summary

Topologies are often defined by taking Uclosure of a basis. In this notebook it is shown that the class of topologies can be obtained by UCLOSURE from the class CAPclosed. The key step is to show that CAPclosed is invariant under UCLOSURE.

forall[z, implies(member[z, CAPclosed], member[Uclosure[z, CAPclosed]])] // assert
subclass[image[UCLOSURE, CAPclosed], CAPclosed]

This will be our first task.

a consequence of monotonicity

The lemma derived here just uses monotonicity.

Map[implies[and[subclass[x, z], subclass[y, z]], #] &,
SubstTest[implies, subclass[u, v], subclass[image[CAP, u], image[CAP, v]],
{u -> cart[x, y], v -> cart[z, z]}]]

or[not[subclass[x, z]], not[subclass[y, z]],
subclass[image[CAP, cart[x, y]], image[CAP, cart[z, z]]]] == True

or[not[subclass[x, z]], not[subclass[y, z]],
subclass[image[CAP, cart[x, y]], image[CAP, cart[z, z]]]] == True

Map[not, SubstTest[and, implies[p1, p2], implies[p3, p4], implies[and[p2, p4], p5],
not[implies[and[p1, p3], p5]], {p1 -> and[subclass[x, z], subclass[y, z]]],
p2 -> subclass[image[CAP, cart[x, y]], image[CAP, cart[z, z]]],
p3 -> equal[z, image[CAP, cart[z, z]]],
p4 -> subclass[image[CAP, cart[x, y]], z],
p5 -> subclass[image[CAP, cart[x, y]], z]]]

or[not[equal[z, image[CAP, cart[z, z]]]], not[subclass[x, z]],
not[subclass[y, z]], subclass[image[CAP, cart[x, y]], z]] == True
The key formula here is a distributive law:

\[
U[\text{image}[\text{CAP}, \text{cart}[x, y]]] \\
\text{intersection}[U[x], U[y]]
\]

\[
\text{SubstTest}[\text{implies}, \text{and}[\text{subclass}[w, z], \text{member}[w, V]], \text{member}[U[w], \text{Uclosure}[z]], w \rightarrow \text{image}[\text{CAP}, \text{cart}[x, y]]]
\]

\[
\text{or}[\text{member}[\text{intersection}[U[x], U[y]], \text{Uclosure}[z]], \text{not}[\text{member}[\text{image}[\text{CAP}, \text{cart}[x, y]], V]], \text{not}[\text{subclass}[\text{image}[\text{CAP}, \text{cart}[x, y]], z]]] := \text{True}
\]

\[
\text{Lemma:}
\]

\[
\text{SubstTest}[\text{implies}, \text{and}[\text{FUNCTION}[z], \text{member}[w, V]], \text{member}[\text{image}[z, w], V], z \rightarrow \text{CAP}]
\]

\[
\text{or}[\text{member}[\text{image}[\text{CAP}, w], V], \text{not}[\text{member}[w, V]]] := \text{True}
\]

\[
\text{Lemma:}
\]

\[
\text{Map}[\text{implies}[\text{and}[\text{member}[x, V], \text{member}[y, V]], \# \&], \text{SubstTest}[\text{implies}, \text{member}[w, V], \text{member}[\text{image}[\text{CAP}, w], V], w \rightarrow \text{cart}[x, y]]]
\]

\[
\text{or}[\text{member}[\text{image}[\text{CAP}, \text{cart}[x, y]], V], \text{not}[\text{member}[x, V]], \text{not}[\text{member}[y, V]]] := \text{True}
\]

\[
\text{Lemma:}
\]

\[
\text{Map}[\text{not}, \text{SubstTest}[\text{and}, \text{implies}[p1, p2], \text{implies}[\text{and}[p3, p4], p5], \text{implies}[\text{and}[p2, p5], p6], \text{implies}[\text{and}[p1, p3, p4], p6]], \text{and}[\text{member}[x, V], \text{member}[y, V]], p2 \rightarrow \text{image}[\text{CAP}, \text{cart}[x, y]], V], p3 \rightarrow \text{equal}[z, \text{image}[\text{CAP}, \text{cart}[z, z]]], p4 \rightarrow \text{and}[\text{subclass}[x, z], \text{subclass}[y, z]], p5 \rightarrow \text{subclass}[\text{image}[\text{CAP}, \text{cart}[x, y]], z], p6 \rightarrow \text{member}[\text{intersection}[U[x], U[y]], \text{Uclosure}[z]]]]
\]

\[
\text{or}[\text{member}[\text{intersection}[U[x], U[y]], \text{Uclosure}[z]], \text{not}[\text{equal}[z, \text{image}[\text{CAP}, \text{cart}[z, z]]]], \text{not}[\text{member}[x, V]], \text{not}[\text{member}[y, V]], \text{not}[\text{subclass}[x, z]], \text{not}[\text{subclass}[y, z]]] := \text{True}
\]

This simplifies when \( z \) is a set:

\[
\text{Map}[\text{implies}[\text{member}[z, V], \# \&], \% // \text{MapNotNot}
\]

\[
\text{or}[\text{member}[\text{intersection}[U[x], U[y]], \text{Uclosure}[z]], \text{not}[\text{equal}[z, \text{image}[\text{CAP}, \text{cart}[z, z]]]], \text{not}[\text{member}[z, V]], \text{not}[\text{subclass}[x, z]], \text{not}[\text{subclass}[y, z]]] := \text{True}
\]
or [member[intersection[U[x_], U[y_], Uclosure[z_]],
not[equal[z_, image[CAP, cart[z_, z_]]]], not[member[z_, V]],
not[subclass[x_, z_]], not[subclass[y_, z_]]] := True

This proves:

implies[and[member[z, CAPclosed], member[x, P[z]], member[y, P[z]]],
member[intersection[U[x], U[y], Uclosure[z]]]

True

---

The variables x and y can now be removed by universal quantifiers:

SubstTest[assert, forall[x, y, implies[member[z, u], member[pair[x, y], v]]],
{u -> CAPclosed, v -> union[complement[cart[P[z], P[z]]], image[cross[inverse[BIGCUP],
inverse[BIGCUP]], image[inverse[CAP], Uclosure[z]]]}] // Reverse

or [not[equal[z, image[CAP, cart[z, z]]]], not[member[z, V]], subclass[cart[P[z], P[z]],
composite[inverse[BIGCUP], image[inverse[CAP], Uclosure[z]], BIGCUP]]] := True

or [not[equal[z_, image[CAP, cart[z_, z_]]]],
not[member[z_, V]], subclass[cart[P[z_], P[z_]],
composite[inverse[BIGCUP], image[inverse[CAP], Uclosure[z_]], BIGCUP]]] := True

Lemma.
SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
{u -> cart[P[z], P[z]],
v -> composite[inverse[BIGCUP], image[inverse[CAP], Uclosure[z]], BIGCUP],
w -> cross[BIGCUP, BIGCUP]}]

or[notsubclass[cart[P[z], P[z]],
composite[inverse[BIGCUP], image[inverse[CAP], Uclosure[z]], BIGCUP]],
subclass[image[CAP, cart[Uclosure[z], Uclosure[z]]], Uclosure[z]]] := True

Replacing one inclusion by an equality:

Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
{p1 -> member[z, CAPclosed], p2 -> subclass[cart[P[z], P[z]],
composite[inverse[BIGCUP], image[inverse[CAP], Uclosure[z]], BIGCUP]],
p3 -> subclass[image[CAP, cart[Uclosure[z], Uclosure[z]]], Uclosure[z]]]]

or[not[equal[z, image[CAP, cart[z, z]]]], not[member[z, V]],
subclass[image[CAP, cart[Uclosure[z], Uclosure[z]]], Uclosure[z]]] := True

or[not[equal[z_, image[CAP, cart[z_, z_]]]], not[member[z_, V]],
subclass[image[CAP, cart[Uclosure[z_, Uclosure[z_]]], Uclosure[z_]]], Uclosure[z_]] := True

Replacing the other inclusion by an equality:

SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
{p -> member[z, CAPclosed],
u -> image[CAP, cart[Uclosure[z], Uclosure[z]]], v -> Uclosure[z]]] // Reverse

or[equal[image[CAP, cart[Uclosure[z], Uclosure[z]]], Uclosure[z]],
not[equal[z, image[CAP, cart[z, z]]]], not[member[z, V]]] := True

or[equal[image[CAP, cart[Uclosure[z], Uclosure[z]]], Uclosure[z]],
not[equal[z_, image[CAP, cart[z_, z_]]]], not[member[z_, V]]] := True

Eliminating the variable z.

Map[equal[V, #] &, SubstTest[class, z,
implies[member[z, w], member[Uclosure[z], w]], w -> CAPclosed]] // Reverse

subclass[image[UCLOSURE, CAPclosed], CAPclosed] := True

That is, the class CAPclosed is invariant under UCLOSURE.

subclass[image[UCLOSURE, CAPclosed], CAPclosed] := True

reformulation as an equation

The goal in this section is to recast the inclusion derived in the preceding section as an equation.

SubstTest[subclass, u, intersection[v, w],
{u -> image[UCLOSURE, CAPclosed], v -> CAPclosed, w -> fix[UCLOSURE]}]

subclass[image[UCLOSURE, CAPclosed], TOPS] := True

subclass[image[UCLOSURE, CAPclosed], TOPS] := True
ImageComp[UCLOSURE, id[fix[UCLOSURE]], TOPS] // Reverse

image[UCLOSURE, TOPS] == TOPS

image[UCLOSURE, TOPS] := TOPS

Reverse inclusion:

SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
{u -> TOPS, v -> CAPclosed, w -> UCLOSURE}]
subclass[TOPS, image[UCLOSURE, CAPclosed]] == True

subclass[TOPS, image[UCLOSURE, CAPclosed]] := True

An equational version:

SubstTest[and, subclass[u, v], subclass[v, u],
{u -> TOPS, v -> image[UCLOSURE, CAPclosed]}]

True == equal[TOPS, image[UCLOSURE, CAPclosed]]

This equation can be made into a rewrite rule.

image[UCLOSURE, CAPclosed] := TOPS