

Impossibility of the Dissection of a Cube

Thomas Holme Surlykke

March 19, 2025

Abstract

This entry formalizes Littlewood’s argument [2], demonstrating that a 3-dimensional cube cannot be dissected into a finite collection of smaller cubes, each of a different size. The formalization addresses theorem #82, “Dissection of Cubes (J.E. Littlewood’s ‘elegant’ proof)” from Freek Wiedijk’s “100 Mathematical Theorems” list [1], and is based upon a prior formalization in Lean [3].

Contents

1	Basic definitions	2
1.1	point and cube definitions	2
1.2	Calculations with sets from cubes	3
1.2.1	Point membership	3
1.2.2	Cubes subset of each other, by <i>side</i>	4
2	Cubing	5
2.1	Properties of <i>is-dissection</i>	6
3	Hole	6
3.1	Definitions	6
3.2	Properties of a hole	7
3.3	Properties of cubes on a hole	7
4	Bottom of <i>unit-cube</i> is a hole	8
5	Minimum cube on hole is interior	9
5.1	Definition: Minimum cube on <i>h</i>	9
5.2	Minimum cube on hole is interior	10
6	Minimum cube of hole induces hole on top	10

7 The main result

11

theory *Cube-Dissection*

imports *Complex-Main HOL-Library.Disjoint-Sets HOL-Library.Infinite-Set*
begin

Proof that a cube can't be dissected into a finite number of subcubes of different size This formalization is heavily inspired by the Lean proof of the same fact, [3]. One goal of this project, is that by restricting to cubes of dimension 3 the logic will be easier to follow

1 Basic definitions

1.1 point and cube definitions

record *point* = *px*:: *real* *py*::*real* *pz*::*real*
record *cube* = *point*:: *point* *width*::*real*
datatype *axis* = *x* | *y* | *z*
abbreviation *coordinate* \equiv *case-axis px py pz*

abbreviation *is-valid* :: *cube* \Rightarrow *bool* **where** *is-valid c* \equiv (*width c* > 0)

Min value of cube along given axis

hide-const (**open**) *min*

abbreviation *min* :: *axis* \Rightarrow *cube* \Rightarrow *real* **where** *min ax c* \equiv *coordinate ax (point c)*

Max value (supremum) along given axis

hide-const (**open**) *max*

abbreviation *max* :: *axis* \Rightarrow *cube* \Rightarrow *real* **where** *max ax c* \equiv *min ax c* + *width c*

Sides of a cube. Half-open intervals, so that a dissection both is a cover, and consists of disjoint cubes

abbreviation *side* :: *axis* \Rightarrow *cube* \Rightarrow *real set* **where**
side ax c \equiv {*min ax c* ..< *max ax c*}

Sets of points generated from cubes

definition *to-set* :: *cube* \Rightarrow *point set* **where**

to-set c = {*p. px p* \in *side x c* \wedge *py p* \in *side y c* \wedge *pz p* \in *side z c*}

definition *bot* :: *cube* \Rightarrow *point set* **where**

bot c = {*p. px p* \in *side x c* \wedge *py p* \in *side y c* \wedge *pz p* = *min z c*}

definition *top* :: *cube* \Rightarrow *point set* **where**

top c = {*p. px p* \in *side x c* \wedge *py p* \in *side y c* \wedge *pz p* = *max z c*}

Moves a cube its width down (so top face to bottom face)

definition *shift-down* :: *cube* \Rightarrow *cube* **where**

shift-down c = *c* (*point* := *point c* (*pz* := *min z c* - *width c*))

1.2 Calculations with sets from cubes

A bunch of statements we need about how cubes can be compared by *side*

lemma *top-shift-down-eq-bot*: $top (shift-down\ c) = bot\ c$
<proof>

Sets not empty

lemma *non-empty*: $is-valid\ c \implies to-set\ c \neq \{\}$
<proof>

lemma *top-non-empty*: $is-valid\ c \implies top\ c \neq \{\}$
<proof>

min of a cube is in corresponding *side*

lemma *min-in-side*: $is-valid\ c \implies min\ ax\ c \in side\ ax\ c$
<proof>

lemma *min-ne-max*: $is-valid\ c \implies min\ ax\ c \neq max\ ax\ c$
<proof>

lemma *min-lt-max*: $is-valid\ c \implies min\ ax\ c < max\ ax\ c$
<proof>

lemma *bot-subset*: $bot\ c \subseteq to-set\ c$
<proof>

1.2.1 Point membership

Points in a cube's set, by looking at membership of *side*

lemma *in-set-by-side*: $p \in to-set\ c \iff$
 $px\ p \in side\ x\ c \wedge py\ p \in side\ y\ c \wedge pz\ p \in side\ z\ c$
<proof>

lemma *in-set-by-side-2*: $(\lfloor px=x0, py=y0, pz=z0 \rfloor) \in to-set\ c \iff$
 $x0 \in side\ x\ c \wedge y0 \in side\ y\ c \wedge z0 \in side\ z\ c$
<proof>

lemma *in-bot-by-side*: $p \in bot\ c \iff$
 $px\ p \in side\ x\ c \wedge py\ p \in side\ y\ c \wedge pz\ p = min\ z\ c$
<proof>

lemma *in-bot-by-side-2*: $(\lfloor px=x0, py=y0, pz=z0 \rfloor) \in bot\ c \iff$
 $x0 \in side\ x\ c \wedge y0 \in side\ y\ c \wedge z0 = min\ z\ c$
<proof>

lemma *in-top-by-side*: $p \in top\ c \iff$
 $px\ p \in side\ x\ c \wedge py\ p \in side\ y\ c \wedge pz\ p = max\ z\ c$
<proof>

lemma *in-top-by-side-2*: $(\lfloor px=x0, py=y0, pz=z0 \rfloor) \in top\ c \iff$
 $x0 \in side\ x\ c \wedge y0 \in side\ y\ c \wedge z0 = max\ z\ c$
<proof>

lemma *all-point-iff*: $(\forall p. P\ p) \iff (\forall x1\ y1\ z1. P\ (\lfloor px = x1, py = y1, pz = z1 \rfloor))$

<proof>

Intersection by *side*

lemma *set-intersect-by-side*: $to\text{-set } c1 \cap to\text{-set } c2 \neq \{\}$ \longleftrightarrow
 $side\ x\ c1 \cap side\ x\ c2 \neq \{\} \wedge side\ y\ c1 \cap side\ y\ c2 \neq \{\} \wedge side\ z\ c1 \cap side\ z\ c2$
 $\neq \{\}$
<proof>

lemma *bot-intersect-by-side*: $bot\ c1 \cap bot\ c2 \neq \{\}$
 $\longleftrightarrow side\ x\ c1 \cap side\ x\ c2 \neq \{\} \wedge side\ y\ c1 \cap side\ y\ c2 \neq \{\} \wedge min\ z\ c1 = min$
 $z\ c2$
<proof>

lemma *bot-top-intersect-by-side*: $bot\ c1 \cap top\ c2 \neq \{\}$
 $\longleftrightarrow side\ x\ c1 \cap side\ x\ c2 \neq \{\} \wedge side\ y\ c1 \cap side\ y\ c2 \neq \{\} \wedge min\ z\ c1 = max$
 $z\ c2$
<proof>

1.2.2 Cubes subset of each other, by *side*

lemma *set-subset-by-side*: $to\text{-set } c1 \subseteq to\text{-set } c2 \longleftrightarrow$
 $side\ x\ c1 \subseteq side\ x\ c2 \wedge side\ y\ c1 \subseteq side\ y\ c2 \wedge side\ z\ c1 \subseteq side\ z\ c2$
<proof>

lemma *set-eq-by-side*: $to\text{-set } c1 = to\text{-set } c2 \longleftrightarrow$
 $side\ x\ c1 = side\ x\ c2 \wedge side\ y\ c1 = side\ y\ c2 \wedge side\ z\ c1 = side\ z\ c2$
<proof>

lemma *bot-eq-by-side*: $is\text{-valid } c1 \implies bot\ c1 = bot\ c2 \longleftrightarrow$
 $side\ x\ c1 = side\ x\ c2 \wedge side\ y\ c1 = side\ y\ c2 \wedge min\ z\ c1 = min\ z\ c2$
<proof>

lemma *bot-top-subset-by-side*: $is\text{-valid } c1 \implies bot\ c1 \subseteq top\ c2 \longleftrightarrow$
 $side\ x\ c1 \subseteq side\ x\ c2 \wedge side\ y\ c1 \subseteq side\ y\ c2 \wedge min\ z\ c1 = max\ z\ c2$
<proof>

lemma *bot-top-eq-by-side*: $is\text{-valid } c1 \implies bot\ c1 = top\ c2 \longleftrightarrow$
 $side\ x\ c1 = side\ x\ c2 \wedge side\ y\ c1 = side\ y\ c2 \wedge min\ z\ c1 = max\ z\ c2$
<proof>

lemma *width-eq-if-side-eq*: $\llbracket is\text{-valid } c1; side\ ax\ c1 = side\ ax\ c2 \rrbracket \implies width\ c1 =$
 $width\ c2$
<proof>

to-set is injective

lemma *to-set-inj*:
assumes $is\text{-valid } c1$ $to\text{-set } c1 = to\text{-set } c2$
shows $c1 = c2$
<proof>

Cube-Dissection.bot is also injective

lemma *bot-inj*: **assumes** *is-valid c1 bot c1 = bot c2* **shows** *c1 = c2*
 ⟨*proof*⟩

2 Cubing

We in this section introduce a dissection C of the unit cube

The cube we show there is no dissection of

definition *unit-cube* **where** *unit-cube* = (\lfloor *point*=(\lfloor *px=0, py=0, pz=0 \rfloor), *width=1 \rfloor)**

lemma *min-unit-cube-0*: *min ax unit-cube = 0*
 ⟨*proof*⟩

lemma *unit-cube-valid[simp]*: *is-valid unit-cube*
 ⟨*proof*⟩

What we want to show doesn't exist. C is a set of cubes which satisfy:

1. All cubes are valid ($\text{width} > 0$)
2. All cubes are disjoint
3. The union of the cubes in C equal *unit-cube* (hence, all cubes are contained in *unit-cube*)
4. All cubes in C have different width
5. There are at least two cubes in C
6. There are a finite number of cubes in C

definition *is-dissection* :: *cube set* \Rightarrow *bool* **where**
is-dissection $C \iff$
 ($\forall c \in C. \text{is-valid } c$)
 \wedge *disjoint* (*image to-set* C)
 \wedge \bigcup (*image to-set* C) = *to-set unit-cube*
 \wedge *inj-on width* C — All cubes are of different size
 \wedge *card* $C \geq 2$ — At least two cubes
 \wedge *finite* C

From now on, C is some fixed dissection of *unit-cube*, and 'dissection' refers to this fact

context *fixes* C **assumes** *dissection*: *is-dissection* C
begin

2.1 Properties of *is-dissection*

lemma *valid-if-dissection[simp]*: $c \in C \implies \text{is-valid } c$
 ⟨*proof*⟩

lemma *side-unit-cube*:
 $\text{side } ax \text{ unit-cube} = \{0..<1\}$
 ⟨*proof*⟩

lemma *subset-unit-cube-if-dissection*: $c \in C \implies \text{to-set } c \subseteq \text{to-set unit-cube}$
 ⟨*proof*⟩

lemma *subset-unit-cube-by-side*:
 $c \in C \implies \text{side } ax \ c \subseteq \{0..<1\}$
 ⟨*proof*⟩

lemma *eq-iff-intersect*: $\llbracket c1 \in C; c2 \in C \rrbracket \implies c1 = c2 \iff \text{to-set } c1 \cap \text{to-set } c2 \neq \{\}$
 ⟨*proof*⟩

Whenever we have a point in *unit-cube*, there exists a (unique) cube in C containing that point

lemma *obtain-cube*: $p \in \text{to-set unit-cube} \implies \exists c \in C. p \in \text{to-set } c$
 ⟨*proof*⟩

If the top of c doesn't touch the top of *unit-cube*, then top of c must be covered by bottoms of cubes in C

lemma *top-cover-by-bot*:
assumes $c \in C \text{ max } z \ c < 1$
shows $\text{top } c \subseteq \bigcup (\text{image bot } C)$
 ⟨*proof*⟩

3 Hole

A hole h is a special kind of cube, where any cube whose bottom 'touches' the top of v must in fact have its bottom contained in the top of v . If $h \in C$, then this happens because all the other cubes surrounding h go up taller, forming a hole on top of v . Note that we don't require that $h \in C$, but this is only so we can prove that *unit-cube* shifted down by 1 is a hole - all other holes will in fact lie in C . The concept of a hole is inspired by the 'Valley' definition from [3]

3.1 Definitions

definition *is-hole* :: $\text{cube} \Rightarrow \text{bool}$ **where**
 $\text{is-hole } h \iff$
 $\text{is-valid } h$
 $\wedge \text{top } h \subseteq \bigcup (\text{image bot } C)$

$\wedge (\forall c \in C . \text{bot } c \cap \text{top } h \neq \{\} \longrightarrow \text{bot } c \subseteq \text{top } h)$

— v could be a cube in C (and most often is), but any other cube must be different width. Also, this assumption is not actually needed (as it follows from $v, c \in C$), but without it we have to do a special-case proof for the bottom of the *unit-cube*

$\wedge (\forall c \in C . c \neq h \longrightarrow \text{width } c \neq \text{width } h)$

Subset of C which are on a given hole h

definition *is-on-hole* :: *cube* \Rightarrow *cube* \Rightarrow *bool* **where**

is-on-hole h $c \iff \text{bot } c \subseteq \text{top } h$

definition *filter-on-hole* :: *cube* \Rightarrow *cube set* **where**

filter-on-hole $h = \text{Set.filter } (\text{is-on-hole } h) C$

3.2 Properties of a hole

Terminology: 'on hole' means cube c with: *Cube-Dissection.bot* $c \subseteq \text{Cube-Dissection.top } h$. 'in hole' means point p with: $p \in \text{Cube-Dissection.top } h$

local.filter-on-hole $h \subseteq C$

lemma *dissection-if-on-hole[simp]*: $c \in \text{filter-on-hole } h \implies c \in C$

<proof>

Holes, and cubes on them, are valid

lemma *valid-if-hole[simp]*: *is-hole* $h \implies \text{is-valid } h$

<proof>

lemma *valid-if-on-hole[simp]*: $c \in \text{filter-on-hole } h \implies \text{is-valid } c$

<proof>

lemma *on-hole-finite*: *is-hole* $h \implies \text{finite } (\text{filter-on-hole } h)$

<proof>

lemma *on-hole-if-in-filter-on-hole*: $c \in \text{filter-on-hole } h \implies \text{is-on-hole } h c$

<proof>

lemma *on-hole-cover*: **assumes** *is-hole* h **shows** $\text{top } h \subseteq \bigcup (\text{image bot } (\text{filter-on-hole } h))$

<proof>

Whenever we have a point p in the top of a hole h , there exists a (unique) cube $c \in \text{local.filter-on-hole } h$, such that $p \in \text{Cube-Dissection.bot } c$

lemma *obtain-cube-if-in-hole*: $\llbracket \text{is-hole } h; p \in \text{top } h \rrbracket$

$\implies \exists c \in \text{filter-on-hole } h . p \in \text{bot } c$

<proof>

lemma *on-hole-inj-on-width*: *is-hole* $h \implies \text{inj-on width } (\text{filter-on-hole } h)$

<proof>

3.3 Properties of cubes on a hole

lemma *neq-hole-if-on-hole*: $c \in \text{filter-on-hole } h \implies c \neq h$

<proof>

lemma *subset-if-on-hole*: $c \in \text{filter-on-hole } h \implies \text{bot } c \subseteq \text{top } h$
<proof>

lemma *side-subset-if-on-hole*: $\llbracket c \in \text{filter-on-hole } h; ax \in \{x,y\} \rrbracket \implies \text{side } ax \ c \subseteq \text{side } ax \ h$
<proof>

lemma *min-z-eq-max-z-hole-if-on-hole*:
 $c \in \text{filter-on-hole } h \implies \text{min } z \ c = \text{max } z \ h$
<proof>

lemma *z-eq-if-on-hole*:
 $\llbracket c1 \in \text{filter-on-hole } h; c2 \in \text{filter-on-hole } h \rrbracket \implies \text{min } z \ c1 = \text{min } z \ c2$
<proof>

Do not need to care about z-coordinate

lemma *eq-iff-side-eq-if-on-hole*: $\llbracket c1 \in \text{filter-on-hole } h; c2 \in \text{filter-on-hole } h \rrbracket$
 $\implies c1 = c2 \iff \text{side } x \ c1 = \text{side } x \ c2 \wedge \text{side } y \ c1 = \text{side } y \ c2$
<proof>

Disjointness-lemmas:

lemma *eq-iff-bot-intersect-if-on-hole*:
assumes $c1 \in \text{filter-on-hole } h \ c2 \in \text{filter-on-hole } h$
shows $c1 = c2 \iff \text{bot } c1 \cap \text{bot } c2 \neq \{\}$
<proof>

lemma *eq-iff-side-intersect-if-on-hole*:
 $\llbracket c1 \in \text{filter-on-hole } h; c2 \in \text{filter-on-hole } h \rrbracket$
 $\implies c1 = c2 \iff \text{side } x \ c1 \cap \text{side } x \ c2 \neq \{\} \wedge \text{side } y \ c1 \cap \text{side } y \ c2 \neq \{\}$
<proof>

lemma *width-on-hole-lt-width-hole*:
assumes $\text{is-hole } h \ c \in \text{filter-on-hole } h$ **shows** $\text{width } c < \text{width } h$
<proof>

lemma *strict-subset-if-on-hole*: **assumes** $\text{is-hole } h \ c \in \text{filter-on-hole } h$
shows $\text{bot } c \subset \text{top } h$
<proof>

lemma *on-hole-non-empty*: $\text{is-hole } h \implies \text{filter-on-hole } h \neq \{\}$
<proof>

4 Bottom of *unit-cube* is a hole

lemma *bot-unit-cube-cover-by-bot*: $\text{bot } \text{unit-cube} \subseteq \bigcup (\text{image } \text{bot } C)$
<proof>

lemma *eq-if-width-eq-if-subset*:
assumes $\text{width } c1 = \text{width } c2$ $\text{to-set } c1 \subseteq \text{to-set } c2$
shows $\text{to-set } c1 = \text{to-set } c2$
 $\langle \text{proof} \rangle$

lemma *width-ne-one*:
assumes $c \in C$
shows $\text{width } c \neq 1$
 $\langle \text{proof} \rangle$

Combines the previous lemmas, to show that the bottom of *unit-cube* is a hole

proposition *hole-unit-cube: is-hole (shift-down unit-cube)*
 $\langle \text{proof} \rangle$

5 Minimum cube on hole is interior

context
fixes h **assumes** *hole: is-hole h*
begin

For this section, we fix a hole h , and define *cmin* to be the smallest cube on this hole. Theorem *hole* refers to this fact. The goal of this section is then to show that *cmin* itself is a hole.

5.1 Definition: Minimum cube on h

cmin is the smallest cube on the hole h

definition *cmin:: cube*
where $cmin = (\text{ARG-MIN width } c . c \in \text{filter-on-hole } h)$

lemma *arg-min-exist*: $\llbracket \text{finite } C'; C' \neq \{\} \rrbracket \implies (\text{ARG-MIN width } c . c \in C') \in C'$
 $\langle \text{proof} \rangle$

This lemma also shows that *local.cmin* exists

lemma *cmin-on-h*: $cmin \in \text{filter-on-hole } h$
 $\langle \text{proof} \rangle$

lemma *cmin-valid[simp]*: *is-valid cmin*
 $\langle \text{proof} \rangle$

lemma *arg-min-minimal*: $\llbracket \text{finite } C'; c \in C' \rrbracket \implies \text{width } (\text{ARG-MIN width } c . c \in C') \leq \text{width } c$
 $\langle \text{proof} \rangle$

lemma *cmin-minimal*: $c \in \text{filter-on-hole } h \implies \text{width } cmin \leq \text{width } c$
 $\langle \text{proof} \rangle$

lemma *cmin-minimal-strict*:
assumes $c \in \text{filter-on-hole } h \ c \neq \text{cmin}$
shows $\text{width } \text{cmin} < \text{width } c$
 $\langle \text{proof} \rangle$

lemma *cmin-max-z-neq-one*: $\text{max } z \ \text{cmin} < 1$
 $\langle \text{proof} \rangle$

5.2 Minimum cube on hole is interior

All squares on the boundary of h

definition *is-on-boundary* :: $\text{axis} \Rightarrow \text{cube} \Rightarrow \text{bool}$ **where**
 $\text{is-on-boundary } ax \ c \longleftrightarrow \text{min } ax \ h = \text{min } ax \ c \vee \text{max } ax \ h = \text{max } ax \ c$

Shows that IF local.cmin is on a boundary ax , then we find some ax -coordinate r , which is further from the boundary than the edge of local.cmin , but closer than the edge of any other cube sufficiently close to the boundary.

lemma *cmin-on-boundary*:
assumes $\text{is-on-boundary } ax \ \text{cmin} \ ax \in \{x, y\}$
shows $\exists r .$
 $r \in (\text{side } ax \ h - (\text{side } ax \ \text{cmin})) \wedge$
 $(\forall c \in \text{filter-on-hole } h . c \neq \text{cmin} \longrightarrow \text{side } ax \ \text{cmin} \cap \text{side } ax \ c \neq \{\}) \longrightarrow r \in$
 $\text{side } ax \ c)$
 $\langle \text{proof} \rangle$

Using the previous lemma, we show that local.cmin being on the boundary leads to a contradiction

lemma *cmin-not-on-boundary-by-axis*:
assumes $ax \in \{x, y\}$
shows $\neg \text{is-on-boundary } ax \ \text{cmin}$
 $\langle \text{proof} \rangle$

Previous result, written as inequalities instead

proposition *cmin-not-on-boundary*:
 $\text{min } x \ h < \text{min } x \ \text{cmin} \wedge \text{max } x \ \text{cmin} < \text{max } x \ h$
 $\wedge \text{min } y \ h < \text{min } y \ \text{cmin} \wedge \text{max } y \ \text{cmin} < \text{max } y \ h$
 $\langle \text{proof} \rangle$

6 Minimum cube of hole induces hole on top

The main result of this proof - the minimum cube on a hole is itself a hole!

proposition *hole-cmin*:
shows $\text{is-hole } \text{cmin}$
 $\langle \text{proof} \rangle$

The main purpose of the previous result: From the proposition, hole-cmin when given the hole h induce another hole h' (i.e., local.cmin), which is in C and is strictly smaller.

lemma *recursive-step*: $\exists h'. h' \in C \wedge \text{is-hole } h' \wedge \text{width } h' < \text{width } h$
 ⟨proof⟩

Here we end the context in which h is some fixed hole (and hence also the specific *local.cmin*)

end

7 The main result

We combine the previous lemmas inductively as follows: 0: Start with the bottom of *unit-cube*, which we showed is a hole. n: For each hole, take the minimum cube on this hole, which is then a new hole, strictly smaller, and in C . Hence, C is infinite.

definition *next-hole*:: *cube* \Rightarrow *cube* **where**
next-hole $h = (\text{SOME } h'. h' \in C \wedge \text{is-hole } h' \wedge \text{width } h' < \text{width } h)$

lemma *next-hole-exist*: *is-hole* h
 $\Rightarrow \text{next-hole } h \in C \wedge \text{is-hole } (\text{next-hole } h) \wedge \text{width } (\text{next-hole } h) < \text{width } h$
 ⟨proof⟩

For following proof, we want the image of *nth-hole* to be contained in C , hence we start at $1 (= \text{Suc } 0)$. *nth-hole* is a function from \mathbb{N} to C

definition *nth-hole* :: *nat* \Rightarrow *cube* **where**
nth-hole $n = (\text{next-hole } \widehat{\sim} \text{Suc } n)$ (*shift-down unit-cube*)

Each cube in the image of *local.nth-hole* is a hole

lemma *nth-hole-is-hole*: *is-hole* (*nth-hole* n)
 ⟨proof⟩

uminus is *uminus*, and *strict-mono* means strictly increasing (not strictly monotonous, as the name might suggest)

lemma *nth-hole-strict-decreasing*: *strict-mono* (*uminus* \circ *width* \circ *nth-hole*)
 ⟨proof⟩

local.nth-hole is injective

lemma *nth-hole-inj* : *inj* *nth-hole*
 ⟨proof⟩

The image (range) of *local.nth-hole* is contained in C

lemma *nth-hole-in*: *nth-hole* $n \in C$
 ⟨proof⟩

Same as previous lemma, but written with a quantifier

lemma *nth-hole-in-forall*: $\forall n . \text{nth-hole } n \in C$
 ⟨proof⟩

The assumption made in this context (*is-dissection* C) leads to *False* (since *local.nth-hole* generates an infinite subset of C)

theorem *false-if-dissection*: *False*

<proof>

end — Here we end the *is-dissection C* context

Main result (spelling out the definition of *is-dissection*).

theorem *dissection-does-not-exist*:

$\# C. (\forall c \in C. \textit{is-valid } c)$

$\wedge \textit{disjoint (image to-set } C)$

$\wedge \bigcup (\textit{image to-set } C) = \textit{to-set unit-cube}$

$\wedge \textit{inj-on width } C$

$\wedge \textit{card } C \geq 2$

$\wedge \textit{finite } C$

<proof>

end

References

- [1] Formalizing 100 Theorems. <https://www.cs.ru.nl/~freek/100/>, accessed 2024-11-22.
- [2] J. E. Littlewood and B. Bollobas. *Littlewood's Miscellany*. Cambridge University Press, Cambridge [Cambridgeshire] ; New York, rev. ed edition, 1986.
- [3] F. van Doorn. Archive.Wiedijk100Theorems.CubingACube. https://leanprover-community.github.io/mathlib4_docs/Archive/Wiedijk100Theorems/CubingACube.html#Theorems100.%C2%AB82%C2%BB.cannot_cube_a_cube, accessed 2024-11-22.