The Kuratowski Closure-Complement Theorem

Peter Gammie and Gianpaolo Gioiosa

October 27, 2017

Contents

1 Introduction

We discuss a topological curiosity discovered by Kuratowski (1922): the fact that the number of distinct operators on a topological space generated by compositions of closure and complement never exceeds 14, and is exactly 14 in the case of $\mathbb{R}$. In addition, we prove a theorem due to Chagrov (1982) that classifies topological spaces according to the number of such operators they support.

Kuratowski's result, which is exposited in Whitty (2015) and Chapter 7 of Chamberland (2015), has already been treated in Mizar — see Bagińska and Grabowski (2003) and Grabowski (2004). To the best of our knowledge, we are the first to mechanize Chagrov's result.

Our work is based on a presentation of Kuratowski's and Chagrov's results by Gardner and Jackson (2008).

We begin with some preliminary facts pertaining to the relationship between interiors of unions and unions of interiors (§2) and the relationship between $\mathbb{Q}$ and $\mathbb{R}$ (§3). We then prove Kuratowski's result (§4) and the corollary that at most 7 distinct operators on a topological space can be generated by compositions of closure and interior (§5). Finally, we prove Chagrov's result (§6).

2 Interiors and unions

definition
  $boundary :: 'a::topological\_space\ set \Rightarrow 'a\ set$
where
  $boundary\ X = closure\ X - interior\ X$

```
lemma boundary_empty:
  shows boundary {} = {}
unfolding boundary_def by simp

definition
  exterior :: 'a::topological_space set ⇒ 'a set
where
  exterior X = − (interior X ∪ boundary X)

lemma interior_union_boundary:
  shows interior (X ∪ Y) = interior X ∪ interior Y
        ⟷ boundary X ∩ boundary Y ⊆ boundary (X ∪ Y) (is (?lhs1 = ?lhs2) ⟷ ?rhs)
proof(rule iffI[OF _ subset_antisym[OF subsetI]])
  assume ?lhs1 = ?lhs2 then show ?rhs by (force simp: boundary_def)
next
  fix x
  assume ?rhs and x ∈ ?lhs1
  have x ∈ ?lhs2 if x ∉ interior X x ∉ interior Y
  proof(cases x ∈ boundary X ∩ boundary Y)
    case True with ‹?rhs› ‹x ∈ ?lhs1› show ?thesis by (simp add: boundary_def subset_iff)
  next
    case False then consider (X) x ∉ boundary X | (Y) x ∉ boundary Y by blast
    then show ?thesis
    proof cases
      case X
      from X ‹x ∉ interior X› have x ∈ exterior X by (simp add: exterior_def)
      from ‹x ∉ boundary X› ‹x ∈ exterior X› ‹x ∉ interior X›
      obtain U where open U U ⊆ − X x ∈ U
        by (metis ComplI DiffI boundary_def closure_interior interior_subset open_interior)
      from ‹x ∈ interior (X ∪ Y)› obtain U' where open U' U' ⊆ X ∪ Y x ∈ U'
        by (meson interiorE)
      from ‹U ⊆ − X› ‹U' ⊆ X ∪ Y› have U ∩ U' ⊆ Y by blast
      with ‹x ∉ interior Y› ‹open U'› ‹open U› ‹x ∈ U'› ‹x ∈ U› show ?thesis
        by (meson IntI interiorI open_Int)
    next
      case Y
      from Y ‹x ∉ interior Y› have x ∈ exterior Y by (simp add: exterior_def)
      from ‹x ∉ boundary Y› ‹x ∈ exterior Y› ‹x ∉ interior Y›
      obtain U where open U U ⊆ − Y x ∈ U
        by (metis ComplI DiffI boundary_def closure_interior interior_subset open_interior)
      from ‹x ∈ interior (X ∪ Y)› obtain U' where open U' U' ⊆ X ∪ Y x ∈ U'
        by (meson interiorE)
      from ‹U ⊆ −Y› ‹U' ⊆ X ∪ Y› have U ∩ U' ⊆ X by blast
      with ‹x ∉ interior X› ‹open U'› ‹open U› ‹x ∈ U'› ‹x ∈ U› show ?thesis
        by (meson IntI interiorI open_Int)
    qed
  qed
  with ‹x ∈ ?lhs1› show x ∈ ?lhs2 by blast
next
  show ?lhs2 ⊆ ?lhs1 by (simp add: interior_mono)
qed

lemma interior_union_closed_intervals:
  fixes a :: 'a::ordered_euclidean_space
  assumes b < c
  shows interior ({a..b} ∪ {c..d}) = interior {a..b} ∪ interior {c..d}
using assms by (subst interior_union_boundary; auto simp: boundary_def)
```

3 Additional facts about the rationals and reals

lemma *Rat_real_limpt*:
  fixes x :: *real*
  shows x *islimpt* $\mathbb{Q}$
proof(*rule islimptI*)
  fix T **assume** $x \in T$ *open* T
  then obtain e **where** $0 < e$ **and** *ball*: $|x' - x| < e \longrightarrow x' \in T$ **for** x' **by** (*auto simp*: *open_real*)
  from ‹$0 < e$› **obtain** q **where** $x <$ *real_of_rat* $q \wedge$ *real_of_rat* $q < x + e$ **using** *of_rat_dense* **by** *force*
  with *ball* **show** $\exists y \in \mathbb{Q}.\ y \in T \wedge y \neq x$ **by** *force*
qed

lemma *Rat_closure*:
  shows *closure* $\mathbb{Q}$ = (*UNIV* :: *real set*)
unfolding *closure_def* **using** *Rat_real_limpt* **by** *blast*

lemma *Rat_interval_closure*:
  fixes x :: *real*
  assumes $x < y$
  shows *closure* $(\{x<..<y\} \cap \mathbb{Q}) = \{x..y\}$
using *assms*
by (*metis* (*no_types*, *lifting*) *Rat_closure closure_closure closure_greaterThanLessThan closure_mono inf_le1 inf_top.right_ne*
open_Int_closure_subset open_real_greaterThanLessThan subset_antisym)

lemma *Rat_not_open*:
  fixes T :: *real set*
  assumes *open* T
  assumes $T \neq \{\}$
  shows $\neg T \subseteq \mathbb{Q}$
using *assms* **by** (*simp add*: *countable_rat open_minus_countable subset_eq*)

lemma *Irrat_dense_in_real*:
  fixes x :: *real*
  assumes $x < y$
  shows $\exists r \in -\mathbb{Q}.\ x < r \wedge r < y$
using *assms Rat_not_open*[**where** $T=\{x<..<y\}$] **by** *force*

lemma *closed_interval_Int_compl*:
  fixes x :: *real*
  assumes $x < y$
  assumes $y < z$
  shows $- \{x..y\} \cap - \{y..z\} = - \{x..z\}$
using *assms* **by** *auto*

4 Kuratowski's result

We prove that at most 14 distinct operators can be generated by compositions of *closure* and complement. For convenience, we give these operators short names and try to avoid pointwise reasoning. We treat the *interior* operator at the same time.

declare *o_apply*[*simp del*]

definition C :: $'a$::*topological_space set* $\Rightarrow$ $'a$ *set* **where** $C\ X = -\ X$

definition K :: $'a$::*topological_space set* $\Rightarrow$ $'a$ *set* **where** $K\ X$ = *closure* X

definition I :: $'a$::*topological_space set* $\Rightarrow$ $'a$ *set* **where** $I\ X$ = *interior* X

lemma *C_C*:
  shows $C \circ C = id$
by (*simp add*: *fun_eq_iff C_def o_apply*)

lemma *K_K*:
  shows $K \circ K = K$
by (*simp add*: *fun_eq_iff K_def o_apply*)

lemma *I_I*:
  shows $I \circ I = I$
unfolding *I_def* **by** (*simp add*: *o_def*)

lemma *I_K*:
  shows $I = C \circ K \circ C$
unfolding *C_def I_def K_def* **by** (*simp add*: *o_def interior_closure*)

lemma *K_I*:
  shows $K = C \circ I \circ C$
unfolding *C_def I_def K_def* **by** (*simp add*: *o_def interior_closure*)

lemma *K_I_K_I*:
  shows $K \circ I \circ K \circ I = K \circ I$
unfolding *C_def I_def K_def*
by (*clarsimp simp*: *fun_eq_iff o_apply closure_minimal closure_mono closure_subset interior_maximal interior_subset subset_antisym*)

lemma *I_K_I_K*:
  shows $I \circ K \circ I \circ K = I \circ K$
unfolding *C_def I_def K_def*
by (*simp add*: *fun_eq_iff o_apply*)
   (*metis* (*no_types*) *closure_closure closure_mono closure_subset interior_maximal interior_mono interior_subset open_interior subset_antisym*)

lemma *K_mono*:
  assumes $x \subseteq y$
  shows $K\ x \subseteq K\ y$
using *assms* **unfolding** *K_def* **by** (*simp add*: *closure_mono*)

The following lemma embodies the crucial observation about compositions of C and K:

lemma *KCKCKCK_KCK*:
  shows $K \circ C \circ K \circ C \circ K \circ C \circ K = K \circ C \circ K$ (**is** *?lhs* = *?rhs*)
proof(*rule ext*[*OF equalityI*])
  fix x
  have $(C \circ K \circ C \circ K \circ C \circ K)\ x \subseteq$ *?rhs* x **by** (*simp add*: *C_def K_def closure_def o_apply*)
  then have $(K \circ (C \circ K \circ C \circ K \circ C \circ K))\ x \subseteq (K \circ$ *?rhs*$)\ x$ **by** (*simp add*: *K_mono o_apply*)
  then show *?lhs* $x \subseteq$ *?rhs* x **by** (*simp add*: *K_K o_assoc*)
next
  fix x :: *'a::topological_space set*
  have $(C \circ K \circ C \circ K)\ x \subseteq K\ x$ **by** (*simp add*: *C_def K_def closure_def o_apply*)
  then have $(K \circ (C \circ K \circ C \circ K))\ x \subseteq (K \circ K)\ x$ **by** (*simp add*: *K_mono o_apply*)
  then have $(C \circ (K \circ K))\ x \subseteq (C \circ (K \circ (C \circ K \circ C \circ K)))\ x$ **by** (*simp add*: *C_def o_apply*)
  then have $(K \circ (C \circ (K \circ K)))\ x \subseteq (K \circ (C \circ (K \circ (C \circ K \circ C \circ K))))\ x$ **by** (*simp add*: *K_mono o_apply*)
  then show *?rhs* $x \subseteq$ *?lhs* x **by** (*simp add*: *K_K o_assoc*)
qed

The inductive set CK captures all operators that can be generated by compositions of C and K. We shallowly embed the operators; that is, we identify operators up to extensional equality.

inductive *CK* :: (*'a::topological_space set* $\Rightarrow$ *'a set*) $\Rightarrow$ *bool* **where**

CK C
| *CK K*
| ⟦ *CK f*; *CK g* ⟧ ⟹ *CK* (*f* ∘ *g*)

declare *CK.intros*[*intro*!]

lemma *CK_id*[*intro*!]:
CK id
by (*metis CK.intros*(*1*) *CK.intros*(*3*) *C_C*)

The inductive set *CK_nf* captures the normal forms for the 14 distinct operators.

inductive *CK_nf* :: (′*a*::*topological_space set* ⇒ ′*a set*) ⇒ *bool* **where**
CK_nf id
| *CK_nf C*
| *CK_nf K*
| *CK_nf* (*C* ∘ *K*)
| *CK_nf* (*K* ∘ *C*)
| *CK_nf* (*C* ∘ *K* ∘ *C*)
| *CK_nf* (*K* ∘ *C* ∘ *K*)
| *CK_nf* (*C* ∘ *K* ∘ *C* ∘ *K*)
| *CK_nf* (*K* ∘ *C* ∘ *K* ∘ *C*)
| *CK_nf* (*C* ∘ *K* ∘ *C* ∘ *K* ∘ *C*)
| *CK_nf* (*K* ∘ *C* ∘ *K* ∘ *C* ∘ *K*)
| *CK_nf* (*C* ∘ *K* ∘ *C* ∘ *K* ∘ *C* ∘ *K*)
| *CK_nf* (*K* ∘ *C* ∘ *K* ∘ *C* ∘ *K* ∘ *C*)
| *CK_nf* (*C* ∘ *K* ∘ *C* ∘ *K* ∘ *C* ∘ *K* ∘ *C*)

declare *CK_nf.intros*[*intro*!]

lemma *CK_nf_set*:
shows {*f* . *CK_nf f*} = {*id*, *C*, *K*, *C* ∘ *K*, *K* ∘ *C*, *C* ∘ *K* ∘ *C*, *K* ∘ *C* ∘ *K*, *C* ∘ *K* ∘ *C* ∘ *K*, *K* ∘ *C* ∘ *K* ∘ *C*, *C* ∘ *K* ∘ *C* ∘ *K* ∘ *C*, *K* ∘ *C* ∘ *K* ∘ *C* ∘ *K*, *C* ∘ *K* ∘ *C* ∘ *K* ∘ *C* ∘ *K*, *K* ∘ *C* ∘ *K* ∘ *C* ∘ *K* ∘ *C*, *C* ∘ *K* ∘ *C* ∘ *K* ∘ *C* ∘ *K* ∘ *C*}
by (*auto simp*: *CK_nf.simps*)

That each operator generated by compositions of C and K is extensionally equivalent to one of the normal forms captured by *CK_nf* is demonstrated by means of an induction over the construction of *CK_nf* and an appeal to the facts proved above.

theorem *CK_nf*:
CK f ⟷ *CK_nf f*
proof(*rule iffI*)
assume *CK f* **then show** *CK_nf f*
by *induct*
(*elim CK_nf.cases*; *clarsimp simp*: *id_def*[*symmetric*] *C_C K_K KCKCKCK_KCK o_assoc*; *simp add*: *o_assoc*[*symmetric*]; *clarsimp simp*: *C_C K_K KCKCKCK_KCK o_assoc*
| *blast*)+
next
assume *CK_nf f* **then show** *CK f* **by** *induct* (*auto simp*: *id_def*[*symmetric*])
qed

theorem *CK_card*:
shows *card* {*f*. *CK f*} ≤ *14*
by (*auto simp*: *CK_nf CK_nf_set card_insert intro*!: *le_trans*[*OF card_Diff1_le*])

We show, using the following subset of ℝ (an example taken from Rusin (2001)) as a witness, that there exist topological spaces on which all 14 operators are distinct.

definition
RRR :: *real set*

```
where
  RRR = {0<..<1} ∪ {1<..<2} ∪ {3} ∪ ({5<..<7} ∩ ℚ)
```

The following facts allow the required proofs to proceed by *simp*:

```
lemma RRR_closure:
  shows closure RRR = {0..2} ∪ {3} ∪ {5..7}
unfolding RRR_def by (force simp: closure_insert Rat_interval_closure)

lemma RRR_interior:
  interior RRR = {0<..<1} ∪ {1<..<2} (is ?lhs = ?rhs)
proof(rule equalityI[OF subsetI subsetI])
  fix x assume x ∈ ?lhs
  then obtain T where open T and x ∈ T and T ⊆ RRR by (blast elim: interiorE)
  then obtain e where 0 < e and ball x e ⊆ T by (blast elim!: openE)
  from ‹x ∈ T› ‹0 < e› ‹ball x e ⊆ T› ‹T ⊆ RRR›
  have False if x = 3
    using that unfolding RRR_def ball_def
    by (auto dest!: subsetD[where c=min (3 + e/2) 4] simp: dist_real_def)
  moreover
  from Irrat_dense_in_real[where x=x and y=x + e/2] ‹0 < e›
  obtain r where r∈− ℚ ∧ x < r ∧ r < x + e / 2 by auto
  with ‹x ∈ T›  ‹ball x e ⊆ T› ‹T ⊆ RRR›
  have False if x ∈ {5<..<7} ∩ ℚ
    using that unfolding RRR_def ball_def
    by (force simp: dist_real_def dest: subsetD[where c=r])
  moreover note ‹x ∈ interior RRR›
  ultimately show x ∈ ?rhs
    unfolding RRR_def by (auto dest: subsetD[OF interior_subset])
next
  fix x assume x ∈ ?rhs
  then show x ∈ ?lhs
    unfolding RRR_def interior_def by (auto intro: open_real_greaterThanLessThan)
qed

lemma RRR_interior_closure[simplified]:
  shows interior ({0::real..2} ∪ {3} ∪ {5..7}) = {0<..<2} ∪ {5<..<7} (is ?lhs = ?rhs)
proof −
  have ?lhs = interior ({0..2} ∪ {5..7})
   by (metis (no_types, lifting) Un_assoc Un_commute closed_Un closed_eucl_atLeastAtMost interior_closed_Un_empty_inter
interior_singleton)
  also have ... = ?rhs
    by (simp add: interior_union_closed_intervals)
  finally show ?thesis .
qed
```

The operators can be distinguished by testing which of the points in $\{1,2,3,4,6\}$ belong to their results.

```
definition
  test :: (real set ⇒ real set) ⇒ bool list
where
  test f ≡ map (λx. x ∈ f RRR) [1,2,3,4,6]

lemma RRR_test:
  assumes f RRR = g RRR
  shows test f = test g
unfolding test_def using assms by simp

lemma nf_RRR:
  shows
```

test id = [False, False, True, False, True]
test C = [True, True, False, True, False]
test K = [True, True, True, False, True]
test (K ∘ C) = [True, True, True, True, True]
test (C ∘ K) = [False, False, False, True, False]
test (C ∘ K ∘ C) = [False, False, False, False, False]
test (K ∘ C ∘ K) = [False, True, True, True, False]
test (C ∘ K ∘ C ∘ K) = [True, False, False, False, True]
test (K ∘ C ∘ K ∘ C) = [True, True, False, False, False]
test (C ∘ K ∘ C ∘ K ∘ C) = [False, False, True, True, True]
test (K ∘ C ∘ K ∘ C ∘ K) = [True, True, False, False, True]
test (C ∘ K ∘ C ∘ K ∘ C ∘ K) = [False, False, True, True, False]
test (K ∘ C ∘ K ∘ C ∘ K ∘ C) = [False, True, True, True, True]
test (C ∘ K ∘ C ∘ K ∘ C ∘ K ∘ C) = [True, False, False, False, False]
unfolding test_def C_def K_def
by (simp_all add: RRR_closure RRR_interior RRR_interior_closure closure_complement closed_interval_Int_compl o_apply)
(simp_all add: RRR_def)

theorem CK_nf_real_card:
shows card ((λ f. f RRR) ‘ {f . CK_nf f}) = 14
by (simp add: CK_nf_set) ((subst card_insert_disjoint; auto dest!: RRR_test simp: nf_RRR id_def[symmetric])[1])+

theorem CK_real_card:
shows card {f::real set ⇒ real set. CK f} = 14 (**is** ?lhs = ?rhs)
proof(rule antisym[OF CK_card])
show ?rhs ≤ ?lhs
unfolding CK_nf
by (rule le_trans[OF eq_imp_le[OF CK_nf_real_card[symmetric]] card_image_le])
(simp add: CK_nf_set)
qed

5 A corollary of Kuratowski's result

We show that it is a corollary of *CK_real_card* that at most 7 distinct operators on a topological space can be generated by compositions of closure and interior. In the case of $\mathbb{R}$, exactly 7 distinct operators can be so generated.

inductive IK :: ('a::topological_space set ⇒ 'a set) ⇒ bool **where**
IK id
| IK I
| IK K
| ⟦ IK f; IK g ⟧ ⟹ IK (f ∘ g)

inductive IK_nf :: ('a::topological_space set ⇒ 'a set) ⇒ bool **where**
IK_nf id
| IK_nf I
| IK_nf K
| IK_nf (I ∘ K)
| IK_nf (K ∘ I)
| IK_nf (I ∘ K ∘ I)
| IK_nf (K ∘ I ∘ K)

declare IK.intros[intro!]

declare IK_nf.intros[intro!]

lemma IK_nf_set:
{f . IK_nf f} = {id, I, K, I ∘ K, K ∘ I, I ∘ K ∘ I, K ∘ I ∘ K}

by (*auto simp*: *IK_nf.simps*)

theorem *IK_nf*:
  $IK\ f \longleftrightarrow IK\_nf\ f$
proof(*rule iffI*)
  assume $IK\ f$ **then show** $IK\_nf\ f$
    by *induct*
      (*elim IK_nf.cases*; *clarsimp simp*: *id_def*[*symmetric*] *o_assoc*; *simp add*: *I_I K_K o_assoc*[*symmetric*]; *clarsimp simp*: *K_I_K_I I_K_I_K o_assoc*
      | *blast*)+
next
  assume $IK\_nf\ f$ **then show** $IK\ f$ **by** *induct blast*+
qed

theorem *IK_card*:
  shows $card\ \{f.\ IK\ f\} \le 7$
by (*auto simp*: *IK_nf IK_nf_set card_insert intro*!: *le_trans*[*OF card_Diff1_le*])

theorem *IK_nf_real_card*:
  shows $card\ ((\lambda\ f.\ f\ RRR)\ `\ \{f\ .\ IK\_nf\ f\}) = 7$
by (*simp add*: *IK_nf_set*) ((*subst card_insert_disjoint*; *auto dest*!: *RRR_test simp*: *nf_RRR I_K id_def*[*symmetric*] *o_assoc*)[*1*])+

theorem *IK_real_card*:
  shows $card\ \{f{::}real\ set \Rightarrow real\ set.\ IK\ f\} = 7$ (**is** $?lhs = ?rhs$)
proof(*rule antisym*[*OF IK_card*])
  show $?rhs \le ?lhs$
    unfolding *IK_nf*
    by (*rule le_trans*[*OF eq_refl*[*OF IK_nf_real_card*[*symmetric*]] *card_image_le*])
      (*simp add*: *IK_nf_set*)
qed

6 Chagrov's result

Chagrov's theorem, which is discussed in Section 2.1 of Gardner and Jackson (2008), states that the number of distinct operators on a topological space that can be generated by compositions of closure and complement is one of 2, 6, 8, 10 or 14.

We begin by observing that the set of normal forms *CK_nf* can be split into two disjoint sets, *CK_nf_pos* and *CK_nf_neg*, which we define in terms of interior and closure.

inductive *CK_nf_pos* :: $('a{::}topological\_space\ set \Rightarrow 'a\ set) \Rightarrow bool$ **where**
  CK_nf_pos id
| *CK_nf_pos I*
| *CK_nf_pos K*
| $CK\_nf\_pos\ (I \circ K)$
| $CK\_nf\_pos\ (K \circ I)$
| $CK\_nf\_pos\ (I \circ K \circ I)$
| $CK\_nf\_pos\ (K \circ I \circ K)$

declare *CK_nf_pos.intros*[*intro*!]

lemma *CK_nf_pos_set*:
  shows $\{f\ .\ CK\_nf\_pos\ f\} = \{id,\ I,\ K,\ I \circ K,\ K \circ I,\ I \circ K \circ I,\ K \circ I \circ K\}$
by (*auto simp*: *CK_nf_pos.simps*)

definition
  CK_nf_neg :: $('a{::}topological\_space\ set \Rightarrow 'a\ set) \Rightarrow bool$
where

$CK\_nf\_neg\ f \longleftrightarrow (\exists\, g.\ CK\_nf\_pos\ g \wedge f = C \circ g)$

lemma *CK_nf_pos_neg_disjoint*:
assumes *CK_nf_pos f*
assumes *CK_nf_neg g*
shows $f \neq g$
using *assms* **unfolding** *CK_nf_neg_def*
by (*clarsimp simp*: *CK_nf_pos.simps*; *elim disjE*; *metis comp_def C_def I_def K_def Compl_iff closure_UNIV interior_UNIV id_apply*)

lemma *CK_nf_pos_neg_CK_nf*:
CK_nf f $\longleftrightarrow$ *CK_nf_pos f* $\vee$ *CK_nf_neg f* (**is** *?lhs* $\longleftrightarrow$ *?rhs*)
proof(*rule iffI*)
assume *?lhs* **then show** *?rhs*
unfolding *CK_nf_neg_def*
by (*rule CK_nf.cases*; *metis* (*no_types*, *lifting*) *CK_nf_pos.simps C_C I_K K_I comp_id o_assoc*)
next
assume *?rhs* **then show** *?lhs*
unfolding *CK_nf_neg_def*
by (*auto elim!*: *CK_nf_pos.cases simp*: *I_K C_C o_assoc*)
qed

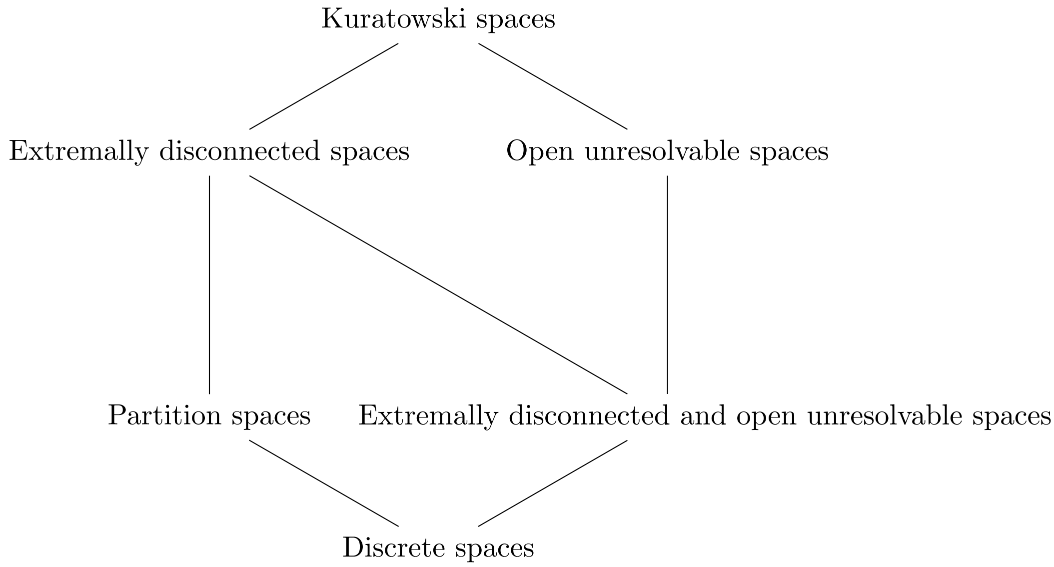
We now focus on *CK_nf_pos*. In particular, we show that its cardinality for any given topological space is one of 1, 3, 4, 5 or 7.

The proof consists of exhibiting normal forms for the operators supported by each of six classes of topological spaces. These are sublattices of the following lattice of *CK_nf_pos* operators:

lemmas *K_I_K_subseteq_K* = *closure_mono*[*OF interior_subset*, *of closure X*, *simplified*] **for** *X*

lemma *CK_nf_pos_lattice*:
shows
$I \leq (id :: 'a::topological\_space\ set \Rightarrow 'a\ set)$
$id \leq (K :: 'a::topological\_space\ set \Rightarrow 'a\ set)$
$I \leq I \circ K \circ (I :: 'a::topological\_space\ set \Rightarrow 'a\ set)$
$I \circ K \circ I \leq I \circ (K :: 'a::topological\_space\ set \Rightarrow 'a\ set)$
$I \circ K \circ I \leq K \circ (I :: 'a::topological\_space\ set \Rightarrow 'a\ set)$
$I \circ K \leq K \circ I \circ (K :: 'a::topological\_space\ set \Rightarrow 'a\ set)$
$K \circ I \leq K \circ I \circ (K :: 'a::topological\_space\ set \Rightarrow 'a\ set)$
$K \circ I \circ K \leq (K :: 'a::topological\_space\ set \Rightarrow 'a\ set)$
unfolding *I_def K_def*
by (*simp_all add*: *interior_subset closure_subset interior_maximal closure_mono o_apply interior_mono K_I_K_subseteq_K le_funI*)

We define the six classes of topological spaces in question, and show that they are related by inclusion in the following way (as shown in Figure 2.3 of Gardner and Jackson (2008)):



6.1 Discrete spaces

definition
discrete $(X :: {}'a{::}topological\_space\ set) \longleftrightarrow I = (id{::}{}'a\ set \Rightarrow {}'a\ set)$

lemma *discrete_eqs*:
assumes *discrete* $(X :: {}'a{::}topological\_space\ set)$
shows
$I\ = (id{::}{}'a\ set \Rightarrow {}'a\ set)$
$K\ = (id{::}{}'a\ set \Rightarrow {}'a\ set)$
using *assms* **unfolding** *discrete_def* **by** (*auto simp*: *C_C K_I*)

lemma *discrete_card*:
assumes *discrete* $(X :: {}'a{::}topological\_space\ set)$
shows *card* $\{f.\ CK\_nf\_pos\ (f{::}{}'a\ set \Rightarrow {}'a\ set)\} = 1$
using *discrete_eqs*[*OF assms*] *CK_nf_pos_lattice*[**where** $'a{=}'a$] **by** (*simp add*: *CK_nf_pos_set*)

lemma *discrete_discrete_topology*:
fixes $X :: {}'a{::}topological\_space\ set$
assumes $\bigwedge Y{::}{}'a\ set.\ open\ Y$
shows *discrete X*
using *assms* **unfolding** *discrete_def I_def interior_def islimpt_def* **by** (*auto simp*: *fun_eq_iff*)

6.2 Partition spaces

definition
part $(X :: {}'a{::}topological\_space\ set) \longleftrightarrow K \circ I = (I :: {}'a\ set \Rightarrow {}'a\ set)$

lemma *discrete_part*:
assumes *discrete X*
shows *part X*
using *assms* **unfolding** *discrete_def part_def* **by** (*simp add*: *C_C K_I*)

lemma *part_eqs*:
assumes *part* $(X :: {}'a{::}topological\_space\ set)$
shows
$K \circ I = (I :: {}'a\ set \Rightarrow {}'a\ set)$
$I \circ K = (K :: {}'a\ set \Rightarrow {}'a\ set)$
using *assms* **unfolding** *part_def* **by** (*assumption*, *metis* (*no_types*, *hide_lams*) *I_I K_I o_assoc*)

lemma *part_not_discrete_card*:
  assumes *part* $(X :: 'a::topological\_space\ set)$
  assumes $\neg$*discrete X*
  shows *card* $\{f.\ CK\_nf\_pos\ (f::'a\ set \Rightarrow 'a\ set)\} = 3$
using *part_eqs*[*OF* ⟨*part X*⟩] ⟨$\neg$*discrete X*⟩ *CK_nf_pos_lattice*[**where** *'a='a*]
unfolding *discrete_def*
by (*simp add*: *CK_nf_pos_set card_insert_if C_C I_K K_K o_assoc*; *metis comp_id*)

A partition space is a topological space whose basis consists of the empty set and the equivalence classes of points of the space induced by some equivalence relation R on the underlying set of the space. Equivalently, a partition space is one in which every open set is closed. Thus, for example, the class of partition spaces includes every topological space whose open sets form a boolean algebra.

datatype *part_witness* = *a* | *b* | *c*

lemma *part_witness_UNIV*:
  shows *UNIV* = *set* [*a*, *b*, *c*]
using *part_witness.exhaust* **by** *auto*

lemmas *part_witness_pow* = *subset_subseqs*[*OF subset_trans*[*OF subset_UNIV Set.equalityD1*[*OF part_witness_UNIV*]]]

lemmas *part_witness_Compl* = *Compl_eq_Diff_UNIV*[**where** *'a=part_witness*, *unfolded part_witness_UNIV*, *simplified*]

instantiation *part_witness* :: *topological_space*
begin

definition *open_part_witness X* $\longleftrightarrow$ $X \in \{\{\}, \{a\}, \{b, c\}, \{a, b, c\}\}$

lemma *part_witness_ball*:
  $(\forall s{\in}S.\ s \in \{\{\}, \{a\}, \{b, c\}, \{a, b, c\}\}) \longleftrightarrow S \subseteq set\ [\{\}, \{a\}, \{b, c\}, \{a, b, c\}]$
by *simp blast*

lemmas *part_witness_subsets_pow* = *subset_subseqs*[*OF iffD1*[*OF part_witness_ball*]]

instance proof *standard*
  fix *K* :: *part_witness set set*
  assume $\forall S{\in}K.$ *open S* **then show** *open* $(\bigcup K)$
    unfolding *open_part_witness_def*
    by − (*drule part_witness_subsets_pow*; *clarsimp*; *elim disjE*; *simp add*: *insert_commute*)
qed (*auto simp*: *open_part_witness_def part_witness_UNIV*)

end

lemma *part_witness_interior_simps*:
  shows
    interior $\{a\} = \{a\}$
    interior $\{b\} = \{\}$
    interior $\{c\} = \{\}$
    interior $\{a, b\} = \{a\}$
    interior $\{a, c\} = \{a\}$
    interior $\{b, c\} = \{b, c\}$
    interior $\{a, b, c\} = \{a, b, c\}$
unfolding *interior_def open_part_witness_def* **by** *auto*

lemma *part_witness_part*:
  fixes *X* :: *part_witness set*
  shows *part X*
proof −

```
  have closure (interior Y) = interior Y for Y :: part_witness set
    using part_witness_pow[where X=Y]
    by (auto simp: closure_interior part_witness_interior_simps part_witness_Compl insert_Diff_if)
  then show ?thesis
    unfolding part_def I_def K_def by (simp add: o_def)
qed

lemma part_witness_not_discrete:
  fixes X :: part_witness set
  shows ¬discrete X
unfolding discrete_def I_def
by (clarsimp simp: o_apply fun_eq_iff exI[where x={b}] part_witness_interior_simps)

lemma part_witness_card:
  shows card {f. CK_nf_pos (f::part_witness set ⇒ part_witness set)} = 3
by (rule part_not_discrete_card[OF part_witness_part part_witness_not_discrete])
```

6.3 Extremally disconnected and open unresolvable spaces

```
definition
  ed_ou (X :: 'a::topological_space set) ⟷ I ∘ K = K ∘ (I :: 'a set ⇒ 'a set)

lemma discrete_ed_ou:
  assumes discrete X
  shows ed_ou X
using assms unfolding discrete_def ed_ou_def by simp

lemma ed_ou_eqs:
  assumes ed_ou (X :: 'a::topological_space set)
  shows
    I ∘ K ∘ I = K ∘ (I :: 'a set ⇒ 'a set)
    K ∘ I ∘ K = K ∘ (I :: 'a set ⇒ 'a set)
    I ∘ K = K ∘ (I :: 'a set ⇒ 'a set)
using assms unfolding ed_ou_def by (metis I_I K_K o_assoc)+

lemma ed_ou_neqs:
  assumes ed_ou (X :: 'a::topological_space set)
  assumes ¬discrete X
  shows
    I ≠ (K :: 'a set ⇒ 'a set)
    I ≠ K ∘ (I :: 'a set ⇒ 'a set)
    K ≠ K ∘ (I :: 'a set ⇒ 'a set)
    I ≠ (id :: 'a set ⇒ 'a set)
    K ≠ (id :: 'a set ⇒ 'a set)
using assms CK_nf_pos_lattice[where 'a='a]
unfolding ed_ou_def discrete_def
by (metis (no_types, lifting) C_C I_K K_I comp_id o_assoc antisym)+

lemma ed_ou_not_discrete_card:
  assumes ed_ou (X :: 'a::topological_space set)
  assumes ¬discrete X
  shows card {f. CK_nf_pos (f::'a set ⇒ 'a set)} = 4
using ed_ou_eqs[OF ‹ed_ou X›] ed_ou_neqs[OF assms]
by (subst CK_nf_pos_set) (subst card_insert_disjoint; (auto)[1])+
```

We consider an example extremally disconnected and open unresolvable topological space.

```
datatype ed_ou_witness = a | b | c | d | e
```

```
lemma ed_ou_witness_UNIV:
  shows UNIV = set [a, b, c, d, e]
using ed_ou_witness.exhaust by auto

lemmas ed_ou_witness_pow = subset_subseqs[OF subset_trans[OF subset_UNIV Set.equalityD1[OF ed_ou_witness_UNIV]]]

lemmas ed_ou_witness_Compl = Compl_eq_Diff_UNIV[where 'a=ed_ou_witness, unfolded ed_ou_witness_UNIV,
simplified]

instance ed_ou_witness :: finite
by standard (simp add: ed_ou_witness_UNIV)

instantiation ed_ou_witness :: topological_space
begin

inductive open_ed_ou_witness :: ed_ou_witness set ⇒ bool where
  open_ed_ou_witness {}
| open_ed_ou_witness {a}
| open_ed_ou_witness {b}
| open_ed_ou_witness {e}
| open_ed_ou_witness {a, c}
| open_ed_ou_witness {b, d}
| open_ed_ou_witness {a, c, e}

| open_ed_ou_witness {a, b}
| open_ed_ou_witness {a, e}
| open_ed_ou_witness {b, e}
| open_ed_ou_witness {a, b, c}
| open_ed_ou_witness {a, b, d}
| open_ed_ou_witness {a, b, e}
| open_ed_ou_witness {b, d, e}
| open_ed_ou_witness {a, b, c, d}
| open_ed_ou_witness {a, b, c, e}
| open_ed_ou_witness {a, b, d, e}
| open_ed_ou_witness {a, b, c, d, e}

declare open_ed_ou_witness.intros[intro!]

lemma ed_ou_witness_inter:
  fixes S :: ed_ou_witness set
  assumes open S
  assumes open T
  shows open (S ∩ T)
using assms by (auto elim!: open_ed_ou_witness.cases)

lemma ed_ou_witness_union:
  fixes X :: ed_ou_witness set set
  assumes ∀x∈X. open x
  shows open (⋃X)
using finite[of X] assms
by (induct, force)
  (clarsimp; elim open_ed_ou_witness.cases; simp add: open_ed_ou_witness.simps subset_insertI2 insert_commute;
metis Union_empty_conv)

instance
by standard (auto simp: ed_ou_witness_UNIV intro: ed_ou_witness_inter ed_ou_witness_union)

end
```

lemma *ed_ou_witness_interior_simps*:
shows
interior {*a*} = {*a*}
interior {*b*} = {*b*}
interior {*c*} = {}
interior {*d*} = {}
interior {*e*} = {*e*}
interior {*a*, *b*} = {*a*, *b*}
interior {*a*, *c*} = {*a*, *c*}
interior {*a*, *d*} = {*a*}
interior {*a*, *e*} = {*a*, *e*}
interior {*b*, *c*} = {*b*}
interior {*b*, *d*} = {*b*, *d*}
interior {*b*, *e*} = {*b*, *e*}
interior {*c*, *d*} = {}
interior {*c*, *e*} = {*e*}
interior {*d*, *e*} = {*e*}
interior {*a*, *b*, *c*} = {*a*, *b*, *c*}
interior {*a*, *b*, *d*} = {*a*, *b*, *d*}
interior {*a*, *b*, *e*} = {*a*, *b*, *e*}
interior {*a*, *c*, *d*} = {*a*, *c*}
interior {*a*, *c*, *e*} = {*a*, *c*, *e*}
interior {*a*, *d*, *e*} = {*a*, *e*}
interior {*b*, *c*, *d*} = {*b*, *d*}
interior {*b*, *c*, *e*} = {*b*, *e*}
interior {*b*, *d*, *e*} = {*b*, *d*, *e*}
interior {*c*, *d*, *e*} = {*e*}
interior {*a*, *b*, *c*, *d*} = {*a*, *b*, *c*, *d*}
interior {*a*, *b*, *c*, *e*} = {*a*, *b*, *c*, *e*}
interior {*a*, *b*, *d*, *e*} = {*a*, *b*, *d*, *e*}
interior {*a*, *b*, *c*, *d*, *e*} = {*a*, *b*, *c*, *d*, *e*}
interior {*a*, *c*, *d*, *e*} = {*a*, *c*, *e*}
interior {*b*, *c*, *d*, *e*} = {*b*, *d*, *e*}
unfolding *interior_def* **by** *safe* (*clarsimp simp*: *open_ed_ou_witness.simps*; *blast*)+

lemma *ed_ou_witness_not_discrete*:
fixes *X* :: *ed_ou_witness set*
shows ¬*discrete X*
unfolding *discrete_def I_def* **using** *ed_ou_witness_interior_simps* **by** (*force simp*: *fun_eq_iff*)

lemma *ed_ou_witness_ed_ou*:
fixes *X* :: *ed_ou_witness set*
shows *ed_ou X*
unfolding *ed_ou_def I_def K_def*
proof(*clarsimp simp*: *o_apply fun_eq_iff*)
fix *x* :: *ed_ou_witness set*
from *ed_ou_witness_pow*[*of x*]
show *interior* (*closure x*) = *closure* (*interior x*)
by − (*simp*; *elim disjE*; *simp add*: *closure_interior ed_ou_witness_interior_simps ed_ou_witness_Compl insert_Diff_if*)
qed

lemma *ed_ou_witness_card*:
shows *card* {*f*. *CK_nf_pos* (*f*::*ed_ou_witness set* ⇒ *ed_ou_witness set*)} = *4*
by (*rule ed_ou_not_discrete_card*[*OF ed_ou_witness_ed_ou ed_ou_witness_not_discrete*])

6.4 Extremally disconnected spaces

definition
extremally_disconnected $(X :: 'a{::}topological\_space\ set) \longleftrightarrow K \circ I \circ K = I \circ (K :: 'a\ set \Rightarrow 'a\ set)$

lemma *ed_ou_part_extremally_disconnected*:
assumes *ed_ou X*
assumes *part X*
shows *extremally_disconnected X*
using *assms* **unfolding** *extremally_disconnected_def ed_ou_def part_def* **by** *simp*

lemma *extremally_disconnected_eqs*:
fixes $X :: 'a{::}topological\_space\ set$
assumes *extremally_disconnected X*
shows
$I \circ K \circ I = K \circ (I :: 'a\ set \Rightarrow 'a\ set)$
$K \circ I \circ K = I \circ (K :: 'a\ set \Rightarrow 'a\ set)$
using *assms* **unfolding** *extremally_disconnected_def* **by** (*metis K_I_K_I*)+

lemma *extremally_disconnected_not_part_not_ed_ou_card*:
fixes $X :: 'a{::}topological\_space\ set$
assumes *extremally_disconnected X*
assumes $\neg$*part X*
assumes $\neg$*ed_ou X*
shows *card* $\{f.\ CK\_nf\_pos\ (f{::}'a\ set \Rightarrow 'a\ set)\} = 5$
using *extremally_disconnected_eqs*[*OF* ‹*extremally_disconnected X*›] *CK_nf_pos_lattice*[**where** *'a='a*] *assms*(*2*,*3*)
unfolding *part_def ed_ou_def*
by (*simp add*: *CK_nf_pos_set C_C I_K K_K o_assoc card_insert_if*; *metis* (*no_types*) *C_C K_I id_comp o_assoc*)

Any topological space having an infinite underlying set and whose topology consists of the empty set and every cofinite subset of the underlying set is extremally disconnected. We consider an example such space having a countably infinite underlying set.

datatype *'a cofinite* = *cofinite 'a*

instantiation *cofinite* :: (*type*) *topological_space*
begin

definition *open_cofinite* = $(\lambda X{::}'a\ cofinite\ set.\ finite\ (-X) \lor X = \{\})$

instance
by *standard* (*auto simp*: *open_cofinite_def uminus_Sup*)

end

lemma *cofinite_closure_finite*:
fixes $X :: 'a\ cofinite\ set$
assumes *finite X*
shows *closure X = X*
using *assms* **by** (*simp add*: *closed_open open_cofinite_def*)

lemma *cofinite_closure_infinite*:
fixes $X :: 'a\ cofinite\ set$
assumes *infinite X*
shows *closure X = UNIV*
using *assms* **by** (*metis Compl_empty_eq closure_subset double_compl finite_subset interior_complement open_cofinite_def open_interior*)

lemma *cofinite_interior_finite*:

```
  fixes X :: 'a cofinite set
  assumes finite X
  assumes infinite (UNIV::'a cofinite set)
  shows interior X = {}
using assms cofinite_closure_infinite[where X=−X] by (simp add: interior_closure)

lemma cofinite_interior_infinite:
  fixes X :: 'a cofinite set
  assumes infinite X
  assumes infinite (−X)
  shows interior X = {}
using assms cofinite_closure_infinite[where X=−X] by (simp add: interior_closure)

abbreviation evens :: nat cofinite set ≡ {cofinite n | n. ∃ i. n=2∗i}

lemma evens_infinite:
  shows infinite evens
proof(rule iffD2[OF infinite_iff_countable_subset], rule exI, rule conjI)
  let ?f = λn::nat. cofinite (2∗n)
  show inj ?f by (auto intro: inj_onI)
  show range ?f ⊆ evens by auto
qed

lemma cofinite_nat_infinite:
  shows infinite (UNIV::nat cofinite set)
using evens_infinite finite_Diff2 by fastforce

lemma evens_Compl_infinite:
  shows infinite (− evens)
proof(rule iffD2[OF infinite_iff_countable_subset], rule exI, rule conjI)
  let ?f = λn::nat. cofinite (2∗n+1)
  show inj ?f by (auto intro: inj_onI)
  show range ?f ⊆ −evens by clarsimp presburger
qed

lemma evens_closure:
  shows closure evens = UNIV
using evens_infinite by (rule cofinite_closure_infinite)

lemma evens_interior:
  shows interior evens = {}
using evens_infinite evens_Compl_infinite by (rule cofinite_interior_infinite)

lemma cofinite_not_part:
  fixes X :: nat cofinite set
  shows ¬part X
unfolding part_def I_def K_def
using cofinite_nat_infinite
by (clarsimp simp: fun_eq_iff o_apply)
   (metis (no_types) cofinite_closure_finite cofinite_interior_finite double_compl finite.emptyI finite.insertI insert_not_empty interior_closure)

lemma cofinite_not_ed_ou:
  fixes X :: nat cofinite set
  shows ¬ed_ou X
unfolding ed_ou_def I_def K_def
by (clarsimp simp: fun_eq_iff o_apply evens_closure evens_interior exI[where x=evens])
```

lemma *cofinite_extremally_disconnected_aux*:
  fixes X :: *nat cofinite set*
  shows *closure* (*interior* (*closure* X)) $\subseteq$ *interior* (*closure* X)
by (*metis subsetI closure_closure closure_complement closure_def closure_empty finite_Un interior_eq open_cofinite_def open_interior*)

lemma *cofinite_extremally_disconnected*:
  fixes X :: *nat cofinite set*
  shows *extremally_disconnected* X
unfolding *extremally_disconnected_def I_def K_def*
by (*auto simp*: *fun_eq_iff o_apply dest*: *subsetD*[*OF closure_subset*] *subsetD*[*OF interior_subset*] *subsetD*[*OF cofinite_extremally_disconnected_aux*])

lemma *cofinite_card*:
  shows *card* {*f*. *CK_nf_pos* (*f*::*nat cofinite set* $\Rightarrow$ *nat cofinite set*)} = *5*
by (*rule extremally_disconnected_not_part_not_ed_ou_card*[*OF cofinite_extremally_disconnected cofinite_not_part cofinite_not_ed_ou*])

6.5 Open unresolvable spaces

definition
  open_unresolvable (X :: *'a*::*topological_space set*) $\longleftrightarrow$ $K \circ I \circ K = K \circ (I$ :: *'a set* $\Rightarrow$ *'a set*)

lemma *ed_ou_open_unresolvable*:
  assumes *ed_ou* X
  shows *open_unresolvable* X
using *assms* **unfolding** *open_unresolvable_def* **by** (*simp add*: *ed_ou_eqs*)

lemma *open_unresolvable_eqs*:
  assumes *open_unresolvable* (X :: *'a*::*topological_space set*)
  shows
    $I \circ K \circ I = I \circ (K$ :: *'a set* $\Rightarrow$ *'a set*)
    $K \circ I \circ K = K \circ (I$ :: *'a set* $\Rightarrow$ *'a set*)
using *assms* **unfolding** *open_unresolvable_def* **by** − (*metis I_K_I_K o_assoc*; *simp*)

lemma *not_ed_ou_neqs*:
  assumes ¬*ed_ou* (X :: *'a*::*topological_space set*)
  shows
    $I \neq I \circ (K$ :: *'a set* $\Rightarrow$ *'a set*)
    $K \neq K \circ (I$ :: *'a set* $\Rightarrow$ *'a set*)
using *assms* **unfolding** *ed_ou_def*
by (*simp_all add*: *fun_eq_iff I_K K_def C_def o_apply*)
   (*metis* (*no_types*, *hide_lams*) *closure_eq_empty disjoint_eq_subset_Compl double_complement interior_Int interior_complement set_eq_subset*)+

lemma *open_unresolvable_not_ed_ou_card*:
  assumes *open_unresolvable* (X :: *'a*::*topological_space set*)
  assumes ¬*ed_ou* X
  shows *card* {*f*. *CK_nf_pos* (*f*::*'a set* $\Rightarrow$ *'a set*)} = *5*
using *open_unresolvable_eqs*[*OF* ‹*open_unresolvable* X›] *not_ed_ou_neqs*[*OF* ‹¬*ed_ou* X›] ‹¬*ed_ou* X›
unfolding *ed_ou_def* **by** (*auto simp*: *CK_nf_pos_set card_insert_if*)

We show that the class of open unresolvable spaces is non-empty by exhibiting an example of such a space.

datatype *ou_witness* = *a* | *b* | *c*

lemma *ou_witness_UNIV*:
  shows *UNIV* = *set* [*a*, *b*, *c*]
using *ou_witness.exhaust* **by** *auto*

instantiation *ou_witness* :: *topological_space*
begin

definition *open_ou_witness X* $\longleftrightarrow$ *a* $\notin$ *X* $\vee$ *X* = *UNIV*

instance
by *standard* (*auto simp*: *open_ou_witness_def*)

end

lemma *ou_witness_closure_simps*:
  shows
    closure {*a*} = {*a*}
    closure {*b*} = {*a*, *b*}
    closure {*c*} = {*a*, *c*}
    closure {*a*, *b*} = {*a*, *b*}
    closure {*a*, *c*} = {*a*, *c*}
    closure {*a*, *b*, *c*} = {*a*, *b*, *c*}
    closure {*b*, *c*} = {*a*, *b*, *c*}
unfolding *closure_def islimpt_def open_ou_witness_def* **by** *force+*

lemma *ou_witness_open_unresolvable*:
  fixes *X* :: *ou_witness set*
  shows *open_unresolvable X*
unfolding *open_unresolvable_def I_def K_def*
by (*clarsimp simp*: *o_apply fun_eq_iff*)
  (*metis* (*no_types*, *lifting*) *Compl_iff K_I_K_subseteq_K closure_complement closure_interior closure_mono closure_subset interior_eq interior_maximal open_ou_witness_def subset_antisym*)

lemma *ou_witness_not_ed_ou*:
  fixes *X* :: *ou_witness set*
  shows ¬*ed_ou X*
unfolding *ed_ou_def I_def K_def*
by (*clarsimp simp*: *o_apply fun_eq_iff*)
  (*metis UNIV_I insert_iff interior_eq open_ou_witness_def singletonD*
      ou_witness.distinct(*4*,*5*) *ou_witness.simps*(*2*) *ou_witness_closure_simps*(*2*))

lemma *ou_witness_card*:
  shows *card* {*f*. *CK_nf_pos* (*f*::*ou_witness set* ⇒ *ou_witness set*)} = *5*
by (*rule open_unresolvable_not_ed_ou_card*[*OF ou_witness_open_unresolvable ou_witness_not_ed_ou*])

6.6 Kuratowski spaces

definition
  kuratowski (*X* :: *'a*::*topological_space set*) $\longleftrightarrow$
    ¬*extremally_disconnected X* ∧ ¬*open_unresolvable X*

A Kuratowski space distinguishes all 7 positive operators.

lemma *part_closed_open*:
  fixes *X* :: *'a*::*topological_space set*
  assumes *I* ∘ *K* ∘ *I* = (*I*::*'a set* ⇒ *'a set*)
  assumes *closed X*
  shows *open X*
proof(*rule Topological_Spaces.openI*)
  fix *x* **assume** *x* ∈ *X*
  let *?S* = *I* (−{*x*})
  let *?G* = −*K ?S*

```
  have x ∈ ?G
  proof -
    from ‹I ∘ K ∘ I = I› have I (K (I ?S)) = ?S I ?S = ?S
      unfolding I_def K_def by (simp_all add: o_def fun_eq_iff)
    then have K (I ?S) ≠ UNIV
      unfolding I_def K_def using interior_subset by fastforce
    moreover have G ⊆ ?S ∨ x ∈ G if open G for G
      using that unfolding I_def by (meson interior_maximal subset_Compl_singleton)
    ultimately show ?thesis
      unfolding I_def K_def
       by clarsimp (metis (no_types, lifting) ComplD Compl_empty_eq closure_interior closure_subset ex_in_conv
open_interior subset_eq)
  qed
  moreover from ‹I ∘ K ∘ I = I› have open ?G
    unfolding I_def K_def by (auto simp: fun_eq_iff o_apply)
  moreover have ?G ⊆ X
  proof -
    have ?G ⊆ K ?G unfolding K_def using closure_subset by fastforce
    also from ‹I ∘ K ∘ I = I› have ... = K {x}
      unfolding I_def K_def by (metis closure_interior comp_def double_complement)
    also from ‹closed X› ‹x ∈ X› have ... ⊆ X
      unfolding K_def by clarsimp (meson closure_minimal contra_subsetD empty_subsetI insert_subset)
    finally show ?thesis .
  qed
  ultimately show ∃ T. open T ∧ x ∈ T ∧ T ⊆ X by blast
qed

lemma part_I_K_I:
  assumes I ∘ K ∘ I = (I::'a::topological_space set ⇒ 'a set)
  shows I ∘ K = (K::'a set ⇒ 'a set)
using interior_open[OF part_closed_open[OF assms closed_closure]] unfolding I_def K_def o_def by simp

lemma part_K_I_I:
  assumes I ∘ K ∘ I = (I::'a::topological_space set ⇒ 'a set)
  shows K ∘ I = (I::'a set ⇒ 'a set)
using part_I_K_I[OF assms] assms by simp

lemma kuratowski_neqs:
  assumes kuratowski (X :: 'a::topological_space set)
  shows
    I ≠ I ∘ K ∘ (I :: 'a set ⇒ 'a set)
    I ∘ K ∘ I ≠ K ∘ (I :: 'a set ⇒ 'a set)
    I ∘ K ∘ I ≠ I ∘ (K :: 'a set ⇒ 'a set)
    I ∘ K ≠ K ∘ I ∘ (K :: 'a set ⇒ 'a set)
    K ∘ I ≠ K ∘ I ∘ (K :: 'a set ⇒ 'a set)
    K ∘ I ∘ K ≠ (K :: 'a set ⇒ 'a set)
    I ∘ K ≠ K ∘ (I :: 'a set ⇒ 'a set)
    I ≠ (id :: 'a set ⇒ 'a set)
    K ≠ (id :: 'a set ⇒ 'a set)
    I ∘ K ∘ I ≠ (id :: 'a set ⇒ 'a set)
    K ∘ I ∘ K ≠ (id :: 'a set ⇒ 'a set)
using assms unfolding kuratowski_def extremally_disconnected_def open_unresolvable_def
by (metis (no_types, lifting) I_K K_K I_K_I_K K_I_K_I part_I_K_I part_K_I_I o_assoc comp_id)+

lemma kuratowski_card:
  assumes kuratowski (X :: 'a::topological_space set)
  shows card {f. CK_nf_pos (f::'a set ⇒ 'a set)} = 7
using CK_nf_pos_lattice[where 'a='a] kuratowski_neqs[OF assms] assms
```

unfolding *kuratowski_def extremally_disconnected_def open_unresolvable_def*
by (*subst CK_nf_pos_set*) (*subst card_insert_disjoint*; (*auto*)[*1*])+

$\mathbb{R}$ is a Kuratowski space.

lemma *kuratowski_reals*:
  shows *kuratowski* ($\mathbb{R}$ :: *real set*)
unfolding *kuratowski_def extremally_disconnected_def open_unresolvable_def*
by (*rule conjI*)
  (*metis* (*no_types*, *lifting*) *I_K list.inject nf_RRR*(*11*) *nf_RRR*(*8*) *o_assoc*,
   metis (*no_types*, *lifting*) *I_K fun.map_comp list.inject nf_RRR*(*11*) *nf_RRR*(*9*))

6.7 Chagrov's theorem

theorem *chagrov*:
  fixes X :: *'a::topological_space set*
  obtains *discrete* X
    | ¬*discrete* X ∧ *part* X
    | ¬*discrete* X ∧ *ed_ou* X
    | ¬*ed_ou* X ∧ *open_unresolvable* X
    | ¬*ed_ou* X ∧ ¬*part* X ∧ *extremally_disconnected* X
    | *kuratowski* X
unfolding *kuratowski_def* **by** *metis*

corollary *chagrov_card*:
  shows *card* {*f*. *CK_nf_pos* (*f*::*'a::topological_space set* ⇒ *'a set*)} ∈ {*1*,*3*,*4*,*5*,*7*}
using *discrete_card part_not_discrete_card ed_ou_not_discrete_card open_unresolvable_not_ed_ou_card*
    extremally_disconnected_not_part_not_ed_ou_card kuratowski_card
by (*cases rule*: *chagrov*) *blast*+

References

L. K. Bagińska and A. Grabowski. On the Kuratowski closure-complement problem. *Journal of Formalized Mathematics*, 15, 2003. URL http://mizar.org/JFM/Vol15/kurato_1.html. MML Identifier: KURATO_1.

A. V. Chagrov. Kuratowski numbers. *Application of functional analysis in approximation theory*, page 186–190, 1982. Gos. Univ., Kalinin [Russian].

M. Chamberland. *Single Digits: In Praise of Small Numbers.* Princeton University Press, 2015.

B.J. Gardner and M. Jackson. The Kuratowski closure-complement theorem. *New Zealand Journal of Mathematics*, 38:9–44, 2008. URL http://nzjm.math.auckland.ac.nz/images/6/63/The_Kuratowski_Closure-Complement_Theorem.pdf.

A. Grabowski. Solving two problems in general topology via types. In J.-C. Filliâtre, C. Paulin-Mohring, and B. Werner, editors, *TYPES 2004*, volume 3839 of *LNCS*, pages 138–153. Springer, 2004.

K. Kuratowski. Sur l'opération Ā de l'analysis situs. *Fundamenta Mathematicae*, (3):182–199, 1922.

D. Rusin, June 2001. URL http://web.archive.org/web/20031011151110/http://www.math.niu.edu/~rusin/known-math/94/kuratowski.

R. Whitty. Kuratowski's 14-set theorem, 2015. URL http://www.theoremoftheday.org/Topology/Kuratowski14/TotDKuratowski14.pdf.