *funcset-types* *S F T I*

**shows** (*I* = {}  $\rightarrow$  *topspace* *W* = {})  $\wedge$  (*I*  $\neq$  {}  $\rightarrow$  *topspace* *W* = *S*)

**proof** (*cases* *I* = {})

**case** *True*

**hence** *weak-generators*  $S F T I = \{\}$  **using** *assms(1)* *weak-generators-def*[of  $S F T I$ ] **by** *auto*  
**hence** *topspace*  $W = \{\}$  **using** *assms(1)* *weak-generators-topspace*[of  $W S F T I$ ] **by** *simp*  
**then show** *?thesis* **using** *True* **by** *simp*  
**next**  
**case** *False*  
**then obtain**  $i$  **where** *iprops*:  $i \in I$  **by** *auto*  
**hence**  $(F i) ' S \subseteq \text{topspace } (T i)$   
**using** *assms(2)* *unfolding* *funcset-types-def* **by** *auto*  
**hence**  $\text{inverse}' (F i) S (\text{topspace } (T i)) = S$   
**using** *inverse'-def*[of  $F i S \text{topspace } (T i)$ ] **by** *auto*  
**moreover have** *openin*  $(T i) (\text{topspace } (T i))$  **using** *weak-generators-def* **by** *simp*  
**ultimately have**  $S \in \text{open-sets-induced-by-func } (F i) S (T i)$   
**using** *open-sets-induced-by-func-def*[of  $F i S T i$ ] *assms(2)* *iprops* *unfolding* *funcset-types-def*  
**by** *auto*  
**hence**  $S \in \text{weak-generators } S F T I$   
**using** *weak-generators-def*[of  $S F T I$ ] *iprops* **by** *auto*  
**hence**  $S \subseteq \text{topspace } W$   
**using** *weak-generators-topspace*[of  $W S F T I$ ] *assms* **by** *auto*

**moreover have** *topspace*  $W \subseteq S$   
**proof** –  
**have** *openin*  $W (\text{topspace } W)$  **by** *auto*  
**hence** *topspace*  $W \in \text{opens-generated-by } (\text{weak-generators } S F T I)$   
**using** *assms(1)* *unfolding* *weak-topology-def*  
**using** *opens-eq-generated-topology*[of *weak-generators*  $S F T I$  *topspace*  $W$ ]  
**by** *simp*  
**then obtain**  $A$  **where** *topspace*  $W = \bigcup A \wedge A \subseteq \{\bigcap B \mid B. \text{finite}' B \wedge B \subseteq \text{weak-generators } S F T I\}$   
**using** *opens-generated-unfolded*[of *weak-generators*  $S F T I$ ]  
**by** *auto*  
**thus** *?thesis* **using** *assms(2)*  
**unfolding** *weak-generators-def* *open-sets-induced-by-func-def* *inverse'-def* *funcset-types-def*  
**by** *blast*  
**qed**  
**ultimately show** *?thesis* **using** *False* **by** *auto*  
**qed**

**lemma** *weak-opens-nhood-base*:  
**assumes**  $W = \text{weak-topology } S F T I$   
**and** *openin*  $W U$   
**and**  $x \in U$   
**shows**  $\exists b \in \text{weak-base } S F T I . x \in b \wedge b \subseteq U$   
**proof** –  
**define**  $G$  **where**  $G = \text{weak-generators } S F T I$

**hence**  $Wprops: U \in \text{opens-generated-by } G$   
**using**  $\text{weak-topology-def}[ \text{of } S F T I ] \text{ opens-eq-generated-topology}[ \text{of } G ] \text{ assms}(1)$   
 $\text{assms}(2)$   
**by**  $\text{presburger}$   
**then obtain**  $B$  **where**  $Bprops: B \subseteq \text{base-generated-by } G \wedge U = \bigcup B$   
**unfolding**  $\text{opens-generated-by-def arbitrary-unions-of-def}$  **by**  $\text{auto}$   
**then obtain**  $b$  **where**  $b \in \text{base-generated-by } G \wedge x \in b$   
**using**  $\text{assms}(3)$  **by**  $\text{blast}$   
**thus**  $?thesis$  **using**  $G\text{-def weak-base-def}[ \text{of } S F T I ]$   
**by**  $(\text{metis Union-iff } Bprops \text{ assms}(3) \text{ subset-eq})$   
**qed**

**lemma**  $\text{opens-generate-opens}$ :  
**assumes**  $\forall b \in S . \text{openin } T b$   
**shows**  $\forall U \in \text{opens-generated-by } S . \text{openin } T U$   
**by**  $(\text{metis assms generate-topology-on-coarsest istopology-openin openin-topology-generated-by}$   
 $\text{opens-eq-generated-topology})$

**lemma**  $\text{weak-topology-is-weakest}$ :  
**assumes**  $W = \text{weak-topology } S F T I$   
**and**  $\text{funcset-types } S F T I$   
**and**  $\text{topspace } X = \text{topspace } W$   
**and**  $\forall i \in I . \text{continuous-map } X (T i) (F i)$   
**and**  $\text{openin } W U$   
**shows**  $\text{openin } X U$   
**proof** –  
**{** **fix**  $b$  **assume**  $bprops: b \in \text{weak-generators } S F T I$   
**then obtain**  $i$  **where**  $iprops: i \in I \wedge b \in \text{open-sets-induced-by-func } (F i) S$   
 $(T i)$   
**using**  $\text{weak-generators-def}[ \text{of } S F T I ]$  **by**  $\text{auto}$   
**hence**  $Sprops: S = \text{topspace } X$   
**using**  $\text{assms}(1) \text{ assms}(2) \text{ weak-topology-topspace}[ \text{of } W S F T I ]$   
**unfolding**  $\text{funcset-types-def assms}(3)$   
**by**  $\text{auto}$   
**obtain**  $V$  **where**  $Vprops: \text{openin } (T i) V \wedge b = \text{inverse}' (F i) S V$   
**using**  $iprops \text{ open-sets-induced-by-func-def}[ \text{of } F i S T i ]$  **by**  $\text{auto}$   
**have**  $\text{cts}: \text{continuous-map } X (T i) (F i)$  **using**  $iprops \text{ assms}(4)$  **by**  $\text{auto}$   
**hence**  $\forall U . \text{openin } (T i) U \longrightarrow \text{openin } X \{x \in \text{topspace } X . F i x \in U\}$   
**unfolding**  $\text{continuous-map-def}$  **by**  $\text{simp}$   
**hence**  $\text{openin } X \{x \in \text{topspace } X . F i x \in V\}$  **using**  $Vprops$  **by**  $\text{auto}$   
**hence**  $\text{openin } X b$  **using**  $Vprops Sprops$  **unfolding**  $\text{inverse}'\text{-def}$  **by**  $\text{auto}$   
**}**  
**hence**  $\forall b \in \text{weak-generators } S F T I . \text{openin } X b$  **by**  $\text{auto}$   
**hence**  $\forall c \in \text{weak-opens } S F T I . \text{openin } X c$   
**using**  $\text{assms}(5) \text{ weak-opens-def}[ \text{of } S F T I ] \text{ opens-generate-opens}[ \text{of weak-generators}$   
 $S F T I X ]$   
**by**  $\text{auto}$   
**moreover have**  $U \in \text{weak-opens } S F T I$

```

    using assms(1) weak-topology-def[of S F T I] weak-opens-def[of S F T I]
      opens-eq-generated-topology[of weak-generators S F T I U] assms(5)
    by auto
    ultimately show ?thesis by auto
  qed

lemma weak-generators-continuous:
  assumes W = weak-topology S F T I
  and     funcset-types S F T I
  and     i ∈ I
  shows   continuous-map W (T i) (F i)
  proof -
    have S = topspace W using assms(1) assms(2) assms(3) weak-topology-topspace[of
      W S F T I]
      unfolding funcset-types-def by auto
    hence F i ∈ topspace W → topspace (T i)
      using assms funcset-types-def[of S F T I] by auto
    moreover have  $\forall V . \text{openin } (T i) V \longrightarrow \text{openin } W \{x \in \text{topspace } W. (F i) x \in V\}$ 
    proof -
      { fix V assume Vprops: openin (T i) V
        { assume hyp: inverse' (F i) (topspace W) V ≠ {}
          have  $\{x \in \text{topspace } W. (F i) x \in V\} = \text{inverse}' (F i) (\text{topspace } W) V$ 
            using inverse'-def[of F i topspace W V] by simp
          moreover have  $(\text{inverse}' (F i) (\text{topspace } W) V) \in \text{open-sets-induced-by-func}$ 
            (F i) S (T i)
            using Vprops assms weak-topology-topspace[of W S F T I] hyp
            unfolding open-sets-induced-by-func-def funcset-types-def
            by fastforce
          ultimately have  $\{x \in \text{topspace } W. (F i) x \in V\} \in \text{weak-generators } S F T I$ 
            using weak-generators-def[of S F T I] assms(3) by auto
          hence openin W {x ∈ topspace W. (F i) x ∈ V}
            using assms(1) weak-topology-def[of S F T I]
              generators-are-open[of weak-generators S F T I]
              opens-eq-generated-topology[of weak-generators S F T I {x ∈ topspace
                W. (F i) x ∈ V}]
            by auto
        }
      }
    hence  $\text{inverse}' (F i) (\text{topspace } W) V \neq \{\} \longrightarrow \text{openin } W \{x \in \text{topspace } W. (F i) x \in V\}$ 
      by auto
    moreover have  $\text{inverse}' (F i) (\text{topspace } W) V = \{\} \longrightarrow \text{openin } W \{x \in \text{topspace } W. (F i) x \in V\}$ 
      by (metis openin-empty inverse'-def)
    ultimately have openin W {x ∈ topspace W. (F i) x ∈ V} by auto
  }
  thus ?thesis by auto
  qed

```

**ultimately show** *?thesis* **using** *continuous-map-def* **by** *blast*  
**qed**

**lemma** *funcset-types-on-empty*:  
**shows** *funcset-types*  $\{\}$  *F T I*  
**unfolding** *funcset-types-def* **by** *simp*

**lemma** *weak-topology-on-empty*:  
**assumes**  $W = \text{weak-topology } \{\}$  *F T I*  
**shows**  $\forall U . \text{openin } W U \longleftrightarrow U = \{\}$   
**proof** –  
**have** *topspace*  $W = \{\}$   
**using** *assms(1)* *weak-topology-topspace*[of  $W \{\}$  *F T I*] *funcset-types-on-empty*[of  
*F T I*]  
**by** *blast*  
**thus** *?thesis* **by** *simp*  
**qed**

## 2.1 Tychonov spaces carry the weak topology induced by $C^*(X)$

**abbreviation** *tych-space* :: 'a topology  $\Rightarrow$  bool  
**where** *tych-space*  $X \equiv \text{t1-space } X \wedge \text{completely-regular-space } X$

**abbreviation** *compact-Hausdorff* :: 'a topology  $\Rightarrow$  bool  
**where** *compact-Hausdorff*  $X \equiv \text{compact-space } X \wedge \text{Hausdorff-space } X$

**lemma** *compact-Hausdorff-imp-tych*:  
**assumes** *compact-Hausdorff*  $K$   
**shows** *tych-space*  $K$   
**by** (*simp add: Hausdorff-imp-t1-space assms compact-Hausdorff-or-regular-imp-normal-space*  
*normal-imp-completely-regular-space-A*)

**lemma** *tych-space-imp-Hausdorff*:  
**assumes** *tych-space*  $X$   
**shows** *Hausdorff-space*  $X$   
**proof** –  
**have** *Hausdorff-space euclideanreal* **by** *auto*  
**moreover** **have**  $(0::\text{real}) \neq (1::\text{real})$  **by** *simp*  
**moreover** **have**  $(0::\text{real}) \in \text{topspace euclideanreal} \wedge (1::\text{real}) \in \text{topspace euclideanreal}$  **by** *simp*  
**ultimately** **have**  $\exists U V . \text{openin euclideanreal } U \wedge \text{openin euclideanreal } V \wedge$   
 $(0::\text{real}) \in U \wedge (1::\text{real}) \in V \wedge \text{disjnt } U V$   
**using** *Hausdorff-space-def*[of *euclideanreal*] **by** *blast*  
**then** **obtain**  $U V$   
**where**  $UV\text{props: openin euclideanreal } U \wedge \text{openin euclideanreal } V \wedge (0::\text{real})$   
 $\in U \wedge (1::\text{real}) \in V \wedge \text{disjnt } U V$   
**by** *auto*

```

{ fix x y assume xyprops: x ∈ topspace X ∧ y ∈ topspace X ∧ x ≠ y
  hence closedin X {y} ∧ x ∈ topspace X - {y}
  using assms(1) by (simp add: t1-space-closedin-finite)
  then obtain f
    where fprops: continuous-map X (top-of-set {0..1}) f ∧ f x = (0::real) ∧ f
y ∈ {1::real}
  using assms(1) completely-regular-space-def[of X] by blast
  hence freal: continuous-map X euclideanreal f ∧ f x = 0 ∧ f y = 1
  using continuous-map-into-fulltopology by auto

  define U' where U' = { v ∈ topspace X . f v ∈ U }
  define V' where V' = { v ∈ topspace X . f v ∈ V }
  have openin X U' ∧ openin X V'
    using U'-def V'-def UVprops freal continuous-map-def[of X euclideanreal f]
    by auto
  moreover have U' ∩ V' = {} using UVprops U'-def V'-def disjnt-def[of U
V] by auto
  moreover have x ∈ U' ∧ y ∈ V' using UVprops U'-def V'-def fprops xyprops
  by auto
  ultimately have ∃ U' V' . openin X U' ∧ openin X V' ∧ x ∈ U' ∧ y ∈ V'
  ∧ disjnt U' V'
    using disjnt-def[of U' V'] by auto
  }
  hence ∀ x y . x ∈ topspace X ∧ y ∈ topspace X ∧ x ≠ y
    → (∃ U' V' . openin X U' ∧ openin X V' ∧ x ∈ U' ∧ y ∈ V' ∧
disjnt U' V')
  by auto
  thus ?thesis using Hausdorff-space-def[of X] by blast
qed

```

**lemma** *cstar-range-restricted*:

```

assumes f ∈ C* X
and U ⊆ topspace euclideanreal
shows inverse' f (topspace X) U = inverse' f (topspace X) (U ∩ range' X f)
proof -
  define U' where U' = U ∩ range' X f
  hence inverse' f (topspace X) U' ⊆ inverse' f (topspace X) U
  unfolding inverse'-def U'-def by auto
  moreover have inverse' f (topspace X) U ⊆ inverse' f (topspace X) U'
  proof -
    { fix x assume hyp: x ∈ inverse' f (topspace X) U
      hence f x ∈ U ∩ (f ' topspace X) unfolding inverse'-def by auto
      hence f x ∈ U ∩ range' X f
        unfolding range'-def
      by (metis Int-iff closure-of-subset-Int inf.orderE inf-top-left topspace-euclidean)
      hence x ∈ inverse' f (topspace X) U'
        unfolding inverse'-def
    }
  }

```

```

    using U'-def hyp inverse'-alt by fastforce
  }
  thus ?thesis
    by (simp add: subsetI)
qed
ultimately show ?thesis using U'-def by simp
qed

lemma weak-restricted-topology-eq-weak:
shows weak-topology (topspace X) (cstar-id X) ( $\lambda f \in C^* X . euclideanreal$ ) (C* X)
      = weak-topology (topspace X) (cstar-id X) ( $\lambda f \in C^* X . subtopology euclideanreal (range' X f)$ ) (C* X)
proof -
  define T where T = ( $\lambda f \in C^* X . euclideanreal$ )
  define T' where T' = ( $\lambda f \in C^* X . subtopology euclideanreal (range' X f)$ )
  define W where W = weak-topology (topspace X) (cstar-id X) T (C* X)
  define W' where W' = weak-topology (topspace X) (cstar-id X) T' (C* X)

  have  $\forall f \in C^* X . f \in topspace X \rightarrow topspace (T f)$ 
    using T-def unfolding continuous-map-def T-def by auto

  have generators: weak-generators (topspace X) (cstar-id X) T (C* X)
    = weak-generators (topspace X) (cstar-id X) T' (C* X)
  proof -

    have weak-generators (topspace X) (cstar-id X) T (C* X)
       $\subseteq$  weak-generators (topspace X) (cstar-id X) T' (C* X)
    proof -
      have weak-generators (topspace X) (cstar-id X) T (C* X)
         $\subseteq$  weak-generators (topspace X) (cstar-id X) T' (C* X)
      proof -
        { fix U assume Uprops: U  $\in$  weak-generators (topspace X) (cstar-id X) T (C* X)
          then obtain f where fprops: f  $\in$  (C* X)  $\wedge$  U  $\in$  open-sets-induced-by-func f (topspace X) (T f)
            unfolding weak-generators-def using cstar-id-def[of X]
              by (smt (verit) Union-iff mem-Collect-eq restrict-apply')
          then obtain V where Vprops: U = inverse' f (topspace X) V  $\wedge$  openin (T f) V
            unfolding open-sets-induced-by-func-def by blast
          hence U = inverse' f (topspace X) V by auto
          hence rtp1: U  $\subseteq$  topspace X unfolding inverse'-def by auto

          have rtp2: openin (T' f) (V  $\cap$  range' X f)
          proof -
            have openin euclideanreal V using fprops Vprops T-def by auto
            hence openin (subtopology euclideanreal (range' X f)) (V  $\cap$  range' X f)
              by (simp add: openin-subtopology-Int)
          }
        }
      }
    }
  }

```

```

    thus ?thesis using fprops T'-def by auto
  qed

  have rtp3:  $f \in \text{topspace } X \rightarrow \text{topspace } (T' f)$ 
  proof -
    have  $f' : \text{topspace } X \subseteq \text{topspace euclideanreal}$  using fprops by auto
    hence  $f' : \text{topspace } X \subseteq \text{range}' X f$  unfolding range'-def
      by (meson closure-of-subset)
    thus ?thesis using T'-def fprops by auto
  qed

  hence rtp4:  $U = \text{inverse}' f (\text{topspace } X) (V \cap \text{range}' X f)$ 
  proof -
    have  $\text{inverse}' f (\text{topspace } X) (V \cap \text{range}' X f) \subseteq U$ 
      using Vprops fprops unfolding inverse'-def by auto
    moreover have  $U \subseteq \text{inverse}' f (\text{topspace } X) (V \cap \text{range}' X f)$ 
    proof -
      { fix u assume uprops:  $u \in U$ 
        hence  $f u \in V$  using Vprops unfolding inverse'-def by auto
        moreover have  $f u \in \text{range}' X f$  using uprops rtp1 unfolding
range'-def
          by (metis closure-of-subset-Int imageI inf-top-left subset-iff
topspace-euclidean)
        ultimately have  $u \in \text{inverse}' f (\text{topspace } X) (V \cap \text{range}' X f)$ 
          unfolding inverse'-def range'-def using rtp1 uprops by force
      }
      thus ?thesis by auto
    qed
    ultimately show ?thesis by auto
  qed

  have  $U \in \text{open-sets-induced-by-func } f (\text{topspace } X) (T' f)$ 
    using rtp1 rtp2 rtp3 rtp4 unfolding open-sets-induced-by-func-def
    by blast
  hence  $U \in \text{weak-generators } (\text{topspace } X) (\text{cstar-id } X) T' (C* X)$ 
    using fprops weak-generators-def[of  $(\text{topspace } X) (\text{cstar-id } X) T' (C* X)$ ]
    cstar-id-def[of X]
    by (smt (verit, best) Sup-upper in-mono mem-Collect-eq restrict-apply')
  }
  thus ?thesis by auto
  qed
  thus ?thesis by auto
  qed

  moreover have  $\text{weak-generators } (\text{topspace } X) (\text{cstar-id } X) T' (C* X)$ 
     $\subseteq \text{weak-generators } (\text{topspace } X) (\text{cstar-id } X) T (C* X)$ 
  proof -
    { fix U assume Uprops:  $U \in \text{weak-generators } (\text{topspace } X) (\text{cstar-id } X) T'$ 
 $(C* X)$ 

```

**then obtain  $f$  where  $fprops: f \in (C* X) \wedge U \in \text{open-sets-induced-by-func } f \text{ (topspace } X) \text{ (} T' f)$**   
**unfolding  $\text{weak-generators-def}$  using  $\text{cstar-id-def[of } X]$**   
**by  $(\text{smt (verit) Union-iff mem-Collect-eq restrict-apply'})$**

**then obtain  $V$  where  $Vprops: U = \text{inverse}' f \text{ (topspace } X) V \wedge \text{openin (} T' f) V$**   
**unfolding  $\text{open-sets-induced-by-func-def}$  by  $\text{blast}$**

**have  $T' f = \text{subtopology (} T f) \text{ (topspace (} T' f))$**   
**using  $T\text{-def } T'\text{-def } fprops$  unfolding  $\text{range}'\text{-def}$  by  $\text{auto}$**   
**moreover have  $\text{openin (} T' f) V$  using  $Vprops$  by  $\text{simp}$**   
**ultimately obtain  $Vbig$  where  $Vbigprops: \text{openin (} T f) Vbig \wedge V = Vbig \cap \text{(topspace (} T' f))$**   
**using  $\text{openin-subtopology[of } T f \text{ topspace (} T' f)]$**   
**by  $\text{auto}$**

**have  $Vrestrict: Vbig \cap \text{topspace (} T' f) = Vbig \cap \text{range}' X f$**   
**using  $T'\text{-def } fprops$  by  $\text{auto}$**

**have  $Vrange: \text{inverse}' f \text{ (topspace } X) (Vbig \cap \text{range}' X f) = \text{inverse}' f \text{ (topspace } X) Vbig$**   
**proof –**  
**{ fix  $x$  assume  $x \in \text{inverse}' f \text{ (topspace } X) Vbig$**   
**hence  $x \in \text{topspace } X \wedge f x \in Vbig \cap \text{range}' X f$**   
**using  $\text{range}'\text{-def[of } X f]$**   
**by  $(\text{metis Int-iff closure-of-subset image-subset-iff inverse}'\text{-alt subset-UNIV}$**   
  
 **$\text{topspace-euclidean vimage-eq}$**   
**hence  $x \in \text{inverse}' f \text{ (topspace } X) (Vbig \cap \text{range}' X f)$  unfolding**  
 **$\text{inverse}'\text{-def}$  by  $\text{auto}$**   
**}**  
**hence  $\text{inverse}' f \text{ (topspace } X) Vbig \subseteq \text{inverse}' f \text{ (topspace } X) (Vbig \cap \text{range}' X f)$  by  $\text{auto}$**   
**thus  $?thesis$  unfolding  $\text{inverse}'\text{-def}$  by  $\text{auto}$**   
**qed**  
**hence  $U = \text{inverse}' f \text{ (topspace } X) Vbig \wedge \text{openin (} T f) Vbig$**   
**by  $(\text{simp add: } Vbigprops Vprops Vrestrict)$**   
**moreover have  $fcstar: f \in C* X$  using  $fprops$  by  $\text{simp}$**   
**ultimately have  $U \in \text{open-sets-induced-by-func } f \text{ (topspace } X) \text{ (} T f)$**   
**using  $\text{open-sets-induced-by-func-def[of } f \text{ topspace } X \text{ euclideanreal}] T\text{-def}$**   
**by  $\text{auto}$**   
**hence  $U \in \text{open-sets-induced-by-func (cstar-id } X f) \text{ (topspace } X) \text{ (} T f) \wedge f \in C* X$**   
**using  $fcstar \text{ cstar-id-def[of } X]$  by  $\text{auto}$**   
**hence  $U \in \text{weak-generators (topspace } X) \text{ (cstar-id } X) T \text{ (} C* X)$**   
**using  $fcstar$  unfolding  $\text{weak-generators-def}$  by  $\text{auto}$**   
**}**  
**thus  $?thesis$  by  $\text{auto}$**

qed  
ultimately show *?thesis* by *auto*  
qed  
thus *?thesis* by (*simp add: T-def T'-def weak-topology-def cstar-id-def*)  
qed

## 2.2 A topology is a weak topology if it admits a continuous function set that separates points from closed sets

**definition** *funcset-separates-points* :: '*a topology*  $\Rightarrow$  ('*b*  $\Rightarrow$  '*a*  $\Rightarrow$  '*c*)  $\Rightarrow$  '*b set*  $\Rightarrow$  *bool*  
**where** *funcset-separates-points* *X F I*  
 $= (\forall x \in \text{topspace } X . \forall y \in \text{topspace } X . x \neq y \longrightarrow (\exists i \in I . F i x \neq F i y))$

**definition** *funcset-separates-points-from-closed-sets* ::  
'*a topology*  $\Rightarrow$  ('*b*  $\Rightarrow$  '*a*  $\Rightarrow$  '*c*)  $\Rightarrow$  ('*b*  $\Rightarrow$  '*c topology*)  $\Rightarrow$  '*b set*  $\Rightarrow$  *bool*  
**where** *funcset-separates-points-from-closed-sets* *X F T I*  
 $= (\forall x . \forall A . \text{closedin } X A \wedge x \in (\text{topspace } X - A) \longrightarrow (\exists i \in I . F i x \notin (T i) \text{ closure-of } (F i ' A)))$

**lemma** *funcset-separates-points-from-closed-sets-imp-weak*:

**assumes** *funcset-separates-points-from-closed-sets* *X F T I*

**and**  $\forall i \in I . \text{continuous-map } X (T i) (F i)$

**and**  $W = \text{weak-topology } (\text{topspace } X) F T I$

**and** *funcset-types* (*topspace* *X*) *F T I*

**shows**  $X = W$

**proof** –

{ **fix** *U* **assume** *Uhyp: openin* *X U*

{ **fix** *x* **assume** *xhyp: x*  $\in$  *U*

**define** *A* **where**  $A = (\text{topspace } X) - U$

**have** *xinX: x*  $\in$  *topspace* *X* **using** *Uhyp xhyp openin-subset* **by** *auto*

**moreover** **have** *Aprops: closedin* *X A*  $\wedge$   $x \notin A$  **using** *Uhyp xhyp A-def* **by**

*auto*

**ultimately obtain** *i* **where** *iprops: i*  $\in$  *I*  $\wedge$   $F i x \notin (T i) \text{ closure-of } (F i ' A)$

**using** *assms(1) funcset-separates-points-from-closed-sets-def*[of *X F T I*] **by** *auto*

**define** *V* **where**  $V = \text{topspace } (T i) - (T i) \text{ closure-of } (F i ' A)$

**define** *R* **where**  $R = \{ p \in (\text{topspace } X) . F i p \in V \}$

**have** *Vopen: openin* (*T i*) *V*  $\wedge$   $F i x \in V$  **using** *iprops xinX V-def*

**by** (*metis DiffI Int-iff assms(2) closedin-closure-of continuous-map-preimage-topspace*

*openin-diff openin-topspace vimage-eq*)

**hence**  $x \in R$  **using** *R-def assms(2) xinX* **by** *simp*

**moreover** **have**  $R \subseteq U$

**proof** –

**have**  $F i ' R \subseteq V$  **using** *R-def* **by** *auto*

**hence**  $F i \cdot R \cap (T i) \text{ closure-of } (F i \cdot A) = \{\}$  **using**  $V\text{-def}$  **by**  $auto$   
**moreover have**  $F i \cdot A \subseteq (T i) \text{ closure-of } (F i \cdot A)$   
**by** ( $metis$   $Aprops$   $assms(2)$   $\text{closure-of-eq}$   $\text{continuous-map-image-closure-subset}$   $iprops$ )  
**ultimately have**  $F i \cdot R \cap (F i \cdot A) = \{\}$  **by**  $auto$   
**hence**  $R \cap A = \{\}$  **by**  $auto$   
**thus**  $?thesis$  **using**  $A\text{-def}$   $R\text{-def}$  **by**  $auto$   
**qed**  
**moreover have**  $openin\ W\ R$   
**proof** –  
**have**  $R = inverse' (F i) (topspace\ X)\ V$   
**by** ( $simp$   $add$ :  $R\text{-def}$   $inverse'\text{-def}$ )  
**hence**  $R \in open\text{-sets-induced-by-func } (F i) (topspace\ X) (T i)$   
**using**  $open\text{-sets-induced-by-func-def}$ [ $of\ F\ i\ topspace\ X\ T\ i$ ]  $Vopen$   
 $assms(2)$   $\text{continuous-map-funspace}$   $iprops$  **by**  $fastforce$   
**hence**  $R \in weak\text{-generators } (topspace\ X)\ F\ T\ I$   
**using**  $weak\text{-generators-def}$ [ $of\ topspace\ X\ F\ T\ I$ ]  $iprops$  **by**  $auto$   
**thus**  $?thesis$  **using**  $generators\text{-are-open}$ [ $of\ weak\text{-generators } (topspace\ X)\ F$   
 $T\ I$ ]  
 $opens\text{-eq-generated-topology}$ [ $of\ weak\text{-generators } (topspace\ X)\ F\ T\ I\ R$ ]  
 $assms(3)$   
**by** ( $simp$   $add$ :  $topology\text{-generated-by-Basis}$   $weak\text{-topology-def}$ )  
**qed**  
**ultimately have**  $x \in R \wedge R \subseteq U \wedge openin\ W\ R$  **by**  $auto$   
**hence**  $\exists R . x \in R \wedge R \subseteq U \wedge openin\ W\ R$  **by**  $auto$   
**}**  
**hence**  $\forall x . x \in U \longrightarrow (\exists R . x \in R \wedge R \subseteq U \wedge openin\ W\ R)$   
**by**  $auto$   
**hence**  $openin\ W\ U$  **by** ( $meson$   $openin\text{-subopen}$ )  
**}**  
**hence**  $XimpW$ :  $\forall U . openin\ X\ U \longrightarrow openin\ W\ U$  **by**  $auto$   
  
**moreover have**  $\forall U . openin\ W\ U \longrightarrow openin\ X\ U$   
**proof** –  
**have**  $topspace\ X = topspace\ W$   
**using**  $assms(3)$   $assms(4)$   $weak\text{-topology-topospace}$ [ $of\ W\ topspace\ X\ F\ T\ I$ ]  
**by** ( $metis$   $XimpW$   $openin\text{-topospace}$   $openin\text{-topospace-empty}$   $subtopology\text{-eq-discrete-topology-empty}$ )  
**thus**  $?thesis$   
**using**  $assms(3)$   $assms(4)$   $assms(2)$   $weak\text{-topology-is-weakest}$ [ $of\ W\ topspace\ X$   
 $F\ T\ I\ X$ ]  
**by**  $blast$   
**qed**  
**ultimately show**  $?thesis$  **by** ( $meson$   $topology\text{-eq}$ )  
**qed**

The canonical functions on a product space: evaluation and projection

**definition**  $evaluation\text{-map} :: 'a\ topology \Rightarrow ('b \Rightarrow 'a \Rightarrow 'c) \Rightarrow 'b\ set \Rightarrow 'a \Rightarrow 'b \Rightarrow 'c$

**where**  $evaluation\text{-map}\ X\ F\ I = (\lambda x \in topspace\ X . (\lambda i \in I . F\ i\ x))$

**definition** *product-projection* :: ('a  $\Rightarrow$  'b topology)  $\Rightarrow$  'a set  $\Rightarrow$  'a  $\Rightarrow$  ('a  $\Rightarrow$  'b)  $\Rightarrow$  'b  
**where** *product-projection* T I = ( $\lambda$  i  $\in$  I . ( $\lambda$  p  $\in$  topspace (product-topology T I) . p i))

**lemma** *product-projection*:  
**shows**  $\forall$  i  $\in$  I .  $\forall$  p  $\in$  topspace (product-topology T I) . *product-projection* T I i p = p i  
**using** *product-projection-def*[of T I] **by** *simp*

**lemma** *evaluation-then-projection*:  
**assumes**  $\forall$  i  $\in$  I . F i  $\in$  topspace X  $\rightarrow$  topspace (T i)  
**shows**  $\forall$  i  $\in$  I .  $\forall$  x  $\in$  topspace X . ((*product-projection* T I i) o (*evaluation-map* X F I)) x = F i x  
**proof** –  
{ **fix** i **assume** *iprops*: i  $\in$  I  
{ **fix** x **assume** *xprops*: x  $\in$  topspace X  
**have** *Fix*: ( $\lambda$  i  $\in$  I . F i x)  $\in$  topspace (product-topology T I) **using** *xprops*  
*assms*(1) **by** *auto*  
**have** ((*product-projection* T I i) o (*evaluation-map* X F I)) x  
= (*product-projection* T I i) (( $\lambda$  x  $\in$  topspace X . ( $\lambda$  i  $\in$  I . F i x)) x)  
**unfolding** *evaluation-map-def* **by** *auto*  
**moreover** **have** ... = (*product-projection* T I i) ( $\lambda$  i  $\in$  I . F i x) **using**  
*xprops* **by** *simp*  
**moreover** **have** ... = ( $\lambda$  p  $\in$  topspace (product-topology T I) . p i) ( $\lambda$  i  $\in$  I . F i x)  
**unfolding** *product-projection-def* **using** *iprops* **by** *auto*  
**moreover** **have** ... = F i x **using** *Fix iprops* **by** *simp*  
**ultimately** **have** ((*product-projection* T I i) o (*evaluation-map* X F I)) x =  
F i x **by** *auto*  
}  
**hence**  $\forall$  x  $\in$  topspace X . ((*product-projection* T I i) o (*evaluation-map* X F I)) x = F i x  
**by** *auto*  
}  
**thus** *?thesis* **by** *auto*  
**qed**

### 2.3 A product topology is the weak topology induced by its projections if the projections separate points from closed sets.

**lemma** *projections-continuous*:  
**assumes** P = product-topology T I  
**and** F = ( $\lambda$  i  $\in$  I . *product-projection* T I i)  
**shows**  $\forall$  i  $\in$  I . *continuous-map* P (T i) (F i)  
**using** *assms*(1) *assms*(2) *product-projection-def*[of T I]

by *fastforce*

**lemma** *product-topology-eq-weak-topology*:  
**assumes**  $P = \text{product-topology } T \ I$   
**and**  $F = (\lambda i \in I . \text{product-projection } T \ I \ i)$   
**and**  $W = \text{weak-topology } (\text{topspace } P) \ F \ T \ I$   
**and**  $\text{funcset-types } (\text{topspace } P) \ F \ T \ I$   
**and**  $\text{funcset-separates-points-from-closed-sets } P \ F \ T \ I$   
**shows**  $P = W$   
**using** *assms product-projection-def*[of  $T \ I$ ] *projections-continuous*  
*funcset-separates-points-from-closed-sets-imp-weak*[of  $P \ F \ T \ I \ W$ ]  
**by** *simp*

Reducing the domain and minimising the range of continuous functions, and related results concerning weak topologies.

**lemma** *continuous-map-reduced*:  
**assumes** *continuous-map*  $X \ Y \ f$   
**shows** *continuous-map* (*subtopology*  $X \ S$ ) (*subtopology*  $Y \ (f'S)$ ) (*restrict*  $f \ S$ )  
**using** *assms continuous-map-from-subtopology continuous-map-in-subtopology* **by**  
*fastforce*

**lemma** *inj-on-imp*:  
**assumes** *inj-on*  $f \ S$   
**shows**  $\forall y . (y \in f'S) \longleftrightarrow (\exists x \in S . y = f \ x)$   
**by** (*simp add: image-iff*)

**lemma** *injection-on-intersection*:  
**assumes** *inj-on*  $f \ S$   
**and**  $B \neq \{\}$   
**and**  $\forall b \in B . b \subseteq S$   
**shows**  $f'(\bigcap B) = \bigcap \{ f' \ b \mid b . b \in B \}$   
*(is ?lhs = ?rhs)*  
**proof** –  
**have**  $?lhs \subseteq ?rhs$  **by** *auto*  
**moreover have**  $?rhs \subseteq ?lhs$   
**proof** –  
**{** **fix**  $y$  **assume**  $rhs: y \in ?rhs$   
**then obtain**  $b$  **where**  $bprops: y \in f' \ b \wedge b \in B$   
**by** (*smt (verit, del-insts) Inter-iff assms(2) ex-in-conv mem-Collect-eq*)  
**then obtain**  $x$  **where**  $xprops: x \in b \wedge b \in B \wedge y = f \ x$  **by** *auto*  
  
**have**  $\forall b \in B . y \in f' \ b$  **using**  $rhs$  **by** *auto*  
**hence**  $\forall b \in B . f \ x \in f' \ b$  **using**  $xprops$  **by** *auto*  
**hence**  $\forall b \in B . x \in b$  **using**  $assms(1)$   
**by** (*meson assms(3) in-mono inj-on-image-mem-iff xprops*)  
**hence**  $x \in \bigcap B$  **by** *auto*  
**hence**  $y \in ?lhs$  **using**  $xprops$  **by** *auto*  
**}**

```

    thus ?thesis by auto
  qed
  ultimately show ?thesis by auto
qed

```

## 2.4 Evaluation is an embedding for weak topologies

**lemma** *evaluation-is-embedding*:

```

  assumes  $X = \text{weak-topology } (\text{topspace } X) F T I$ 
  and  $P = \text{product-topology } T I$ 
  and  $\text{funcset-types } (\text{topspace } X) F T I$ 
  and  $\text{funcset-separates-points } X F I$ 
  shows  $\text{embedding-map } X P (\text{evaluation-map } X F I)$ 
  proof -
    define  $ev$  where  $ev = \text{evaluation-map } X F I$ 
    define  $proj$  where  $proj = \text{product-projection } T I$ 
    define  $R$  where  $R = ev \text{ ` } \text{topspace } X$ 
    define  $Rtop$  where  $Rtop = \text{subtopology } P R$ 

```

**have** *injective*:  $\text{inj-on } ev (\text{topspace } X)$

**proof** -

```

  have  $\text{sigs: } \forall i \in I . F i \in (\text{topspace } X) \rightarrow (\text{topspace } (T i))$ 
    using  $\text{assms}(3) \text{ funcset-types-def}[of \text{topspace } X F T I]$ 
    by blast

```

```

{ fix  $x y$  assume  $xyprops: x \in \text{topspace } X \wedge y \in \text{topspace } X$ 
  { assume  $\text{hyp: } x \neq y$ 
    then obtain  $i$  where  $iprops: i \in I \wedge F i x \neq F i y$ 
      using  $\text{assms}(4) \text{ funcset-separates-points-def}[of X F I] \text{ hyp } xyprops$ 
      by blast
    hence  $(proj i) (ev x) \neq (proj i) (ev y)$ 
      using  $\text{evaluation-then-projection}[of I F X T] \text{ proj-def } ev\text{-def}$ 
      by ( $\text{simp add: sigs } xyprops$ )
    hence  $ev x \neq ev y$  by auto
  }
  hence  $ev x = ev y \longrightarrow x = y$  by auto
}

```

**thus** ?thesis using *inj-on-def* by blast

qed

**moreover** have *ev-cts*:  $\text{continuous-map } X Rtop ev$

**proof** -

```

  have  $\text{main: } \forall i \in I . \forall x \in \text{topspace } X . (proj i o ev) x = F i x$ 
    using  $\text{proj-def } ev\text{-def } \text{product-projection-def}[of T I] \text{ evaluation-then-projection}[of I F X T]$ 
       $\text{evaluation-map-def}[of X F I]$ 
    by ( $\text{metis } \text{assms}(1) \text{ assms}(3) \text{ continuous-map-funspace } \text{weak-generators-continuous}$ )
  moreover have  $\forall i \in I . \text{continuous-map } X (T i) (F i)$ 

```

**using** *weak-generators-continuous*[of  $X$  *topspace*  $X$   $F$   $T$   $I$ ] *assms* **by** *auto*  
**moreover have**  $\forall i \in I . \forall x \in \text{topspace } X . F i x = \text{ev } x i$   
**using** *product-projection-def*[of  $T$   $I$ ] *main* *ev-def*  
**by** (*simp* *add: evaluation-map-def*[of  $X$   $F$   $I$ ])  
**moreover have**  $\text{ev } \text{' } \text{topspace } X \subseteq \text{extensional } I$   
**using** *ev-def* *extensional-def* *assms* *evaluation-map-def*[of  $X$   $F$   $I$ ]  
**by** *fastforce*  
**ultimately have** *continuous-map*  $X$   $P$   $ev$   
**using** *assms* *proj-def* *ev-def* *Rtop-def* *continuous-map-componentwise*[of  $X$   $T$   
 $I$   $ev$ ]  
*continuous-map-eq* **by** *fastforce*  
**thus** *?thesis*  
**using** *Rtop-def* *R-def* *continuous-map-in-subtopology* **by** *blast*  
**qed**

**moreover have** *open-map*  $X$   $Rtop$   $ev$   
**proof** –  
**have** *open-map-on-gens*:  $\forall U \in \text{weak-generators } (\text{topspace } X) F T I . \text{openin}$   
 $Rtop$  ( $ev$   $\text{' } U$ )  
**proof** –  
**{ define**  $R_s$  **where**  $R_s = (\lambda i \in I . (F i \text{' } \text{topspace } X))$   
**define**  $Rtops$  **where**  $Rtops = (\lambda i \in I . \text{subtopology } (T i) (R_s i))$

**fix**  $U$  **assume**  $U \in \text{weak-generators } (\text{topspace } X) F T I$   
**then obtain**  $i$  **where** *iprops*:  $i \in I \wedge U \in \text{open-sets-induced-by-func } (F i)$   
 $(\text{topspace } X) (T i)$   
**using** *assms* *weak-generators-def*[of *topspace*  $X$   $F$   $T$   $I$ ] **by** *auto*  
**then obtain**  $V$   
**where**  $Vprops$ :  $\text{openin } (T i) V \wedge U = \text{inverse}' (F i) (\text{topspace } X) V$   
**using** *open-sets-induced-by-func-def*[of  $F i$  *topspace*  $X$   $T i$ ]  
**by** *blast*  
**hence**  $Uprops$ :  $\text{openin } (T i) V \wedge U = \{ x \in \text{topspace } X . F i x \in V \}$   
**using** *inverse'-def*[of  $F i$  *topspace*  $X$   $V$ ] **by** *auto*  
**moreover have**  $\forall x \in \text{topspace } X . F i x = ((\text{proj } i) o \text{ev}) x$   
**using** *evaluation-then-projection*[of  $I$   $F$   $X$   $T$ ] *assms*(3)  
*funcset-types-def*[of *topspace*  $X$   $F$   $T$   $I$ ] *iprops*  
*proj-def* *ev-def*  
**by** *auto*  
**hence**  $U = \{ x \in \text{topspace } X . ((\text{proj } i) o \text{ev}) x \in V \}$  **using**  $Uprops$  **by**  
*auto*

**hence**  $ev \text{' } U = \{ y \in R . (\text{proj } i) y \in V \}$  **using**  $R$ -*def* **by** *auto*  
**moreover have**  $\{ y \in R . (\text{proj } i) y \in V \} = R \cap ((\text{proj } i) \text{' } V)$   
**by** *auto*  
**moreover have** *continuous-map*  $P$   $(T i)$   $(\text{proj } i)$   
**using** *continuous-map-product-projection*[of  $i$   $I$   $T$ ] *iprops* *proj-def*  
*product-projection-def*[of  $T$   $I$ ] *assms*(2) **by** *auto*  
**ultimately have** *summary*:  $\text{openin } (T i) V \wedge \text{continuous-map } P (T i) (\text{proj}$   
 $i)$   
 $\wedge (ev \text{' } U) = R \cap ((\text{proj } i) \text{' } V)$  **by** *auto*

```

hence  $\forall U. \text{openin } (T \ i) \ U \longrightarrow \text{openin } P \ \{x \in \text{topspace } P. \text{proj } i \ x \in U\}$ 
using continuous-map-def[of  $P \ T \ i \ \text{proj } i$ ] by auto
hence  $\text{openin } P \ ((\text{proj } i \ -' \ V) \cap \text{topspace } P)$ 
using summary by blast
moreover have  $R \subseteq \text{topspace } P$ 
using R-def ev-def evaluation-map-def[of  $X \ F \ I$ ] assms(3)
funcset-types-def[of  $\text{topspace } X \ F \ T \ I$ ]
by (metis Rtop-def ev-cts continuous-map-image-subset-topospace
continuous-map-into-fulltopology)
ultimately have  $\text{openin } Rtop \ ((\text{proj } i \ -' \ V) \cap R)$ 
using Rtop-def
by (metis inf.absorb-iff2 inf-assoc openin-subtopology)
hence  $\text{openin } Rtop \ (ev \ -' \ U)$  using summary
by (simp add: inf-commute)
}
thus ?thesis by auto
qed

have open-map-on-basics:  $\forall U \in \text{weak-base } (\text{topspace } X) \ F \ T \ I . \text{openin } Rtop$ 
 $(ev \ -' \ U)$ 
proof –
have Ugens:  $\bigcup (\text{weak-generators } (\text{topspace } X) \ F \ T \ I) = \text{topspace } X$ 
using assms(1) weak-generators-topospace by blast

{ fix  $U$  assume bprops:  $U \in \text{weak-base } (\text{topspace } X) \ F \ T \ I$ 
hence  $U \in \text{finite-intersections-of } (\text{weak-generators } (\text{topspace } X) \ F \ T \ I)$ 
by (simp add: base-generated-by-def weak-base-def)
then obtain  $b$  where bprops:  $b \subseteq \text{weak-generators } (\text{topspace } X) \ F \ T \ I \wedge$ 
finite'  $b \wedge U = \bigcap b$ 
unfolding finite-intersections-of-def
by auto
hence finite'  $b \wedge (\forall g \in b . \text{openin } Rtop \ (ev \ -' \ g))$  using open-map-on-gens
by auto
hence  $\text{openin } Rtop \ (\bigcap \{ (ev \ -' \ g) \mid g . g \in b \})$  by auto
hence  $\text{openin } Rtop \ (ev \ -' \ \bigcap b)$ 
using injection-on-intersection[of  $ev \ \text{topspace } X \ b$ ] bprops
by (metis (no-types, lifting) Ugens Union-upper in-mono injective)
hence  $\text{openin } Rtop \ (ev \ -' \ U)$  using bprops by metis
}
thus ?thesis by auto
qed
hence open-map-on-opens:  $\forall U \in \text{weak-opens } (\text{topspace } X) \ F \ T \ I . \text{openin}$ 
 $Rtop \ (ev \ -' \ U)$ 
by (smt (verit, ccfv-SIG) image-iff image-mono openin-subopen weak-opens-nhood-base

weak-topology-alt)
thus ?thesis
using opens-eq-generated-topology[of  $\text{weak-generators } (\text{topspace } X) \ F \ T \ I$ ]
assms(1)

```

**unfolding** *weak-topology-def* **using** *open-map-def*[of  $X$   $Rtop$ ]  
**by** (*simp add: weak-opens-def*)  
**qed**

**ultimately have** *homeomorphic-map*  $X$   $Rtop$  *ev*  
**by** (*metis R-def Rtop-def bijective-open-imp-homeomorphic-map continuous-map-image-subset-topospace*  
*continuous-map-into-fulltopology topspace-subtopology-subset*)  
**thus** *?thesis* **using** *embedding-map-def*[of  $X$   $P$  *ev*] *ev-def R-def Rtop-def*  
**by** *auto*  
**qed**

### 3 Compactification

#### 3.1 Definition

**lemma** *embedding-map-id*:  
**assumes**  $S \subseteq topspace\ X$   
**shows** *embedding-map* (*subtopology*  $X$   $S$ )  $X$  *id*  
**using** *assms embedding-map-def topspace-subtopology-subset*  
**by** *fastforce*

**definition** *compactification-via* :: ( $'a \Rightarrow 'b$ )  $\Rightarrow$   $'a$  *topology*  $\Rightarrow$   $'b$  *topology*  $\Rightarrow$  *bool*  
**where** *compactification-via*  $f$   $X$   $K \equiv compact-space\ K \wedge dense-embedding\ X\ K\ f$

**definition** *compactification* ::  $'a$  *topology*  $\Rightarrow$   $'b$  *topology*  $\Rightarrow$  *bool*  
**where** *compactification*  $X$   $K = (\exists f . compactification-via\ f\ X\ K)$

**lemma** *compactification-compactification-via*:  
**assumes** *compactification-via*  $f$   $X$   $K$   
**shows** *compactification*  $X$   $K$   
**using** *assms unfolding compactification-def* **by** *fastforce*

#### 3.2 Example: The Alexandroff compactification of a non-compact locally-compact Hausdorff space

**lemma** *Alexandroff-is-compactification-via-Some*:  
**assumes**  $\neg compact-space\ X \wedge Hausdorff-space\ X \wedge locally-compact-space\ X$   
**shows** *compactification-via* *Some*  $X$  (*Alexandroff-compactification*  $X$ )  
**using** *assms compact-space-Alexandroff-compactification*  
*embedding-map-Some*  
*Alexandroff-compactification-dense*  
*compactification-via-def*  
**by** (*metis dense-in-def*)

### 3.3 Example: The closure of a subset of a compact space

**lemma** *compact-closure-is-compactification*:

**assumes** *compact-space*  $K$

**and**  $S \subseteq \text{topspace } K$

**shows**  $\text{compactification-via id (subtopology } K \ S) \ (\text{subtopology } K \ (K \ \text{closure-of } S))$

**proof** –

**define** *big* **where**  $\text{big} = \text{subtopology } K \ (K \ \text{closure-of } S)$

**define** *small* **where**  $\text{small} = \text{subtopology } K \ S$

**have** *dense-in big (id ‘ topspace small) (topspace big)*

**by** (*metis dense-in-def big-def small-def assms(2) closedin-topspace closure-of-minimal*

*closure-of-subset closure-of-subtopology-open id-def image-id inf.orderE  
openin-imp-subset openin-subtopology-refl topspace-subtopology-subset*)

**moreover have** *embedding-map small big id*

**by** (*metis assms(2) big-def closure-of-subset-Int embedding-map-in-subtopology*

*id-apply*

*embedding-map-id image-id small-def topspace-subtopology*)

**ultimately have** *dense-embedding small big id by blast*

**moreover have** *compact-space big*

**by** (*simp add: big-def assms(1) closedin-compact-space compact-space-subtopology*)

**ultimately show** *?thesis*

**unfolding** *compactification-via-def using small-def big-def by blast*

**qed**

### 3.4 Example: A compact space is a compactification of itself

**lemma** *compactification-of-compact*:

**assumes** *compact-space*  $K$

**shows**  $\text{compactification-via id } K \ K$

**using** *compact-closure-is-compactification[of  $K$  topspace  $K$ ]*

**by** (*simp add: assms*)

### 3.5 Example: A closed non-trivial real interval is a compactification of its interior

**lemma** *closed-interval-interior*:

**shows**  $\{a::\text{real} \ <..< \ b\} = \text{interior } \{a..b\}$

**by** *auto*

**lemma** *open-interval-closure*:

**shows**  $(a < (b::\text{real})) \longrightarrow \{a .. b\} = \text{closure } \{a <..< b\}$

**using** *closure-greaterThanLessThan[of  $a$   $b$ ] by simp*

**lemma** *closed-interval-compactification*:

**assumes**  $(a::\text{real}) < b$

**and**  $\text{open-interval} = \text{subtopology euclideanreal } \{a <..< b\}$

**and**  $\text{closed-interval} = \text{subtopology euclideanreal } \{a..b\}$

**shows**  $\text{compactification open-interval closed-interval}$

**proof** –  
**have** *compact-space closed-interval* **using** *assms(3)*  
**using** *compact-space-subtopology compactin-euclidean-iff* **by** *blast*  
**moreover have** *Hausdorff-space closed-interval*  
**by** (*simp add: Hausdorff-space-subtopology assms(3)*)  
**moreover have**  $\{a < .. < b\} \subseteq \text{topspace closed-interval}$   
**by** (*simp add: assms(3) greaterThanLessThan-subseteq-atLeastAtMost-iff*)  
**ultimately have** *compactification-via id open-interval closed-interval*  
**using** *compact-closure-is-compactification[of closed-interval {a < .. < b}]*  
*open-interval-closure[of a b]*  
**by** (*metis assms closedin-self closedin-subtopology-refl closure-of-subtopology*  
*euclidean-closure-of subtopology-subtopology subtopology-topospace*  
*topospace-subtopology-subset*)  
**thus** *?thesis* **using**  
*compactification-compactification-via[of id open-interval closed-interval]*  
**by** *auto*  
**qed**

## 4 The Stone-Čech compactification of a Tychonov space

**lemma** *compact-range'*:  
**assumes**  $f \in C^* X$   
**shows** *compact (range' X f)*  
**proof** –  
**obtain**  $m M$  **where**  $mM: \forall y \in \text{topspace } X . f y \in \{m..M\}$  **using** *assms* **by**  
*auto*  
**hence**  $f ' \text{topspace } X \subseteq \{m..M\}$  **by** *auto*  
**hence**  $\text{range}' X f \subseteq \text{euclideanreal closure-of } \{m..M\}$   
**unfolding** *range'-def* **by** (*meson closure-of-mono*)  
**moreover have** *compact {m..M}* **by** *auto*  
**ultimately show** *?thesis*  
**by** (*metis closed-Int-compact closed-atLeastAtMost closed-closedin closedin-closure-of*  
*closure-of-closedin inf.order-iff range'-def*)  
**qed**

**lemma** *c-range-nonempty*:  
**assumes**  $f \in C(X)$   
**and**  $\text{topspace } X \neq \{\}$   
**shows**  $\text{range}' X f \neq \{\}$   
**proof** –  
**have**  $f ' \text{topspace } X \neq \{\}$  **using** *assms* **by** *blast*  
**thus** *?thesis* **unfolding** *range'-def* **by** *simp*  
**qed**

**lemma** *cstar-range-nonempty*:  
**assumes**  $f \in C^* X$

**and**  $\text{topspace } X \neq \{\}$   
**shows**  $\text{range}' X f \neq \{\}$   
**using** *assms c-range-nonempty*[of  $f X$ ]  
**by** *auto*

**lemma** *cstar-separates-tych-space*:

**assumes** *tych-space*  $X$

**shows** *funcset-separates-points-from-closed-sets*  $X$  (*cstar-id*  $X$ ) ( $\lambda f \in C^* X$ . *euclideanreal*) ( $C^* X$ )

$\wedge$  *funcset-separates-points*  $X$  (*cstar-id*  $X$ ) ( $C^* X$ )

**proof** –

{ **fix**  $x S$  **assume** *closedin*  $X S \wedge x \in \text{topspace } X - S$

**then obtain**  $f$

**where** *fprops*: *continuous-map*  $X$  (*top-of-set*  $\{0..(1::\text{real})\}$ )  $f \wedge f x = 0 \wedge f ' S \subseteq \{1\}$

$S \subseteq \{1\}$

**using** *assms completely-regular-space-def*[of  $X$ ]

**by** *presburger*

**hence**  $f \in C^* X$

**using** *continuous-map-into-fulltopology*[of  $X$  *euclideanreal*  $\{0..(1::\text{real})\}$   $f$ ]

**by** *auto*

**moreover have** *fbounded*  $f X$

**by** (*meson atLeastAtMost-iff continuous-map-upper-lower-semicontinuous-le-gen fprops*)

**ultimately have** *f-in-cstar*:  $f \in (C^* X)$  **by** *auto*

**moreover have** *f-separates*:  $f x \notin (\text{euclideanreal closure-of } (f ' S))$

**proof** –

**have** *closedin euclideanreal*  $(f ' S)$

**by** (*metis closed-closedin closed-empty closed-singleton fprops subset-singletonD*)

**moreover have**  $f x \notin f ' S$  **using** *fprops* **by** *auto*

**thus** *?thesis* **using** *calculation* **by** *auto*

**qed**

**ultimately have**  $\exists f \in C^* X$  .  $f x \notin \text{euclideanreal closure-of } (f ' S)$  **by** *auto*

}

**hence** *rtp1*: *funcset-separates-points-from-closed-sets*  $X$  (*cstar-id*  $X$ ) ( $\lambda f \in C^* X$ . *euclideanreal*) ( $C^* X$ )

**using** *cstar-id-def*[of  $X$ ] **unfolding** *funcset-separates-points-from-closed-sets-def* **by** *auto*

**moreover have** *funcset-separates-points*  $X$  (*cstar-id*  $X$ ) ( $C^* X$ )

**proof** –

{ **fix**  $x y$  **assume**  $\{x, y\} \subseteq \text{topspace } X \wedge x \neq y$

**hence** *closedin*  $X \{y\} \wedge x \in \text{topspace } X - \{y\}$

**using** *assms* **by** (*simp add: t1-space-closedin-finite*)

**hence**  $\exists f \in C^* X$  . *cstar-id*  $X f x \notin (\lambda f \in C^* X$  . *euclideanreal*)  $f$  *closure-of cstar-id*  $X f ' \{y\}$

**using** *funcset-separates-points-from-closed-sets-def*[of  $X$  *cstar-id*  $X$   $\lambda f \in C^* X$  . *euclideanreal*  $C^* X$ ]

```

      rtp1 by presburger
    hence  $\exists f \in C^* X . f x \neq f y$ 
      using cstar-id-def[of X] t1-space-closedin-finite[of euclideanreal] by auto
    }
  thus ?thesis using cstar-id-def[of X] unfolding funcset-separates-points-def
by auto
qed
ultimately show ?thesis by auto
qed

```

The product topology induced by  $C^*(X)$  on a Tychonov space.

```

definition scT :: 'a topology  $\Rightarrow$  ('a  $\Rightarrow$  real)  $\Rightarrow$  real topology
  where scT X = ( $\lambda f \in C^* X .$  subtopology euclideanreal (range' X f))

```

```

definition scT-full :: 'a topology  $\Rightarrow$  ('a  $\Rightarrow$  real)  $\Rightarrow$  real topology
  where scT-full X = ( $\lambda f \in C^* X .$  euclideanreal)

```

```

definition scProduct :: 'a topology  $\Rightarrow$  (('a  $\Rightarrow$  real)  $\Rightarrow$  real) topology
  where scProduct X = product-topology (scT X) (C* X)

```

```

definition scProject :: 'a topology  $\Rightarrow$  ('a  $\Rightarrow$  real)  $\Rightarrow$  (('a  $\Rightarrow$  real)  $\Rightarrow$  real)  $\Rightarrow$  real
  where scProject X = product-projection (scT X) (C* X)

```

```

definition scEmbed :: 'a topology  $\Rightarrow$  'a  $\Rightarrow$  ('a  $\Rightarrow$  real)  $\Rightarrow$  real
  where scEmbed X = evaluation-map X (cstar-id X) (C* X)

```

```

lemma scT-images-compact-Hausdorff:
  shows  $\forall f \in C^* X .$  compact-Hausdorff (scT X f)
proof –
  have T:  $\forall f \in C^* X .$  scT X f = subtopology euclideanreal (range' X f)
    unfolding scT-def by simp
  thus ?thesis using range'-def[of X f]
    by (simp add: Hausdorff-space-subtopology compact-range' compact-space-subtopology)
qed

```

```

lemma scT-images-bounded:
  shows  $\forall f \in C^* X .$  bounded (topspace (scT X f))
  using scT-images-compact-Hausdorff[of X] scT-def[of X]
  by (simp add: compact-imp-bounded compact-range')

```

```

lemma scProduct-compact-Hausdorff:
  shows compact-Hausdorff (scProduct X)
  unfolding scProduct-def using scT-images-compact-Hausdorff[of X]
  using compact-space-product-topology
  by (metis (no-types, lifting) compact-Hausdorff-imp-regular-space regular-space-product-topology)

```

*regular-t0-eq-Hausdorff-space t0-space-product-topology*)

The Stone-Ćech compactification of a Tychonov space and its extension properties

**lemma** *tych-space-weak*:

**assumes** *tych-space X*

**shows**  $X = \text{weak-topology } (\text{topspace } X) (\text{cstar-id } X) (\text{scT } X) (C^* X)$

**proof** (*cases topspace X = {}*)

**case** *True*

**then show** *?thesis*

**using** *weak-topology-on-empty*[*of weak-topology (topspace X) (cstar-id X) (scT X) (C\* X)*]

*topology-eq* **by** *fastforce*

**next**

**case** *False*

**define** *W* **where**  $W = \text{weak-topology } (\text{topspace } X) (\text{cstar-id } X) (\text{scT } X) (C^* X)$

**hence** *topspace W = topspace X*

**using** *cstar-types-restricted*[*of X*] *scT-def*[*of X*] *W-def* *cstar-nonempty*[*of X*] *weak-topology-topospace*[*of W topspace X cstar-id X scT X C\* X*]

**by** *auto*

**moreover have**  $\forall f. \text{continuous-map } X \text{ euclideanreal } f \wedge \text{fbounded } f X \longrightarrow \text{continuous-map } X (\text{top-of-set } (\text{closure } (f \text{ ' } \text{topspace } X))) f$

**using** *closure-subset continuous-map-into-subtopology image-subset-iff* **by** *fastforce*

**ultimately have** *openin X U* **if** *openin W U* **for** *U*

**using** *cstar-types-restricted*[*of X*] *scT-def*[*of X*] *cstar-id-def*[*of X*] *weak-topology-is-weakest* [*OF W-def*] *that*

**by** (*simp add: range'-def*)

**moreover have** *openin W U* **if** *U: openin X U* **for** *U*

**proof** –

**have**  $\exists V. x \in V \wedge V \subseteq U \wedge \text{openin } W V$  **if**  $x \in U$  **for**  $x$

**proof** –

**have** *x-in-X: x ∈ topspace X*

**using** *openin-subset U x* **by** *fastforce*

**define** *S* **where**  $S = \text{topspace } X - U$

**hence** *props': x ∈ topspace X - S ∧ closedin X S*

**using** *U openin-closedin-eq x* **by** *fastforce*

**then obtain** *f* **where** *fprops: continuous-map X (top-of-set {0..1::real}) f ∧ f x = 0 ∧ f ' S ⊆ {1}*

**using** *assms(1) completely-regular-space-def*[*of X*]

**by** *meson*

**then obtain** *ffull*

**where** *ffullprops: (ffull ∈ C X) ∧ ffull x = (0::real) ∧ ffull ' S ⊆ {1}*

**using** *continuous-map-into-fulltopology*

```

    by (metis mem-Collect-eq)

define F where F = fbound ffull 0 1
hence Fcstar: F ∈ C* X using ffullprops fbound-cstar[of ffull X 0 1] by
auto
hence Ftype: F ∈ topspace X → topspace euclideanreal
    unfolding continuous-map-def by auto

define I where I = {(-1) <..< 1::real}
hence Iprops: openin euclideanreal I
    by (simp add: openin-delete)

define V where V = inverse' F (topspace X) I

have crprops: F x = 0 ∧ F ' S ⊆ {1}
    using ffullprops F-def
    unfolding fbound-def fmid-def fmin-def fmax-def min-def max-def
    by auto
hence V ⊆ U
    by (auto simp: S-def I-def inverse'-def V-def)
moreover have x ∈ V
    using crprops I-def x-in-X unfolding inverse'-def V-def by auto
moreover have openin W V
proof -
    have V ∈ open-sets-induced-by-func F (topspace X) euclideanreal
        unfolding open-sets-induced-by-func-def using Ftype V-def Iprops
        by blast
    moreover have open-sets-induced-by-func F (topspace X) euclideanreal ⊆
        weak-generators (topspace X) (cstar-id X) (scT-full X) (C* X)
        using weak-generators-def[of topspace X (cstar-id X) scT-full X C* X]
        scT-full-def[of X] cstar-id-def[of X] Fcstar
        by (smt (verit, ccfv-SIG) Sup-upper mem-Collect-eq restrict-apply')
    ultimately have V ∈ weak-generators (topspace X) (cstar-id X) (scT-full
X) (C* X)
        by auto
    hence openin (topology-generated-by (weak-generators (topspace X) (cstar-id
X) (scT-full X) (C* X))) V
        using generators-are-open[of weak-generators (topspace X) (cstar-id X)
(scT-full X) (C* X)]
        topology-generated-by-Basis by blast
    thus ?thesis
        using W-def weak-restricted-topology-eq-weak[of X]
        unfolding scT-def scT-full-def weak-topology-def
        by simp
qed
ultimately show ?thesis by auto
qed
thus ?thesis by (meson openin-subopen)
qed

```

ultimately have  $\forall U . \text{openin } X \ U \longleftrightarrow \text{openin } W \ U$  **by** *auto*  
hence  $X = W$  **by** (*simp add: topology-eq*)  
thus *?thesis* **using** *W-def* **by** *simp*  
**qed**

#### 4.1 Definition of $\beta X$

**definition** *scEmbeddedCopy* :: *'a topology*  $\Rightarrow$  (*'a*  $\Rightarrow$  *real*)  $\Rightarrow$  *real*) *set*  
**where** *scEmbeddedCopy* *X* = *scEmbed* *X* ' *topspace* *X*

**definition** *scCompactification* :: *'a topology*  $\Rightarrow$  (*'a*  $\Rightarrow$  *real*)  $\Rightarrow$  *real*) *topology* ( $\beta$   
 $\rightarrow$ )  
**where** *scCompactification* *X*  
= *subtopology* (*scProduct* *X*) ((*scProduct* *X*) *closure-of* (*scEmbeddedCopy*  
*X*))

**lemma** *sc-topspace*:

**shows** *topspace* ( $\beta$  *X*) = (*scProduct* *X*) *closure-of* (*scEmbeddedCopy* *X*)  
**using** *scCompactification-def*[*of X*] *closure-of-subset-topspace* **by** *force*

**lemma** *scProject'*:

**shows**  $\forall f \in C^* \ X . \forall p \in \text{topspace } (\beta \ X) . \text{scProject } X \ f \ p = p \ f$

**proof** –

**have** *topspace* ( $\beta$  *X*)  $\subseteq$  *topspace* (*scProduct* *X*) **unfolding** *scCompactification-def*  
**by** *auto*

**thus** *?thesis*

**unfolding** *scProject-def* *product-projection-def* *scProduct-def*

**by** *auto*

**qed**

Evaluation densely embeds Tychonov  $X$  in  $\beta X$

**lemma** *dense-embedding-scEmbed*:

**assumes** *tych-space* *X*

**shows** *dense-embedding* *X* ( $\beta$  *X*) (*scEmbed* *X*)

**proof** –

**define** *W* **where** *W* = *weak-topology* (*topspace* *X*) (*cstar-id* *X*) ( $\lambda f \in C^* \ X .$   
*euclideanreal*) (*C\** *X*)

**hence**  $X = W$  **using** *assms* *tych-space-weak*[*of X*]

**by** (*metis* (*mono-tags*, *lifting*) *scT-def* *weak-restricted-topology-eq-weak*)

**hence** *Xweak*:  $X = \text{weak-topology } (\text{topspace } X) (\text{cstar-id } X) (\text{scT } X) (C^* \ X)$

**using** *scT-def*[*of X*] *W-def* *cstar-id-def*[*of X*]

*weak-restricted-topology-eq-weak*[**where**  $X = X$ ] **by** *auto*

**moreover** **have** *scProduct* *X* = *product-topology* (*scT* *X*) (*C\** *X*) **using** *scProd-*  
*uct-def*[*of X*] **by** *auto*

**moreover** **have** *funcset-types* (*topspace* *X*) (*cstar-id* *X*) (*scT* *X*) (*C\** *X*)

**unfolding** *scT-def* **using** *cstar-types-restricted*[*of X*] **by** *auto*

**moreover** **have** *funcset-separates-points* *X* (*cstar-id* *X*) (*C\** *X*)

**using** *cstar-separates-tych-space*[*of X*] *assms*(1) **by** *auto*

**moreover have**  $(C^* X) \neq \{\}$  **using** *cstar-nonempty* **by** *auto*  
**ultimately have** *embedding-map*  $X$   $(scProduct X)$   $(scEmbed X)$   
**using** *evaluation-is-embedding*[of  $X$  *cstar-id*  $X$  *scT*  $X$   $C^* X$  *scProduct*  $X$ ]  
**unfolding** *scProduct-def* *scEmbed-def*  
**by** *auto*  
**hence embeds:** *embedding-map*  $X$   $(\beta X)$   $(scEmbed X)$   
**unfolding** *scCompactification-def*  
**by** *(metis closure-of-subset embedding-map-in-subtopology scEmbeddedCopy-def*  
*subtopology-topspace)*  
**moreover have** *dense-in*  $(\beta X)$   $(scEmbed X \text{ ' } topspace X)$   $(topspace (\beta X))$   
**unfolding** *dense-in-def* **using** *scCompactification-def*[of  $X$ ] *scEmbeddedCopy-def*[of  
 $X$ ]  
**by** *(metis Int-absorb1 closure-of-subset closure-of-subset-topspace closure-of-subtopology*  
  
*embedding-map-in-subtopology embeds set-eq-subset subtopology-topspace*  
  
*topspace-subtopology-subset)*  
**ultimately show** *?thesis* **by** *auto*  
**qed**

## 4.2 $\beta X$ is a compactification of $X$

**lemma** *scCompactification-compact-Hausdorff*:  
**assumes** *tych-space*  $X$   
**shows** *compact-Hausdorff*  $(\beta X)$   
**using** *scCompactification-def*[of  $X$ ] *scProduct-compact-Hausdorff*[of  $X$ ]  
**by** *(simp add: Hausdorff-space-subtopology closedin-compact-space compact-space-subtopology)*

**lemma** *scCompactification-is-compactification-via-scEmbed*:  
**assumes** *tych-space*  $X$   
**shows** *compactification-via*  $(scEmbed X)$   $X$   $(\beta X)$   
**using** *compactification-via-def*[of  $scEmbed X$   $X$   $\beta X$ ]  
*scCompactification-compact-Hausdorff*[of  $X$ ]  
*dense-embedding-scEmbed*[of  $X$ ] *assms*  
**by** *auto*

**lemma** *scCompactification-is-compactification*:  
**assumes** *tych-space*  $X$   
**shows** *compactification*  $X$   $(\beta X)$   
**using** *assms compactification-compactification-via*  
*scCompactification-is-compactification-via-scEmbed*  
**by** *blast*

**lemma** *scEvaluation-range*:  
**assumes**  $x \in topspace X$   
**and** *tych-space*  $X$   
**shows**  $(\lambda f \in C^* X . f x) \in topspace (product-topology (scT X) C^* X)$   
**proof** –  
**have** *funcset-types*  $(topspace X)$   $(cstar-id X)$   $(\lambda f \in C^* X . top-of-set (range' X$

f))  $C^* X$   
**using** *cstar-types-restricted*[of  $X$ ] **by auto**  
**hence**  $\forall f \in C^* X . f \in \text{topspace } X \rightarrow \text{topspace } (\text{scT } X f)$   
**unfolding** *funcset-types-def scT-def cstar-id-def*[of  $X$ ] **by auto**  
**thus** *?thesis using topspace-product-topology*[of  $\text{scT } X C^* X$ ] *assms(1)* **by auto**  
**qed**

**lemma** *scEmbed-then-project*:

**assumes**  $f \in C^* X$   
**and**  $x \in \text{topspace } X$   
**and** *tych-space*  $X$   
**shows**  $\text{scProject } X f (\text{scEmbed } X x) = f x$   
**proof** –  
**have** *fequiv*:  $\forall y \in \text{topspace } X . (\lambda g \in C^* X . (\text{cstar-id } X) g y) = (\lambda g \in C^* X . g y)$   
**proof** –  
{ **fix**  $y$  **assume** *yprops*:  $y \in \text{topspace } X$   
**hence**  $\forall g \in C^* X . (\text{cstar-id } X) g y = g y$  **unfolding** *cstar-id-def* **by auto**  
**hence**  $(\lambda g \in C^* X . (\text{cstar-id } X) g y) = (\lambda g \in C^* X . g y)$   
**by** (*meson restrict-ext*)  
}  
**thus** *?thesis* **by auto**  
**qed**

**have**  $\text{scProject } X f (\text{scEmbed } X x) = \text{scProject } X f (\text{evaluation-map } X (\text{cstar-id } X) (C^* X) x)$   
**unfolding** *scEmbed-def* **by auto**  
**also have**  $\dots = \text{scProject } X f (\lambda g \in C^* X . g x)$   
**unfolding** *evaluation-map-def* **using** *assms(2)* *fequiv* **by auto**  
**also have**  $\dots = (\lambda g \in C^* X . \lambda p \in \text{topspace } (\text{product-topology } (\text{scT } X) (C^* X)). p g) f (\lambda g \in C^* X . g x)$   
**unfolding** *product-projection-def scProject-def* **by auto**  
**also have**  $\dots = (\lambda p \in \text{topspace } (\text{product-topology } (\text{scT } X) (C^* X)). p f) (\lambda g \in C^* X . g x)$   
**using** *assms(1)* **by auto**  
**also have**  $\dots = f x$  **using** *scEvaluation-range*[of  $x X$ ] *assms* **by auto**  
**ultimately show** *?thesis* **by auto**  
**qed**

### 4.3 Evaluation is a $C^*$ -embedding of $X$ into $\beta X$

**definition** *scExtend* ::  $'a \text{ topology} \Rightarrow ('a \Rightarrow \text{real}) \Rightarrow (( 'a \Rightarrow \text{real}) \Rightarrow \text{real}) \Rightarrow \text{real}$   
**where**  $\text{scExtend } X = (\lambda f \in C^* X . \text{restrict } (\text{scProject } X f) (\text{topspace } (\beta X)))$

**proposition** *scExtend-extends*:

**assumes** *tych-space*  $X$   
**shows**  $\forall f \in C^* X . \forall x \in \text{topspace } X . f x = (\text{scExtend } X f) (\text{scEmbed } X x)$   
**proof** –

```

{ fix  $f$  assume  $fprops: f \in C^* X$ 
  have  $\forall x \in \text{topspace } X . (\text{scProject } X f) (\text{scEmbed } X x) = (\text{scExtend } X f)$ 
   $(\text{scEmbed } X x)$ 
proof –
  { fix  $x$  assume  $xprops: x \in \text{topspace } X$ 
    define  $p$  where  $pprops: p = \text{scEmbed } X x$ 

    hence  $\text{scExtend } X f p = (\text{restrict } (\text{scProject } X f) (\text{topspace } \beta X)) p$ 
    using  $xprops fprops$  unfolding  $\text{scExtend-def}$  by  $\text{auto}$ 
    moreover have  $p \in \text{topspace } \beta X$ 
    using  $assms(1) pprops$   $\text{dense-embedding-scEmbed[of } X]$ 
     $\text{scCompactification-def[of } X]$   $\text{scEmbeddedCopy-def[of } X]$ 
    by  $(metis (\text{no-types}, \text{lifting}) \text{embedding-map-in-subtopology image-eqI}$ 
     $\text{in-mono subtopology-topospace } xprops)$ 
    ultimately have  $\text{scExtend } X f p = \text{scProject } X f p$ 
    using  $pprops$   $\text{scEmbeddedCopy-def[of } X]$   $\text{scEmbed-def[of } X]$   $\text{evaluation-map-def}$  by  $\text{auto}$ 
  }
  thus  $?thesis$  by  $\text{auto}$ 
qed
hence  $\forall x \in \text{topspace } X . f x = (\text{scExtend } X f) (\text{scEmbed } X x)$ 
using  $\text{scEmbed-then-project[of } f X]$   $assms(1) fprops$  by  $\text{auto}$ 
}
thus  $?thesis$  by  $\text{auto}$ 
qed

```

**lemma**  $\text{scExtend-extends-cstar}$ :

```

assumes  $\text{tych-space } X$ 
shows  $\forall f \in C^* X . (\forall x \in \text{topspace } X . f x = (\text{scExtend } X f) (\text{scEmbed } X x)) \wedge \text{scExtend } X f \in C^* (\beta X)$ 
proof –
define  $e$  where  $e = \text{scExtend } X$ 
{ fix  $f$  assume  $fprops: f \in C^* X$ 
  hence  $\text{continuous-map } (\text{scProduct } X) (\text{scT } X f) (\text{scProject } X f)$ 
  using  $\text{scProduct-def[of } X]$   $\text{scProject-def[of } X]$ 
   $\text{projections-continuous[of scProduct } X \text{scT } X C^* X \text{scProject } X]$ 
   $\text{product-projection-def[of scT } X C^* X]$ 
  by  $(metis (\text{no-types}, \text{lifting}) \text{restrict-extensional extensional-restrict})$ 
hence  $\text{continuous-map } (\beta X) (\text{scT } X f) (\text{scProject } X f)$ 
  by  $(simp \text{ add: continuous-map-from-subtopology scCompactification-def})$ 
hence  $c\text{-embedded-f: continuous-map } (\beta X) (\text{scT } X f) (\text{scExtend } X f)$ 
  using  $\text{scExtend-def[of } X]$   $fprops$  by  $\text{force}$ 
moreover have  $f\text{bounded-f:fbounded } (\text{scExtend } X f) (\beta X)$ 
proof –
  obtain  $m M$  where  $f \text{ ' } \text{topspace } X \subseteq \{m..M\}$  using  $fprops$  by  $\text{force}$ 
  hence  $\text{extend-on-embedded: } e f \text{ ' } (\text{scEmbeddedCopy } X) \subseteq \{m..M\}$ 
  using  $\text{scExtend-extends[of } X]$   $e\text{-def}$ 
  by  $(smt (\text{verit}, \text{ccfv-SIG}) fprops assms(1) \text{image-cong image-image scEmbeddedCopy-def})$ 

```

**hence**  $e f ' (topspace (\beta X)) \subseteq \{m..M\}$   
**proof** –  
{ **fix**  $p$  **assume**  $pprops: p \in e f ' (topspace (\beta X))$   
**then obtain**  $v$  **where**  $vprops: v \in topspace (\beta X) \wedge p = e f v$  **by** *auto*  
{ **fix**  $U$  **assume**  $Uprops: openin (scT X f) U \wedge p \in U$   
**define**  $V$  **where**  $V = inverse' (e f) (topspace (\beta X)) U$   
**hence**  $openin (\beta X) V$   
**using** *c-embedded-f Uprops e-def unfolding continuous-map-def*  
*inverse'-def*  
**by** *auto*  
**moreover have**  $topspace (\beta X) = (\beta X) \text{ closure-of } scEmbeddedCopy X$   
**using** *scCompactification-def[of X] closure-of-subset-topspace[of  $\beta X$*   
*scEmbeddedCopy X]*  
*dense-embedding-scEmbed[of X] scEmbeddedCopy-def[of X]*  
**by** (*metis assms closure-of-subtopology-open embedding-map-in-subtopology*  
  
*subtopology-topspace topspace-subtopology*)  
**moreover have**  $v \in V \wedge v \in topspace (\beta X)$   
**using**  $vprops$  *V-def Uprops unfolding inverse'-def* **by** *auto*  
**ultimately obtain**  $x$  **where**  $xprops: x \in scEmbeddedCopy X \wedge x \in V$   
**using** *in-closure-of[of v  $\beta X$  scEmbeddedCopy X]*  
**by** *presburger*  
**define**  $w$  **where**  $w = e f x$   
**hence**  $w \in \{m..M\}$  **using** *extend-on-embedded xprops* **by** *blast*  
**moreover have**  $w \in U$  **using** *w-def xprops vprops V-def*  
**by** (*simp add: inverse'-alt*)  
**ultimately have**  $\exists w. w \in U \cap \{m..M\}$  **by** *auto*  
}
} **hence**  $\forall U. openin (scT X f) U \wedge p \in U \longrightarrow (\exists w. w \in U \cap \{m..M\})$   
**by** *auto*  
**moreover have**  $p \in topspace (scT X f)$   
**by** (*metis e-def Int-iff vprops c-embedded-f continuous-map-preimage-topspace*  
*vimageE*)  
**ultimately have**  $p \in (scT X f) \text{ closure-of } \{m..M\}$   
**using** *in-closure-of[of p scT X f {m..M}]*  
**by** *auto*  
**hence**  $p \in euclideanreal \text{ closure-of } \{m..M\}$   
**using** *scT-def[of X] range'-def[of X f]*  
**by** (*metis (no-types, lifting) closure-of-subtopology-subset fprops re-*  
*strict-apply' subsetD*)  
**hence**  $p \in \{m..M\}$  **by** *auto*  
}
} **thus** *?thesis* **by** *auto*  
**qed**  
**thus** *?thesis* **by** (*metis e-def atLeastAtMost-iff image-subset-iff*)  
**qed**  
**ultimately have**  $scExtend X f \in C^* (\beta X)$   
**using** *scT-def[of X] continuous-map-into-fulltopology fprops* **by** *auto*  
}

hence  $\forall f \in C^* X . scExtend X f \in C^* (\beta X)$  by *auto*  
 thus *?thesis using assms scExtend-extends by blast*  
 qed

**lemma** *cstar-embedding-scEmbed*:  
 assumes *tych-space X*  
 shows *cstar-embedding X ( $\beta X$ ) (scEmbed X)*  
 using *assms scExtend-extends-cstar[of X] dense-embedding-scEmbed[of X]*  
 by *meson*

A compact Hausdorff space is its own Stone-Cech compactification

**lemma** *scCompactification-of-compact-Hausdorff*:  
 assumes *compact-Hausdorff X*  
 shows *homeomorphic-map X ( $\beta X$ ) (scEmbed X)*  
**proof** –  
 have *dense: dense-embedding X ( $\beta X$ ) (scEmbed X)*  
 by (*simp add: assms compact-Hausdorff-imp-tych dense-embedding-scEmbed*)  
 moreover have *closed: closed-map X ( $\beta X$ ) (scEmbed X)*  
 by (*meson T1-Spaces.continuous-imp-closed-map assms compact-Hausdorff-imp-tych*)

*continuous-map-in-subtopology embedding-map-def dense*  
*homeomorphic-eq-everything-map scCompactification-compact-Hausdorff*)  
 moreover have *open-map X ( $\beta X$ ) (scEmbed X)*  
 by (*metis closed closure-of-subset-eq dense-in-def embedding-imp-closed-map-eq*  
*embedding-map-def homeomorphic-imp-open-map local.dense subtopol-*  
*ogy-superset*)  
 thus *?thesis*  
 by (*metis closed closure-of-subset-eq dense-in-def embedding-imp-closed-map-eq*  
*embedding-map-def local.dense subtopology-superset*)

qed

#### 4.4 The Stone-Čech Extension Property: Any continuous map from $X$ to a compact Hausdorff space $K$ extends uniquely to a continuous map from $\beta X$ to $K$ .

**proposition** *gof-cstar*:  
 assumes *compact-Hausdorff K*  
 and *continuous-map X K f*  
 shows  $\forall g \in C^* K . (g \circ f) \in C^* X$   
**proof** –  
 have *tych-K: tych-space K*  
 using *assms(1) compact-Hausdorff-imp-tych by auto*  
 { **fix**  $g$  **assume** *gprops:  $g \in C^* K$*   
 have *continuous-map K (scT K g) g*  
 using *closure-subset gprops by (fastforce simp: continuous-map-in-subtopology scT-def range'-def)*

**hence** *cts-scT*: *continuous-map*  $X$  (*scT*  $K$   $g$ ) ( $g \circ f$ )  
**using** *assms* **by** (*simp* *add*: *continuous-map-compose*)  
**hence** *gofprops*:  $(g \circ f) \in (C\ X)$   
**using** *scT-def*[*of*  $K$ ] *range'-def*[*of*  $K$ ]  
**by** (*metis* (*mono-tags*, *lifting*) *continuous-map-in-subtopology* *gprops* *mem-Collect-eq* *restrict-apply'*)  
**moreover** **have** *fbounded* ( $g \circ f$ )  $X$   
**proof** –  
**have** *compact* ( $g \text{ ' } \textit{topspace}$   $K$ ) **using** *assms*(1) *gprops*  
**using** *compact-space-def* *compactin-euclidean-iff* *image-compactin* **by** *blast*  
**hence** *bounded* ( $g \text{ ' } \textit{topspace}$   $K$ )  
**by** (*simp* *add*: *compact-imp-bounded*)  
**moreover** **have** ( $g \circ f$ )  $\text{ ' } \textit{topspace}$   $X \subseteq g \text{ ' } \textit{topspace}$   $K$   
**by** (*metis* *assms*(2) *continuous-map-image-subset-topospace* *image-comp* *image-mono*)  
**ultimately** **have** *bounded* ( $(g \circ f) \text{ ' } \textit{topspace}$   $X$ )  
**by** (*metis* *bounded-subset*)  
**thus** *?thesis* **using** *bounded-range-iff-fbounded*[*of*  $g \circ f$   $X$ ] *gofprops* **by** *auto*  
**qed**  
**ultimately** **have** ( $g \circ f$ )  $\in C^* X$  **by** *auto*  
**}**  
**thus** *?thesis* **by** *auto*  
**qed**

**proposition** *scEmbed-range*:  
**assumes** *tych-space*  $X$   
**and**  $x \in \textit{topspace}$   $X$   
**shows** *scEmbed*  $X$   $x \in \textit{topspace}$  ( $\beta$   $X$ )  
**using** *assms*(1) *assms*(2) *dense-embedding-scEmbed* *embedding-map-in-subtopology*  
**by** *fastforce*

**proposition** *scEmbed-range'*:  
**assumes** *tych-space*  $X$   
**and**  $x \in \textit{topspace}$   $X$   
**shows** *scEmbed*  $X$   $x \in \textit{topspace}$  (*scProduct*  $X$ )  
**using** *assms*(1) *assms*(2) *scEmbed-range*[*of*  $X$ ]  
**by** (*simp* *add*: *scCompactification-def*)

**proposition** *scProjection*:  
**shows**  $\forall f \in C^* X. \forall p \in \textit{topspace}$  (*scProduct*  $X$ ) . *scProject*  $X$   $f$   $p = p$   $f$   
**using** *scProject-def*[*of*  $X$ ] *scProduct-def*[*of*  $X$ ] *product-projection*[*of*  $C^* X$  *scT*  $X$ ]  
**by** *simp*

**proposition** *scProjections-continuous*:  
**shows**  $\forall f \in C^* X$  . *continuous-map* (*scProduct*  $X$ ) (*scT*  $X$   $f$ ) (*scProject*  $X$   $f$ )  
**proof** –  
**have**  $\forall f \in C^* X$  . *continuous-map* (*scProduct*  $X$ ) (*scT*  $X$   $f$ ) (*scProject*  $X$   $f$ )  
**using** *scProduct-def*[*of*  $X$ ] *scProject-def*[*of*  $X$ ]  
**by** (*metis* (*mono-tags*, *lifting*) *projections-continuous* *restrict-apply'*)

**thus** *?thesis* **using** *scCompactification-def*[of *X*] **by** *simp*  
**qed**

**proposition** *continuous-embedding-inverse*:

**assumes** *embedding-map* *X Y e*  
**shows**  $\exists e' . \text{continuous-map } (\text{subtopology } Y (e \text{ ' } \text{topspace } X)) X e' \wedge (\forall x \in \text{topspace } X . e' (e x) = x)$   
**by** (*meson* *assms embedding-map-def homeomorphic-map-maps homeomorphic-maps-def*)

**lemma** *scExtension-exists*:

**assumes** *tych-space* *X*  
**and** *compact-Hausdorff* *K*  
**shows**  $\forall f \in \text{cts}[X, K] . \exists F \in \text{cts}[\beta X, K] . (\forall x \in \text{topspace } X . F (\text{scEmbed } X x) = f x)$   
**proof** –  
{ **fix** *f* **assume** *fprops*: *f*  $\in \text{cts}[X, K]$

**have** *tych-K*: *tych-space* *K* **using** *assms(2)* *compact-Hausdorff-imp-tych*[of *K*]  
**by** *simp*

**define** *Xspace* **where** *Xspace* = *topspace* (*scProduct* *X*)  
**define** *Kspace* **where** *Kspace* = *topspace* (*scProduct* *K*)

**define** *H* **where** *H* =  $(\lambda p \in Xspace . \lambda g \in C^* K . \text{scProject } X (g \circ f) p)$

**have** *H-of-scEmbed*:  $\forall x \in \text{topspace } X . H (\text{scEmbed } X x) = \text{scEmbed } K (f x)$

**proof** –  
{ **fix** *x* **assume** *xprops*: *x*  $\in \text{topspace } X$   
**hence** *H* (*scEmbed* *X x*) =  $(\lambda p \in Xspace . \lambda g \in C^* K . \text{scProject } X (g \circ f) p) (\text{scEmbed } X x)$   
**using** *H-def* **by** *auto*  
**moreover** **have** (*scEmbed* *X x*)  $\in Xspace$   
**using** *Xspace-def* *assms(1)* *scEmbed-range'*[of *X x*] *xprops* **by** *auto*  
**ultimately** **have** *H* (*scEmbed* *X x*) =  $(\lambda g \in C^* K . \text{scProject } X (g \circ f) (\text{scEmbed } X x))$   
**by** *auto*  
**also** **have**  $\dots = (\lambda g \in C^* K . (g \circ f) x)$   
**using** *assms(2)* *gof-cstar*[of *K X f*] *xprops* *fprops* *assms(1)*  
*scEmbed-then-project***where** *x=x* **and** *X=X*  
**by** (*metis* (*no-types*, *lifting*) *mem-Collect-eq restrict-ext*)  
**also** **have**  $\dots = (\lambda g \in C^* K . g (f x))$  **by** *auto*  
**finally** **have** *H* (*scEmbed* *X x*) = *scEmbed* *K* (*f x*)  
**using** *scEmbed-def*[of *K*] *cstar-id-def*[of *K*] *evaluation-map-def*[of *K cstar-id*  
*K C^\* K*]

**by** (*smt* (*verit*) *continuous-map-image-subset-topspace fprops xprops image-subset-iff*  
*mem-Collect-eq restrict-apply' restrict-ext xprops*)  
**}**  
**thus** *?thesis* **by** *auto*  
**qed**  
**hence** *H-on-embedded: H ' scEmbeddedCopy X ⊆ scEmbeddedCopy K*  
**proof** –  
**{** **fix** *p* **assume** *p ∈ H ' scEmbeddedCopy X*  
**then obtain** *q* **where** *qprops: q ∈ scEmbeddedCopy X ∧ p = H q* **by** *auto*  
**then obtain** *x* **where** *xprops: x ∈ topspace X ∧ q = scEmbed X x*  
**using** *scEmbeddedCopy-def[of X]* **by** *auto*  
**hence** *p = scEmbed K (f x)* **using** *qprops H-of-scEmbed* **by** *auto*  
**hence** *p ∈ scEmbeddedCopy K*  
**using** *scEmbeddedCopy-def[of K] xprops qprops fprops*  
**by** (*metis continuous-map-image-subset-topspace image-eqI in-mono mem-Collect-eq*)  
**}**  
**thus** *?thesis* **by** *auto*  
**qed**

**have** *components-cts: ∀ g ∈ C\* K . continuous-map (scProduct X) (scT K g)*  
*(λx ∈ Xspace . H x g)*  
**proof** –  
**{** **fix** *g* **assume** *gprops: g ∈ C\* K*  
  
**have** *continuous-map (scProduct X) (scT X (g o f)) (λ x ∈ Xspace . H x*  
*g)*  
**proof** –  
**have** *∀ f ∈ C\* X . continuous-map (scProduct X) (scT X f) (scProject X*  
*f)*  
**using** *scProjections-continuous[of X]* **by** *simp*  
**hence** *continuous-map (scProduct X) (scT X (g o f)) (scProject X (g o*  
*f))*  
**using** *assms(2) fprops gprops gof-cstar[of K X f]* **by** *auto*  
**moreover have** *∀ x ∈ Xspace. H x g = (scProject X (g o f)) x*  
**using** *gprops H-def Xspace-def*  
**by** *auto*  
**ultimately show** *?thesis*  
**using** *Xspace-def continuous-map-eq* **by** *fastforce*  
**qed**

**moreover have** *scT X (g o f) = subtopology (scT K g) (range' X (g o f))*  
**proof** –  
**have** *(g o f) ' topspace X ⊆ g ' topspace K*  
**using** *gprops fprops unfolding continuous-map-def* **by** *auto*  
**hence** *range' X (g o f) ⊆ range' K g*  
**unfolding** *range'-def* **by** (*meson closure-of-mono*)

```

hence top-of-set (range' X (g o f))
  = subtopology (top-of-set (range' K g)) (range' X (g o f))
by (simp add: inf.absorb-iff2 subtopology-subtopology)
hence scT X (g o f) = subtopology (scT K g) (range' X (g o f))
  using scT-def[of X] scT-def[of K] gprops assms(2) gof-cstar[of K X f]
fprops by auto
  thus ?thesis by auto
qed
ultimately have continuous-map (scProduct X) (scT K g) (λ x ∈ Xspace
. H x g)
  using continuous-map-in-subtopology by auto
}
thus ?thesis by auto
qed

hence Hcts: continuous-map (scProduct X) (scProduct K) H
  using continuous-map-coordinatewise-then-product[of C* K scProduct X scT
K H]
  scProduct-def[of X] scProduct-def[of K] H-def Xspace-def
by (smt (verit, del-insts) continuous-map-eq restrict-apply)

have H-on-beta: H ' topspace (β X) ⊆ scEmbeddedCopy K
proof –
  have H ' scEmbeddedCopy X ⊆ scEmbeddedCopy K using H-on-embedded
by auto
  hence H ' topspace (β X) ⊆ scProduct K closure-of scEmbeddedCopy K
  using scCompactification-def[of X] Hcts closure-of-mono
  continuous-map-eq-image-closure-subset by fastforce
  thus ?thesis using scEmbeddedCopy-def
by (metis assms(2) closure-of-subset-topospace homeomorphic-imp-surjective-map

scCompactification-def scCompactification-of-compact-Hausdorff
topospace-subtopology-subset)
qed

have embeds: dense-embedding K (β K) (scEmbed K) using dense-embedding-scEmbed[of
K] tych-K by auto
  have closed: closedin (scProduct K) (scEmbeddedCopy K)
  using assms(2) scEmbeddedCopy-def[of X] scCompactification-def[of K]
  scCompactification-compact-Hausdorff[of K]
  by (metis closure-of-eq closure-of-subset-topospace closure-of-topospace dense-in-def
embeds
homeomorphic-map-closure-of scCompactification-of-compact-Hausdorff
scEmbeddedCopy-def
topospace-subtopology-subset)
hence onto: scEmbeddedCopy K = topspace (β K)
  using scCompactification-def[of K]
by (metis closure-of-eq closure-of-subset-topospace topspace-subtopology-subset)
then obtain e'

```

**where**  $e'_{\text{props}}$ : *continuous-map*  $(\beta K) K e'$   
 $\wedge (\forall x \in \text{topspace } K . e' (\text{scEmbed } K x) = x)$   
**by** (*metis continuous-embedding-inverse embeds scEmbeddedCopy-def*  
*subtopology-topospace*)

**define**  $F$  **where**  $F = e' o (\lambda p \in \text{topspace } (\beta X) . \text{restrict } H (\text{topspace } \beta X) p)$

**have**  $F_{\text{cts}}$ :  $F \in \text{cts}[\beta X, K]$   
**proof** –  
**have**  $(\lambda p \in \text{topspace } (\beta X) . \text{restrict } H (\text{topspace } \beta X) p) \in \text{cts}[\beta X, \text{scProduct } K]$   
**using**  $H_{\text{cts}}$   $X_{\text{space-def}}$  *continuous-map-from-subtopology*  $\text{scCompactification-def}$   
**by** (*metis closedin-subset closedin-topospace mem-Collect-eq restrict-continuous-map*)  
**moreover** **have**  $H \in (\text{topspace } \beta X) \rightarrow \text{topspace } (\beta K)$   
**using**  $X_{\text{space-def}}$   $H_{\text{on-beta}}$   $X_{\text{space-def}}$   $\text{scCompactification-def}$ [*of*  $K$ ] *onto*  
**by** *blast*  
**ultimately** **have**  $(\lambda p \in \text{topspace } (\beta X) . \text{restrict } H (\text{topspace } \beta X) p) \in \text{cts}[\beta X, \beta K]$   
**using**  $\text{scCompactification-def}$ [*of*  $K$ ] **by** (*simp add: Pi-iff continuous-map-into-subtopology*)  
**moreover** **have**  $e' \in \text{cts}[\beta K, K]$  **using**  $e'_{\text{props}}$  **by** *simp*  
**ultimately** **show** *?thesis*  
**using**  $F_{\text{def}}$  *continuous-map-compose*[*of*  $\beta X \beta K (\lambda p \in \text{topspace } (\beta X) . \text{restrict } H (\text{topspace } \beta X) p)$ ]  
**by** *auto*  
**qed**

**moreover** **have**  $F_{\text{extends}}$ :  $\forall x \in \text{topspace } X . (F o \text{scEmbed } X) x = f x$   
**proof** –  
**{** **fix**  $x$  **assume**  $x_{\text{props}}$ :  $x \in \text{topspace } X$   
**have**  $(F o \text{scEmbed } X) x = F (\text{scEmbed } X x)$  **by** *auto*  
**moreover** **have**  $\text{scEmbed } X x \in \text{topspace } (\beta X)$   
**using**  $\text{assms}(1)$   $\text{scEmbed-range}$ [*of*  $X x$ ]  $x_{\text{props}}$  **by** *auto*  
**ultimately** **have**  $(F o \text{scEmbed } X) x = (e' o (\lambda p \in \text{topspace } (\beta X) . \text{restrict } H (\text{topspace } \beta X) p)) (\text{scEmbed } X x)$   
**using**  $F_{\text{def}}$  **by** *simp*  
**also** **have**  $\dots = (e' o (\lambda p \in \text{topspace } (\beta X) . H p)) (\text{scEmbed } X x)$  **by** *auto*  
**finally** **have**  $\text{step1}$ :  $(F o \text{scEmbed } X) x = e' ((\lambda p \in \text{topspace } (\beta X) . H p) (\text{scEmbed } X x))$  **by** *auto*  
**have**  $(\lambda p \in \text{topspace } (\beta X) . H p) (\text{scEmbed } X x) = H (\text{scEmbed } X x)$   
**using**  $\text{scEmbed-range}$ [*of*  $X x$ ]  $\text{assms}(1)$   $x_{\text{props}}$  **by** *auto*  
**also** **have**  $\dots = \text{scEmbed } K (f x)$  **using**  $H_{\text{of-scEmbed}}$   $x_{\text{props}}$  **by** *auto*  
**finally** **have**  $\text{step2}$ :  $(\lambda p \in \text{topspace } (\beta X) . H p) (\text{scEmbed } X x) = \text{scEmbed } K (f x)$   
**by** *auto*  
**have**  $(F o \text{scEmbed } X) x = e' ((\lambda p \in \text{topspace } (\beta X) . H p) (\text{scEmbed } X x))$

**using** *step1* **by** *simp*  
**also have**  $\dots = e' (scEmbed\ K\ (f\ x))$  **using** *step2* **by** *auto*  
**finally have**  $(F\ o\ scEmbed\ X)\ x = f\ x$   
**using** *e'props* *tych-K* *scEmbed-range[of\ K\ f\ x]* *xprops* *fprops*  
**by** (*metis* *continuous-map-image-subset-topospace* *image-subset-iff* *mem-Collect-eq*)  
**}**  
**thus** *?thesis* **by** *auto*  
**qed**

**ultimately have**  $F \in cts[\beta\ X,\ K] \wedge (\forall\ x \in\ topspace\ X.\ F\ (scEmbed\ X\ x) = f\ x)$   
**by** *auto*  
**hence**  $\exists\ F \in cts[\beta\ X,\ K]. (\forall\ x \in\ topspace\ X.\ F\ (scEmbed\ X\ x) = f\ x)$  **by**  
*auto*  
**}**  
**thus** *?thesis* **by** *auto*  
**qed**

**lemma** *scExtension-unique*:  
**assumes**  $F \in cts[\beta\ X,\ K] \wedge (\forall\ x \in\ topspace\ X.\ F\ (scEmbed\ X\ x) = f\ x)$   
**and** *compact-Hausdorff\ K*  
**shows**  $(\forall\ G.\ G \in cts[\beta\ X,\ K] \wedge (\forall\ x \in\ topspace\ X.\ G\ (scEmbed\ X\ x) = f\ x))$   
 $\longrightarrow (\forall\ p \in\ topspace\ (\beta\ X).\ F\ p = G\ p)$

**proof** –  
**{** **fix** *G* **assume** *Gprops*:  $G \in cts[\beta\ X,\ K] \wedge (\forall\ x \in\ topspace\ X.\ G\ (scEmbed\ X\ x) = f\ x)$   
**have**  $\forall\ p \in\ scEmbeddedCopy\ X.\ F\ p = G\ p$   
**proof** –  
**{** **fix** *p* **assume** *pprops*:  $p \in\ scEmbeddedCopy\ X$   
**then obtain** *x* **where** *xprops*:  $x \in\ topspace\ X \wedge p = scEmbed\ X\ x$   
**using** *scEmbeddedCopy-def[of\ X]* **by** *auto*  
**hence**  $F\ p = G\ p$  **using** *assms* *Gprops* **by** *auto*  
**}**  
**thus** *?thesis* **by** *auto*  
**qed**

**moreover have** *dense-in*  $(\beta\ X)\ (scEmbeddedCopy\ X)\ (topspace\ (\beta\ X))$   
**by** (*metis* *closure-of-subset-topospace* *dense-in-closure* *dense-in-def* *scCompactification-def*  
*topospace-subtopology-subset*)

**moreover have** (*cts-on*  $\beta\ X$  *to-shared* *Hausdorff-space*)  $\{F, G\}$   
**proof** –  
**have** *Hausdorff-space* *K* **using** *assms(2)* **by** *auto*  
**moreover have**  $\forall\ g \in\ \{F, G\}.\ g \in\ cts[\beta\ X,\ K]$   
**using** *assms* *Gprops* **by** *auto*  
**ultimately have**  $\exists\ K.\ Hausdorff-space\ K \wedge \{F, G\} \subseteq cts[\beta\ X, K]$  **by** *auto*  
**thus** *?thesis* **by** *auto*  
**qed**

**ultimately have**  $(\forall\ p \in\ topspace\ (\beta\ X).\ F\ p = G\ p)$

```

    using continuous-maps-on-dense-subset[of F G β X scEmbeddedCopy X]
    by auto
  }
  thus ?thesis by auto
qed

lemma scExtension-property:
  assumes tych-space X
  and     compact-Hausdorff K
  shows    $\forall f \in \text{cts}[X, K] . \exists ! F \in \text{cts}_E[\beta X, K] . (\forall x \in \text{topspace } X . F (\text{scEmbed } X x) = f x)$ 
  proof -
    { fix f assume fprops: f ∈ cts[X, K]
      define P where  $P = (\lambda g . g \in \text{cts}_E[\beta X, K] \wedge (\forall x \in \text{topspace } X . g (\text{scEmbed } X x) = f x))$ 
      then obtain F where Fprops: F ∈ cts[β X, K] ∧ (∀ x ∈ topspace X . F (scEmbed X x) = f x)
      using scExtension-exists[of X K] assms fprops by auto
      define F' where  $F' = \text{restrict } F (\text{topspace } \beta X)$ 

      have  $F \in (\text{topspace } \beta X) \rightarrow \text{topspace } K$  using Fprops continuous-map-def[of β X K F] by auto
      hence F'ext: F' ∈ (topspace β X) →E topspace K
      using F'-def restrict-def[of F topspace β X] extensional-def[of topspace β X]
      by auto
      moreover have F'cts: F' ∈ cts[β X, K]
      proof -
        have  $F' \in (\text{topspace } \beta X) \rightarrow \text{topspace } K$  using F'ext by auto
        moreover have  $\forall U . \{x \in \text{topspace } \beta X . F' x \in U\} = \{x \in \text{topspace } \beta X . F' x \in U\}$ 
        using F'-def by auto
        ultimately show ?thesis using Fprops unfolding continuous-map-def by
        auto
      qed
      ultimately have  $F' \in \text{cts}_E[\beta X, K]$  by auto
      moreover have F'embed: (∀ x ∈ topspace X . F' (scEmbed X x) = f x)
      proof -
        have  $\forall x \in \text{topspace } X . \text{scEmbed } X x \in \text{topspace } \beta X$ 
        using assms(1) scEmbed-range[of X] by blast
        thus ?thesis using F'-def Fprops by fastforce
      qed
      ultimately have P F' using P-def by auto

      moreover have  $\forall G . P G \longrightarrow G = F'$ 
      proof -
        { fix G assume Gprops: P G
          { fix p
            have  $F' p = G p$ 
            proof (cases p ∈ topspace β X)

```

```

    case True
    hence  $F' \in \text{cts}[\beta X, K] \wedge (\forall x \in \text{topspace } X. F' (\text{scEmbed } X x) = f x)$ 
      using  $F' \text{cts } F' \text{embed}$  by auto
    moreover have  $G \in \text{cts}[\beta X, K] \wedge (\forall x \in \text{topspace } X. G (\text{scEmbed } X x)$ 
=  $f x)$ 
      using  $G \text{props } P\text{-def}$  by auto
    ultimately show ?thesis
      using  $\text{assms}(2)$   $\text{scExtension-unique}[\text{of } F' X K f]$  True by blast
  next
  case False
  hence  $F' p = \text{undefined}$  using  $F'\text{-def}$  by auto
  moreover have  $G p = \text{undefined}$ 
    using  $G \text{props } P\text{-def extensional-def}[\text{of } \text{topspace } \beta X]$  False by auto
  ultimately show ?thesis by auto
qed
}
hence  $\forall p. F' p = G p$  by auto
}
thus ?thesis by auto
qed
ultimately have  $\exists! F'. P F'$  by blast
hence  $\exists! F \in \text{cts}_E[\beta X, K]. (\forall x \in \text{topspace } X. F (\text{scEmbed } X x) = f x)$ 
  using  $P\text{-def}$  by auto
}
thus ?thesis by auto
qed

end

```

## References

- [Wal74] Russell C. Walker. *The Stone-Čech Compactification*. Springer-Verlag, 1974.
- [Wil70] Stephen Willard. *General Topology*. Addison-Wesley, 1970.