

The Topology of Lazy Lists

Stefan Friedrich

December 12, 2009

Abstract

This directory contains two theories. The first, `Topology`, develops the basic notions of general topology. The second, `LList.Topology`, develops the topology of lazy lists.

Contents

| | | |
|----------|--|-----------|
| 1 | A bit of general topology | 2 |
| 1.1 | Preliminaries | 2 |
| 1.2 | Definition | 2 |
| 1.3 | Neighbourhoods | 7 |
| 1.4 | Closed sets | 8 |
| 1.5 | Core, closure, and frontier of a set | 9 |
| 1.5.1 | Core | 10 |
| 1.5.2 | Closure | 11 |
| 1.5.3 | Frontier | 12 |
| 1.5.4 | Adherent points | 13 |
| 1.6 | More about closure and core | 13 |
| 1.7 | Dense sets | 14 |
| 1.8 | Continuous functions | 14 |
| 1.9 | Filters | 18 |
| 1.10 | Convergence | 21 |
| 1.11 | Separation | 22 |
| 1.11.1 | T0 spaces | 22 |
| 1.11.2 | T1 spaces | 22 |
| 1.11.3 | T2 spaces (Hausdorff spaces) | 23 |
| 1.11.4 | T3 axiom and regular spaces | 25 |
| 1.11.5 | T4 axiom and normal spaces | 25 |
| 2 | The topology of llists | 27 |
| 2.1 | The topology of all llists | 27 |
| 2.2 | The topology of infinite llists | 28 |
| 2.3 | The topology of non-empty llists | 30 |

1 A bit of general topology

```
theory Topology
imports FuncSet Zorn
begin
```

This theory gives a formal account of basic notions of general topology as they can be found in various textbooks, e.g. in [2] or in [3]. The development includes open and closed sets, neighbourhoods, as well as closure, open core, frontier, and adherent points of a set, dense sets, continuous functions, filters, ultra filters, convergence, and various separation axioms.

We use the theory on “Pi and Function Sets” by Florian Kammüller and Lawrence C Paulson.

1.1 Preliminaries

```
lemma seteqI:
```

```
"[[  $\bigwedge x. x \in A \implies x \in B$ ;  $\bigwedge x. x \in B \implies x \in A$  ]]  $\implies A = B$ "
<proof>
```

```
lemma subset_mono: " $A \subseteq B \implies M \subseteq A \longrightarrow M \subseteq B$ "
```

```
<proof>
```

```
lemma diff_diff:
```

```
" $C - (A - B) = (C - A) \cup (C \cap B)$ "
```

```
<proof>
```

```
lemma diff_diff_inter: " $[[B \subseteq A; B \subseteq X]] \implies (X - (A - B)) \cap A = B$ "
```

```
<proof>
```

```
lemmas diffsimps = double_diff Diff_Un vimage_Diff
Diff_Int_distrib Diff_Int
```

```
lemma vimage_compose:
```

```
" $f : A \rightarrow B \implies A \cap (f^{-1} B \cap f^{-1} g^{-1} m) = A \cap (g \circ f)^{-1} m$ "
```

```
<proof>
```

```
lemma funcset_comp:
```

```
"[[  $f : A \rightarrow B$ ;  $g : B \rightarrow C$  ]]  $\implies g \circ f : A \rightarrow C$ "
```

```
<proof>
```

1.2 Definition

A topology is defined by a set of sets (the open sets) that is closed under finite intersections and infinite unions.

```
types
```

```
'a top = "'a set set"
```

```
definition
```

```
carr :: "'a top  $\Rightarrow$  'a set" ("carrier $\iota$ ") where
```

```
"carr T =  $\bigcup T$ "
```

definition

```
is_open :: "'a top  $\Rightarrow$  'a set  $\Rightarrow$  bool" ("_ openz" [50] 50) where
  "is_open T s  $\longleftrightarrow$  s  $\in$  T"
```

locale carrier =

```
fixes T :: "'a top" (structure)
```

lemma (in carrier) openI:

```
"m  $\in$  T  $\Longrightarrow$  m open"
<proof>
```

lemma (in carrier) openE:

```
"[ m open; m  $\in$  T  $\Longrightarrow$  R ]  $\Longrightarrow$  R"
<proof>
```

lemma (in carrier) carrierI [intro]:

```
"[ t open; x  $\in$  t ]  $\Longrightarrow$  x  $\in$  carrier"
<proof>
```

lemma (in carrier) carrierE [elim]:

```
"[ x  $\in$  carrier;
   $\bigwedge$ t. [ t open; x  $\in$  t ]  $\Longrightarrow$  R
]  $\Longrightarrow$  R"
<proof>
```

lemma (in carrier) subM:

```
"[ t  $\in$  M; M  $\subseteq$  T ]  $\Longrightarrow$  t open"
<proof>
```

lemma (in carrier) topeqI [intro!]:

```
fixes S (structure)
shows "[  $\bigwedge$ m. m open1  $\Longrightarrow$  m open2;
         $\bigwedge$ m. m open2  $\Longrightarrow$  m open1 ]
        $\Longrightarrow$  T = S"
```

<proof>

locale topology = carrier T for T (structure) +

```
assumes Int_open [intro!]: "[ x open; y open ]  $\Longrightarrow$  x  $\cap$  y open"
and union_open [intro]: " $\forall$ m  $\in$  M. m open  $\Longrightarrow$   $\bigcup$  M open"
```

lemma topologyI:

```
"[  $\bigwedge$  x y. [ is_open T x; is_open T y ]  $\Longrightarrow$  is_open T (x  $\cap$  y);
   $\bigwedge$  M.  $\forall$  m  $\in$  M. is_open T m  $\Longrightarrow$  is_open T ( $\bigcup$  M)
]  $\Longrightarrow$  topology T"
<proof>
```

lemma (in topology) Un_open [intro!]:

```
assumes abopen: "A open" "B open"
shows "A  $\cup$  B open"
```

<proof>

Common definitions of topological spaces require that the empty set and the carrier set of the space be open. With our definition, however, the carrier is implicitly given as the union of all open sets; therefore it is trivially open. The empty set is open by the laws of HOLs typed set theory.

```
lemma (in topology) empty_open [iff]: "{} open"
  <proof>
```

```
lemma (in topology) carrier_open [iff]: "carrier open"
  <proof>
```

```
lemma (in topology) open_kriterion:
  assumes t_contains_open: " $\bigwedge x. x \in t \implies \exists t'. t' \text{ open} \wedge x \in t' \wedge t' \subseteq t$ "
  shows "t open"
  <proof>
```

We can obtain a topology from a set of basic open sets by closing the set under finite intersections and arbitrary unions.

```
inductive_set
  topo :: "'a set set  $\Rightarrow$  'a top"
  for B :: "'a set set"
where
  basic [intro]: "x  $\in$  B  $\implies$  x  $\in$  topo B"
| inter [intro]: "[x  $\in$  topo B; y  $\in$  topo B]  $\implies$  x  $\cap$  y  $\in$  topo B"
| union [intro]: " $(\bigwedge x. x \in M \implies x \in \text{topo B}) \implies \bigcup M \in \text{topo B}$ "

locale topobase = carrier T for B and T (structure) +
  defines "T  $\equiv$  topo B"
```

```
lemma (in topobase) topo_open:
  "t open = (t  $\in$  topo B)"
  <proof>
```

```
lemma (in topobase)
  basic [intro]: "t  $\in$  B  $\implies$  t open" and
  inter [intro]: "[x open; y open]  $\implies$  (x  $\cap$  y) open" and
  union [intro]: " $(\bigwedge t. t \in M \implies t \text{ open}) \implies \bigcup M \text{ open}$ "
  <proof>
```

```
lemma (in topobase) topo_induct
  [case_names basic inter union, induct set: topo, consumes 1]:
  assumes opn: "x open"
  and bas: " $\bigwedge x. x \in B \implies P x$ "
  and int: " $\bigwedge x y. [x \text{ open}; P x; y \text{ open}; P y] \implies P (x \cap y)$ "
  and uni: " $\bigwedge M. (\forall t \in M. t \text{ open} \wedge P t) \implies P (\bigcup M)$ "
  shows "P x"
  <proof>
```

```
lemma topo_topology [iff]:
  "topology (topo B)"
  <proof>
```

```

lemma topo_mono:
  assumes asubb: "A  $\subseteq$  B"
  shows "topo A  $\subseteq$  topo B"
<proof>

```

```

lemma topo_open_imp:
  fixes A and S (structure) defines "S  $\equiv$  topo A"
  fixes B and T (structure) defines "T  $\equiv$  topo B"
  shows "[[ A  $\subseteq$  B; x open1 ]  $\implies$  x open2]" (is "PROP ?P")
<proof>

```

```

lemma (in topobase) carrier_topo: "carrier =  $\bigcup$ B"
<proof>

```

Topological subspace

```

locale subtopology = carrier S + carrier T for S (structure) and T (structure) +
  assumes subtop[iff]: "s open = ( $\exists$ t. t open2  $\wedge$  s = t  $\cap$  carrier)"

```

```

lemma subtopologyI:
  fixes S (structure)
  fixes T (structure)
  assumes H1: " $\bigwedge$ s. s open  $\implies$   $\exists$ t. t open2  $\wedge$  s = t  $\cap$  carrier"
  and      H2: " $\bigwedge$ t. t open2  $\implies$  t  $\cap$  carrier open"
  shows "subtopology S T"
<proof>

```

```

lemma (in subtopology) subtopologyE [elim]:
  assumes major: "s open"
  and      minor: " $\bigwedge$ t. [ t open2; s = t  $\cap$  carrier ]  $\implies$  R"
  shows "R"
<proof>

```

```

lemma (in subtopology) subtopI [intro]:
  "t open2  $\implies$  t  $\cap$  carrier open"
<proof>

```

```

lemma (in subtopology) carrier_subset:
  "carrier1  $\subseteq$  carrier2"
<proof>

```

```

lemma (in subtopology) subtop_sub:
  assumes "topology T"
  assumes carrSopen: "carrier1 open2"
  and s_open: "s open1"
  shows "s open2"
<proof>

```

```

lemma (in subtopology) subtop_topology [iff]:
  assumes "topology T"
  shows "topology S"
<proof>

```

```

lemma subtop_lemma:

```

```

fixes A and S (structure) defines "S  $\equiv$  topo A"
fixes B and T (structure) defines "T  $\equiv$  topo B"
assumes Asub: "A = ( $\bigcup_{t \in B}. \{ t \cap \bigcup A \}$ )"
shows "subtopology S T"
<proof>

Sample topologies

definition
  trivial_top :: "'a top" where
    "trivial_top = {{}}"

definition
  discrete_top :: "'a set  $\Rightarrow$  'a set set" where
    "discrete_top X = Pow X"

definition
  indiscrete_top :: "'a set  $\Rightarrow$  'a set set" where
    "indiscrete_top X = {{}, X}"

definition
  order_base :: "('a::order) set  $\Rightarrow$  'a set set" where
    "order_base A = ( $\bigcup_{x \in A}. \{y. y \in A \wedge x \leq y\}$ )"

definition
  order_top :: "('a::order) set  $\Rightarrow$  'a set set" where
    "order_top X = topo(order_base X)"

locale trivial = carrier +
  defines "T  $\equiv$  {{}}"

lemma (in trivial) open_iff [iff]:
  "m open = (m = {})"
  <proof>

lemma trivial_topology:
  fixes T (structure) defines "T  $\equiv$  {{}}"
  shows "topology T"
  <proof>

lemma empty_carrier_implies_trivial:
  fixes S (structure) assumes "topology S"
  fixes T (structure) defines "T  $\equiv$  {{}}"
  shows "carrier = {}  $\implies$  S = T" (is "PROP ?P")
  <proof>

locale discrete = carrier T for X and T (structure) +
  defines "T  $\equiv$  discrete_top X"

lemma (in discrete) carrier:
  "carrier = X"
  <proof>

lemma (in discrete) open_iff [iff]:

```

```

    "t open = (t ∈ Pow carrier)"
  ⟨proof⟩

lemma discrete_topology: "topology (discrete_top X)"
  ⟨proof⟩

locale indiscrete = carrier T for X and T (structure) +
  defines "T ≡ indiscrete_top X"

lemma (in indiscrete) carrier:
  "X = carrier"
  ⟨proof⟩

lemma (in indiscrete) open_iff [iff]:
  "t open = (t = {} ∨ t = carrier)"
  ⟨proof⟩

lemma indiscrete_topology: "topology (indiscrete_top X)"
  ⟨proof⟩

locale orderbase =
  fixes X and B
  defines "B ≡ order_base X"

locale ordertop1 = orderbase X B + topobase B T for X and B and T (structure)

locale ordertop = carrier T for X and T (structure) +
  defines "T ≡ order_top X"

lemma (in ordertop) ordertop_open:
  "t open = (t ∈ order_top X)"
  ⟨proof⟩

lemma ordertop_topology [iff]:
  "topology (order_top X)"
  ⟨proof⟩



### 1.3 Neighbourhoods

definition
  nhd :: "'a top ⇒ 'a ⇒ 'a set set" ( "nhds" ) where
  "nhd T x = {U. U ⊆ carr T ∧ (∃ m. is_open T m ∧ x ∈ m ∧ m ⊆ U)}"

lemma (in carrier) nhdI [intro]:
  "[[ U ⊆ carrier; m open; x ∈ m; m ⊆ U ] ] ⇒ U ∈ nhds x"
  ⟨proof⟩

lemma (in carrier) nhdE [elim]:
  "[[ U ∈ nhds x; ∧m. [ U ⊆ carrier; m open; x ∈ m; m ⊆ U ] ] ] ⇒ R ] ⇒ R"
  ⟨proof⟩

lemma (in carrier) elem_in_nhd:
  "U ∈ nhds x ⇒ x ∈ U"

```

```

    <proof>

lemma (in carrier) carrier_nhd [intro]: "x ∈ carrier ⇒ carrier ∈ nhds x"
  <proof>

lemma (in carrier) empty_not_nhd [iff]:
  "{} ∉ nhds x "
  <proof>

lemma (in carrier) nhds_greater:
  "[[V ⊆ carrier; U ⊆ V; U ∈ nhds x]] ⇒ V ∈ nhds x"
  <proof>

lemma (in topology) nhds_inter:
  assumes nhdU: "U ∈ nhds x"
  and      nhdV: "V ∈ nhds x"
  shows "(U ∩ V) ∈ nhds x"
  <proof>

lemma (in carrier) sub_nhd:
  "U ∈ nhds x ⇒ ∃ V ∈ nhds x. V ⊆ U ∧ (∀ z ∈ V. U ∈ nhds z)"
  <proof>

lemma (in ordertop1) l1:
  assumes mopen: "m open"
  and xpoint: "x ∈ X"
  and ypoint: "y ∈ X"
  and xley: "x ≤ y"
  and xinm: "x ∈ m"
  shows "y ∈ m"
  <proof>

lemma (in ordertop1)
  assumes xpoint: "x ∈ X" and ypoint: "y ∈ X" and xley: "x ≤ y"
  shows "nhds x ⊆ nhds y"
  <proof>

```

1.4 Closed sets

A set is closed if its complement is open.

definition

```

is_closed :: "'a top ⇒ 'a set ⇒ bool" ("_ closed" [50] 50) where
  "is_closed T s ↔ is_open T (carr T - s)"

```

```

lemma (in carrier) closedI:
  "(carrier - s) open ⇒ s closed"
  <proof>

```

```

lemma (in carrier) closedE:
  "[[ s closed; (carrier - s) open ⇒ R ]] ⇒ R"
  <proof>

```

```

lemma (in topology) empty_closed [iff]:
  "{} closed"
  <proof>

lemma (in topology) carrier_closed [iff]:
  "carrier closed"
  <proof>

lemma (in carrier) compl_open_closed:
  assumes mopen: "m open"
  shows "(carrier - m) closed"
  <proof>

lemma (in carrier) compl_open_closed1:
  "[[ m ⊆ carrier; (carrier - m) closed ] ⇒ m open"
  <proof>

lemma (in carrier) compl_closed_iff [iff]:
  "m ⊆ carrier ⇒ (carrier - m) closed = (m open)"
  <proof>

lemma (in topology) Un_closed [intro!]:
  "[[ x closed; y closed ] ⇒ x ∪ y closed"
  <proof>

lemma (in topology) inter_closed:
  assumes xsclosed: "∧x. x∈S ⇒ x closed"
  shows "∩S closed"
  <proof>

corollary (in topology) Int_closed [intro!]:
  assumes abclosed: "A closed" "B closed"
  shows "A ∩ B closed"
  <proof>

lemma (in topology) closed_diff_open:
  assumes aclosed: "A closed"
  and bopen: "B open"
  shows "A - B closed"
  <proof>

lemma (in topology) open_diff_closed:
  assumes aclosed: "A closed"
  and bopen: "B open"
  shows "B - A open"
  <proof>

```

1.5 Core, closure, and frontier of a set

definition

```

cor :: "'a top ⇒ 'a set ⇒ 'a set"           ("corez") where
"cor T s = (∪{m. is_open T m ∧ m ⊆ s})"

```

definition

```

clsr :: "'a top  $\Rightarrow$  'a set  $\Rightarrow$  'a set"          ("closure $\iota$ ") where
"clsr T a = ( $\bigcap$ {c. is_closed T c  $\wedge$  a  $\subseteq$  c})"

```

definition

```

firt :: "'a top  $\Rightarrow$  'a set  $\Rightarrow$  'a set"          ("frontier $\iota$ ") where
"firt T s = clsr T s - cor T s"

```

1.5.1 Core**lemma (in carrier) coreI:**

```

"[[m open; m  $\subseteq$  s; x  $\in$  m]]  $\implies$  x  $\in$  core s"
<proof>

```

lemma (in carrier) coreE:

```

"[[ x  $\in$  core s;  $\bigwedge$ m. [[m open; m  $\subseteq$  s; x  $\in$  m]]  $\implies$  R]]  $\implies$  R"
<proof>

```

lemma (in topology) core_open [iff]:

```

"core a open"
<proof>

```

lemma (in carrier) core_subset:

```

"core a  $\subseteq$  a"
<proof>

```

lemmas (in carrier) core_subsetD = subsetD [OF core_subset, standard]

lemma (in carrier) core_greatest:

```

"[[ m open; m  $\subseteq$  a]]  $\implies$  m  $\subseteq$  core a"
<proof>

```

lemma (in carrier) core_idem [simp]:

```

"core (core a) = core a"
<proof>

```

lemma (in carrier) open_core_eq [simp]:

```

"a open  $\implies$  core a = a"
<proof>

```

lemma (in topology) core_eq_open:

```

"core a = a  $\implies$  a open"
<proof>

```

lemma (in topology) core_iff:

```

"a open = (core a = a)"
<proof>

```

lemma (in carrier) core_mono:

```

"a  $\subseteq$  b  $\implies$  core a  $\subseteq$  core b"
<proof>

```

lemma (in topology) core_Int [simp]:

```

"core (a ∩ b) = core a ∩ core b"
⟨proof⟩

lemma (in carrier) core_nhds:
  "⟦ U ⊆ carrier; x ∈ core U ⟧ ⇒ U ∈ nhds x"
  ⟨proof⟩

lemma (in carrier) nhds_core:
  "U ∈ nhds x ⇒ x ∈ core U"
  ⟨proof⟩

lemma (in carrier) core_nhds_iff:
  "U ⊆ carrier ⇒ (x ∈ core U) = (U ∈ nhds x)"
  ⟨proof⟩

1.5.2 Closure

lemma (in carrier) closureI [intro]:
  "(∧c. ⟦c closed; a ⊆ c⟧ ⇒ x ∈ c) ⇒ x ∈ closure a"
  ⟨proof⟩

lemma (in carrier) closureE [elim]:
  "⟦ x ∈ closure a; ¬ c closed ⇒ R; ¬ a ⊆ c ⇒ R; x ∈ c ⇒ R ⟧ ⇒ R"
  ⟨proof⟩

lemma (in carrier) closure_least:
  "s closed ⇒ closure s ⊆ s"
  ⟨proof⟩

lemma (in carrier) subset_closure:
  "s ⊆ closure s"
  ⟨proof⟩

lemma (in topology) closure_carrier [simp]:
  "closure carrier = carrier"
  ⟨proof⟩

lemma (in topology) closure_subset:
  "A ⊆ carrier ⇒ closure A ⊆ carrier"
  ⟨proof⟩

lemma (in topology) closure_closed [iff]:
  "closure a closed"
  ⟨proof⟩

lemma (in carrier) closure_idem [simp]:
  "closure (closure s) = closure s"
  ⟨proof⟩

lemma (in carrier) closed_closure_eq [simp]:
  "a closed ⇒ closure a = a"
  ⟨proof⟩

```

```

lemma (in topology) closure_eq_closed:
  "closure a = a  $\implies$  a closed"
  <proof>

lemma (in topology) closure_iff:
  "a closed = (closure a = a)"
  <proof>

lemma (in carrier) closure_mono1:
  "mono (closure)"
  <proof>

lemma (in carrier) closure_mono:
  "a  $\subseteq$  b  $\implies$  closure a  $\subseteq$  closure b"
  <proof>

lemma (in topology) closure_Un [simp]:
  "closure (a  $\cup$  b) = closure a  $\cup$  closure b"
  <proof>

```

1.5.3 Frontier

```

lemma (in carrier) frontierI:
  "[[x  $\in$  closure s; x  $\in$  core s  $\implies$  False]]  $\implies$  x  $\in$  frontier s"
  <proof>

lemma (in carrier) frontierE:
  "[[ x  $\in$  frontier s; [[x  $\in$  closure s; x  $\in$  core s  $\implies$  False]]  $\implies$  R ]]  $\implies$  R"
  <proof>

lemma (in topology) frontier_closed [iff]:
  "frontier s closed"
  <proof>

lemma (in carrier) frontier_Un_core:
  "frontier s  $\cup$  core s = closure s"
  <proof>

lemma (in carrier) frontier_Int_core:
  "frontier s  $\cap$  core s = {}"
  <proof>

lemma (in topology) closure_frontier [simp]:
  "closure (frontier a) = frontier a"
  <proof>

lemma (in topology) frontier_carrier [simp]:
  "frontier carrier = {}"
  <proof>

```

Hence frontier is not monotone. Also $\text{core}_{\top}(\text{frontier}_{\top} A) = \{\}$ is not a theorem as illustrated by the following counter example. By the way: could the counter example be proved using

an instantiation?

```
lemma counter_example_core_frontier:
  fixes X defines [simp]: "X  $\equiv$  (UNIV::nat set)"
  fixes T (structure) defines "T  $\equiv$  indiscrete_top X"
  shows "core (frontier {0}) = X"
<proof>
```

1.5.4 Adherent points

definition

```
adhs :: "'a top  $\Rightarrow$  'a  $\Rightarrow$  'a set  $\Rightarrow$  bool"      (infix "adh2" 50) where
"adhs T x A  $\longleftrightarrow$  ( $\forall$  U  $\in$  nhds T x. U  $\cap$  A  $\neq$  {})"
```

```
lemma (in carrier) adhCE [elim?]:
"[[x adh A; U  $\notin$  nhds x  $\implies$  R; U  $\cap$  A  $\neq$  {}  $\implies$  R]]  $\implies$  R"
<proof>
```

```
lemma (in carrier) adhI [intro]:
"( $\bigwedge$ U. U  $\in$  nhds x  $\implies$  U  $\cap$  A  $\neq$  {})  $\implies$  x adh A"
<proof>
```

```
lemma (in carrier) closure_imp_adh:
  assumes asub: "A  $\subseteq$  carrier"
  and closure: "x  $\in$  closure A"
  shows "x adh A"
<proof>
```

```
lemma (in carrier) adh_imp_closure:
  assumes xpoint: "x  $\in$  carrier"
  and adh: "x adh A"
  shows "x  $\in$  closure A"
<proof>
```

```
lemma (in topology) closed_adh:
  assumes Asub: "A  $\subseteq$  carrier"
  shows "A closed = ( $\forall$  x  $\in$  carrier. x adh A  $\longrightarrow$  x  $\in$  A)"
<proof>
```

```
lemma (in carrier) adh_closure_iff:
"[[ A  $\subseteq$  carrier; x  $\in$  carrier ]]  $\implies$  (x adh A) = (x  $\in$  closure A)"
<proof>
```

1.6 More about closure and core

```
lemma (in topology) closure_complement [simp]:
  shows "closure (carrier - A) = carrier - core A"
<proof>
```

```
lemma (in carrier) core_complement [simp]:
  assumes asub: "A  $\subseteq$  carrier"
  shows "core (carrier - A) = carrier - closure A"
<proof>
```

```

lemma (in carrier) core_closure_diff_empty [simp]:
  assumes asub: "A  $\subseteq$  carrier"
  shows "core (closure A - A) = {}"
<proof>

```

1.7 Dense sets

definition

```

is_densein :: "'a top  $\Rightarrow$  'a set  $\Rightarrow$  'a set  $\Rightarrow$  bool" (infix "denseinτ" 50) where
"is_densein T A B  $\longleftrightarrow$  B  $\subseteq$  clsr T A"

```

definition

```

is_dense :: "'a top  $\Rightarrow$  'a set  $\Rightarrow$  bool" ("_ denseτ" [50] 50) where
"is_dense T A = is_densein T A (carr T)"

```

```

lemma (in carrier) densinI [intro!]: "B  $\subseteq$  closure A  $\Longrightarrow$  A densein B"
<proof>

```

```

lemma (in carrier) denseinE [elim!]: "[ A densein B; B  $\subseteq$  closure A  $\Longrightarrow$  R ]  $\Longrightarrow$  R"
<proof>

```

```

lemma (in carrier) denseI [intro!]: "carrier  $\subseteq$  closure A  $\Longrightarrow$  A dense"
<proof>

```

```

lemma (in carrier) denseE [elim]: "[ A dense; carrier  $\subseteq$  closure A  $\Longrightarrow$  R ]  $\Longrightarrow$  R"
<proof>

```

```

lemma (in topology) dense_closure_eq [dest]:
" [ A dense; A  $\subseteq$  carrier ]  $\Longrightarrow$  closure A = carrier"
<proof>

```

```

lemma (in topology) dense_lemma:
"A  $\subseteq$  carrier  $\Longrightarrow$  carrier - (closure A - A) dense"
<proof>

```

```

lemma (in topology) ex_dense_closure_inter:
  assumes ssub: "S  $\subseteq$  carrier"
  shows " $\exists$  D C. D dense  $\wedge$  C closed  $\wedge$  S = D  $\cap$  C"
<proof>

```

```

lemma (in topology) ex_dense_closure_interE:
  assumes ssub: "S  $\subseteq$  carrier"
  and H: " $\bigwedge$  D C. [ D  $\subseteq$  carrier; C  $\subseteq$  carrier; D dense; C closed; S = D  $\cap$  C ]  $\Longrightarrow$  R"
  shows "R"
<proof>

```

1.8 Continuous functions

definition

```

INJ :: "'a set  $\Rightarrow$  'b set  $\Rightarrow$  ('a  $\Rightarrow$  'b) set" where

```

"INJ A B = {f. f : A → B ∧ inj_on f A}"

definition

SUR :: "'a set ⇒ 'b set ⇒ ('a ⇒ 'b) set" where
 "SUR A B = {f. f : A → B ∧ (∀ y∈B. ∃ x∈A. y = f x)}"

definition

BIJ :: "'a set ⇒ 'b set ⇒ ('a ⇒ 'b) set" where
 "BIJ A B = INJ A B ∩ SUR A B"

definition

cnt :: "'a top ⇒ 'b top ⇒ ('a ⇒ 'b) set" where
 "cnt S T = {f. f : carr S → carr T ∧
 (∀ m. is_open T m → is_open S (carr S ∩ (f -' m)))}"

definition

HOM :: "'a top ⇒ 'b top ⇒ ('a ⇒ 'b) set" where
 "HOM S T = {f. f ∈ cnt S T ∧ inv f ∈ cnt T S ∧ f ∈ BIJ (carr S) (carr T)}"

definition

homeo :: "'a top ⇒ 'b top ⇒ bool" where
 "homeo S T ↔ (∃ h ∈ BIJ (carr S) (carr T). h ∈ cnt S T ∧ inv h ∈ cnt T S)"

definition

fimg :: "'b top ⇒ ('a ⇒ 'b) ⇒ 'a set set ⇒ 'b set set" where
 "fimg T f F = {v. v ⊆ carr T ∧ (∃ u ∈ F. f'u ⊆ v)}"

lemma domain_subset_vimage:

"f : A → B ⇒ A ⊆ f-'B"
 <proof>

lemma domain_inter_vimage:

"f : A → B ⇒ A ∩ f-'B = A"
 <proof>

lemma funcset_vimage_diff:

"f : A → B ⇒ A - f-'(B - C) = A ∩ f-'C"
 <proof>

locale func = S: carrier S + T: carrier T

for f and S (structure) and T (structure) and fimage +
assumes func [iff]: "f : carrier₁ → carrier₂"
defines "fimage ≡ fimg T f"
notes func_mem [simp, intro] = funcset_mem [OF func]
and domain_subset_vimage [iff] = domain_subset_vimage [OF func]
and domain_inter_vimage [simp] = domain_inter_vimage [OF func]
and vimage_diff [simp] = funcset_vimage_diff [OF func]

lemma (in func) fimageI [intro!]:

shows "[v ⊆ carrier₂; u ∈ F; f'u ⊆ v] ⇒ v ∈ fimage F"
 <proof>

lemma (in func) fimageE [elim!]:
 "[v ∈ fimage F; $\bigwedge u. [v \subseteq \text{carrier}_2 ; u \in F; f' u \subseteq v] \implies R] \implies R$ "
 <proof>

lemma cntI:
 "[f : carr S → carr T;
 ($\bigwedge m. \text{is_open } T \ m \implies \text{is_open } S \ (\text{carr } S \cap (f \text{ -' } m))$)]
 $\implies f \in \text{cnt } S \ T$ "
 <proof>

lemma cntE:
 "[f ∈ cnt S T;
 [f : carr S → carr T;
 $\forall m. \text{is_open } T \ m \longrightarrow \text{is_open } S \ (\text{carr } S \cap (f \text{ -' } m))] \implies P$
] $\implies P$ "
 <proof>

lemma cntCE:
 "[f ∈ cnt S T;
 [$\neg \text{is_open } T \ m; f : \text{carr } S \rightarrow \text{carr } T] \implies P$;
 [$\text{is_open } S \ (\text{carr } S \cap (f \text{ -' } m)); f : \text{carr } S \rightarrow \text{carr } T] \implies P$
] $\implies P$ "
 <proof>

lemma cnt_fun:
 "f ∈ cnt S T $\implies f : \text{carr } S \rightarrow \text{carr } T$ "
 <proof>

lemma cntD1:
 "[f ∈ cnt S T; x ∈ carr S] $\implies f \ x \in \text{carr } T$ "
 <proof>

lemma cntD2:
 "[f ∈ cnt S T; is_open T m] $\implies \text{is_open } S \ (\text{carr } S \cap (f \text{ -' } m))$ "
 <proof>

locale continuous = func +
 assumes continuous [dest, simp]:
 "m open₂ $\implies \text{carrier} \cap (f \text{ -' } m)$ open"

lemma continuousI:
 fixes S (structure)
 fixes T (structure)
 assumes "f : carrier₁ → carrier₂"
 " $\bigwedge m. m \text{ open}_2 \implies \text{carrier} \cap (f \text{ -' } m)$ open"
 shows "continuous f S T"
 <proof>

lemma continuousE:
 fixes S (structure)
 fixes T (structure)
 shows

```

"[[ continuous f S T;
  [[ f : carrier1 → carrier2;
    ∀m. m open2 → carrier1 ∩ (f -' m) open ]] ⇒ P
  ]] ⇒ P"
<proof>

lemma continuousCE:
  fixes S (structure)
  fixes T (structure)
  shows
  "[[ continuous f S T;
    [[ ¬ m open2; f : carrier1 → carrier2 ]] ⇒ P;
    [[ carrier1 ∩ (f -' m) open1; f : carrier1 → carrier2 ]] ⇒ P
  ]] ⇒ P"
<proof>

lemma (in continuous) closed_vimage [intro, simp]:
  assumes csubset: "c ⊆ carrier2 "
  and cclosed: "c closed2"
  shows "f -' c closed"
<proof>

lemma continuousI2:
  fixes S (structure)
  fixes T (structure)
  assumes func: "f : carrier1 → carrier2"
  assumes R: "∧c. [[ c ⊆ carrier2; c closed2 ]] ⇒ f -' c closed"
  shows "continuous f S T"
<proof>

lemma cnt_compose:
  "[[ f ∈ cnt S T; g ∈ cnt T U ]] ⇒ (g ∘ f) ∈ cnt S U "
<proof>

lemma continuous_compose:
  "[[ continuous f S T; continuous g T U ]] ⇒ continuous (g ∘ f) S U"
<proof>

lemma id_continuous:
  fixes T (structure)
  shows "continuous id T T"
<proof>

lemma (in discrete) continuous:
  fixes f and S (structure) and fimage
  assumes "func f T S" defines "fimage ≡ fimg S f"
  shows "continuous f T S"
<proof>

lemma (in indiscrete) continuous:
  fixes S (structure)

```

```

assumes "topology S"
fixes f and fimage
assumes "func f S T" defines "fimage  $\equiv$  fimg T f"
shows "continuous f S T"
<proof>

```

1.9 Filters

definition

```

fbase :: "'a top  $\Rightarrow$  'a set set  $\Rightarrow$  bool" ("fbase $\imath$ ") where
"fbase T B  $\longleftrightarrow$   $\{\} \notin B \wedge B \neq \{\} \wedge$ 
  ( $\forall a \in B. \forall b \in B. \exists c \in B. c \subseteq a \cap b$ )"

```

definition

```

filters :: "'a top  $\Rightarrow$  'a set set set" ("Filters $\imath$ ") where
"filters T = { F.  $\{\} \notin F \wedge \bigcup F \subseteq \text{carr T} \wedge$ 
  ( $\forall A B. A \in F \wedge B \in F \longrightarrow A \cap B \in F$ )  $\wedge$ 
  ( $\forall A B. A \in F \wedge A \subseteq B \wedge B \subseteq \text{carr T} \longrightarrow B \in F$ ) }"

```

definition

```

ultr :: "'a top  $\Rightarrow$  'a set set  $\Rightarrow$  bool" ("ultra $\imath$ ") where
"ultr T F  $\longleftrightarrow$  ( $\forall A. A \subseteq \text{carr T} \longrightarrow A \in F \vee (\text{carr T} - A) \in F$ )"

```

lemma filtersI [intro]:

```

fixes T (structure)
assumes a1: " $\{\} \notin F$ "
and a2: " $\bigcup F \subseteq \text{carrier}$ "
and a3: " $\bigwedge A B. [ A \in F; B \in F ] \Longrightarrow A \cap B \in F$ "
and a4: " $\bigwedge A B. [ A \in F; A \subseteq B; B \subseteq \text{carrier} ] \Longrightarrow B \in F$ "
shows "F  $\in$  Filters"
<proof>

```

lemma filtersE:

```

assumes a1: "F  $\in$  filters T"
and R: "[  $\{\} \notin F$ ;
   $\bigcup F \subseteq \text{carr T}$ ;
   $\forall A B. A \in F \wedge B \in F \longrightarrow A \cap B \in F$ ;
   $\forall A B. A \in F \wedge A \subseteq B \wedge B \subseteq \text{carr T} \longrightarrow B \in F$ 
]  $\Longrightarrow$  R"
shows "R"
<proof>

```

lemma filtersD1:

```

"F  $\in$  filters T  $\Longrightarrow$   $\{\} \notin F$ "
<proof>

```

lemma filtersD2:

```

"F  $\in$  filters T  $\Longrightarrow$   $\bigcup F \subseteq \text{carr T}$ "
<proof>

```

```

lemma filtersD3:
  "[ F ∈ filters T; A ∈ F; B ∈ F ] ⇒ A ∩ B ∈ F"
  <proof>

lemma filtersD4:
  "[ F ∈ filters T; A ⊆ B; B ⊆ carr T; A ∈ F ] ⇒ B ∈ F"
  <proof>

locale filter = carrier T for F and T (structure) +
  assumes F_filter: "F ∈ Filters"
  notes not_empty [iff]          = filtersD1 [OF F_filter]
  and union_carr [iff]          = filtersD2 [OF F_filter]
  and filter_inter [intro!, simp] = filtersD3 [OF F_filter]
  and filter_greater [dest] = filtersD4 [OF F_filter]

lemma (in filter) elem_carrier [elim]:
  assumes A: "A ∈ F"
  assumes R: "[ A ⊆ carrier; A ≠ {} ] ⇒ R"
  shows "R"
  <proof>

lemma empty_filter [iff]: "{} ∈ filters T"
  <proof>

lemma (in filter) contains_carrier [intro, simp]:
  assumes F_not_empty: "F ≠ {}"
  shows "carrier ∈ F"
  <proof>

lemma nonempty_filter_implies_nonempty_carrier:
  fixes T (structure)
  assumes F_filter: "F ∈ Filters"
  and F_not_empty: "F ≠ {}"
  shows "carrier ≠ {}"
  <proof>

lemma carrier_singleton_filter:
  fixes T (structure)
  shows "carrier ≠ {} ⇒ {carrier} ∈ Filters"
  <proof>

lemma (in topology) nhds_filter:
  "nhds x ∈ Filters"
  <proof>

lemma fimage_filter:
  fixes f and S (structure) and T (structure) and fimage
  assumes "func f S T" defines "fimage ≡ fimg T f"
  fixes F assumes "filter F S"

```

```

  shows "fimage F ∈ Filters2"
⟨proof⟩

lemma Int_filters:
  fixes F and T (structure) assumes "filter F T"
  fixes E assumes "filter E T"
  shows "F ∩ E ∈ Filters"
⟨proof⟩

lemma ultraCI [intro!]:
  fixes T (structure)
  shows "( $\bigwedge A. [A \subseteq \text{carrier}; \text{carrier} - A \notin F] \implies A \in F) \implies \text{ultra } F"$ 
⟨proof⟩

lemma ultraE:
  fixes T (structure)
  shows "[ultra F; A ⊆ carrier;
  A ∈ F ⟹ R;
  carrier - A ∈ F ⟹ R
  ] ⟹ R"
⟨proof⟩

lemma ultraD:
  fixes T (structure)
  shows "[ultra F; A ⊆ carrier; A ∉ F] ⟹ (carrier - A) ∈ F"
⟨proof⟩

locale ultra_filter = filter +
  assumes ultra: "ultra F"
  notes ultraD = ultraD [OF ultra]
  notes ultraE [elim] = ultraE [OF ultra]

lemma (in ultra_filter) max:
  fixes E assumes "filter E T"
  assumes fsube: "F ⊆ E"
  shows "E ⊆ F"
⟨proof⟩

lemma (in filter) max_ultra:
  assumes carrier_not_empty: "carrier ≠ {}"
  and fmax: " $\forall E \in \text{Filters}. F \subseteq E \longrightarrow F = E"$ "
  shows "ultra F"
⟨proof⟩

lemma filter_chain_lemma:
  fixes T (structure) and F
  assumes "filter F T"
  assumes C_chain: "C ∈ chain {V. V ∈ Filters ∧ F ⊆ V}" (is "C ∈ chain ?FF")
  shows " $\bigcup (C \cup \{F\}) \in \text{Filters}$ " (is "?E ∈ Filters")
⟨proof⟩

```

```

lemma expand_filter_ultra:
  fixes T (structure)
  assumes carrier_not_empty: "carrier  $\neq$  {}"
  and F_filter: "F  $\in$  Filters"
  and R: " $\bigwedge U. [ U \in \text{Filters}; F \subseteq U; \text{ultra } U ] \implies R$ "
  shows "R"
<proof>

```

1.10 Convergence

definition

```

converges :: "'a top  $\Rightarrow$  'a set set  $\Rightarrow$  'a  $\Rightarrow$  bool" ("(_  $\dashrightarrow$   $\imath$  _)" [55, 55] 55) where
"converges T F x  $\iff$  nhd T x  $\subseteq$  F"

```

notation (xsymbols)

```

converges ("(_  $\vdash$  _  $\longrightarrow$  _)" [55, 55, 55] 55)

```

definition

```

cnvgnt :: "'a top  $\Rightarrow$  'a set set  $\Rightarrow$  bool" ("_ convergent $\imath$ " [50] 50) where
"cnvgnt T F  $\iff$  ( $\exists x \in \text{carr } T. \text{converges } T F x$ )"

```

definition

```

limites :: "'a top  $\Rightarrow$  'a set set  $\Rightarrow$  'a set" ("lims $\imath$ ") where
"limites T F = {x. x  $\in$  carr T  $\wedge$  T  $\vdash$  F  $\longrightarrow$  x}"

```

definition

```

limes :: "'a top  $\Rightarrow$  'a set set  $\Rightarrow$  'a" ("lim $\imath$ ") where
"limes T F = (THE x. x  $\in$  carr T  $\wedge$  T  $\vdash$  F  $\longrightarrow$  x)"

```

lemma (in carrier) convergesI [intro]:

```

"nhds x  $\subseteq$  F  $\implies$  F  $\dashrightarrow$  x"

```

<proof>

lemma (in carrier) convergesE [elim]:

```

"[ F  $\dashrightarrow$  x; nhds x  $\subseteq$  F  $\implies$  R ]  $\implies$  R"

```

<proof>

lemma (in carrier) convergentI [intro?]:

```

"[ F  $\dashrightarrow$  x; x  $\in$  carrier ]  $\implies$  F convergent"

```

<proof>

lemma (in carrier) convergentE [elim]:

```

"[ F convergent;
 $\bigwedge x. [ F \dashrightarrow x; x \in \text{carrier} ] \implies R$ 
]  $\implies$  R"

```

<proof>

lemma (in continuous) fimage_converges:

```

assumes xpoint: "x  $\in$  carrier"

```

```

and conv: "F  $\dashrightarrow_1$  x"

```

```

shows "fimage F  $\dashrightarrow_2$  (f x)"

```

<proof>

corollary (in continuous) fimage_convergent [intro!]:
 "F convergent₁ \implies fimage F convergent₂"
<proof>

lemma (in topology) closure_convergent_filter:
 assumes xclosure: "x \in closure A"
 and xpoint: "x \in carrier"
 and asub: "A \subseteq carrier"
 and H: " $\bigwedge F. [\![F \in \text{Filters}; F \longrightarrow x; A \in F \!] \implies R$ "
 shows "R"
<proof>

lemma convergent_filter_closure:
 fixes F and T (structure)
 assumes "filter F T"
 assumes converge: "F \longrightarrow x"
 and xpoint: "x \in carrier"
 and AF: "A \in F"
 shows "x \in closure A"
<proof>

1.11 Separation

1.11.1 T0 spaces

locale T0 = topology +
 assumes T0: " $\forall x \in \text{carrier}. \forall y \in \text{carrier}. x \neq y \longrightarrow$
 ($\exists u \in \text{nhds } x. y \notin u$) \vee ($\exists v \in \text{nhds } y. x \notin v$)"

lemma (in T0) T0_eqI:
 assumes points: "x \in carrier" "y \in carrier"
 and R1: " $\bigwedge u. u \in \text{nhds } x \implies y \in u$ "
 and R2: " $\bigwedge v. v \in \text{nhds } y \implies x \in v$ "
 shows "x = y"
<proof>

lemma (in T0) T0_neqE [elim]:
 assumes x_neq_y: "x \neq y"
 and points: "x \in carrier" "y \in carrier"
 and R1: " $\bigwedge u. [\![u \in \text{nhds } x; y \notin u \!] \implies R$ "
 and R2: " $\bigwedge v. [\![v \in \text{nhds } y; x \notin v \!] \implies R$ "
 shows "R"
<proof>

1.11.2 T1 spaces

locale T1 = T0 +
 assumes DT01: " $\forall x \in \text{carrier}. \forall y \in \text{carrier}. x \neq y \longrightarrow$
 ($\exists u \in \text{nhds } x. y \notin u$) = ($\exists v \in \text{nhds } y. x \notin v$)"

```

lemma (in T1) T1_neqE [elim]:
  assumes x_neq_y: "x ≠ y"
  and points: "x ∈ carrier" "y ∈ carrier"
  and R [intro] : "∧u v. [ u ∈ nhds x; v ∈ nhds y; y ∉ u; x ∉ v ] ⇒ R"
  shows "R"
<proof>

```

```

declare (in T1) T0_neqE [rule del]

```

```

lemma (in T1) T1_eqI:
  assumes points: "x ∈ carrier" "y ∈ carrier"
  and R1: "∧u v. [ u ∈ nhds x; v ∈ nhds y; y ∉ u ] ⇒ x ∈ v"
  shows "x = y"
<proof>

```

```

lemma (in T1) singleton_closed [iff]: "{x} closed"
<proof>

```

```

lemma (in T1) finite_closed:
  assumes finite: "finite A"
  shows "A closed"
<proof>

```

1.11.3 T2 spaces (Hausdorff spaces)

```

locale T2 = T1 +
  assumes T2: "∀ x ∈ carrier. ∀ y ∈ carrier. x ≠ y
  → (∃ u ∈ nhds x. ∃ v ∈ nhds y. u ∩ v = {})"

```

```

lemma T2_axiomsI:
  fixes T (structure)
  shows
  "(∧x y. [ x ∈ carrier; y ∈ carrier; x ≠ y ] ⇒
  ∃ u ∈ nhds x. ∃ v ∈ nhds y. u ∩ v = {})
  ⇒ T2_axioms T"
<proof>

```

```

declare (in T2) T1_neqE [rule del]

```

```

lemma (in T2) neqE [elim]:
  assumes neq: "x ≠ y"
  and points: "x ∈ carrier" "y ∈ carrier"
  and R: "∧ u v. [ u ∈ nhds x; v ∈ nhds y; u ∩ v = {} ] ⇒ R"
  shows R
<proof>

```

```

lemma (in T2) neqE2 [elim]:
  assumes neq: "x ≠ y"
  and points: "x ∈ carrier" "y ∈ carrier"
  and R: "∧ u v. [ u ∈ nhds x; v ∈ nhds y; z ∉ u ∨ z ∉ v ] ⇒ R"
  shows R

```

<proof>

lemma T2_axiom_implies_T1_axiom:
 fixes T (structure)
 assumes T2: " $\forall x \in \text{carrier}. \forall y \in \text{carrier}. x \neq y$
 $\longrightarrow (\exists u \in \text{nhds } x. \exists v \in \text{nhds } y. u \cap v = \{\})$ "
 shows "T1_axioms T"

<proof>

lemma T2_axiom_implies_T0_axiom:
 fixes T (structure)
 assumes T2: " $\forall x \in \text{carrier}. \forall y \in \text{carrier}. x \neq y$
 $\longrightarrow (\exists u \in \text{nhds } x. \exists v \in \text{nhds } y. u \cap v = \{\})$ "
 shows "T0_axioms T"

<proof>

lemma T2I:
 fixes T (structure) **assumes** "topology T"
 assumes I: " $\bigwedge x y. [\![x \in \text{carrier}; y \in \text{carrier}; x \neq y]\!] \implies$
 $\exists u \in \text{nhds } x. \exists v \in \text{nhds } y. u \cap v = \{\}$ "
 shows "T2 T"

<proof>

lemmas T2E = T2.neqE
lemmas T2E2 = T2.neqE2

lemma (in T2) unique_convergence:
fixes F **assumes** "filter F T"
assumes points: "x \in carrier" "y \in carrier"
 and Fx: "F \longrightarrow x"
 and Fy: "F \longrightarrow y"
 shows "x = y"

<proof>

lemma (in topology) unique_convergence_implies_T2 [rule_format]:
 assumes unique_convergence:
 " $\bigwedge x y F. [\![x \in \text{carrier}; y \in \text{carrier}; F \in \text{Filters};$
 F \longrightarrow x; F \longrightarrow y]\!] \implies x = y"
 shows "T2 T"

<proof>

lemma (in T2) limI [simp]:
 assumes filter: "F \in Filters"
 and point: "x \in carrier"
 and converges: "F \longrightarrow x"
 shows "lim F = x"

<proof>

lemma (in T2) convergent_limE:
 assumes convergent: "F convergent"
 and filter: "F \in Filters"

and R: " $\llbracket \lim F \in \text{carrier}; F \dashrightarrow \lim F \rrbracket \implies R$ "
 shows "R"
<proof>

lemma image_lim_subset_lim_fimage:
 fixes f and S (structure) and T (structure) and fimage
 defines "fimage \equiv fimg T f"
 assumes "continuous f S T"
 shows " $F \in \text{Filters}_1 \implies f'(\text{lims } F) \subseteq \text{lims}_2 (\text{fimage } F)$ "
<proof>

1.11.4 T3 axiom and regular spaces

locale T3 = topology +
 assumes T3: " $\forall A. \forall x \in \text{carrier} - A. A \subseteq \text{carrier} \wedge A \text{ closed} \longrightarrow$
 $(\exists B. \exists U \in \text{nhds } x. B \text{ open} \wedge A \subseteq B \wedge B \cap U = \{\})$ "

lemma (in T3) T3E:
 assumes H: "A \subseteq carrier" "A closed" "x \in carrier" "x \notin A"
 and R: " $\bigwedge B U. \llbracket A \subseteq B; B \text{ open}; U \in \text{nhds } x; B \cap U = \{\} \rrbracket \implies R$ "
 shows "R"
<proof>

locale regular = T1 + T3

lemma regular_implies_T2:
 fixes T (structure)
 assumes "regular T"
 shows "T2 T"
<proof>

1.11.5 T4 axiom and normal spaces

locale T4 = topology +
 assumes T4: " $\forall A B. A \text{ closed} \wedge A \subseteq \text{carrier} \wedge B \text{ closed} \wedge B \subseteq \text{carrier} \wedge$
 $A \cap B = \{\} \longrightarrow (\exists U V. U \text{ open} \wedge A \subseteq U \wedge V \text{ open} \wedge B \subseteq V \wedge U \cap V = \{\})$ "

lemma (in T4) T4E:
 assumes H: "A closed" "A \subseteq carrier" "B closed" "B \subseteq carrier" "A \cap B = {"
 and R: " $\bigwedge U V. \llbracket U \text{ open}; A \subseteq U; V \text{ open}; B \subseteq V; U \cap V = \{\} \rrbracket \implies R$ "
 shows "R"
<proof>

locale normal = T1 + T4

lemma normal_implies_regular:
 fixes T (structure)
 assumes "normal T"
 shows "regular T"
<proof>

end

2 The topology of llists

```
theory LList_Topology
imports Topology "../Lazy-Lists-II/LList2"
begin
```

2.1 The topology of all llists

This theory introduces the topologies of all llists, of infinite llists, and of the non-empty llists. For all three cases it is proved that safety properties are closed sets and that liveness properties are dense sets. Finally, we prove in each of the the three different topologies the respective theorem of Alpern and Schneider [1], which states that every property can be represented as an intersection of a safety property and a liveness property.

definition

```
ttop :: "'a set  $\Rightarrow$  'a llist top" where
"ttop A = topo ( $\bigcup$  s $\in$ A*. {suff A s})"
```

```
lemma ttop_topology [iff]: "topology (ttop A)"
<proof>
```

```
locale suffixes =
fixes A and B
defines [simp]: "B  $\equiv$  ( $\bigcup$  s $\in$ A*. {suff A s})"
```

```
locale trace_top = suffixes + topobase
```

```
lemma (in trace_top) open_iff [iff]:
"m open = (m  $\in$  topo ( $\bigcup$  s $\in$ A*. {suff A s}))"
<proof>
```

```
lemma (in trace_top) suff_open [intro!]:
"r  $\in$  A*  $\implies$  suff A r open"
<proof>
```

```
lemma (in trace_top) ttop_carrier: "A $^\infty$  = carrier"
<proof>
```

```
lemma (in trace_top) suff_nhd_base:
assumes unhd: "u  $\in$  nhds t"
and H: " $\bigwedge$ r. [ $r \in$  finpref A t; suff A r  $\subseteq$  u ]  $\implies$  R"
shows "R"
<proof>
```

```
lemma (in trace_top) nhds_LNil [simp]: "nhds LNil = {A $^\infty$ }"
<proof>
```

```
lemma (in trace_top) adh_lemma:
assumes xpoint: "x  $\in$  A $^\infty$ "
and PA: "P  $\subseteq$  A $^\infty$ "
shows "(x adh P) = ( $\forall$  r  $\in$  finpref A x.  $\exists$  s  $\in$  A $^\infty$ . r @@ s  $\in$  P)"
```

```

<proof>

lemma (in trace_top) topology [iff]:
  "topology T"
<proof>

lemma (in trace_top) safety_closed_iff:
  "P ⊆ A∞ ⇒ safety A P = (P closed)"
<proof>

lemma (in trace_top) liveness_dense_iff:
  assumes P: "P ⊆ A∞"
  shows "liveness A P = (P dense)"
<proof>

lemma (in trace_top) LNil_safety: "safety A {LNil}"
<proof>

lemma (in trace_top) LNil_closed: "{LNil} closed"
<proof>

theorem (in trace_top) alpern_schneider:
  assumes Psub: "P ⊆ A∞"
  shows "∃ S L. safety A S ∧ liveness A L ∧ P = S ∩ L"
<proof>

```

2.2 The topology of infinite llists

definition

```

itop :: "'a set ⇒ 'a llist top" where
  "itop A = topo (⋃ s∈A*. {infsuff A s})"

```

```

locale infsuffixes =

```

```

  fixes A and B
  defines [simp]: "B ≡ (⋃ s∈A*. {infsuff A s})"

```

```

locale itrace_top = infsuffixes + topobase

```

```

lemma (in itrace_top) open_iff [iff]:
  "m open = (m ∈ topo (⋃ s∈A*. {infsuff A s}))"
<proof>

```

```

lemma (in itrace_top) topology [iff]: "topology T"
<proof>

```

```

lemma (in itrace_top) infsuff_open [intro!]:
  "r ∈ A* ⇒ infsuff A r open"
<proof>

```

```

lemma (in itrace_top) itop_carrier: "carrier = Aω"
<proof>

```

```

lemma itop_sub_ttop_base:
  fixes A :: "'a ⇒ bool"
    and B :: "('a llist ⇒ bool) ⇒ bool"
    and C :: "('a llist ⇒ bool) ⇒ bool"
  defines [simp]: "B ≡ ⋃s∈A*. {suff A s}" and [simp]: "C ≡ ⋃s∈A*. {infsuff A s}"
  shows "C = (⋃ t∈B. {t ∩ C})"
  ⟨proof⟩

lemma itop_sub_ttop [folded ttop_def itop_def]:
  fixes A and C and S (structure)
  defines "C ≡ ⋃s∈A*. {infsuff A s}" and "S ≡ topo C"
  fixes B and T (structure)
  defines "B ≡ ⋃s∈A*. {suff A s}" and "T ≡ topo B"
  shows "subtopology S T"
  ⟨proof⟩

lemma (in itrace_top) infsuff_nhd_base:
  assumes unhd: "u ∈ nhds t"
  and H: "⋀r. [ r ∈ finpref A t; infsuff A r ⊆ u ] ⇒ R"
  shows "R"
  ⟨proof⟩

lemma (in itrace_top) hausdorff [iff]: "T2 T"
  ⟨proof⟩

corollary (in itrace_top) unique_convergence:
  "[ x ∈ carrier;
  y ∈ carrier;
  F ∈ Filters ;
  F → x;
  F → y ] ⇒ x = y"
  ⟨proof⟩

lemma (in itrace_top) adh_lemma:
  assumes xpoint: "x ∈ Aω"
  and PA: "P ⊆ Aω"
  shows "x adh P = (∀ r ∈ finpref A x. ∃ s ∈ Aω. r @@ s ∈ P)"
  ⟨proof⟩

lemma (in itrace_top) infsafety_closed_iff:
  "P ⊆ Aω ⇒ infsafety A P = (P closed)"
  ⟨proof⟩

lemma (in itrace_top) empty:
  "A = {} ⇒ T = {{{}}"
  ⟨proof⟩

lemma itop_empty: "itop {} = {{{}}"
  ⟨proof⟩

lemma infliveness_empty:

```

```

"infliveness {} P  $\implies$  False"
<proof>

lemma (in trivial) dense:
  "P dense"
  <proof>

lemma (in itrace_top) infliveness_dense_iff:
  assumes notempty: "A  $\neq$  {"
  and P: "P  $\subseteq$  A $^\omega$ "
  shows "infliveness A P = (P dense)"
  <proof>

theorem (in itrace_top) alpern_schneider:
  assumes notempty: "A  $\neq$  {"
  and Psub: "P  $\subseteq$  A $^\omega$ "
  shows " $\exists$  S L. infsafety A S  $\wedge$  infliveness A L  $\wedge$  P = S  $\cap$  L"
  <proof>

2.3 The topology of non-empty llists

definition
  ptop :: "'a set  $\Rightarrow$  'a llist top" where
  "ptop A  $\equiv$  topo ( $\bigcup$  s $\in$ A $^\clubsuit$ . {suff A s})"

locale possuffixes =
  fixes A and B
  defines [simp]: "B  $\equiv$  ( $\bigcup$  s $\in$ A $^\clubsuit$ . {suff A s})"

locale ptrace_top = possuffixes + topobase

lemma (in ptrace_top) open_iff [iff]:
  "m open = (m  $\in$  topo ( $\bigcup$  s $\in$ A $^\clubsuit$ . {suff A s}))"
  <proof>

lemma (in ptrace_top) topology [iff]: "topology T"
  <proof>

lemma (in ptrace_top) ptop_carrier: "carrier = A $^\clubsuit$ "
  <proof>

lemma ptop_subtop_ttop:
  fixes S (structure)
  fixes A and B and T (structure)
  defines "B  $\equiv$   $\bigcup$  s $\in$ A $^\clubsuit$ . {suff A s}" and "T  $\equiv$  topo B"
  defines "S  $\equiv$   $\bigcup$  t  $\in$  T. {t - {LNil}}"
  shows "subtopology S T"
  <proof>

lemma ptop_top:
  fixes S (structure)

```

```

fixes A and B and T (structure)
defines "B  $\equiv \bigcup_{s \in A^*}. \{\text{suff } A \ s\}$ " and "T  $\equiv \text{topo } B$ "
defines "S  $\equiv \bigcup t \in T. \{t - \{\text{LNil}\}\}$ "
shows "topology ( $\bigcup t \in T. \{t - \{\text{LNil}\}\}$ )"
<proof>

lemma (in ptrace_top) suff_open [intro!]:
  "r  $\in A^\blacklozenge \implies \text{suff } A \ r \text{ open}$ "
  <proof>

lemma (in ptrace_top) suff_ptop_nhd_base:
  assumes unhd: "u  $\in \text{nhd } t$ "
  and H: " $\bigwedge r. [\ r \in \text{pfinpref } A \ t; \text{suff } A \ r \subseteq u \ ] \implies R$ "
  shows "R"
  <proof>

lemma pfinpref_LNil [simp]: "pfinpref A LNil = {}"
  <proof>

lemma (in ptrace_top) adh_lemma:
  assumes xpoint: "x  $\in A^\blacklozenge$ "
  and P_subset_A: "P  $\subseteq A^\blacklozenge$ "
  shows "x adh P = ( $\forall r \in \text{pfinpref } A \ x. \exists s \in A^\infty. r \ @\@ \ s \in P$ )"
  <proof>

lemma (in ptrace_top) possafety_closed_iff:
  "P  $\subseteq A^\blacklozenge \implies \text{possafety } A \ P = (P \text{ closed})$ "
  <proof>

lemma (in ptrace_top) posliveness_dense_iff:
  assumes P: "P  $\subseteq A^\blacklozenge$ "
  shows "posliveness A P = (P dense)"
  <proof>

theorem (in ptrace_top) alpern_schneider:
assumes Psub: "P  $\subseteq A^\blacklozenge$ "
  shows " $\exists S \ L. \text{possafety } A \ S \wedge \text{posliveness } A \ L \wedge P = S \cap L$ "
  <proof>

end

```

References

- [1] B. Alpern and F. B. Schneider. Defining Liveness. *Information Processing Letters*, 21(4):181–185, Oct. 1985.
- [2] G. McCarty. *Topology : an introduction with application to topological groups*. International series in pure and applied mathematics. Graw-Hill, New York, 1967.

- [3] B. von Querenburg. *Mengentheoretische Topologie*. Springer, Heidelberg, 3. edition, 2001.