Pick's Theorem

Sage Binder and Katherine Kosaian

November 9, 2026

Abstract

We formalize Pick's theorem for finding the area of a simple polygon whose vertices are integral lattice points [1]. We are inspired by John Harrison's formalization of Pick's theorem in HOL Light [2], but tailor our proof approach to avoid a primary challenge point in his formalization, which is proving that any polygon with more than three vertices can be split (in its interior) by a line between some two vertices. Our formalization involves augmenting the existing geometry libraries in various foundational ways (e.g., by adding the definition of a polygon and formalizing some key properties thereof).

Contents

1 Misc. Linear Algebra Setup **3**

2 Integral Bijective Matrix Determinant **4**

3 Polygon Definitions **5**

4 Jordan Curve Theorem for Polygons **6**

5 Properties of make polygonal path, pathstart and pathfinish of a polygon **8**

6 Loop Free Properties **10**

7 Explicit Linepath Characterization of Polygonal Paths **11**

8 A Triangle is a Polygon **13**

9 Polygon Vertex Rotation **14**

10 Translating a Polygon **18**

11 Misc. properties **19**

12 Properties of Sublists of Polygonal Path Vertex Lists **20**

13 Reversing Polygonal Path Vertex List **24**

14 Collinearity Properties **25**

15 Linepath Properties **26**

16 Measure of linepaths **27**

17 Misc. Convex Polygon Properties **28**

18 Vertices on Convex Frontier Implies Polygon is Convex **30**

19 Polygon Splitting **32**

20 Triangles **37**

21 Measure Setup **40**

22 Unit Triangle **40**

23 Unit Square **41**

24 Unit Triangle Area is 1/2 **42**

25 Area of Elementary Triangle is 1/2 **42**

26 Setup **43**
26.1 Integral Points Cardinality Properties 43

27 Pick splitting **44**

28 Convex Hull Has Good Linepath **46**

29 Pick's Theorem **47**
29.1 Pick's Theorem Triangle Case 47
29.2 Pocket properties 50
29.3 Arbitrary Polygon Case 61

theory *Integral-Matrix*
imports
Complex-Main
HOL−Analysis.Finite-Cartesian-Product
HOL−Analysis.Linear-Algebra
HOL−Analysis.Determinants
begin

1 Misc. Linear Algebra Setup

lemma *vec-scaleR-2*: $(c::real) *_R ((vector\ [a,\ b])::real\hat{}2) = vector\ [a * c,\ b * c]$
⟨*proof*⟩

definition *is-int* :: *real* ⇒ *bool* **where**
is-int $x \longleftrightarrow (\exists\, n::int.\ x = n)$

lemma *is-int-sum*: *is-int* $x \wedge$ *is-int* $y \longrightarrow$ *is-int* $(x + y)$
⟨*proof*⟩

lemma *is-int-minus*: *is-int* $x \wedge$ *is-int* $y \longrightarrow$ *is-int* $(x - y)$
⟨*proof*⟩

lemma *is-int-mult*: *is-int* $x \wedge$ *is-int* $y \longrightarrow$ *is-int* $(x * y)$
⟨*proof*⟩

definition *integral-vec* :: *real*$\hat{}2$ ⇒ *bool* **where**
integral-vec $v \longleftrightarrow$ (*is-int* $(v\$1) \wedge$ *is-int* $(v\$2)$)

lemma *integral-vec-sum*: *integral-vec* $v \wedge$ *integral-vec* $w \longrightarrow$ *integral-vec* $(v + w)$
⟨*proof*⟩

lemma *integral-vec-minus*: *integral-vec* $v \longrightarrow$ *integral-vec* $(-v)$
⟨*proof*⟩

lemma *real-2-inner*:
shows $((vector\ [a,\ b])::(real\hat{}2)) \cdot ((vector\ [c,\ d])::(real\hat{}2)) = a*c + b*d$
(**is** *?v* · *?w* = $a*c + b*d$)
⟨*proof*⟩

lemma *integral-vec-2*:
fixes $a\ b$ :: *int*
assumes $v = vector\ [a,\ b]$
shows *integral-vec v*
⟨*proof*⟩

definition *matrix-inv* :: *real*$\hat{}2\hat{}2$ ⇒ *real*$\hat{}2\hat{}2$ ⇒ *bool* **where**
matrix-inv A A′ $\longleftrightarrow$ (A ** $A' = mat\ 1 \wedge A'$ ** $A = mat\ 1$)

lemma *mat-vec-mult-2*:
fixes v :: *real*$\hat{}2$ **and**
T :: *real*$\hat{}2\hat{}2$
defines x: $x \equiv v\$1$ **and** y: $y \equiv v\$2$ **and**
a: $a \equiv T\$1\$1$ **and** b: $b \equiv T\$1\$2$ **and**
c: $c \equiv T\$2\$1$ **and** d: $d \equiv T\$2\$2$
shows $(T *v v) = vector\ [x*a + y*b,\ x*c + y*d]$
⟨*proof*⟩

definition *integral-mat* :: *real^2^2* ⇒ *bool* **where**
integral-mat T ⟷ (∀ *v*. *integral-vec v* ⟶ *integral-vec* (*T* ∗*v v*))

definition *integral-mat-surj* :: *real^2^2* ⇒ *bool* **where**
integral-mat-surj T ⟷ (∀ *v*. *integral-vec v* ⟶ (∃ *w*. *integral-vec w* ∧ *T* ∗*v w* = *v*))

definition *integral-mat-bij* :: *real^2^2* ⇒ *bool* **where**
integral-mat-bij T ⟷ *integral-mat T* ∧ *integral-mat-surj T*

lemma *integral-mat-integral-vec*: *integral-mat A* ⟶ *integral-vec v* ⟶ *integral-vec* (*A* ∗*v v*)
⟨*proof*⟩

lemma *integral-mat-int-entries*:
fixes *T* :: *real^2^2*
assumes *integral-mat T*
defines *a*: *a* ≡ *T$1$1* **and** *b*: *b* ≡ *T$1$2* **and**
c: *c* ≡ *T$2$1* **and** *d*: *d* ≡ *T$2$2*
shows *is-int a* ∧ *is-int b* ∧ *is-int c* ∧ *is-int d*
⟨*proof*⟩

2 Integral Bijective Matrix Determinant

lemma *integral-mat-int-det*:
fixes *T* :: *real^2^2*
assumes *integral-mat T*
shows *is-int* (*det T*)
⟨*proof*⟩

lemma *integral-mat-bij-inv*:
fixes *T* :: *real^2^2*
assumes *integral-mat-bij T*
obtains *Tinv* **where** *invertible T* ∧ *integral-mat-bij Tinv* ∧ *matrix-inv T Tinv*
⟨*proof*⟩

lemma *integral-mat-bij-det-pm1*:
fixes *T* :: *real^2^2*
assumes *integral-mat-bij T*
shows *det T* = *1* ∨ *det T* = −*1*
⟨*proof*⟩

end
theory *Polygon-Jordan-Curve*
imports
HOL−Analysis.Cartesian-Space
HOL−Analysis.Path-Connected

Poincare-Bendixson.Poincare-Bendixson
Integral-Matrix

begin

3 Polygon Definitions

type-synonym *R-to-R2* = (*real* ⇒ *real^2*)

definition *closed-path* :: *R-to-R2* ⇒ *bool* **where**
closed-path g ⟷ *path g* ∧ *pathstart g* = *pathfinish g*

definition *path-inside* :: *R-to-R2* ⇒ (*real^2*) *set* **where**
path-inside g = *inside* (*path-image g*)

definition *path-outside* :: *R-to-R2* ⇒ (*real^2*) *set* **where**
path-outside g = *outside* (*path-image g*)

fun *make-polygonal-path* :: (*real^2*) *list* ⇒ *R-to-R2* **where**
make-polygonal-path [] = *linepath 0 0*
| *make-polygonal-path* [*a*] = *linepath a a*
| *make-polygonal-path* [*a*,*b*] = *linepath a b*
| *make-polygonal-path* (*a* # *b* # *xs*) = (*linepath a b*) +++ *make-polygonal-path* (*b* # *xs*)

definition *polygonal-path* :: *R-to-R2* ⇒ *bool* **where**
polygonal-path g ⟷ *g* ∈ *make-polygonal-path*‘{*xs* :: (*real^2*) *list*. *True*}

definition *all-integral* :: (*real^2*) *list* ⇒ *bool* **where**
all-integral l = (∀ *x* ∈ *set l*. *integral-vec x*)

definition *polygon* :: *R-to-R2* ⇒ *bool* **where**
polygon g ⟷ *polygonal-path g* ∧ *simple-path g* ∧ *closed-path g*

definition *integral-polygon* :: *R-to-R2* ⇒ *bool* **where**
integral-polygon g ⟷
(*polygon g* ∧ (∃ *vts*. *g* = *make-polygonal-path vts* ∧ *all-integral vts*))

definition *make-triangle* :: *real^2* ⇒ *real^2* ⇒ *real^2* ⇒ *R-to-R2* **where**
make-triangle a b c = *make-polygonal-path* [*a*, *b*, *c*, *a*]

definition *polygon-of* :: *R-to-R2* ⇒ (*real^2*) *list* ⇒ *bool* **where**
polygon-of p vts ⟷ *polygon p* ∧ *p* = *make-polygonal-path vts*

definition *good-linepath* :: *real^2* ⇒ *real^2* ⇒ (*real^2*) *list* ⇒ *bool* **where**
good-linepath a b vts ⟷ (*let p* = *make-polygonal-path vts in*
a ≠ *b* ∧ {*a*, *b*} ⊆ *set vts* ∧ *path-image* (*linepath a b*) ⊆ *path-inside p* ∪ {*a*, *b*})

definition *good-polygonal-path* :: *real^2* ⇒ (*real^2*) *list* ⇒ *real^2* ⇒ (*real^2*) *list*

$\Rightarrow$ *bool* **where**
good-polygonal-path a cutvts b vts $\longleftrightarrow$ (
let p = make-polygonal-path vts in
let p-cut = make-polygonal-path ([*a*] @ *cutvts* @ [*b*]) *in*
($a \neq b \wedge \{a, b\} \subseteq$ *set vts* $\wedge$ *path-image* (*p-cut*) $\subseteq$ *path-inside p* $\cup \{a, b\} \wedge$ *loop-free p-cut*))

4 Jordan Curve Theorem for Polygons

definition *inside-outside* :: *R-to-R2* $\Rightarrow$ (*real^2*) *set* $\Rightarrow$ (*real^2*) *set* $\Rightarrow$ *bool* **where**
inside-outside p ins outs $\longleftrightarrow$
(*ins* $\neq \{\}$ $\wedge$ *open ins* $\wedge$ *connected ins* $\wedge$
outs $\neq \{\}$ $\wedge$ *open outs* $\wedge$ *connected outs* $\wedge$
bounded ins $\wedge \neg$ *bounded outs* $\wedge$
ins $\cap$ *outs* $= \{\}$ $\wedge$ *ins* $\cup$ *outs* $=$ $-$ *path-image p* $\wedge$
frontier ins = *path-image p* $\wedge$ *frontier outs* = *path-image p*)

lemma *Jordan-inside-outside-real2*:
fixes *p* :: *real* $\Rightarrow$ *real^2*
assumes *simple-path p pathfinish p* = *pathstart p*
shows *inside*(*path-image p*) $\neq \{\}$ $\wedge$
open(*inside*(*path-image p*)) $\wedge$
connected(*inside*(*path-image p*)) $\wedge$
outside(*path-image p*) $\neq \{\}$ $\wedge$
open(*outside*(*path-image p*)) $\wedge$
connected(*outside*(*path-image p*)) $\wedge$
bounded(*inside*(*path-image p*)) $\wedge$
$\neg$ *bounded*(*outside*(*path-image p*)) $\wedge$
inside(*path-image p*) $\cap$ *outside*(*path-image p*) $= \{\}$ $\wedge$
inside(*path-image p*) $\cup$ *outside*(*path-image p*) =
$-$ *path-image p* $\wedge$
frontier(*inside*(*path-image p*)) = *path-image p* $\wedge$
frontier(*outside*(*path-image p*)) = *path-image p*
⟨*proof*⟩

lemma *inside-outside-polygon*:
fixes *p* :: *R-to-R2*
assumes *polygon*: *polygon p*
shows *inside-outside p* (*path-inside p*) (*path-outside p*)
⟨*proof*⟩

lemma *inside-outside-unique*:
fixes *p* :: *R-to-R2*
assumes *polygon p*
assumes *io1*: *inside-outside p inside1 outside1*
assumes *io2*: *inside-outside p inside2 outside2*
shows *inside1* = *inside2* $\wedge$ *outside1* = *outside2*

⟨*proof*⟩

lemma *polygon-jordan-curve*:
fixes *p* :: *R-to-R2*
assumes *polygon p*
obtains *inside outside* **where**
inside-outside p inside outside
⟨*proof*⟩

lemma *connected-component-image*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes *linear f bij f*
shows *f* ‘ (*connected-component-set S x*) = *connected-component-set* (*f* ‘ *S*) (*f x*)
⟨*proof*⟩

lemma *bounded-map*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes *linear f bij f*
shows *bounded* (*f* ‘ *S*) = *bounded S*
⟨*proof*⟩

lemma *inside-bijective-linear-image*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
fixes *c* :: *real* ⇒ *'a*
assumes *c-simple*:*path c*
assumes *linear f bij f*
shows *inside* (*f* ‘ (*path-image c*)) = *f* ‘ (*inside*(*path-image c*))
⟨*proof*⟩

lemma *bij-image-intersection*:
assumes *path-image c1* ∩ *path-image c2* = *S*
assumes *bij f*
assumes *c* ∈ *path-image* (*f* ∘ *c1*) ∩ *path-image* (*f* ∘ *c2*)
shows *c* ∈ *f* ‘ *S*
⟨*proof*⟩

theorem (**in** *c1-on-open-R2*) *split-inside-simple-closed-curve-locale*:
fixes *c* :: *real* ⇒ *'a*
assumes *c1-simple*:*simple-path c1* **and** *c1-start*: *pathstart c1* = *a* **and** *c1-end*: *pathfinish c1* = *b*
assumes *c2-simple*: *simple-path c2* **and** *c2-start*: *pathstart c2* = *a* **and** *c2-end*: *pathfinish c2* = *b*
assumes *c-simple*: *simple-path c* **and** *c-start*: *pathstart c* = *a* **and** *c-end*: *pathfinish c* = *b*
assumes *a-neq-b*: *a* ≠ *b*
and *c1c2*: *path-image c1* ∩ *path-image c2* = {*a*,*b*}
and *c1c*: *path-image c1* ∩ *path-image c* = {*a*,*b*}

and *c2c*: *path-image c2* ∩ *path-image c* = {*a*,*b*}
and *ne-12*: *path-image c* ∩ *inside*(*path-image c1* ∪ *path-image c2*) ≠ {}
obtains *inside*(*path-image c1* ∪ *path-image c*) ∩ *inside*(*path-image c2* ∪ *path-image c*) = {}
inside(*path-image c1* ∪ *path-image c*) ∪ *inside*(*path-image c2* ∪ *path-image c*) ∪
(*path-image c* − {*a*,*b*}) = *inside*(*path-image c1* ∪ *path-image c2*)
⟨*proof*⟩

lemma *split-inside-simple-closed-curve-real2*:
fixes *c* :: *real* ⇒ *real*^2
assumes *c1-simple*:*simple-path c1* **and** *c1-start*: *pathstart c1* = *a* **and** *c1-end*: *pathfinish c1* = *b*
assumes *c2-simple*: *simple-path c2* **and** *c2-start*: *pathstart c2* = *a* **and** *c2-end*: *pathfinish c2* = *b*
assumes *c-simple*: *simple-path c* **and** *c-start*: *pathstart c* = *a* **and** *c-end*: *pathfinish c* = *b*
assumes *a-neq-b*: *a* ≠ *b*
and *c1c2*: *path-image c1* ∩ *path-image c2* = {*a*,*b*}
and *c1c*: *path-image c1* ∩ *path-image c* = {*a*,*b*}
and *c2c*: *path-image c2* ∩ *path-image c* = {*a*,*b*}
and *ne-12*: *path-image c* ∩ *inside*(*path-image c1* ∪ *path-image c2*) ≠ {}
obtains *inside*(*path-image c1* ∪ *path-image c*) ∩ *inside*(*path-image c2* ∪ *path-image c*) = {}
inside(*path-image c1* ∪ *path-image c*) ∪ *inside*(*path-image c2* ∪ *path-image c*) ∪
(*path-image c* − {*a*,*b*}) = *inside*(*path-image c1* ∪ *path-image c2*)
⟨*proof*⟩

end
theory *Polygon-Lemmas*
imports
Polygon-Jordan-Curve
HOL−Library.Sublist
HOL.Set-Interval
HOL.Fun

begin

5 Properties of make polygonal path, pathstart and pathfinish of a polygon

lemma *make-polygonal-path-induct*[*case-names Empty Single Two Multiple*]:
fixes *ell* :: (*real*^2) *list*
assumes *empty*: ⋀*ell*. *ell* = [] ⟹ *P ell*
and *single*: ⋀*ell*. ⟦*length ell* = 1⟧ ⟹ *P ell*
and *two*: ⋀*ell*. ⟦*length ell* = 2⟧ ⟹ *P ell*
and *multiple*: ⋀*ell*.

⟦*length ell* > *2*;
P ([(*ell*!*0*), (*ell*!*1*)]);
P ((*ell*!*1*)#(*drop 2 ell*))⟧ ⟹ *P ell*
shows *P ell*
⟨*proof*⟩

lemma *make-polygonal-path-gives-path*:
fixes *v* :: (*real*^*2*) *list*
shows *path* (*make-polygonal-path v*)
⟨*proof*⟩

corollary *polygonal-path-is-path*:
fixes *g* :: *R-to-R2*
assumes *polygonal-path g*
shows *path g*
⟨*proof*⟩

lemma *polygon-to-polygonal-path*:
fixes *p* :: *R-to-R2*
assumes *polygon p*
obtains *ell* **where** *p* = *make-polygonal-path ell*
⟨*proof*⟩

lemma *polygon-pathstart*:
fixes *g* :: *R-to-R2*
assumes *l* ≠ []
assumes *g* = *make-polygonal-path l*
shows *pathstart g* = *l*!*0*
⟨*proof*⟩

lemma *polygon-pathfinish*:
fixes *g* :: *R-to-R2*
assumes *l* ≠ []
assumes *g* = *make-polygonal-path l*
shows *pathfinish g* = *l*!(*length l* − *1*)
⟨*proof*⟩

lemma *make-polygonal-path-image-property*:
assumes *length vts* ≥ *2*
assumes *p-is-path*: *x* ∈ *path-image* (*make-polygonal-path vts*)
shows ∃ *k* < *length vts* − *1*. *x* ∈ *path-image* (*linepath* (*vts* ! *k*) (*vts* ! (*k* + *1*)))
⟨*proof*⟩

lemma *linepaths-subset-make-polygonal-path-image*:
assumes *length vts* ≥ *2*
assumes *k* < *length vts* − *1*
shows *path-image* (*linepath* (*vts* ! *k*) (*vts* ! (*k* + *1*))) ⊆ *path-image* (*make-polygonal-path vts*)

⟨*proof*⟩

lemma *vertices-on-path-image*: **shows** *set vts* ⊆ *path-image* (*make-polygonal-path vts*)
⟨*proof*⟩

lemma *path-image-cons-union*:
assumes p = *make-polygonal-path vts*
assumes p' = *make-polygonal-path vts'*
assumes *vts'* ≠ []
assumes *vts* = *a* # *vts'* ∧ *b* = *vts'*!*0*
shows *path-image p* = *path-image* (*linepath a b*) ∪ *path-image p'*
⟨*proof*⟩

lemma *polygonal-path-image-linepath-union*:
assumes p = *make-polygonal-path vts*
assumes n = *length vts*
assumes $n \geq 2$
shows *path-image p* = (⋃ {*path-image* (*linepath* (*vts*!*i*) (*vts*!(*i*+*1*))) | *i*. *i* ≤ *n* − *2*})
⟨*proof*⟩

6 Loop Free Properties

lemma *constant-linepath-is-not-loop-free*:
shows ¬(*loop-free* ((*linepath a a*)::*real* ⇒ *real*^*2*))
⟨*proof*⟩

lemma *doubling-back-is-not-loop-free*:
assumes *a* ≠ *b*
shows ¬(*loop-free* ((*make-polygonal-path* [*a*, *b*, *a*])::*real* ⇒ *real*^*2*))
⟨*proof*⟩

lemma *not-loop-free-first-component*:
assumes ¬(*loop-free p1*)
shows ¬(*loop-free* (*p1*+++*p2*))
⟨*proof*⟩

lemma *not-loop-free-second-component*:
assumes *pathfinish-pathstart*: *pathfinish p1* = *pathstart p2*
assumes ¬(*loop-free p2*)
shows ¬(*loop-free* (*p1*+++*p2*))
⟨*proof*⟩

lemma *loop-free-subpath*:
assumes *path p*
assumes *u-and-v*: *u* ∈ {*0..1*} *v* ∈ {*0..1*} *u* < *v*
assumes ¬ (*loop-free* (*subpath u v p*))
shows ¬ (*loop-free p*)

⟨*proof*⟩

lemma *loop-free-associative*:
assumes *path p*
assumes *path q*
assumes *path r*
assumes *pathfinish p* = *pathstart q*
assumes *pathfinish q* = *pathstart r*
shows ¬ (*loop-free* ((*p* +++ *q*) +++ *r*)) ⟷ ¬ (*loop-free* (*p* +++ (*q* +++ *r*)))
⟨*proof*⟩

lemma *polygon-at-least-3-vertices*:
assumes *polygon p* **and**
p = *make-polygonal-path vts*
shows *card* (*set vts*) ≥ *3*
⟨*proof*⟩

lemma *polygon-vertices-length-at-least-4*:
assumes *polygon p* **and**
p = *make-polygonal-path vts*
shows *length vts* ≥ *4*
⟨*proof*⟩

lemma *linepath-loop-free*:
assumes *a* ≠ *b*
shows *loop-free* (*linepath a b*)
⟨*proof*⟩

7 Explicit Linepath Characterization of Polygonal Paths

lemma *triangle-linepath-images*:
fixes *x* :: *real*
assumes *vts* = [*a*, *b*, *c*]
assumes *p* = *make-polygonal-path vts*
shows *x* ∈ {*0..1/2*} ⟹ *p x* = ((*linepath a b*)) (*2*x*)
x ∈ {*1/2..1*} ⟹ *p x* = ((*linepath b c*)) (*2*x* − *1*)
⟨*proof*⟩

lemma *polygon-linepath-images1*:
fixes *n*:: *nat*
assumes *n* ≥ *3*
assumes *length ell* = *n*
assumes *x* ∈ {*0..1/2*}
shows *make-polygonal-path ell x* = ((*linepath* (*ell* ! *0*) (*ell* ! *1*))) (*2*x*)
⟨*proof*⟩

lemma *sum-insert* [*simp*]:
assumes $x \notin F$ **and** *finite F*
shows $(\sum y{\in}insert\ x\ F.\ P\ y) = (\sum y{\in}F.\ P\ y) + P\ x$
⟨*proof*⟩

lemma *sum-of-index-diff* [*simp*]:
fixes f:: $nat \Rightarrow 'a$::*comm-monoid-add*
shows $(\sum i{\in}\{a..{<}a{+}b\}.\ f(i{-}a)) = (\sum i{\in}\{..{<}b\}.\ f(i))$
⟨*proof*⟩

lemma *sum-of-index-diff2* [*simp*]:
fixes f :: $nat \Rightarrow 'a$::*comm-monoid-add*
shows $(\sum i{\in}\{a{+}c..b{+}c\}.\ f(i)) = (\sum i{\in}\{a..b\}.\ f(i{+}c))$
⟨*proof*⟩

lemma *sum-split* [*simp*]:
fixes f :: $nat \Rightarrow 'a$::*comm-monoid-add*
assumes $c \in \{a..b\}$
shows $(\sum i \in \{a..b\}.\ f\ i) = (\sum i \in \{a..c\}.\ f\ i) + (\sum i \in \{c{+}1..b\}.\ f\ i)$
⟨*proof*⟩

lemma *summation-helper*:
fixes x :: *real*
fixes k :: *nat*
assumes $1 \leq k$
shows $(2{::}real) * (\sum i = 1..k.\ 1\ /\ 2\ \hat{}\ i) - 1 = (\sum i = 1..(k{-}1).\ (1\ /\ (2\hat{}i)))$
⟨*proof*⟩

lemma *polygon-linepath-images2*:
fixes $n\ k$:: *nat*
fixes *ell*:: (*real*^2) *list*
fixes f :: $nat \Rightarrow real \Rightarrow real$
assumes $n \geq 3$
assumes $0 \leq k \wedge k \leq n - 3$
assumes *length ell* $= n$
assumes p: p = *make-polygonal-path ell*
assumes $f = (\lambda k\ x.\ (x - (\sum i \in \{1..k\}.\ 1/(2\hat{}i))) * (2\hat{}(k{+}1)))$
assumes $x \in \{(\sum i \in \{1..k\}.\ 1/(2\hat{}i))..(\sum i \in \{1..(k + 1)\}.\ 1/(2\hat{}i))\}$
shows $p\ x = ((linepath\ (ell\ !\ k)\ (ell\ !\ (k{+}1))\ (f\ k\ x)))$
⟨*proof*⟩

lemma *polygon-linepath-images3*:
fixes $n\ k$:: *nat*
fixes *ell*:: (*real*^2) *list*
assumes $n \geq 3$
assumes *length ell* $= n$
assumes p = *make-polygonal-path ell*
assumes $x \in \{(\sum i \in \{1..n{-}2\}.\ 1/(2\hat{}i))..1\}$

assumes $f = (\lambda x.\ (x - (\sum i \in \{1..n-2\}.\ 1/(2\hat{}i))) * (2\hat{}(n-2)))$
shows $p\ x = (linepath\ (ell\ !\ (n-2))\ (ell\ !\ (n-1)))\ (f\ x)$
⟨*proof*⟩

8 A Triangle is a Polygon

lemma *not-collinear-linepaths-intersect-helper*:
assumes *not-collinear*: $\neg collinear\ \{a,b,c\}$
assumes $0 \leq k1$
assumes $k1 \leq 1$
assumes $0 \leq k2$
assumes $k2 \leq 1$
assumes *eo*: $k2 = 0 \Longrightarrow k1 \neq 1$
shows $\neg\ ((linepath\ a\ b)\ k1 = (linepath\ b\ c)\ k2)$
⟨*proof*⟩

lemma *not-collinear-linepaths-intersect-helper-2*:
assumes *not-collinear*: $\neg collinear\ \{a,b,c\}$
assumes $0 \leq k1$
assumes $k1 \leq 1$
assumes $0 \leq k2$
assumes $k2 \leq 1$
assumes *eo*: $k1 = 0 \Longrightarrow k2 \neq 1$
shows $\neg\ ((linepath\ a\ b)\ k1 = (linepath\ c\ a)\ k2)$
⟨*proof*⟩

lemma *not-collinear-loopfree-path*: $\bigwedge a\ b\ c{::}real\hat{}2.\ \neg collinear\ \{a,b,c\} \Longrightarrow loop\text{-}free\ ((linepath\ a\ b)\ +\!+\!+\ (linepath\ b\ c))$
⟨*proof*⟩

lemma *triangle-is-polygon*: $\bigwedge a\ b\ c.\ \neg collinear\ \{a,b,c\} \Longrightarrow polygon\ (make\text{-}triangle\ a\ b\ c)$
⟨*proof*⟩

lemma *have-wraparound-vertex*:
assumes *polygon p*
assumes $p = make\text{-}polygonal\text{-}path\ vts$
shows $vts = (take\ (length\ vts - 1)\ vts)@[vts\ !\ 0]$
⟨*proof*⟩

lemma *polygon-at-least-3-vertices-wraparound*:
assumes *polygon p*
assumes $p = make\text{-}polygonal\text{-}path\ vts$
shows $card\ (set\ (take\ (length\ vts - 1)\ vts)) \geq 3$
⟨*proof*⟩

9 Polygon Vertex Rotation

definition *rotate-polygon-vertices*:: *'a list ⇒ nat ⇒ 'a list*
where *rotate-polygon-vertices ell i =*
(let ell1 = rotate i (butlast ell) in ell1 @ [ell1 ! 0])

lemma *rotate-polygon-vertices-same-set*:
assumes *polygon (make-polygonal-path vts)*
shows *set (rotate-polygon-vertices vts i) = set vts*
⟨*proof*⟩

lemma *arb-rotation-as-single-rotation*:
fixes *i*:: *nat*
shows *rotate-polygon-vertices vts (Suc i) = rotate-polygon-vertices (rotate-polygon-vertices vts i) 1*
⟨*proof*⟩

lemma *rotation-sum*:
fixes *i j* :: *nat*
shows *rotate-polygon-vertices vts (i + j) = rotate-polygon-vertices (rotate-polygon-vertices vts i) j*
⟨*proof*⟩

lemma *rotated-polygon-vertices-helper*:
fixes *p* :: *R-to-R2*
assumes *poly-p*: *polygon p*
assumes *p-is-path*: *p = make-polygonal-path vts*
assumes *p'-is*: *p' = make-polygonal-path (rotate-polygon-vertices vts 1)*
shows *(vts ! 0) = (rotate-polygon-vertices vts 1) ! (length (rotate-polygon-vertices vts 1) − 2)*
(rotate-polygon-vertices vts 1) ! (length (rotate-polygon-vertices vts 1) − 1)
= (vts ! 1)
⟨*proof*⟩

lemma *rotate-polygon-vertices-same-length*:
fixes *vts* :: *(real^2) list*
assumes *length vts ≥ 1*
shows *length vts = length (rotate-polygon-vertices vts i)*
⟨*proof*⟩

lemma *rotated-polygon-vertices-helper2*:
assumes *len-gteq*: *length vts ≥ 2*
assumes *i < length vts − 1*
assumes *hd vts = last vts*
shows *(rotate-polygon-vertices vts 1) ! i = vts ! (i+1)*
⟨*proof*⟩

lemma *polygon-rotation-t-translation1*:
assumes *polygon-of p vts*

assumes $p' = make\text{-}polygonal\text{-}path\ (rotate\text{-}polygon\text{-}vertices\ vts\ 1)$
(**is** $p' = make\text{-}polygonal\text{-}path\ ?vts'$)
assumes $x' \in \{(\sum i \in \{1..k\}.\ 1/(2\hat{}i))..(\sum i \in \{1..k+1\}.\ 1/(2\hat{}i))\}$
assumes $n = length\ vts$
assumes $0 \leq k \wedge k \leq n - 4$
assumes $l = x' - (\sum i \in \{1..k\}.\ 1/(2\hat{}i))$
assumes $x = l/2 + (\sum i \in \{1..(k+1)\}.\ 1/(2\hat{}i))$
shows $x \in \{(\sum i \in \{1..k+1\}.\ 1/(2\hat{}i))..(\sum i \in \{1..k+2\}.\ 1/(2\hat{}i))\}$
$p'\ x' = p\ x$
⟨*proof*⟩

lemma *polygon-rotation-t-translation1-strict*:
assumes *polygon-of p vts*
assumes $p' = make\text{-}polygonal\text{-}path\ (rotate\text{-}polygon\text{-}vertices\ vts\ 1)$
(**is** $p' = make\text{-}polygonal\text{-}path\ ?vts'$)
assumes $x' \in \{(\sum i \in \{1..k\}.\ 1/(2\hat{}i))..<(\sum i \in \{1..k+1\}.\ 1/(2\hat{}i))\}$
assumes $n = length\ vts$
assumes $0 \leq k \wedge k \leq n - 4$
assumes $l = x' - (\sum i \in \{1..k\}.\ 1/(2\hat{}i))$
assumes $x = l/2 + (\sum i \in \{1..(k+1)\}.\ 1/(2\hat{}i))$
shows $x \in \{(\sum i \in \{1..k+1\}.\ 1/(2\hat{}i))..<(\sum i \in \{1..k+2\}.\ 1/(2\hat{}i))\}$
$p'\ x' = p\ x$
⟨*proof*⟩

lemma *polygon-rotation-t-translation2*:
assumes *polygon-of p vts*
assumes $p' = make\text{-}polygonal\text{-}path\ (rotate\text{-}polygon\text{-}vertices\ vts\ 1)$
(**is** $p' = make\text{-}polygonal\text{-}path\ ?vts'$)
assumes $n = length\ vts$
assumes $x' \in \{(\sum i \in \{1..(n-3)\}.\ 1/(2\hat{}i))..(\sum i \in \{1..(n-2)\}.\ 1/(2\hat{}i))\}$
assumes $x = x' + 1/(2\hat{}(n-2))$
shows $x \in \{(\sum i \in \{1..n-2\}.\ 1/(2\hat{}i))..1\}$
$p'\ x' = p\ x$
⟨*proof*⟩

lemma *polygon-rotation-t-translation2-strict*:
assumes *polygon-of p vts*
assumes $p' = make\text{-}polygonal\text{-}path\ (rotate\text{-}polygon\text{-}vertices\ vts\ 1)$
(**is** $p' = make\text{-}polygonal\text{-}path\ ?vts'$)
assumes $n = length\ vts$
assumes $x' \in \{(\sum i \in \{1..(n-3)\}.\ 1/(2\hat{}i))..<(\sum i \in \{1..(n-2)\}.\ 1/(2\hat{}i))\}$
assumes $x = x' + 1/(2\hat{}(n-2))$
shows $x \in \{(\sum i \in \{1..n-2\}.\ 1/(2\hat{}i))..<1\}$
$p'\ x' = p\ x$
⟨*proof*⟩

lemma *polygon-rotation-t-translation3*:
assumes *polygon-of p vts*

assumes $p' = make\text{-}polygonal\text{-}path\ (rotate\text{-}polygon\text{-}vertices\ vts\ 1)$
(**is** $p' = make\text{-}polygonal\text{-}path\ ?vts'$)
assumes $x' \in \{(\sum i \in \{1..n-2\}.\ 1/(2\hat{}i))..1\}$
assumes $n = length\ vts$
assumes $l = x' - (\sum i \in \{1..n-2\}.\ 1/(2\hat{}i))$
assumes $x = l * (2\hat{}(n-3))$
shows $x \in \{0..1/2\}$
$p'\ x' = p\ x$
⟨*proof*⟩

lemma *polygon-rotation-t-translation3-strict*:
assumes *polygon-of p vts*
assumes $p' = make\text{-}polygonal\text{-}path\ (rotate\text{-}polygon\text{-}vertices\ vts\ 1)$
(**is** $p' = make\text{-}polygonal\text{-}path\ ?vts'$)
assumes $x' \in \{(\sum i \in \{1..n-2\}.\ 1/(2\hat{}i))..<1\}$
assumes $n = length\ vts$
assumes $l = x' - (\sum i \in \{1..n-2\}.\ 1/(2\hat{}i))$
assumes $x = l * (2\hat{}(n-3))$
shows $x \in \{0..<1/2\}$
$p'\ x' = p\ x$
⟨*proof*⟩

lemma *f-gteq-0-sum-gt*: $\bigwedge f{::}nat \Rightarrow real.\ (\bigwedge i{::}nat.\ (f\ i) > 0) \Longrightarrow a > b \Longrightarrow (\sum i = 1..a.\ (f\ i)) > (\sum i = 1..b.\ (f\ i))$ **for** $a\ b :: nat$
⟨*proof*⟩

lemma *rotation-intervals-disjoint*:
assumes $k1 \neq k2$
shows $\{\sum i = 1..k1.\ 1\ /\ (2\ \hat{}\ i{::}real)..<\sum i = 1..k1+1.\ 1\ /\ 2\ \hat{}\ i\} \cap \{\sum i = 1..k2.\ 1\ /\ (2\ \hat{}\ i{::}real)..<\sum i = 1..k2+1.\ 1\ /\ 2\ \hat{}\ i\} = \{\}$
⟨*proof*⟩

lemma *bounding-interval-helper1*:
shows $(\sum i = 1..k.\ 1\ /\ (2\ \hat{}\ i{::}real)) = (2\hat{}k - 1)/(2\hat{}k)$
⟨*proof*⟩

lemma *bounding-interval-helper2*:
fixes $x :: real$
assumes $x \in \{0..<1\}$
shows $\exists k.\ x < (\sum i = 1..k.\ 1\ /\ (2\ \hat{}\ i{::}real))$
⟨*proof*⟩

lemma *bounding-interval-for-reals-btw01*:
fixes $x{::}real$
assumes $x \in \{0..<1\}$
shows $\exists k.\ x \in \{(\sum i \in \{1..k\}.\ 1/(2\hat{}i{::}real))..<(\sum i \in \{1..(k+1)\}.\ 1/(2\hat{}i))\}$
⟨*proof*⟩

lemma *all-rotation-intervals-between-0and1*:

shows $\{(\sum i \in \{1..k\}.\ 1/(2\hat{}i::real))..(\sum i \in \{1..(k+1)\}.\ 1/(2\hat{}i))\} \subseteq \{0..<1\}$
⟨*proof*⟩

lemma *all-rotation-intervals-between-0and1-strict*:
shows $\{(\sum i \in \{1..k\}.\ 1/(2\hat{}i::real))..<(\sum i \in \{1..(k+1)\}.\ 1/(2\hat{}i))\} \subseteq \{0..<1\}$
⟨*proof*⟩

lemma *one-polygon-rotation-is-loop-free*:
assumes *polygon-of p vts*
assumes $p' =$ *make-polygonal-path* (*rotate-polygon-vertices vts 1*)
(**is** $p' =$ *make-polygonal-path ?vts'*)
shows *loop-free* p'
⟨*proof*⟩

lemma *one-rotation-is-polygon*:
fixes $p ::$ *R-to-R2*
fixes $i ::$ *nat*
assumes *poly-p*: *polygon p* **and**
p-is-path: $p =$ *make-polygonal-path vts* **and**
p'*-is*: $p' =$ *make-polygonal-path* (*rotate-polygon-vertices vts 1*)
(**is** $p' =$ *make-polygonal-path ?vts'*)
shows *polygon* p'
⟨*proof*⟩

lemma *rotation-is-polygon*:
fixes $p ::$ *R-to-R2*
fixes $i::$ *nat*
assumes *polygon p* **and**
$p =$ *make-polygonal-path vts*
shows *polygon* (*make-polygonal-path* (*rotate-polygon-vertices vts i*))
⟨*proof*⟩

lemma *polygon-rotate-mod*:
fixes *vts* :: (*real*^2) *list*
assumes $n =$ *length vts*
assumes $n \geq 2$
assumes *hd vts* = *last vts*
shows *rotate-polygon-vertices vts* $(n - 1) =$ *vts*
⟨*proof*⟩

lemma *polygon-rotate-mod-arb*:
fixes *vts* :: (*real*^2) *list*
assumes $n =$ *length vts*
assumes $n \geq 2$
assumes *hd vts* = *last vts*
shows *rotate-polygon-vertices vts* $((n - 1) * i) =$ *vts*
⟨*proof*⟩

lemma *unrotation-is-polygon*:

fixes p :: *R-to-R2*
fixes i:: *nat*
assumes *polygon* (*make-polygonal-path* (*rotate-polygon-vertices vts i*))
(**is** *polygon* (*make-polygonal-path ?vts'*))
p = *make-polygonal-path vts*
hd vts = *last vts*
shows *polygon p*
⟨*proof*⟩

lemma *rotated-polygon-vertices*:
assumes vts' = *rotate-polygon-vertices vts j*
assumes *hd vts* = *last vts*
assumes *length vts* ≥ *2*
assumes $j \leq i \wedge i <$ *length vts*
shows *vts* ! i = vts' ! $(i - j)$
⟨*proof*⟩

lemma *polygon-path-image*:
assumes *poly-p*: *polygon p*
assumes *p-is-path*: p = *make-polygonal-path vts*
shows *path-image* $p = p$ ‘ {*0* ..< *1*}
⟨*proof*⟩

lemma *polygon-vts-one-rotation*:
fixes p :: *R-to-R2*
assumes *poly-p*: *polygon p* **and**
p-is-path: p = *make-polygonal-path vts* **and**
p'*-is*: p' = *make-polygonal-path* (*rotate-polygon-vertices vts 1*)
shows *path-image p* = *path-image* p'
⟨*proof*⟩

lemma *polygon-vts-arb-rotation*:
fixes p :: *R-to-R2*
assumes *polygon p* **and**
p = *make-polygonal-path vts*
shows *path-image p* = *path-image* (*make-polygonal-path* (*rotate-polygon-vertices vts i*))
⟨*proof*⟩

10 Translating a Polygon

lemma *linepath-translation*:
linepath ((λx. x + u) a) ((λx. x + u) b) = (λx. x + u) ∘ (*linepath a b*)
⟨*proof*⟩

lemma *make-polygonal-path-translate*:
assumes *length vts* ≥ *2*
shows *make-polygonal-path* (*map* (λx. x + u) *vts*) = (λx. x + u) ∘ (*make-polygonal-path vts*)

⟨*proof*⟩

lemma *translation-is-polygon*:
assumes *polygon-of p vts*
shows *polygon-of* ((λ*x*. *x* + *u*) ∘ *p*) (*map* (λ*x*. *x* + *u*) *vts*) (**is** *polygon-of ?p′ ?vts′*)
⟨*proof*⟩

11 Misc. properties

lemma *tail-of-loop-free-polygonal-path-is-loop-free*:
assumes *loop-free* (*make-polygonal-path* (*x*#*tail*)) (**is** *loop-free ?p*) **and**
length tail ≥ *2*
shows *loop-free* (*make-polygonal-path tail*) (**is** *loop-free ?p′*)
⟨*proof*⟩

lemma *tail-of-simple-polygonal-path-is-simple*:
assumes *simple-path* (*make-polygonal-path* (*x*#*tail*)) (**is** *simple-path ?p*) **and**
length tail ≥ *2*
shows *simple-path* (*make-polygonal-path tail*) (**is** *simple-path ?p′*)
⟨*proof*⟩

lemma *interior-vtx-in-path-image-interior*:
fixes *vts* :: (*real*^*2*) *list*
assumes *x* ∈ *set* (*butlast* (*drop 1 vts*))
shows ∃ *t*. *t* ∈ {*0*<..<*1*} ∧ (*make-polygonal-path vts*) *t* = *x*
⟨*proof*⟩

lemma *loop-free-polygonal-path-vts-distinct*:
assumes *loop-free* (*make-polygonal-path vts*)
shows *distinct* (*butlast vts*)
⟨*proof*⟩

lemma *loop-free-polygonal-path-vts-drop1-distinct*:
assumes *loop-free* (*make-polygonal-path vts*)
shows *distinct* (*drop 1 vts*)
⟨*proof*⟩

lemma *simple-polygonal-path-vts-distinct*:
assumes *simple-path* (*make-polygonal-path vts*)
shows *distinct* (*butlast vts*)
⟨*proof*⟩

lemma *edge-subset-path-image*:
assumes *p* = *make-polygonal-path vts* **and**
(*i*::*int*) ∈ {*0*..<((*length vts*) − *1*)} **and**
x = *vts*!*i* **and**

$y = vts!(i+1)$
shows *path-image* (*linepath x y*) $\subseteq$ *path-image p* (**is** *?xy-img* $\subseteq$ *?p-img*)
⟨*proof*⟩

12 Properties of Sublists of Polygonal Path Vertex Lists

lemma *make-polygonal-path-image-append-var*:
assumes *length vts1* $\geq$ *2*
shows *path-image* (*make-polygonal-path* (*vts1* @ [*v*])) = *path-image* (*make-polygonal-path vts1* +++ (*linepath* (*vts1* ! (*length vts1* − *1*)) *v*))
⟨*proof*⟩

lemma *make-polygonal-path-image-append-helper*:
assumes *length vts1* $\geq$ *1* $\wedge$ *length vts2* $\geq$ *1*
shows *path-image* (*make-polygonal-path* (*vts1* @ [*v*] @ [*v*] @ *vts2*)) = *path-image* (*make-polygonal-path* (*vts1* @ [*v*] @ *vts2*))
⟨*proof*⟩

lemma *make-polygonal-path-image-append*:
assumes *length vts1* $\geq$ *2* $\wedge$ *length vts2* $\geq$ *2*
shows *path-image* (*make-polygonal-path* (*vts1* @ *vts2*)) = *path-image* (*make-polygonal-path vts1* +++ (*linepath* (*vts1* ! (*length vts1* − *1*)) (*vts2* ! *0*)) +++ *make-polygonal-path vts2*)
⟨*proof*⟩

lemma *make-polygonal-path-image-append-alt*:
assumes *p* = *make-polygonal-path vts*
assumes *p1* = *make-polygonal-path vts1*
assumes *p2* = *make-polygonal-path vts2*
assumes *last vts1* = *hd vts2*
assumes *length vts1* $\geq$ *2* $\wedge$ *length vts2* $\geq$ *2*
assumes *vts* = *vts1* @ (*tl vts2*)
shows *path-image p* = *path-image* (*p1* +++ *p2*)
⟨*proof*⟩

lemma *cont-incr-interval-image*:
fixes *f* :: *real* ⇒ *real*
assumes *a* $\leq$ *b*
assumes *continuous-on* {*a*..*b*} *f*
assumes $\forall x \in \{a..b\}.\ \forall y \in \{a..b\}.\ x \leq y \longrightarrow f\ x \leq f\ y$
shows *f*‘{*a*..*b*} = {*f a*..*f b*}
⟨*proof*⟩

lemma *two-x-minus-one-image*:
assumes *f* = (λ*x*::*real*. *2*∗*x* − *1*)
assumes *a* $\leq$ *b*
shows *f*‘{*a*..*b*} = {*f a*..*f b*}

⟨*proof*⟩

lemma *vts-split-path-image*:
assumes p = *make-polygonal-path vts*
assumes *p1* = *make-polygonal-path vts1*
assumes *p2* = *make-polygonal-path vts2*
assumes *vts1* = *take i vts*
assumes *vts2* = *drop* (*i*−*1*) *vts*
assumes n = *length vts*
assumes *1* ≤ i ∧ i < n
assumes x = (*2*^(*i*−*1*) − *1*)/(*2*^(*i*−*1*))
shows *path-image p1* = *p*‘{*0*..x} ∧ *path-image p2* = *p*‘{x..*1*}
⟨*proof*⟩

lemma *drop-i-is-loop-free*:
fixes *vts* :: (*real*^*2*) *list*
assumes m = *length vts*
assumes i ≤ m − *2*
assumes *vts*′ = *drop i vts*
assumes p = *make-polygonal-path vts*
assumes p' = *make-polygonal-path vts*′
assumes *loop-free p*
shows *loop-free* p'
⟨*proof*⟩

lemma *joinpaths-tl-transform*:
assumes f = (λx::*real*. *2*∗x − *1*)
assumes *pathfinish g1* = *pathstart g2*
assumes p = *g1* +++ *g2*
assumes x ≥ *1*/*2*
shows p x = *g2* (f x)
⟨*proof*⟩

lemma *joinpaths-tl-image-transform*:
assumes f = (λx::*real*. *2*∗x − *1*)
assumes *pathfinish g1* = *pathstart g2*
assumes p = *g1* +++ *g2*
assumes *1*/*2* ≤ a ∧ a ≤ b
shows *p*‘{a..b} = *g2*‘{f a..f b}
⟨*proof*⟩

lemma *vts-sublist-path-image*:
assumes p = *make-polygonal-path vts*
assumes p' = *make-polygonal-path vts*′
assumes *vts*′ = *take j* (*drop i vts*)
assumes m = *length vts*
assumes n = *length vts*′
assumes k = i + j
assumes k ≤ m − *1* ∧ *2* ≤ j

assumes *x1* = (*2^i* − *1*)/(*2^i*)
assumes *x2* = (*2^*(*k*−*1*) − *1*)/(*2^*(*k*−*1*))
shows *path-image p′* = *p*‘{*x1*..*x2*}
⟨*proof*⟩

lemma *one-append-simple-path*:
fixes *vts* :: (*real^2*) *list*
assumes *vts* = *vts′* @ [*z*]
assumes *n* = *length vts*
assumes *n* ≥ *3*
assumes *p* = *make-polygonal-path vts*
assumes *p′* = *make-polygonal-path vts′*
assumes *simple-path p*
shows *simple-path p′*
⟨*proof*⟩

lemma *take-i-is-loop-free*:
fixes *vts* :: (*real^2*) *list*
assumes *n* = *length vts*
assumes *2* ≤ *i* ∧ *i* ≤ *n*
assumes *vts′* = *take i vts*
assumes *p* = *make-polygonal-path vts*
assumes *p′* = *make-polygonal-path vts′*
assumes *loop-free p*
shows *loop-free p′*
⟨*proof*⟩

lemma *sublist-is-loop-free*:
fixes *vts* :: (*real^2*) *list*
assumes *p* = *make-polygonal-path vts*
assumes *p′* = *make-polygonal-path vts′*
assumes *loop-free p*
assumes *m* = *length vts*
assumes *n* = *length vts′*
assumes *sublist vts′ vts*
assumes *n* ≥ *2* ∧ *m* ≥ *2*
shows *loop-free p′*
⟨*proof*⟩

lemma *diff-points-path-image-set-property*:
fixes *a b*:: *real^2*
assumes *a* ≠ *b*
shows *path-image* (*linepath a b*) ≠ {*a*, *b*}
⟨*proof*⟩

lemma *polygonal-path-vertex-t*:
assumes *p* = *make-polygonal-path vts*
assumes *n* = *length vts*
assumes *n* ≥ *1*

assumes $0 \leq i \wedge i < n - 1$
assumes $x = (2\hat{}i - 1)/(2\hat{}i)$
shows $vts!i = p\ x$
⟨*proof*⟩

lemma *loop-free-split-int*:
assumes $p = make\text{-}polygonal\text{-}path\ vts \wedge loop\text{-}free\ p$
assumes $vts1 = take\ i\ vts$
assumes $vts2 = drop\ (i-1)\ vts$
assumes $c1 = make\text{-}polygonal\text{-}path\ vts1$
assumes $c2 = make\text{-}polygonal\text{-}path\ vts2$
assumes $n = length\ vts$
assumes $1 \leq i \wedge i < n$
shows $(path\text{-}image\ c1) \cap (path\text{-}image\ c2) \subseteq \{pathstart\ c1, pathstart\ c2\}$
(**is** $?C1 \cap ?C2 \subseteq \{pathstart\ c1, pathstart\ c2\}$)
⟨*proof*⟩

lemma *loop-free-arc-split-int*:
assumes $p = make\text{-}polygonal\text{-}path\ vts \wedge loop\text{-}free\ p \wedge arc\ p$
assumes $vts1 = take\ i\ vts$
assumes $vts2 = drop\ (i-1)\ vts$
assumes $c1 = make\text{-}polygonal\text{-}path\ vts1$
assumes $c2 = make\text{-}polygonal\text{-}path\ vts2$
assumes $n = length\ vts$
assumes $1 \leq i \wedge i < n$
shows $(path\text{-}image\ c1) \cap (path\text{-}image\ c2) \subseteq \{pathstart\ c2\}$
(**is** $?C1 \cap ?C2 \subseteq \{pathstart\ c2\}$)
⟨*proof*⟩

lemma *loop-free-append*:
assumes $p = make\text{-}polygonal\text{-}path\ vts$
assumes $p1 = make\text{-}polygonal\text{-}path\ vts1$
assumes $p2 = make\text{-}polygonal\text{-}path\ vts2$
assumes $vts = vts1\ @\ (tl\ vts2)$
assumes $loop\text{-}free\ p1 \wedge loop\text{-}free\ p2$
assumes $path\text{-}image\ p1 \cap path\text{-}image\ p2 \subseteq \{pathstart\ p1, pathstart\ p2\}$
assumes $last\ vts2 \neq hd\ vts1 \longrightarrow path\text{-}image\ p1 \cap path\text{-}image\ p2 \subseteq \{pathstart\ p2\}$
assumes $last\ vts1 = hd\ vts2$
assumes $arc\ p1 \wedge arc\ p2$
shows $loop\text{-}free\ p$
⟨*proof*⟩

lemma *sublist-path-image-subset*:
assumes $sublist\ vts1\ vts2$
assumes $length\ vts1 \geq 1$
shows $path\text{-}image\ (make\text{-}polygonal\text{-}path\ vts1) \subseteq path\text{-}image\ (make\text{-}polygonal\text{-}path\ vts2)$
⟨*proof*⟩

lemma *integral-on-edge-subset-integral-on-path*:
assumes p = *make-polygonal-path vts* **and**
$(i::int) \in \{0..<((length\ vts) - 1)\}$ **and**
$x = vts!i$ **and**
$y = vts!(i+1)$
shows $\{v.\ integral\text{-}vec\ v \wedge v \in path\text{-}image\ (linepath\ x\ y)\}$
$\subseteq \{v.\ integral\text{-}vec\ v \wedge v \in path\text{-}image\ p\}$
⟨*proof*⟩

lemma *sublist-pair-integral-subset-integral-on-path*:
assumes p = *make-polygonal-path vts* **and**
sublist $[x, y]$ *vts*
shows $\{v.\ integral\text{-}vec\ v \wedge v \in path\text{-}image\ (linepath\ x\ y)\}$
$\subseteq \{v.\ integral\text{-}vec\ v \wedge v \in path\text{-}image\ p\}$
⟨*proof*⟩

lemma *sublist-integral-subset-integral-on-path*:
assumes *length ell* $\geq$ *2*
assumes p = *make-polygonal-path vts* **and**
sublist ell vts
shows $\{v.\ integral\text{-}vec\ v \wedge v \in path\text{-}image\ (make\text{-}polygonal\text{-}path\ ell)\}$
$\subseteq \{v.\ integral\text{-}vec\ v \wedge v \in path\text{-}image\ p\}$
⟨*proof*⟩

13 Reversing Polygonal Path Vertex List

lemma *rev-vts-path-image*:
shows *path-image* (*make-polygonal-path* (*rev vts*)) = *path-image* (*make-polygonal-path vts*)
⟨*proof*⟩

lemma *rev-vts-is-loop-free*:
assumes p = *make-polygonal-path vts*
assumes *loop-free p*
shows *loop-free* (*make-polygonal-path* (*rev vts*))
⟨*proof*⟩

lemma *rev-vts-is-polygon*:
assumes *polygon-of p vts*
shows *polygon* (*make-polygonal-path* (*rev vts*))
⟨*proof*⟩

end
theory *Linepath-Collinearity*
imports *Polygon-Lemmas*

begin

14 Collinearity Properties

lemma *points-on-linepath-collinear*:
assumes *exists-c*: $(\exists\, c.\ a - b = c *_R u)$
assumes *x-in-linepath*: $x \in$ *path-image* (*linepath a b*)
shows $(\exists\, c.\ x - a = c *_R u)$ $(\exists\, c.\ b - x = c *_R u)$
⟨*proof*⟩

lemma *three-points-collinear-property*:
fixes *a b*:: *real^2*
assumes *exists-c1*: $(\exists\, c.\ a - x1 = c *_R u)$
assumes *exists-c2*: $(\exists\, c.\ a - x2 = c *_R u)$
shows $\exists\, c.\ x1 - x2 = c *_R u$
⟨*proof*⟩

lemma *in-path-image-imp-collinear*:
fixes *a b*:: *real^2*
assumes $k \in$ *path-image* (*linepath a b*)
shows *collinear* $\{a, b, k\}$
⟨*proof*⟩

lemma *two-linepath-colinearity-property*:
fixes *a b c d*:: *real^2*
assumes $y \neq z \wedge \{y, z\} \subseteq$ (*path-image* (*linepath a b*)) $\cap$ (*path-image* (*linepath c d*))
shows *collinear* $\{a, b, c, d\}$
⟨*proof*⟩

lemma *polygon-vts-not-collinear*:
assumes *polygon-of p vts*
shows $\neg$ *collinear* (*set vts*)
⟨*proof*⟩

lemma *not-collinear-with-subset*:
assumes *collinear A*
assumes $\neg$ *collinear* $(A \cup \{x\})$
assumes *card A* > 2
assumes $a \in A$
shows $\neg$ *collinear* $((A - \{a\}) \cup \{x\})$
⟨*proof*⟩

lemma *vec-diff-scale-collinear*:
fixes *a b c* :: *real^2*
assumes $b - a = m *_R (c - a)$
shows *collinear* $\{a, b, c\}$
⟨*proof*⟩

15 Linepath Properties

lemma *good-linepath-comm*: *good-linepath a b vts* $\Longrightarrow$ *good-linepath b a vts*
⟨*proof*⟩

lemma *finite-set-linepaths*:
assumes *polygon*: *polygon p*
assumes *polygonal-path*: *p* = *make-polygonal-path vts*
shows *finite* {(*a*, *b*). (*a*, *b*) ∈ *set vts* × *set vts*}
⟨*proof*⟩

lemma *linepaths-intersect-once-or-collinear*:
fixes *a b c d* :: *real^2*
assumes *path-image* (*linepath a b*) ∩ *path-image* (*linepath c d*) ≠ {}
shows *collinear* {*a*, *b*, *c*, *d*} ∨ (∃ *x*. *path-image* (*linepath a b*) ∩ *path-image* (*linepath c d*) = {*x*})
⟨*proof*⟩

lemma *linepaths-intersect-once-or-collinear-alt*:
fixes *a b c d* :: *real^2*
assumes *path-image* (*linepath a b*) ∩ *path-image* (*linepath c d*) ≠ {}
shows *collinear* {*a*, *b*, *c*, *d*} ∨ *card* (*path-image* (*linepath a b*) ∩ *path-image* (*linepath c d*)) = *1*
⟨*proof*⟩

lemma *path-image-linepath-union*:
fixes *a b* :: *'a*::*euclidean-space*
assumes *d* ∈ *path-image* (*linepath a b*)
shows *path-image* (*linepath a b*) = *path-image* (*linepath a d*) ∪ *path-image* (*linepath d b*)
⟨*proof*⟩

lemma *path-image-linepath-split*:
assumes *i* < (*length vts*) − *1*
assumes *x* ∈ *path-image* (*linepath* (*vts*!*i*) (*vts*!(*i*+*1*)))
assumes *x-notin*: *x* ∉ *set vts*
shows *path-image* (*make-polygonal-path vts*) = *path-image* (*make-polygonal-path* ((*take* (*i*+*1*) *vts*) @ [*x*] @ (*drop* (*i*+*1*) *vts*)))
⟨*proof*⟩

lemma *linepath-split-is-loop-free*:
assumes *d* ∈ *path-image* (*linepath a b*)
assumes *d* ∉ {*a*, *b*}
shows *loop-free* (*make-polygonal-path* [*a*, *d*, *b*]) (**is** *loop-free ?p*)
⟨*proof*⟩

lemma *loop-free-linepath-split-is-loop-free*:
assumes *p* = *make-polygonal-path vts*
assumes *loop-free p*

assumes $n = length\ vts$
assumes $i < n - 1$
assumes $x \in path\text{-}image\ (linepath\ (vts!i)\ (vts!(i+1))) \wedge x \notin set\ vts$
assumes $vts' = (take\ (i+1)\ vts)\ @\ [x]\ @\ (drop\ (i+1)\ vts)$
assumes $p' = make\text{-}polygonal\text{-}path\ vts'$
shows $loop\text{-}free\ p' \wedge path\text{-}image\ p' = path\text{-}image\ p$
⟨*proof*⟩

lemma *polygon-linepath-split-is-polygon*:
assumes *polygon-of p vts*
assumes $i < (length\ vts) - 1$
assumes $a = vts!i \wedge b = vts!(i+1)$
assumes $x \in path\text{-}image\ (linepath\ a\ b) \wedge x \notin set\ vts$
assumes $vts' = (take\ (i+1)\ vts)\ @\ [x]\ @\ (drop\ (i+1)\ vts)$
shows *polygon* (*make-polygonal-path vts′*)
⟨*proof*⟩

16 Measure of linepaths

lemma *linepath-is-negligible-vertical*:
fixes *a b* :: *real^2*
assumes *a$1* = *b$1*
defines *p* ≡ *linepath a b*
shows *negligible* (*path-image p*)
⟨*proof*⟩

lemma *linepath-is-negligible-non-vertical*:
fixes *a b* :: *real^2*
assumes *a$1* < *b$1*
defines *p* ≡ *linepath a b*
shows *negligible* (*path-image p*)
⟨*proof*⟩

lemma *linepath-is-negligible*:
fixes *a b* :: *real^2*
defines *p* ≡ *linepath a b*
shows *negligible* (*path-image p*)
⟨*proof*⟩

lemma *linepath-has-emeasure-0*:
emeasure lebesgue (*path-image* (*linepath* (*a*::(*real^2*)) (*b*::(*real^2*)))) = *0*
⟨*proof*⟩

lemma *linepath-has-measure-0*:
measure lebesgue (*path-image* (*linepath* (*a*::(*real^2*)) (*b*::(*real^2*)))) = *0*
⟨*proof*⟩

end

theory *Polygon-Convex-Lemmas*
imports
Polygon-Lemmas
Linepath-Collinearity

begin

17 Misc. Convex Polygon Properties

lemma *polygon-path-image-subset-convex*:
assumes *length vts* > *0*
shows *path-image* (*make-polygonal-path vts*) ⊆ *convex hull* (*set vts*) (**is** *path-image ?p* ⊆ *?S*)
⟨*proof*⟩

lemma *convex-contains-simple-closed-path-imp-contains-path-inside*:
assumes *convex S*
assumes *simple-path p* ∧ *closed-path p*
assumes *path-image p* ⊆ *S*
shows *path-inside p* ⊆ *S*
⟨*proof*⟩

lemma *convex-polygon-is-convex-hull*:
assumes *polygon p*
assumes *convex* (*path-inside p* ∪ *path-image p*)
assumes *p* = *make-polygonal-path vts*
shows *convex hull* (*set vts*) = *path-inside p* ∪ *path-image p* (**is** *?hull* = *?poly*)
⟨*proof*⟩

lemma *convex-polygon-inside-is-convex-hull-interior*:
assumes *polygon p*
assumes *convex* (*path-inside p*)
assumes *p* = *make-polygonal-path vts*
shows *interior* (*convex hull* (*set vts*)) = *path-inside p*
⟨*proof*⟩

lemma *convex-polygon-inside-is-convex-hull-interior2*:
assumes *polygon p*
assumes *convex* (*path-inside p* ∪ *path-image p*)
assumes *p* = *make-polygonal-path vts*
shows *interior* (*convex hull* (*set vts*)) = *path-inside p*
⟨*proof*⟩

lemma *polygon-convex-iff*:
assumes *polygon p*
shows *convex* (*path-inside p*) ⟷ *convex* (*path-inside p* ∪ *path-image p*)
⟨*proof*⟩

lemma *convex-polygon-frontier-is-path-image*:

assumes *polygon-of p vts*
assumes *convex* (*path-inside p*)
shows *frontier* (*convex hull* (*set vts*)) = *path-image p*
⟨*proof*⟩

lemma *convex-polygon-frontier-is-path-image2*:
assumes *polygon p*
assumes *convex* (*path-inside p*)
shows *frontier* (*path-image p* ∪ *path-inside p*) = *path-image p*
⟨*proof*⟩

lemma *convex-polygon-frontier-is-path-image3*:
assumes *polygon p*
assumes *convex* (*path-image p* ∪ *path-inside p*)
shows *frontier* (*path-image p* ∪ *path-inside p*) = *path-image p*
⟨*proof*⟩

lemma *polygon-frontier-is-path-image*:
assumes *polygon p*
shows *frontier* (*path-inside p*) = *path-image p*
⟨*proof*⟩

lemma *convex-path-inside-means-convex-polygon*:
assumes *polygon p*
assumes *frontier* (*convex hull* (*set vts*)) = *path-image p*
shows *convex* (*path-inside p*)
⟨*proof*⟩

lemma *convex-hull-of-polygon-is-convex-hull-of-vts*:
assumes *polygon-of p vts*
shows *convex hull* (*path-image p* ∪ *path-inside p*) = *convex hull* (*set vts*)
⟨*proof*⟩

lemma *convex-hull-frontier-polygon*:
assumes *polygon-of p vts*
assumes ¬ *set vts* ⊆ *frontier* (*convex hull* (*set vts*))
shows ¬ *convex* (*path-inside p*)
⟨*proof*⟩

lemma *frontier-int-subset*:
assumes *A* ⊆ *B*
shows (*frontier B*) ∩ *A* ⊆ *frontier A*
⟨*proof*⟩

lemma *in-frontier-in-subset*:
assumes *A* ⊆ *B*
assumes *x* ∈ *frontier B*
assumes *x* ∈ *A*
shows *x* ∈ *frontier A*

⟨*proof*⟩

lemma *in-frontier-in-subset-convex-hull*:
assumes $A \subseteq B$
assumes $x \in$ *frontier* (*convex hull* B)
assumes $x \in$ *convex hull* A
shows $x \in$ *frontier* (*convex hull* A)
⟨*proof*⟩

lemma *convex-hull-two-extreme-points*:
fixes S :: *'a*::*euclidean-space set*
assumes *finite* S
assumes *convex hull* $S \neq \{\}$
assumes $\forall x.$ *convex hull* $S \neq \{x\}$
shows *card* $\{x.\ x$ *extreme-point-of* (*convex hull* $S)\} \geq 2$ (**is** *card ?ep* ≥ 2)
⟨*proof*⟩

lemma *convex-hull-two-vts-on-frontier*:
fixes S :: *'a*::*euclidean-space set*
assumes *card* $S \geq 2$
shows *card* $(S \cap$ *frontier* (*convex hull* $S)) \geq 2$
⟨*proof*⟩

18 Vertices on Convex Frontier Implies Polygon is Convex

lemma *convex-cut-aux*:
assumes $\forall v \in S.\ z \cdot v \leq 0$
shows *convex hull* $S \subseteq \{x.\ z \cdot x \leq 0\}$
⟨*proof*⟩

lemma *convex-cut-aux'*:
assumes $\forall v \in S.\ z \cdot v \geq 0$
shows *convex hull* $S \subseteq \{x.\ z \cdot x \geq 0\}$
⟨*proof*⟩

lemma *convex-cut*:
assumes $z \neq 0$
assumes $\{x.\ z \cdot x = 0\} \cap$ *interior* (*convex hull* $S) \neq \{\}$
obtains *v1 v2* **where** $v1 \neq v2 \wedge \{v1, v2\} \subseteq S \wedge v1 \in \{x.\ z \cdot x < 0\} \wedge v2 \in \{x.\ z \cdot x > 0\}$
⟨*proof*⟩

lemma *affine-2-int-convex*:
fixes S :: *'a*::*euclidean-space set*
assumes $\{a, b\} \subseteq S$
assumes $\{a, b\} \subseteq$ *frontier* (*convex hull* S)
assumes *affine hull* $\{a, b\} \cap$ *interior* (*convex hull* $S) \neq \{\}$

shows *affine hull* $\{a, b\} \cap$ *convex hull* $S =$ *convex hull* $\{a, b\}$
⟨*proof*⟩

lemma *halfplane-frontier-affine-hull*:
fixes $b\ v :: real\hat{}2$
assumes $b \neq 0$
assumes $v \neq 0$
assumes $b \in \{x.\ v \cdot x = 0\}$
shows $\{x.\ v \cdot x = 0\} =$ *affine hull* $\{0, b\}$
⟨*proof*⟩

lemma *vts-on-convex-frontier-aux*:
assumes *polygon-of p vts*
assumes $vts!0 = 0$
assumes *set vts* $\subseteq$ *frontier* (*convex hull* (*set vts*))
shows *path-image* (*linepath* ($vts!0$) ($vts!1$)) $\subseteq$ *frontier* (*convex hull* (*set vts*))
⟨*proof*⟩

lemma *vts-on-convex-frontier-aux′*:
assumes *polygon-of p vts*
assumes *set vts* $\subseteq$ *frontier* (*convex hull* (*set vts*))
shows *path-image* (*linepath* ($vts!0$) ($vts!1$)) $\subseteq$ *frontier* (*convex hull* (*set vts*))
⟨*proof*⟩

lemma *vts-on-convex-frontier*:
assumes *polygon-of p vts*
assumes *set vts* $\subseteq$ *frontier* (*convex hull* (*set vts*))
assumes $i <$ *length vts* $- 1$
shows *path-image* (*linepath* ($vts!i$) ($vts!(i+1)$)) $\subseteq$ *frontier* (*convex hull* (*set vts*))
⟨*proof*⟩

lemma *vts-on-frontier-means-path-image-on-frontier*:
assumes *polygon-of p vts*
assumes *set vts* $\subseteq$ *frontier* (*convex hull* (*set vts*))
shows *path-image p* $\subseteq$ *frontier* (*convex hull* (*set vts*))
⟨*proof*⟩

lemma *vts-on-convex-frontier-interior*:
assumes *polygon-of p vts*
assumes *set vts* $\subseteq$ *frontier* (*convex hull* (*set vts*))
shows *path-inside p* $=$ *interior* (*convex hull* (*set vts*))
⟨*proof*⟩

lemma *vts-subset-frontier*:
assumes *polygon-of p vts*
assumes *set vts* $\subseteq$ *frontier* (*convex hull* (*set vts*))
shows *convex* (*path-image p* $\cup$ *path-inside p*)
⟨*proof*⟩

lemma *convex-hull-of-nonconvex-polygon-strict-subset-ep*:
assumes *polygon-of p vts*
assumes ¬ (*convex* (*path-image p* ∪ *path-inside p*))
shows {*v*. *v extreme-point-of* (*convex hull* (*set vts*))} ⊂ *set vts*
⟨*proof*⟩

lemma *convex-hull-of-nonconvex-polygon-strict-subset*:
assumes *polygon-of p vts*
assumes ¬ (*convex* (*path-image p* ∪ *path-inside p*))
shows ∃ *v* ∈ *set vts*. *v* ∈ *interior* (*convex hull* (*set vts*))
⟨*proof*⟩

lemma *convex-polygon-means-linepaths-inside*:
fixes *p* :: *R-to-R2*
assumes *polygon-of p vts*
assumes *convex-is*: *convex hull* (*set vts*) = (*path-inside p* ∪ *path-image p*)
assumes *a-in*: *a* ∈ (*path-inside p* ∪ *path-image p*)
assumes *b-in*: *b* ∈ (*path-inside p* ∪ *path-image p*)
shows *path-image* (*linepath a b*) ⊆ (*path-inside p* ∪ *path-image p*)
⟨*proof*⟩

end
theory *Polygon-Splitting*
imports
HOL−Analysis.Complete-Measure
Polygon-Jordan-Curve
Polygon-Convex-Lemmas
begin

19 Polygon Splitting

lemma *split-up-a-list-into-3-parts*:
fixes *i j*:: *nat*
assumes *i* < *length vts* ∧ *j* < *length vts* ∧ *i* < *j*
shows
vts = (*take i vts*) @ ((*vts* ! *i*) # ((*take* (*j* − *i* − *1*) (*drop* (*Suc i*) *vts*)) @ (*vts* ! *j*) # *drop* (*j* − *i*) (*drop* (*Suc i*) *vts*)))
⟨*proof*⟩

definition *is-polygon-cut* :: (*real^2*) *list* ⇒ *real^2* ⇒ *real^2* ⇒ *bool* **where**
is-polygon-cut vts x y =
(*x* ≠ *y* ∧
polygon (*make-polygonal-path vts*) ∧
{*x*, *y*} ⊆ *set vts* ∧
path-image (*linepath x y*) ∩ *path-image* (*make-polygonal-path vts*) = {*x*, *y*} ∧
path-image (*linepath x y*) ∩ *path-inside* (*make-polygonal-path vts*) ≠ {})

definition *is-polygon-cut-path* :: (*real^2*) *list* ⇒ *R-to-R2* ⇒ *bool* **where**

is-polygon-cut-path vts cutpath =
(*let x* = *pathstart cutpath* ; *y* = *pathfinish cutpath in*
(*x* ≠ *y* ∧
polygon (*make-polygonal-path vts*) ∧
{*x*, *y*} ⊆ *set vts* ∧
simple-path cutpath ∧
path-image cutpath ∩ *path-image* (*make-polygonal-path vts*) = {*x*, *y*} ∧
path-image cutpath ∩ *path-inside* (*make-polygonal-path vts*) ≠ {}))

definition *is-polygon-split* ::
(*real^2*) *list* ⇒ *nat* ⇒ *nat* ⇒ *bool* **where**
is-polygon-split vts i j =
(*i* < *length vts* ∧ *j* < *length vts* ∧ *i* < *j* ∧
(*let vts1* = (*take i vts*) *in*
let vts2 = (*take* (*j* − *i* − *1*) (*drop* (*Suc i*) *vts*)) *in*
let vts3 = *drop* (*j* − *i*) (*drop* (*Suc i*) *vts*) *in*
let x = *vts* ! *i in*
let y = *vts* ! *j in*
let p = *make-polygonal-path* (*vts*@[*vts*!*0*]) *in*
let p1 = *make-polygonal-path* (*x*#(*vts2*@[*y*, *x*])) *in*
let p2 = *make-polygonal-path* (*vts1* @ [*x*, *y*] @ *vts3* @ [*vts* ! *0*]) *in*
let c1 = *make-polygonal-path* (*x*#(*vts2*@[*y*])) *in*
let c2 = *make-polygonal-path* (*vts1* @ [*x*, *y*] @ *vts3*) *in*
(*is-polygon-cut* (*vts*@[*vts*!*0*]) *x y* ∧
polygon p ∧ *polygon p1* ∧ *polygon p2* ∧
path-inside p1 ∩ *path-inside p2* = {} ∧
path-inside p1 ∪ *path-inside p2* ∪ (*path-image* (*linepath x y*) − {*x*, *y*}) = *path-inside p*
∧ ((*path-image p1*) − (*path-image* (*linepath x y*))) ∩ ((*path-image p2*) − (*path-image* (*linepath x y*)))
= {}
∧ *path-image p*
= ((*path-image p1*) − (*path-image* (*linepath x y*))) ∪ ((*path-image p2*) − (*path-image* (*linepath x y*))) ∪ {*x*, *y*}
)))

definition *is-polygon-split-path* :: (*real^2*) *list* ⇒ *nat* ⇒ *nat* ⇒ (*real^2*) *list* ⇒ *bool* **where**
is-polygon-split-path vts i j cutvts =
(*i* < *length vts* ∧ *j* < *length vts* ∧ *i* < *j* ∧
(*let vts1* = (*take i vts*) *in*
let vts2 = (*take* (*j* − *i* − *1*) (*drop* (*Suc i*) *vts*)) *in*
let vts3 = *drop* (*j* − *i*) (*drop* (*Suc i*) *vts*) *in*
let x = *vts*!*i in*
let y = *vts*!*j in*
let cutpath = *make-polygonal-path* (*x* # *cutvts* @ [*y*]) *in*
let p = *make-polygonal-path* (*vts*@[*vts*!*0*]) *in*
let p1 = *make-polygonal-path* (*x*#(*vts2* @ [*y*] @ (*rev cutvts*) @ [*x*])) *in*

let p2 = make-polygonal-path (vts1 @ ([x] @ cutvts @ [y]) @ vts3 @ [vts ! 0]) in
let c1 = make-polygonal-path (x#(vts2@[y])) in
let c2 = make-polygonal-path (vts1 @ ([x] @ cutvts @ [y]) @ vts3) in
(is-polygon-cut-path (vts@[vts!0]) cutpath ∧
polygon p ∧ polygon p1 ∧ polygon p2 ∧
path-inside p1 ∩ path-inside p2 = {} ∧
path-inside p1 ∪ path-inside p2 ∪ (path-image cutpath − {x, y}) = path-inside p
∧ ((path-image p1) − (path-image cutpath)) ∩ ((path-image p2) − (path-image cutpath)) = {}
∧ path-image p
= ((path-image p1) − (path-image cutpath)) ∪ ((path-image p2) − (path-image cutpath)) ∪ {x, y}
)))

lemma *polygon-split-add-measure*:
fixes *p p1 p2 :: R-to-R2*
assumes *is-polygon-split vts i j*
assumes *vts1 = (take i vts)*
vts2 = (take (j − i − 1) (drop (Suc i) vts))
vts3 = drop (j − i) (drop (Suc i) vts)
x = vts ! i
y = vts ! j
p = make-polygonal-path (vts@[vts!0])
p1 = make-polygonal-path (x#(vts2@[y, x]))
p2 = make-polygonal-path (vts1 @ [x, y] @ vts3 @ [vts ! 0])
defines *M1 ≡ measure lebesgue (path-inside p1)* **and**
M2 ≡ measure lebesgue (path-inside p2) **and**
M ≡ measure lebesgue (path-inside p)
shows *M1 + M2 = M*
⟨*proof*⟩

lemma *polygonal-paths-measurable*:
shows *path-image (make-polygonal-path vts) ∈ sets lebesgue*
⟨*proof*⟩

lemma *polygonal-path-has-emeasure-0*:
shows *emeasure lebesgue (path-image (make-polygonal-path vts)) = 0*
⟨*proof*⟩

lemma *polygon-split-path-add-measure*:
fixes *p p1 p2 :: R-to-R2*
assumes *is-polygon-split-path vts i j cutvts*
assumes *vts1 = (take i vts)*
vts2 = (take (j − i − 1) (drop (Suc i) vts))
vts3 = drop (j − i) (drop (Suc i) vts)
x = vts ! i
y = vts ! j
p = make-polygonal-path (vts@[vts!0])

p1 = *make-polygonal-path* (*x*#(*vts2* @ [*y*] @ (*rev cutvts*) @ [*x*]))
p2 = *make-polygonal-path* (*vts1* @ ([*x*] @ *cutvts* @ [*y*]) @ *vts3* @ [*vts* ! *0*])
defines *M1* ≡ *measure lebesgue* (*path-inside p1*) **and**
M2 ≡ *measure lebesgue* (*path-inside p2*) **and**
M ≡ *measure lebesgue* (*path-inside p*)
shows *M1* + *M2* = *M*
⟨*proof*⟩

lemma *polygon-cut-path-to-split-path-vtx0*:
fixes *p* :: *R-to-R2*
assumes *polygon-p*: *polygon p* **and**
i-gt: *i* > *0* **and**
i-lt: *i* < *length vts* **and**
p-is: *p* = *make-polygonal-path* (*vts* @ [*vts* ! *0*]) **and**
cutpath: *cutpath* = *make-polygonal-path* ([*vts*!*0*] @ *cutvts* @ [*vts*!*i*]) **and**
have-cut: *is-polygon-cut-path* (*vts* @ [*vts*!*0*]) *cutpath*
shows *is-polygon-split-path vts 0 i cutvts*
⟨*proof*⟩

lemma *polygon-cut-path-to-split-path*:
fixes *p* :: *R-to-R2*
assumes *polygon p*
p = *make-polygonal-path* (*vts* @ [*vts* ! *0*])
is-polygon-cut-path (*vts* @ [*vts*!*0*]) *cutpath*
vts1 ≡ (*take i vts*)
vts2 ≡ (*take* (*j* − *i* − *1*) (*drop* (*Suc i*) *vts*))
vts3 ≡ *drop* (*j* − *i*) (*drop* (*Suc i*) *vts*)
x ≡ *vts* ! *i*
y ≡ *vts* ! *j*
cutpath = *make-polygonal-path* ([*x*] @ *cutvts* @ [*y*])
i < *length vts* ∧ *j* < *length vts* ∧ *i* < *j*
p1 ≡ *make-polygonal-path* (*x*#(*vts2*@([*y*] @ (*rev cutvts*) @ [*x*]))) **and**
p2 ≡ *make-polygonal-path* (*vts1* @ ([*x*] @ *cutvts* @ [*y*]) @ *vts3* @ [(*vts1* @ [*x*]) ! *0*])
shows *is-polygon-split-path vts i j cutvts*
⟨*proof*⟩

lemma *good-polygonal-path-implies-polygon-split-path*:
assumes *polygon p*
assumes *p* = *make-polygonal-path* (*vts* @ [*vts*!*0*])
assumes *good-polygonal-path v1 cutvts v2* (*vts* @ [*vts*!*0*])
assumes *i* < *length vts* ∧ *j* < *length vts*
assumes *vts* ! *i* = *v1*
assumes *vts* ! *j* = *v2*
assumes *i* < *j*
shows *is-polygon-split-path vts i j cutvts*
⟨*proof*⟩

lemma *good-path-iff*:
good-linepath a b vts $\longleftrightarrow$ *good-polygonal-path a* [] *b vts*
⟨*proof*⟩

lemma *polygon-cut-iff*: *is-polygon-cut* (*vts* @ [*vts*!*0*]) (*vts*!*i*) (*vts*!*j*)
$\longleftrightarrow$ *is-polygon-cut-path* (*vts* @ [*vts*!*0*]) (*linepath* (*vts*!*i*) (*vts*!*j*))
⟨*proof*⟩

lemma *polygon-split-iff*: *is-polygon-split vts i j* $\longleftrightarrow$ *is-polygon-split-path vts i j* []
⟨*proof*⟩

lemma *polygon-cut-to-split-vtx0*:
fixes *p* :: *R-to-R2*
assumes *polygon-p*: *polygon p* **and**
i-gt: *i* > *0* **and**
i-lt: *i* < *length vts* **and**
p-is: *p* = *make-polygonal-path* (*vts* @ [*vts* ! *0*]) **and**
have-cut: *is-polygon-cut* (*vts* @ [*vts*!*0*]) (*vts*!*0*) (*vts*!*i*)
shows *is-polygon-split vts 0 i*
⟨*proof*⟩

lemma *polygon-cut-to-split*:
fixes *p* :: *R-to-R2*
assumes *is-polygon-cut* (*vts* @ [*vts*!*0*]) (*vts*!*i*) (*vts*!*j*)
i < *length vts* ∧ *j* < *length vts* ∧ *i* < *j*
shows *is-polygon-split vts i j*
⟨*proof*⟩

lemma *good-linepath-implies-polygon-split*:
assumes *polygon p*
assumes *p* = *make-polygonal-path* (*vts* @ [*vts*!*0*])
assumes *good-linepath v1 v2* (*vts* @ [*vts*!*0*])
assumes *i* < *length vts* ∧ *j* < *length vts*
assumes *vts* ! *i* = *v1*
assumes *vts* ! *j* = *v2*
assumes *i* < *j*
shows *is-polygon-split vts i j*
⟨*proof*⟩

end
theory *Triangle-Lemmas*
imports
Polygon-Convex-Lemmas
Integral-Matrix
Affine-Arithmetic.Floatarith-Expression
HOL−Analysis.Topology-Euclidean-Space
HOL−Analysis.Equivalence-Lebesgue-Henstock-Integration
HOL−Analysis.Inner-Product

HOL−Analysis.Line-Segment
HOL−Analysis.Convex-Euclidean-Space
HOL−Analysis.Change-Of-Vars

begin

20 Triangles

definition *elem-triangle* :: *real^2* ⇒ *real^2* ⇒ *real^2* ⇒ *bool* **where**
elem-triangle a b c ⟷
¬ *collinear* {*a*, *b*, *c*}
∧ *integral-vec a* ∧ *integral-vec b* ∧ *integral-vec c*
∧ {*x*. *x* ∈ *convex hull* {*a*, *b*, *c*} ∧ *integral-vec x*} = {*a*, *b*, *c*}

definition *triangle-mat* :: *real^2* ⇒ *real^2* ⇒ *real^2* ⇒ *real^2^2* **where**
triangle-mat a b c = *transpose* (*vector* [*b* − *a*, *c* − *a*])

definition *triangle-linear* :: *real^2* ⇒ *real^2* ⇒ *real^2* ⇒ (*real^2* ⇒ *real^2*) **where**
triangle-linear a b c = (λ*x*. (*triangle-mat a b c*) ∗*v x*)

definition *triangle-affine* :: *real^2* ⇒ *real^2* ⇒ *real^2* ⇒ (*real^2* ⇒ *real^2*) **where**
triangle-affine a b c = (λ*x*. *a* + (*triangle-mat a b c*) ∗*v x*)

abbreviation *unit-square* ≡
(*convex hull* {*vector* [*0*, *0*], *vector* [*0*, *1*], *vector* [*1*, *1*], *vector* [*1*, *0*]})::((*real^2*) *set*)

abbreviation *unit-triangle* ≡
(*convex hull* {*vector* [*0*, *0*], *vector* [*1*, *0*], *vector* [*0*, *1*]})::((*real^2*) *set*)

abbreviation *unit-triangle′* ≡
(*convex hull* {*vector* [*1*, *1*], *vector* [*1*, *0*], *vector* [*0*, *1*]})::((*real^2*) *set*)

lemma *triangle-inside-is-convex-hull-interior*:
assumes *polygon-of p* [*a*, *b*, *c*, *a*]
shows *path-inside p* = *interior* (*convex hull* {*a*, *b*, *c*})
⟨*proof*⟩

lemma *triangle-is-convex*:
assumes *p* = *make-triangle a b c* **and** ¬ *collinear* {*a*, *b*, *c*}
shows *convex* (*path-inside p*) (**is** *convex ?s*)
⟨*proof*⟩

lemma *affine-comp-linear-trans*: *triangle-affine a b c* = (λ*x*. *x* + *a*) ∘ (*triangle-linear a b c*)
⟨*proof*⟩

lemma *triangle-linear-der*:

fixes *a b c* :: *real^2*
defines *T* ≡ *triangle-linear a b c*
shows (*T has-derivative T*) (*at x*)
⟨*proof*⟩

lemma *triangle-affine-der*:
fixes *a b c* :: *real^2*
assumes *S* ∈ *sets lebesgue* **and** *x* ∈ *S*
defines *A* ≡ *triangle-affine a b c* **and** *T* ≡ *triangle-linear a b c*
shows *x* ∈ *S* ⟹ (*A has-derivative T*) (*at x within S*)
⟨*proof*⟩

lemma *triangle-linear-inj*:
fixes *a b c* :: *real^2*
assumes ¬ *collinear* {*a*, *b*, *c*}
defines *L* ≡ *triangle-linear a b c*
shows *inj L*
⟨*proof*⟩

lemma *triangle-affine-inj*:
fixes *a b c* :: *real^2*
assumes ¬ *collinear* {*a*, *b*, *c*}
defines *A* ≡ *triangle-affine a b c*
shows *inj A*
⟨*proof*⟩

lemma *triangle-linear-integrable*:
fixes *a b c* :: *real^2*
assumes *S* ∈ *lmeasurable*
defines *T* ≡ *triangle-linear a b c*
shows (λ*x*. *abs* (*matrix-det* (*matrix* (*T*)))) *integrable-on S* (**is** (λ*x*. *?c*) *integrable-on S*)
⟨*proof*⟩

lemma *measure-differentiable-image-eq-affine*:
fixes *a b c* :: *real^2*
defines *A* ≡ *triangle-affine a b c* **and** *T* ≡ *triangle-linear a b c*
assumes *S* ∈ *lmeasurable* **and** ¬ *collinear* {*a*, *b*, *c*}
shows *measure lebesgue* (*A* ‘ *S*) = *integral S* (λ*x*. *abs* (*matrix-det* (*matrix T*)))
⟨*proof*⟩

lemma *triangle-affine-img*:
fixes *a b c* :: *real^2*
defines *A* ≡ *triangle-affine a b c*
shows *convex hull* {*a*, *b*, *c*} = *A* ‘ *unit-triangle*
⟨*proof*⟩

lemma *triangle-affine-e1-e2*:
fixes *a b c* :: *real^2*

defines $A \equiv$ *triangle-affine a b c*
shows (*triangle-affine a b c*) (*vector* [*0*, *0*]) = *a*
(*triangle-affine a b c*) (*vector* [*1*, *0*]) = *b*
(*triangle-affine a b c*) (*vector* [*0*, *1*]) = *c*
⟨*proof*⟩

lemma *triangle-measure-integral-of-det*:
fixes *a b c* :: *real^2*
defines $S \equiv$ *convex hull* {*a*, *b*, *c*}
assumes ¬ *collinear* {*a*, *b*, *c*}
shows *measure lebesgue S* =
integral unit-triangle (λ(*x*::*real^2*). *abs* (*matrix-det* (*matrix* (*triangle-linear a b c*))))
⟨*proof*⟩

lemma *triangle-affine-preserves-interior*:
assumes *A* = *triangle-affine a b c* **and** *L* = *triangle-linear a b c*
assumes ¬ *collinear* {*a*, *b*, *c*}
shows *A* ‘ (*interior S*) = *interior* (*A* ‘ *S*)
⟨*proof*⟩

lemma *triangle-affine-preserves-affine-hull*:
assumes *A* = *triangle-affine a b c*
assumes ¬ *collinear* {*a*, *b*, *c*}
shows *A* ‘ (*affine hull S*) = *affine hull* (*A* ‘ *S*)
⟨*proof*⟩

lemma *triangle-measure-convex-hull-measure-path-inside-same*:
assumes *p-triangle*: *p* = *make-triangle a b c*
assumes *elem-triangle*: *elem-triangle a b c*
shows *measure lebesgue* (*convex hull* {*a*, *b*, *c*}) = *measure lebesgue* (*path-inside p*)
(**is** *measure lebesgue ?S* = *measure lebesgue ?I*)
⟨*proof*⟩

lemma *on-triangle-path-image-cases*:
assumes *p* = *make-triangle a b c*
assumes *d* ∈ *path-image p*
shows *d* ∈ *path-image* (*linepath a b*) ∨ *d* ∈ *path-image* (*linepath b c*) ∨ *d* ∈ *path-image* (*linepath c a*)
⟨*proof*⟩

lemma *on-triangle-frontier-cases*:
fixes *a b c* :: *real^2*
assumes ¬ *collinear* {*a*, *b*, *c*}
assumes *d* ∈ *frontier* (*convex hull* {*a*, *b*, *c*})
shows *d* ∈ *path-image* (*linepath a b*) ∨ *d* ∈ *path-image* (*linepath b c*) ∨ *d* ∈ *path-image* (*linepath c a*)
⟨*proof*⟩

lemma *triangle-path-image-subset-convex*:
assumes p = *make-triangle a b c*
shows *path-image* $p \subseteq$ *convex hull* $\{a, b, c\}$
⟨*proof*⟩

lemma *triangle-convex-hull*:
assumes p = *make-triangle a b c* **and** ¬ *collinear* $\{a, b, c\}$
shows *convex hull* $\{a, b, c\}$ = (*path-image p*) ∪ (*path-inside p*)
⟨*proof*⟩

end
theory *Unit-Geometry*
imports
HOL−Analysis.Polytope
Polygon-Jordan-Curve
Triangle-Lemmas

begin

21 Measure Setup

lemma *finite-convex-is-measurable*:
fixes p :: (*real^2*) *set*
assumes p = *convex hull l* **and** *finite l*
shows $p \in$ *sets lebesgue*
⟨*proof*⟩

lemma *unit-square-lebesgue*: *unit-square* ∈ *sets lebesgue*
⟨*proof*⟩

lemma *unit-triangle-lebesgue*: *unit-triangle* ∈ *sets lebesgue*
⟨*proof*⟩

lemma *unit-triangle-lmeasurable*: *unit-triangle* ∈ *lmeasurable*
⟨*proof*⟩

22 Unit Triangle

lemma *unit-triangle-vts-not-collinear*:
¬ *collinear* {(*vector* [0, 0])::*real^2*, *vector* [1, 0], *vector* [0, 1]}
(**is** ¬ *collinear* {*?a*, *?b*, *?c*})
⟨*proof*⟩

lemma *unit-triangle-convex*:
assumes p = (*make-polygonal-path* [*vector* [0, 0], *vector* [1, 0], *vector* [0, 1], *vector*[0, 0]])

(**is** p = *make-polygonal-path* [*?O*, *?e1*, *?e2*, *?O*])
shows *convex* (*path-inside* p)
⟨*proof*⟩

lemma *unit-triangle-char*:
shows *unit-triangle* = {x. $0 \leq x \$ 1 \wedge 0 \leq x \$ 2 \wedge x \$ 1 + x \$ 2 \leq 1$}
(**is** *unit-triangle* = *?S*)
⟨*proof*⟩

lemma *unit-triangle-interior-char*:
shows *interior unit-triangle* = {x. $0 < x \$ 1 \wedge 0 < x \$ 2 \wedge x \$ 1 + x \$ 2 < 1$}
(**is** *interior unit-triangle* = *?S*)
⟨*proof*⟩

lemma *unit-triangle-is-elementary*: *elem-triangle* (*vector* [0, 0]) (*vector* [1, 0]) (*vector* [0, 1])
(**is** *elem-triangle ?a ?b ?c*)
⟨*proof*⟩

lemma *unit-triangles-same-area*:
measure lebesgue unit-triangle' = *measure lebesgue unit-triangle*
⟨*proof*⟩

23 Unit Square

lemma *convex-hull-4*:
convex hull {*a*,*b*,*c*,*d*} = { $u *_R a + v *_R b + w *_R c + t *_R d \mid u\ v\ w\ t.\ 0 \leq u \wedge 0 \leq v \wedge 0 \leq w \wedge 0 \leq t \wedge u + v + w + t = 1$}
⟨*proof*⟩

lemma *unit-square-characterization-helper*:
fixes $a\ b$ :: *real*
assumes $0 \leq a \wedge a \leq 1 \wedge 0 \leq b \wedge b \leq 1$ **and**
$a \leq b$
obtains $u\ v\ w\ t$ **where**
vector [*a*, *b*] = $u *_R$ ((*vector* [0, 0])::*real^2*)
$+ v *_R$ (*vector* [0, 1])
$+ w *_R$ (*vector* [1, 1])
$+ t *_R$ (*vector* [1, 0])
$\wedge\ 0 \leq u \wedge 0 \leq v \wedge 0 \leq w \wedge 0 \leq t \wedge u + v + w + t = 1$
⟨*proof*⟩

lemma *unit-square-characterization*:
unit-square = {x. $0 \leq x\$1 \wedge x\$1 \leq 1 \wedge 0 \leq x\$2 \wedge x\$2 \leq 1$} (**is** *unit-square* = *?S*)
⟨*proof*⟩

lemma *e1e2-basis*:

defines *e1* ≡ (*vector* [*1*, *0*])::(*real*^*2*) **and**
e2 ≡ (*vector* [*0*, *1*])::(*real*^*2*)
shows *e1* = *axis 1* (*1*::*real*) **and** *e1* ∈ (*Basis*::((*real*^*2*) *set*)) **and**
e2 = *axis 2* (*1*::*real*) **and** *e2* ∈ (*Basis*::((*real*^*2*) *set*))
⟨*proof*⟩

lemma *unit-square-cbox*: *unit-square* = *cbox* (*vector* [*0*, *0*]) (*vector* [*1*, *1*])
⟨*proof*⟩

lemma *unit-square-area*: *measure lebesgue unit-square* = *1*
⟨*proof*⟩

24 Unit Triangle Area is 1/2

lemma *unit-triangle′-char*:
shows *unit-triangle′* = {*x*. *x* $ *1* ≤ *1* ∧ *x* $ *2* ≤ *1* ∧ *x* $ *1* + *x* $ *2* ≥ *1*}
⟨*proof*⟩

lemma *unit-square-split-diag*:
shows *unit-square* = *unit-triangle* ∪ *unit-triangle′*
⟨*proof*⟩

lemma *unit-triangle-INT-unit-triangle′-measure*:
measure lebesgue (*unit-triangle* ∩ *unit-triangle′*) = *0*
⟨*proof*⟩

lemma *unit-triangle-area*: *measure lebesgue unit-triangle* = *1/2*
⟨*proof*⟩

end
theory *Elementary-Triangle-Area*
imports
Unit-Geometry

begin

25 Area of Elementary Triangle is 1/2

lemma *nonint-in-square-img-IMP-nonint-triangle-img*:
assumes *A* = *triangle-affine a b c*
assumes *x* ∈ *unit-square*
assumes ¬ *integral-vec x*
assumes *integral-vec* (*A x*)
assumes *elem-triangle a b c*
obtains *x′* **where** *x′* ∈ *unit-triangle* ∧ ¬ *integral-vec x′* ∧ *integral-vec* (*A x′*)
⟨*proof*⟩

lemma *elem-triangle-integral-mat-bij*:

fixes *a b c* :: *real^2*
assumes *elem-triangle a b c*
defines *L* ≡ *triangle-mat a b c*
shows *integral-mat-bij L*
⟨*proof*⟩

lemma *elem-triangle-measure-integral-of-1*:
fixes *a b c* :: *real^2*
defines *S* ≡ *convex hull* {*a*, *b*, *c*}
assumes *elem-triangle a b c*
shows *measure lebesgue S* = *integral unit-triangle* (λ(*x*::*real^2*). *1*)
⟨*proof*⟩

lemma *elem-triangle-area-is-half*:
fixes *a b c* :: *real^2*
assumes *elem-triangle a b c*
defines *S* ≡ *convex hull* {*a*, *b*, *c*}
shows *measure lebesgue S* = *1/2* (**is** *?S-area* = *1/2*)
⟨*proof*⟩

end
theory *Pick*
imports
Polygon-Splitting
Elementary-Triangle-Area
begin

26 Setup

26.1 Integral Points Cardinality Properties

lemma *bounded-finite*:
fixes *A*:: (*real^2*) *set*
assumes *bounded A*
shows *finite* {*x*::(*real^2*). *integral-vec x* ∧ *x* ∈ *A*} (**is** *finite ?A-int*)
⟨*proof*⟩

lemma *finite-path-image*:
assumes *polygon p*
shows *finite* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p*}
⟨*proof*⟩

lemma *finite-path-inside*:
assumes *polygon p*
shows *finite* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p*}
⟨*proof*⟩

lemma *bounded-finite-inside*:
fixes *B*:: (*real^2*) *set*

assumes *simple-path p*
shows *bounded* (*path-inside p*)
⟨*proof*⟩

lemma *finite-integral-points-path-image*:
assumes *simple-path p*
shows *finite* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p*}
⟨*proof*⟩

lemma *finite-integral-points-path-inside*:
assumes *simple-path p*
shows *finite* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p*}
⟨*proof*⟩

27 Pick splitting

lemma *pick-split-path-union-main*:
assumes *is-split*: *is-polygon-split-path vts i j cutvts*
assumes *vts1* = (*take i vts*)
assumes *vts2* = (*take* (*j* − *i* − *1*) (*drop* (*Suc i*) *vts*))
assumes *vts3* = *drop* (*j* − *i*) (*drop* (*Suc i*) *vts*)
assumes *x* = *vts*!*i*
assumes *y* = *vts*!*j*
assumes *cutpath* = *make-polygonal-path* (*x* # *cutvts* @ [*y*])
assumes *p*: *p* = *make-polygonal-path* (*vts*@[*vts*!*0*]) (**is** *p* = *make-polygonal-path ?p-vts*)
assumes *p1*: *p1* = *make-polygonal-path* (*x*#(*vts2* @ [*y*] @ (*rev cutvts*) @ [*x*])) (**is** *p1* = *make-polygonal-path ?p1-vts*)
assumes *p2*: *p2* = *make-polygonal-path* (*vts1* @ ([*x*] @ *cutvts* @ [*y*]) @ *vts3* @ [*vts* ! *0*]) (**is** *p2* = *make-polygonal-path ?p2-vts*)
assumes *I1*: *I1* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p1*}
assumes *B1*: *B1* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p1*}
assumes *I2*: *I2* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p2*}
assumes *B2*: *B2* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p2*}
assumes *I*: *I* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p*}
assumes *B*: *B* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p*}
assumes *all-integral-vts*: *all-integral vts*
shows *measure lebesgue* (*path-inside p1*) = *I1* + *B1*/*2* − *1*
⟹ *measure lebesgue* (*path-inside p2*) = *I2* + *B2*/*2* − *1*
⟹ *measure lebesgue* (*path-inside p*) = *I* + *B*/*2* − *1*
measure lebesgue (*path-inside p*) = *I* + *B*/*2* − *1*
⟹ *measure lebesgue* (*path-inside p2*) = *I2* + *B2*/*2* − *1*
⟹ *measure lebesgue* (*path-inside p1*) = *I1* + *B1*/*2* − *1*
measure lebesgue (*path-inside p*) = *I* + *B*/*2* − *1*
⟹ *measure lebesgue* (*path-inside p1*) = *I1* + *B1*/*2* − *1*
⟹ *measure lebesgue* (*path-inside p2*) = *I2* + *B2*/*2* − *1*
⟨*proof*⟩

lemma *pick-split-union*:

assumes *is-split*: *is-polygon-split vts i j*
assumes *vts1* = (*take i vts*)
assumes *vts2* = (*take* (*j* − *i* − *1*) (*drop* (*Suc i*) *vts*))
assumes *vts3* = *drop* (*j* − *i*) (*drop* (*Suc i*) *vts*)
assumes *x* = *vts* ! *i*
assumes *y* = *vts* ! *j*
assumes *p*: *p* = *make-polygonal-path* (*vts*@[*vts*!*0*]) (**is** *p* = *make-polygonal-path ?p-vts*)
assumes *p1*: *p1* = *make-polygonal-path* (*x*#(*vts2*@[*y*, *x*])) (**is** *p1* = *make-polygonal-path ?p1-vts*)
assumes *p2*: *p2* = *make-polygonal-path* (*vts1* @ [*x*, *y*] @ *vts3* @ [*vts* ! *0*]) (**is** *p2* = *make-polygonal-path ?p2-vts*)
assumes *I1*: *I1* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p1*}
assumes *B1*: *B1* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p1*}
assumes *pick1*: *measure lebesgue* (*path-inside p1*) = *I1* + *B1*/*2* − *1*
assumes *I2*: *I2* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p2*}
assumes *B2*: *B2* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p2*}
assumes *pick2*: *measure lebesgue* (*path-inside p2*) = *I2* + *B2*/*2* − *1*
assumes *I*: *I* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p*}
assumes *B*: *B* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p*}
assumes *all-integral-vts*: *all-integral vts*
shows *measure lebesgue* (*path-inside p*) = *I* + *B*/*2* − *1*
measure lebesgue (*path-inside p*) = *measure lebesgue* (*path-inside p1*) + *measure lebesgue* (*path-inside p2*)
⟨*proof*⟩

lemma *pick-split-path-union*:
assumes *is-split*: *is-polygon-split-path vts i j cutvts*
assumes *vts1* = (*take i vts*)
assumes *vts2* = (*take* (*j* − *i* − *1*) (*drop* (*Suc i*) *vts*))
assumes *vts3* = *drop* (*j* − *i*) (*drop* (*Suc i*) *vts*)
assumes *x* = *vts*!*i*
assumes *y* = *vts*!*j*
assumes *cutpath* = *make-polygonal-path* (*x* # *cutvts* @ [*y*])
assumes *p*: *p* = *make-polygonal-path* (*vts*@[*vts*!*0*]) (**is** *p* = *make-polygonal-path ?p-vts*)
assumes *p1*: *p1* = *make-polygonal-path* (*x*#(*vts2* @ [*y*] @ (*rev cutvts*) @ [*x*])) (**is** *p1* = *make-polygonal-path ?p1-vts*)
assumes *p2*: *p2* = *make-polygonal-path* (*vts1* @ ([*x*] @ *cutvts* @ [*y*]) @ *vts3* @ [*vts* ! *0*]) (**is** *p2* = *make-polygonal-path ?p2-vts*)
assumes *I1*: *I1* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p1*}
assumes *B1*: *B1* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p1*}
assumes *pick1*: *measure lebesgue* (*path-inside p1*) = *I1* + *B1*/*2* − *1*
assumes *I2*: *I2* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p2*}
assumes *B2*: *B2* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p2*}
assumes *pick2*: *measure lebesgue* (*path-inside p2*) = *I2* + *B2*/*2* − *1*
assumes *I*: *I* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p*}
assumes *B*: *B* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p*}
assumes *all-integral-vts*: *all-integral vts*

shows *measure lebesgue* (*path-inside p*) = *I* + *B*/*2* − *1*
⟨*proof*⟩

lemma *pick-triangle-basic-split*:
assumes *p* = *make-triangle a b c* **and** *distinct* [*a*, *b*, *c*] **and** ¬ *collinear* {*a*, *b*, *c*} **and**
d-prop: *d* ∈ *path-image* (*linepath a b*) ∧ *d* ∉ {*a*, *b*, *c*}
shows *good-linepath c d* [*a*, *d*, *b*, *c*, *a*]
∧ *path-image* (*make-polygonal-path* [*a*, *d*, *b*, *c*, *a*]) = *path-image p*
⟨*proof*⟩

28 Convex Hull Has Good Linepath

lemma *leq-2-extreme-points-means-collinear*:
fixes *vts* :: *'a*::*euclidean-space set*
assumes *finite vts*
assumes *card* {*v*. *v extreme-point-of* (*convex hull vts*)} ≤ *2*
shows *collinear vts*
⟨*proof*⟩

lemma *convex-hull-non-extreme-point-in-open-seg*:
assumes *H* = *convex hull vts*
assumes *x* ∈ *H* − {*v*. *v extreme-point-of H*}
shows ∃ *a b*. *a* ∈ *H* ∧ *b* ∈ *H* ∧ *x* ∈ *open-segment a b*
⟨*proof*⟩

lemma *convex-hull-extreme-points-vertex-split*:
fixes *vts* :: (*real^2*) *set*
assumes *H* = *convex hull vts*
assumes *finite vts*
assumes *card* {*v*. *v extreme-point-of H*} ≥ *4*
assumes {*a*, *b*, *c*} ⊆ {*v*. *v extreme-point-of H*} ∧ *distinct* [*a*, *b*, *c*]
shows *path-image* (*linepath a b*) ∩ *interior H* ≠ {}
∨ *path-image* (*linepath b c*) ∩ *interior H* ≠ {}
∨ *path-image* (*linepath c a*) ∩ *interior H* ≠ {}
⟨*proof*⟩

lemma *convex-hull-has-vertex-split-helper-wlog*:
assumes *p* = *make-triangle a b c* **and** *distinct* [*a*, *b*, *c*] **and** ¬ *collinear* {*a*, *b*, *c*} **and**
d-prop: *d* ∈ *path-image* (*linepath a b*) ∧ *d* ∉ {*a*, *b*, *c*}
shows *path-image* (*linepath c d*) ∩ *path-inside p* ≠ {}
⟨*proof*⟩

lemma *convex-hull-has-vertex-split-helper*:
assumes *p* = *make-triangle a b c* **and** *distinct* [*a*, *b*, *c*] **and** ¬ *collinear* {*a*, *b*, *c*} **and**
d-prop: *d* ∈ *path-image p* ∧ *d* ∉ {*a*, *b*, *c*}
shows ∃ *x y*. {*x*, *y*} ⊆ {*a*, *b*, *c*, *d*} ∧ *x* ≠ *y* ∧ *path-image* (*linepath x y*) ∩

path-inside p $\neq$ {}
⟨*proof*⟩

lemma *convex-hull-has-vertex-split*:
fixes *vts* :: (*real^2*) *set*
assumes *H* = *convex hull vts*
assumes ¬ *collinear vts*
assumes *card vts* > *3*
assumes *finite vts*
shows ∃ *a b*. {*a*, *b*} ⊆ *vts* ∧ *a* ≠ *b* ∧ *path-image* (*linepath a b*) ∩ *interior H* ≠ {}
⟨*proof*⟩

lemma *convex-polygon-has-good-linepath-helper*:
assumes *polygon-of p vts*
assumes *convex* (*path-inside p* ∪ *path-image p*)
assumes *card* (*set vts*) > *3*
obtains *a b* **where** {*a*, *b*} ⊆ *set vts* ∧ *a* ≠ *b* ∧ ¬ *path-image* (*linepath a b*) ⊆ *path-image p*
⟨*proof*⟩

lemma *convex-polygon-has-good-linepath*:
assumes *convex* (*path-inside p* ∪ *path-image p*)
assumes *polygon p*
assumes *p* = *make-polygonal-path vts*
assumes *card* (*set vts*) > *3*
shows ∃ *a b*. *good-linepath a b vts*
⟨*proof*⟩

29 Pick's Theorem

definition *integral-inside*:
integral-inside p = {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p*}

definition *integral-boundary*:
integral-boundary p = {*x*. *integral-vec x* ∧ *x* ∈ *path-image p*}

29.1 Pick's Theorem Triangle Case

definition *pick-triangle*:
pick-triangle p a b c ⟷
p = *make-triangle a b c*
∧ *all-integral* [*a*, *b*, *c*]
∧ *distinct* [*a*, *b*, *c*]
∧ ¬ *collinear* {*a*, *b*, *c*}

definition *pick-holds*:
pick-holds p ⟷
(*let I* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p*} *in*

let B = *card* $\{x.$ *integral-vec* $x \wedge x \in$ *path-image* $p\}$ *in*
measure lebesgue (*path-inside* p) = $I + B/2 - 1$)

lemma *pick-triangle-wlog-helper*:
assumes *pick-triangle* p a b c **and**
I = *card* (*integral-inside* p) **and**
B = *card* (*integral-boundary* p) **and**
integral-inside $p = \{\}$ **and**
integral-vec $d \wedge d \in$ *path-image* (*linepath* a b) $\wedge$ $d \notin \{a, b, c\}$ **and** $d \notin \{a, b, c\}$ **and**
ih: $\bigwedge p'$ a' b' c'. (*card* (*integral-inside* p') + *card* (*integral-boundary* p') < $I + B$) $\Longrightarrow$ *pick-triangle* p' a' b' $c' \Longrightarrow$ *pick-holds* p'
shows *measure lebesgue* (*path-inside* p) = $I + B/2 - 1$
⟨*proof*⟩

lemma *pick-triangle-helper*:
assumes *pick-triangle* p a b c **and**
I = *card* (*integral-inside* p) **and**
B = *card* (*integral-boundary* p) **and**
integral-inside $p = \{\}$ **and**
integral-vec $d \wedge d \notin \{a, b, c\}$ **and** $d \notin \{a, b, c\}$ **and**
$d \in$ *path-image* (*linepath* a b)
$\vee$ $d \in$ *path-image* (*linepath* b c)
$\vee$ $d \in$ *path-image* (*linepath* c a) **and**
ih: $\bigwedge p'$ a' b' c'. (*card* (*integral-inside* p') + *card* (*integral-boundary* p') < $I + B$) $\Longrightarrow$ *pick-triangle* p' a' b' $c' \Longrightarrow$ *pick-holds* p'
shows *measure lebesgue* (*path-inside* p) = $I + B/2 - 1$
⟨*proof*⟩

lemma *triangle-3-split-helper*:
fixes a b :: *'a*::*euclidean-space*
assumes $a \in$ *frontier* S
assumes $b \in$ *interior* S
assumes *convex* S
assumes *closed* S
shows *path-image* (*linepath* a b) $\cap$ *frontier* $S = \{a\}$
⟨*proof*⟩

lemma *unit-triangle-interior-point-not-collinear-e1-e2*:
assumes p = *make-triangle* (*vector* [0, 0]) (*vector* [1, 0]) (*vector* [0, 1])
(**is** p = *make-triangle ?O ?e1 ?e2*)
assumes $z \in$ *path-inside* p
shows $\neg$ *collinear* $\{?O, ?e1, z\}$
⟨*proof*⟩

lemma *triangle-interior-point-not-collinear-vertices-wlog-helper*:
assumes p = *make-triangle* a b c
assumes *polygon* p
assumes $z \in$ *path-inside* p

shows ¬ *collinear* {*a*, *b*, *z*}
⟨*proof*⟩

lemma *triangle-interior-point-not-collinear-vertices*:
assumes *p* = *make-triangle a b c*
assumes *polygon p*
assumes *z* ∈ *path-inside p*
shows ¬ *collinear* {*a*, *b*, *z*} ∧ ¬ *collinear* {*a*, *c*, *z*} ∧ ¬ *collinear* {*b*, *c*, *z*}
⟨*proof*⟩

lemma *triangle-3-split*:
assumes *p* = *make-triangle a b c*
assumes *polygon p*
assumes *z* ∈ *path-inside p*
shows *is-polygon-split-path* [*a*, *b*, *c*] *0 1* [*z*]
is-polygon-split [*a*, *z*, *b*, *c*] *1 3*
a ∉ *path-image* (*make-triangle z b c*) ∪ *path-inside* (*make-triangle z b c*)
b ∉ *path-image* (*make-triangle a z c*) ∪ *path-inside* (*make-triangle a z c*)
c ∉ *path-image* (*make-triangle a b z*) ∪ *path-inside* (*make-triangle a b z*)
⟨*proof*⟩

lemma *smaller-triangle*:
assumes ¬ *collinear* {*a*, *b*, *c*} ∧ ¬ *collinear* {*a'*, *b'*, *c'*}
assumes *p* = *make-triangle a b c*
assumes *p'* = *make-triangle a' b' c'*
assumes *path-inside p* ⊆ *path-inside p'*
assumes ∃ *d*. *integral-vec d* ∧ *d* ∈ *path-image p'* ∪ *path-inside p'* ∧ *d* ∉ *path-image p* ∪ *path-inside p*
shows *card* (*integral-inside p*) + *card* (*integral-boundary p*) < *card* (*integral-inside p'*) + *card* (*integral-boundary p'*)
⟨*proof*⟩

lemma *pick-elem-triangle*:
fixes *p* :: *R-to-R2*
assumes *p-triangle*: *p* = *make-triangle a b c*
assumes *elem-triangle*: *elem-triangle a b c*
assumes *I* = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p*} **and**
B = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p*}
shows *measure lebesgue* (*path-inside p*) = *I* + *B*/*2* − *1*
⟨*proof*⟩

lemma *pick-triangle-lemma*:
fixes *p* :: *R-to-R2*
assumes *p* = *make-triangle a b c* **and** *all-integral* [*a*, *b*, *c*] **and** *distinct* [*a*, *b*, *c*] **and** ¬ *collinear* {*a*, *b*, *c*}
I = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-inside p*} **and**
B = *card* {*x*. *integral-vec x* ∧ *x* ∈ *path-image p*}
shows *measure lebesgue* (*path-inside p*) = *I* + *B*/*2* − *1*

⟨*proof*⟩

29.2 Pocket properties

definition *index-not-in-set* :: (*real^2*) *list* ⇒ (*real^2*) *set* ⇒ *nat* ⇒ *bool*
where *index-not-in-set vts A i* ⟷ *i* ∈ {*i*. *i* < *length vts* ∧ *vts* ! *i* ∉ *A*}

definition *min-index-not-in-set*:: (*real^2*) *list* ⇒ (*real^2*) *set* ⇒ *nat*
where *min-index-not-in-set vts A* = (*LEAST i*. *index-not-in-set vts A i*)

definition *nonzero-index-in-set* :: (*real^2*) *list* ⇒ (*real^2*) *set* ⇒ *nat* ⇒ *bool*
where
nonzero-index-in-set vts A i ⟷ *i* ∈ {*i*. *0* < *i* ∧ *i* < *length vts* ∧ *vts* ! *i* ∈ *A*}

definition *min-nonzero-index-in-set* :: (*real^2*) *list* ⇒ (*real^2*) *set* ⇒ *nat* **where**
min-nonzero-index-in-set vts A = (*LEAST i*. *nonzero-index-in-set vts A i*)

definition *construct-pocket-0* :: (*real^2*) *list* ⇒ (*real^2*) *set* ⇒ (*real^2*) *list* **where**
construct-pocket-0 vts A = *take* ((*min-nonzero-index-in-set vts A*) + *1*) *vts*

definition *is-pocket-0* :: (*real^2*) *list* ⇒ (*real^2*) *list* ⇒ *bool* **where**
is-pocket-0 vts vts′ ⟷
polygon (*make-polygonal-path vts*)
∧ (∃ *i*. *vts′* = *take i vts*)
∧ *3* ≤ *length vts′* ∧ *length vts′* < *length vts*
∧ *hd vts′* ∈ *frontier* (*convex hull* (*set vts*)) ∧ *last vts′* ∈ *frontier* (*convex hull* (*set vts*))
∧ *set* (*tl* (*butlast vts′*)) ⊆ *interior* (*convex hull* (*set vts*))

definition *fill-pocket-0* :: (*real^2*) *list* ⇒ *nat* ⇒ (*real^2*) *list* **where**
fill-pocket-0 vts i = (*hd vts*) # (*drop* (*i−1*) *vts*)

lemma *min-nonzero-index-in-set-exists*:
assumes *set* (*tl vts*) ∩ *A* ≠ {}
shows ∃ *i*. *nonzero-index-in-set vts A i*
⟨*proof*⟩

lemma *min-nonzero-index-in-set-defined*:
assumes *set* (*tl vts*) ∩ *A* ≠ {}
defines *i* ≡ *min-nonzero-index-in-set vts A*
shows *nonzero-index-in-set vts A i* ∧ (∀ *j* < *i*. ¬ *nonzero-index-in-set vts A j*)
⟨*proof*⟩

lemma *min-index-not-in-set-exists*:
assumes *set vts* ⊃ *A*
shows ∃ *i*. *index-not-in-set vts A i*
⟨*proof*⟩

lemma *min-index-not-in-set-defined*:
assumes *set vts* $\supset$ *A*
defines *i* $\equiv$ *min-index-not-in-set vts A*
shows *index-not-in-set vts A i* $\wedge$ ($\forall$ *j* $<$ *i*. $\neg$ *index-not-in-set vts A j*)
⟨*proof*⟩

lemma *min-nonzero-index-in-set-bound*:
assumes *set* (*tl vts*) $\cap$ *A* $\neq$ {}
shows *min-nonzero-index-in-set vts A* $<$ *length vts*
⟨*proof*⟩

lemma *construct-pocket-0-subset-vts*:
assumes *set* (*tl vts*) $\cap$ *A* $\neq$ {}
shows *set* (*construct-pocket-0 vts A*) $\subseteq$ *set vts*
⟨*proof*⟩

lemma *min-index-not-in-set-0*:
assumes *set vts* $\supset$ *A*
assumes *vts*!*0* $\in$ *A*
defines *i* $\equiv$ *min-index-not-in-set vts A*
defines *r* $\equiv$ *i* $-$ *1*
shows *vts*!*r* $\in$ *A*
⟨*proof*⟩

lemma *construct-pocket-0-last-in-set*:
assumes *set* (*tl vts*) $\cap$ *A* $\neq$ {}
assumes *vts*!*0* $\in$ *A*
defines *p* $\equiv$ *construct-pocket-0 vts A*
shows *last p* $\in$ *A*
⟨*proof*⟩

lemma *construct-pocket-0-first-last-distinct*:
assumes *card A* $\geq$ *2*
assumes *A* $\subseteq$ *set vts*
assumes *distinct* (*butlast vts*)
assumes *hd vts* $=$ *last vts*
shows *hd* (*construct-pocket-0 vts A*) $\neq$ *last* (*construct-pocket-0 vts A*)
⟨*proof*⟩

lemma *construct-pocket-is-pocket*:
assumes *polygon* (*make-polygonal-path vts*)
assumes *vts*!*0* $\in$ *frontier* (*convex hull* (*set vts*))
assumes *vts*!*1* $\notin$ *frontier* (*convex hull* (*set vts*))
shows *is-pocket-0 vts* (*construct-pocket-0 vts* (*set vts* $\cap$ *frontier* (*convex hull* (*set vts*))))
⟨*proof*⟩

lemma *exists-point-above-interior*:
fixes $a :: real\hat{}2$
assumes $a \in interior\ (convex\ hull\ S)$
obtains x **where** $x \in S \wedge x\$2 > a\$2$
⟨*proof*⟩

lemma *exists-point-above-convex-hull-interior*:
fixes $S :: (real\hat{}2)\ set$
assumes $S \neq \{\}$
assumes *compact S*
obtains x **where** $x \in S - (interior\ (convex\ hull\ S)) \wedge (\forall y \in interior\ (convex\ hull\ S).\ x\$2 > y\$2)$
⟨*proof*⟩

lemma *flip-function*:
defines $M \equiv (vector\ [vector\ [1,\ 0],\ vector\ [0,\ -1]])::(real\hat{}2\hat{}2)$
defines $f \equiv \lambda v.\ M *v v$
defines $g \equiv (\lambda v.\ vector\ [v\$1,\ -v\$2])::(real\hat{}2 \Rightarrow real\hat{}2)$
shows $inj\ f\ f = g$
⟨*proof*⟩

lemma *exists-point-below-convex-hull-interior*:
fixes $S :: (real\hat{}2)\ set$
assumes $S \neq \{\}$
assumes *compact S*
obtains x **where** $x \in S - (interior\ (convex\ hull\ S)) \wedge (\forall y \in interior\ (convex\ hull\ S).\ x\$2 < y\$2)$
⟨*proof*⟩

lemma *exists-point-above-all*:
fixes $p\ q :: R\text{-}to\text{-}R2$
defines $H \equiv convex\ hull\ (path\text{-}image\ p \cup path\text{-}image\ q)$
assumes $path\ p \wedge path\ q$
assumes $p`\{0<..<1\} \subseteq interior\ H$
assumes $(p\ 0)\$2 = 0 \wedge (p\ 1)\$2 = 0$
assumes $\exists x \in p`\{0<..<1\}.\ x\$2 \geq 0$
obtains x **where** $x \in path\text{-}image\ q \wedge (\forall y \in path\text{-}image\ p.\ x\$2 > y\$2)$
⟨*proof*⟩

lemma *exists-point-below-all*:
fixes $p\ q :: R\text{-}to\text{-}R2$
defines $H \equiv convex\ hull\ (path\text{-}image\ p \cup path\text{-}image\ q)$
assumes $path\ p \wedge path\ q$
assumes $p`\{0<..<1\} \subseteq interior\ H$
assumes $(p\ 0)\$2 = 0 \wedge (p\ 1)\$2 = 0$
assumes $\exists x \in path\text{-}image\ p \cup path\text{-}image\ q.\ x\$2 < 0$
obtains x **where** $x \in path\text{-}image\ q \wedge (\forall y \in path\text{-}image\ p.\ x\$2 < y\$2)$
⟨*proof*⟩

lemma *pocket-fill-line-int-aux*:
fixes $x\ y\ z :: real\hat{}2$
defines $a \equiv y\$1$
assumes $x = 0$
assumes $a > 0 \wedge y\$2 = 0$
assumes $z\$1 < 0 \vee z\$1 > a$
assumes $z\$2 = 0$
assumes $convex\ A \wedge compact\ A$
assumes $\{x, y, z\} \subseteq A$
assumes $\{x, y\} \subseteq frontier\ A$
shows $z \in frontier\ A \wedge closed\text{-}segment\ x\ y \subseteq frontier\ A$
⟨*proof*⟩

lemma *axis-dist*:
fixes $a\ b :: real\hat{}2$
shows $a\$2 = b\$2 \Longrightarrow dist\ a\ b = dist\ (a\$1)\ (b\$1)\ a\$1 = b\$1 \Longrightarrow dist\ a\ b = dist\ (a\$2)\ (b\$2)$
⟨*proof*⟩

lemma *dist-bound-1*:
fixes $a\ b\ x :: real\hat{}2$
assumes $a\$2 = x\$2$
assumes $b \in ball\ x\ \varepsilon$
assumes $\varepsilon < dist\ a\ x$
shows $a\$1 < x\$1 \Longrightarrow b\$1 > a\$1\ a\$1 > x\$1 \Longrightarrow b\$1 < a\$1$
⟨*proof*⟩

lemma *dist-bound-2*:
fixes $a\ b\ x :: real\hat{}2$
assumes $a\$1 = x\$1$
assumes $b \in ball\ x\ \varepsilon$
assumes $\varepsilon < dist\ a\ x$
shows $a\$2 < x\$2 \Longrightarrow b\$2 > a\$2\ a\$2 > x\$2 \Longrightarrow b\$2 < a\$2$
⟨*proof*⟩

lemma *linepath-bound-1*:
fixes $x\ y :: real\hat{}2$
shows $a < x\$1 \wedge a < y\$1 \Longrightarrow \forall v \in path\text{-}image\ (linepath\ x\ y).\ a < v\$1$
$x\$1 < b \wedge y\$1 < b \Longrightarrow \forall v \in path\text{-}image\ (linepath\ x\ y).\ v\$1 < b$
⟨*proof*⟩

lemma *linepath-bound-2*:
fixes $x\ y :: real\hat{}2$
shows $a < x\$2 \wedge a < y\$2 \Longrightarrow \forall v \in path\text{-}image\ (linepath\ x\ y).\ a < v\$2$
$x\$2 < b \wedge y\$2 < b \Longrightarrow \forall v \in path\text{-}image\ (linepath\ x\ y).\ v\$2 < b$
⟨*proof*⟩

lemma *linepath-int-corner*:
fixes $x\ y\ z :: real\hat{}2$

assumes $x\$2 \neq y\$2$
assumes $y\$2 = z\$2$
shows *path-image* (*linepath x y*) ∩ *path-image* (*linepath y z*) = {*y*}
(**is** *path-image ?l1* ∩ *path-image ?l2* = {*y*})
⟨*proof*⟩

lemma *linepath-int-vertical*:
fixes *w x y z* :: *real^2*
assumes $w\$1 \neq y\$1$
assumes $w\$1 = x\$1$
assumes $y\$1 = z\$1$
shows *path-image* (*linepath w x*) ∩ *path-image* (*linepath y z*) = {}
⟨*proof*⟩

lemma *linepath-int-horizontal*:
fixes *w x y z* :: *real^2*
assumes $w\$2 \neq y\$2$
assumes $w\$2 = x\$2$
assumes $y\$2 = z\$2$
shows *path-image* (*linepath w x*) ∩ *path-image* (*linepath y z*) = {}
⟨*proof*⟩

lemma *linepath-int-columns*:
fixes *w x y z* :: *real^2*
assumes $w\$1 < y\$1 \wedge w\$1 < z\$1$
assumes $x\$1 < y\$1 \wedge x\$1 < z\$1$
shows *path-image* (*linepath w x*) ∩ *path-image* (*linepath y z*) = {}
(**is** *path-image ?l1* ∩ *path-image ?l2* = {})
⟨*proof*⟩

lemma *linepath-int-rows*:
fixes *w x y z* :: *real^2*
assumes $w\$2 < y\$2 \wedge w\$2 < z\$2$
assumes $x\$2 < y\$2 \wedge x\$2 < z\$2$
shows *path-image* (*linepath w x*) ∩ *path-image* (*linepath y z*) = {}
(**is** *path-image ?l1* ∩ *path-image ?l2* = {})
⟨*proof*⟩

lemma *horizontal-segment-at-0*:
assumes *a* > *0*
shows *closed-segment* ((*vector* [*0*, *0*])::(*real^2*)) (*vector* [*a*, *0*]) = {*x*. $x\$2 = 0$
$\wedge$ $x\$1 \in \{0..a\}$}
(**is** *?l* = *?s*)
⟨*proof*⟩

lemma *horizontal-segment-at-0'*:
fixes *x y* :: *real^2*
assumes *a* > *0*
assumes $x\$1 = 0 \wedge x\$2 = 0 \wedge y\$1 = a \wedge y\$2 = 0$

shows *closed-segment x y* = {*x*. *x*$2 = 0 ∧ *x*$1 ∈ {0..*a*}}
⟨*proof*⟩

lemma *pocket-fill-line-int-aux1*:
fixes *p q* :: *R-to-R2*
defines *p0* ≡ *pathstart p*
defines *p1* ≡ *pathfinish p*
defines *q0* ≡ *pathstart q*
defines *q1* ≡ *pathfinish q*
defines *a* ≡ *p1*$*1*
defines *l* ≡ *closed-segment p0 p1*
assumes *simple-path p*
assumes *simple-path q*
assumes *p0*$*1* = 0 ∧ *p0*$*2* = 0 ∧ *p1*$*2* = 0
assumes *a* > 0
assumes *path-image q* ∩ {*x*. *x*$*2* = 0} ⊆ *l*
assumes *path-image p* ∩ {*x*. *x*$*2* = 0} ⊆ *l*
assumes ∀ *v* ∈ *path-image p*. *q0*$*2* ≤ *v*$*2*
assumes ∀ *v* ∈ *path-image p*. *q1*$*2* > *v*$*2*
shows *path-image p* ∩ *path-image q* ≠ {}
⟨*proof*⟩

lemma *pocket-fill-line-int-aux2*:
fixes *p q* :: *R-to-R2*
fixes *A* :: (*real*^*2*) *set*
defines *p0* ≡ *pathstart p*
defines *p1* ≡ *pathfinish p*
defines *a* ≡ *p1*$*1*
defines *l* ≡ *closed-segment p0 p1*
assumes *simple-path p*
assumes *p0*$*1* = 0 ∧ *p0*$*2* = 0 ∧ *p1*$*2* = 0
assumes *a* > 0
assumes *convex A* ∧ *compact A*
assumes {*p0*, *p1*} ⊆ *frontier A*
assumes *p* ‘ {0<..<1} ⊆ *interior A*
shows *path-image p* ∩ {*x*. *x*$*2* = 0} ⊆ *l*
⟨*proof*⟩

lemma *three-points-on-line*:
fixes *a b* :: *'a*::*real-vector*
assumes *A* = *affine hull* {*a*, *b*}
assumes *a* ≠ *b*
assumes {*x*, *y*, *z*} ⊆ *A*
assumes *x* ≠ *y* ∧ *y* ≠ *z* ∧ *x* ≠ *z*
shows *x* ∈ *open-segment y z* ∨ *y* ∈ *open-segment x z* ∨ *z* ∈ *open-segment x y*
⟨*proof*⟩

lemma *pocket-fill-line-int-aux3*:
fixes *A* :: (*real*^*2*) *set*

assumes *convex A* ∧ *compact A*
assumes *v* ≠ *0*
assumes *closed-segment 0 w* ⊆ *frontier A* (**is** *closed-segment ?a ?b* ⊆ -)
assumes *w* · *v* = *0*
assumes *w* ≠ *0*
shows (*A* ⊆ {*x*. *x* · *v* ≤ *0*} ∨ *A* ⊆ {*x*. *x* · *v* ≥ *0*}) (**is** *A* ⊆ *?P1* ∨ *A* ⊆ *?P2*)
⟨*proof*⟩

lemma *pocket-fill-line-int-aux4*:
fixes *p q* :: *R-to-R2*
fixes *A* :: (*real^2*) *set*
defines *p0* ≡ *pathstart p*
defines *p1* ≡ *pathfinish p*
defines *q0* ≡ *pathstart q*
defines *q1* ≡ *pathfinish q*
defines *a* ≡ *p1$1*
defines *l* ≡ *closed-segment p0 p1*
assumes *simple-path p*
assumes *simple-path q*
assumes *path-image p* ∩ *path-image q* = {}
assumes *p0$1* = *0* ∧ *p0$2* = *0* ∧ *p1$2* = *0*
assumes *a* > *0*
assumes ∀ *v* ∈ *path-image p*. *q0$2* ≤ *v$2*
assumes ∀ *v* ∈ *path-image p*. *q1$2* > *v$2*
assumes *convex A* ∧ *compact A*
assumes {*p0*, *p1*} ⊆ *frontier A*
assumes *p‘*{*0*<..<*1*} ⊆ *interior A*
assumes *path-image q* ⊆ *A*
shows *l* ⊆ *frontier A* ∀ *x* ∈ (*path-image p*) ∪ (*path-image q*). *x$2* ≥ *0 q0$2* = *0*
⟨*proof*⟩

lemma *pocket-fill-line-int-aux5*:
fixes *p q* :: *R-to-R2*
fixes *A* :: (*real^2*) *set*
defines *p0* ≡ *pathstart p*
defines *p1* ≡ *pathfinish p*
defines *q0* ≡ *pathstart q*
defines *q1* ≡ *pathfinish q*
defines *a* ≡ *p1$1*
defines *l* ≡ *closed-segment p0 p1*
assumes *simple-path p*
assumes *simple-path q*
assumes *path-image p* ∩ *path-image q* = {*q0*, *q1*}
assumes *p0$1* = *0* ∧ *p0$2* = *0* ∧ *p1$2* = *0*
assumes *a* > *0*
assumes *A* = *convex hull* (*path-image p* ∪ *path-image q*)
assumes {*p0*, *p1*} ⊆ *frontier A*
assumes *p‘*{*0*<..<*1*} ⊆ *interior A*

assumes *path-image q* $\subseteq$ *A*
assumes $\exists\, x \in p\text{`}\{0<..<1\}.\ x\$2 \geq 0$
assumes *q0* = *p1* $\wedge$ *q1* = *p0*
shows *l* $\subseteq$ *frontier A* $\forall\, x \in$ *path-image p* $\cup$ *path-image q*. $x\$2 \geq 0$
⟨*proof*⟩

lemma *pocket-fill-line-int-aux6*:
fixes *p q* :: *R-to-R2*
defines *p0* $\equiv$ *pathstart p*
defines *p1* $\equiv$ *pathfinish p*
defines *q0* $\equiv$ *pathstart q*
defines *q1* $\equiv$ *pathfinish q*
defines *a* $\equiv$ *p1*$1*
assumes *simple-path p*
assumes *simple-path q*
assumes *p0* = *0* $\wedge$ *p1*$2 = *0*
assumes *a* > *0*
assumes *q0*$1 $\in \{0..a\} \wedge$ *q0*$2 = *0*
assumes $\forall\, x \in$ *path-image p*. *q1*$2 > $x\$2$
assumes $\forall\, x \in$ *path-image p* $\cup$ *path-image q*. $x\$2 \geq 0$
shows *path-image p* $\cap$ *path-image q* $\neq \{\}$
⟨*proof*⟩

lemma *pocket-fill-line-int-aux7*:
fixes *p q* :: *R-to-R2*
fixes *A* :: (*real^2*) *set*
defines *p0* $\equiv$ *pathstart p*
defines *p1* $\equiv$ *pathfinish p*
defines *q0* $\equiv$ *pathstart q*
defines *q1* $\equiv$ *pathfinish q*
defines *a* $\equiv$ *p1*$1*
defines *l* $\equiv$ *open-segment p0 p1*
assumes *simple-path p*
assumes *simple-path q*
assumes *path-image p* $\cap$ *path-image q* = $\{q0,\ q1\}$
assumes *p0*$1 = *0* $\wedge$ *p0*$2 = *0* $\wedge$ *p1*$2 = *0*
assumes *a* > *0*
assumes *A* = *convex hull* (*path-image p* $\cup$ *path-image q*)
assumes $\{p0,\ p1\} \subseteq$ *frontier A*
assumes *p*`$\{0<..<1\} \subseteq$ *interior A*
assumes $\exists\, x \in p\text{`}\{0<..<1\}.\ x\$2 \geq 0$
assumes *q0* = *p1* $\wedge$ *q1* = *p0*
shows *path-image q* $\cap$ *l* = $\{\}$ *closed-segment p0 p1* $\subseteq$ *frontier A*
⟨*proof*⟩

lemma *frontier-injective-linear-image*:
fixes *f* :: *'a*::*euclidean-space* $\Rightarrow$ *'a*::*euclidean-space*
assumes *linear f inj f*

shows f ‘ $(frontier\ S) = frontier\ (f$ ‘ $S)$
⟨*proof*⟩

lemma *pocket-fill-line-int-aux8*:
fixes $p\ q$ :: *R-to-R2*
fixes A :: $(real\hat{}2)\ set$
defines $p0 \equiv pathstart\ p$
defines $p1 \equiv pathfinish\ p$
defines $q0 \equiv pathstart\ q$
defines $q1 \equiv pathfinish\ q$
defines $a \equiv p1\$1$
defines $l \equiv open\text{-}segment\ p0\ p1$
assumes *simple-path p*
assumes *simple-path q*
assumes $path\text{-}image\ p \cap path\text{-}image\ q = \{q0, q1\}$
assumes $p0\$1 = 0 \wedge p0\$2 = 0 \wedge p1\$2 = 0$
assumes $a > 0$
assumes $A = convex\ hull\ (path\text{-}image\ p \cup path\text{-}image\ q)$
assumes $\{p0, p1\} \subseteq frontier\ A$
assumes $p`\{0<..<1\} \subseteq interior\ A$
assumes $q0 = p1 \wedge q1 = p0$
shows $path\text{-}image\ q \cap l = \{\} \wedge l \subseteq frontier\ A$
⟨*proof*⟩

lemma *simple-path-linear-image*:
assumes *simple-path p*
assumes $inj\ f \wedge bounded\text{-}linear\ f$
shows $simple\text{-}path\ (f \circ p)$
⟨*proof*⟩

lemma *vts-interior*:
fixes *vts*
defines $p \equiv make\text{-}polygonal\text{-}path\ vts$
assumes *convex H*
assumes $\forall j \in \{0<..<length\ vts - 1\}.\ vts!j \notin frontier\ H$
assumes *loop-free p*
assumes $path\text{-}image\ p \subseteq H$
assumes $length\ vts \geq 3$
shows $p`\{0<..<1\} \subseteq interior\ H$
⟨*proof*⟩

lemma *pocket-fill-line-int-0*:
assumes *polygon-of r vts*
defines $H \equiv convex\ hull\ (set\ vts)$
assumes $2 \leq i \wedge i < length\ vts - 1$
defines $a \equiv hd\ vts$
defines $b \equiv vts!i$
assumes $\{a, b\} \subseteq frontier\ H$
assumes $\forall j \in \{0<..<i\}.\ vts!j \notin frontier\ H$

assumes $a = 0$
shows *path-image* (*linepath a b*) ∩ *path-image r* = {*a*, *b*}
path-image (*linepath a b*) ⊆ *frontier H*
⟨*proof*⟩

lemma *linepath-translation*: (λ*v*. *v* − *a*) ∘ (*linepath x y*) = *linepath* ((λ*v*. *v* − *a*) *x*) ((λ*v*. *v* − *a*) *y*)
⟨*proof*⟩

lemma *linepath-image-translation*:
path-image ((λ*v*. *v* − *a*) ∘ (*linepath x y*)) = *path-image* (*linepath* ((λ*v*. *v* − *a*) *x*) ((λ*v*. *v* − *a*) *y*))
⟨*proof*⟩

lemma *make-polygonal-path-translate*:
assumes *length vts* ≥ *1*
shows (λ*v*. *v* − *a*) ∘ (*make-polygonal-path vts*) = *make-polygonal-path* (*map* (λ*v*. *v* − *a*) *vts*)
⟨*proof*⟩

lemma *pocket-fill-line-int*:
assumes *polygon-of r vts*
defines *H* ≡ *convex hull* (*set vts*)
assumes *2* ≤ *i* ∧ *i* < *length vts* − *1*
defines *a* ≡ *hd vts*
defines *b* ≡ *vts*!*i*
assumes {*a*, *b*} ⊆ *frontier H*
assumes ∀ *j* ∈ {*0*<..<*i*}. *vts*!*j* ∉ *frontier H*
shows *path-image* (*linepath a b*) ∩ *path-image r* = {*a*, *b*}
path-image (*linepath a b*) ⊆ *frontier H*
⟨*proof*⟩

lemma *path-connected-simple-path-endless*:
assumes *simple-path p*
shows *path-connected* (*path-image p* − {*pathstart p*, *pathfinish p*}) (**is** *path-connected ?S*)
⟨*proof*⟩

lemma *simple-loop-split*:
assumes *simple-path p* ∧ *closed-path p*
assumes *simple-path q*
assumes *path-image q* ∩ *path-image p* = {*q 0*, *q 1*}
assumes *path-image q* ∩ *path-inside p* ≠ {}
shows *q*‘{*0*<..<*1*} ⊆ *path-inside p*
⟨*proof*⟩

lemma *pocket-path-interior-aux*:
assumes *simple-path p* ∧ *simple-path q*

assumes *arc p* ∧ *arc q*
assumes *q 0* = *p 1* ∧ *q 1* = *p 0*
assumes *path-image p* ∩ *path-image q* = {*p 0*, *q 0*}
defines *A* ≡ *convex hull* (*path-image p* ∪ *path-image q*)
defines *l* ≡ *linepath* (*p 0*) (*p 1*)
assumes *p*‘{*0*<..<*1*} ⊆ *interior A*
assumes *path-image l* ⊆ *frontier A*
assumes *path-image q* ∩ *path-image l* = {*l 0*, *q 0*}
shows *p*‘{*0*<..<*1*} ∩ *path-inside* (*l* +++ *q*) ≠ {}
simple-path (*l* +++ *q*) ∧ *closed-path* (*l* +++ *q*)
path-image p ∩ *path-image* (*l* +++ *q*) = {*p 0*, *p 1*}
⟨*proof*⟩

lemma *pocket-path-interior*:
assumes *simple-path p* ∧ *simple-path q*
assumes *arc p* ∧ *arc q*
assumes *q 0* = *p 1* ∧ *q 1* = *p 0*
assumes *path-image p* ∩ *path-image q* = {*p 0*, *q 0*}
defines *A* ≡ *convex hull* (*path-image p* ∪ *path-image q*)
defines *l* ≡ *linepath* (*p 0*) (*p 1*)
assumes *p*‘{*0*<..<*1*} ⊆ *interior A*
assumes *path-image l* ⊆ *frontier A*
assumes *path-image q* ∩ *path-image l* = {*l 0*, *q 0*}
shows *p*‘{*0*<..<*1*} ⊆ *path-inside* (*l* +++ *q*)
⟨*proof*⟩

lemma *pocket-path-good*:
assumes *polygon* (*make-polygonal-path vts*)
assumes *vts*!*0* ∈ *frontier* (*convex hull* (*set vts*))
assumes *vts*!*1* ∉ *frontier* (*convex hull* (*set vts*))
assumes ¬ *convex* (*path-image* (*make-polygonal-path vts*) ∪ *path-inside* (*make-polygonal-path vts*))
defines *pocket-path-vts* ≡ *construct-pocket-0 vts* (*set vts* ∩ *frontier* (*convex hull* (*set vts*)))
defines *pocket* ≡ *make-polygonal-path* (*pocket-path-vts* @ [*pocket-path-vts*!*0*])
defines *filled-vts* ≡ *fill-pocket-0 vts* (*length pocket-path-vts*)
defines *filled-p* ≡ *make-polygonal-path filled-vts*
defines *a* ≡ *hd pocket-path-vts*
defines *b* ≡ *last pocket-path-vts*
defines *good-pocket-path-vts* ≡ *tl* (*butlast pocket-path-vts*)
shows *polygon filled-p*
is-polygon-split-path (*butlast filled-vts*) *0 1 good-pocket-path-vts*
polygon pocket
card (*set pocket-path-vts*) < *card* (*set vts*)
card (*set filled-vts*) < *card* (*set vts*)
⟨*proof*⟩

29.3 Arbitrary Polygon Case

lemma *pick-rotate*:
assumes *polygon-of p vts*
assumes *all-integral vts*
obtains p' vts' **where** *polygon-of* p' vts'
$\wedge$ $vts'!0 \in$ *frontier* (*convex hull* (*set* vts'))
$\wedge$ *path-image* p' = *path-image p*
$\wedge$ *all-integral* vts'
$\wedge$ *set* vts' = *set vts*
⟨*proof*⟩

lemma *pick-unrotated*:
fixes *p* :: *R-to-R2*
assumes *polygon*: *polygon p*
assumes *polygonal-path*: *p* = *make-polygonal-path vts*
assumes *int-vertices*: *all-integral vts*
assumes *I-is*: *I* = *card* {*x*. *integral-vec x* $\wedge$ *x* $\in$ *path-inside p*}
assumes *B-is*: *B* = *card* {*x*. *integral-vec x* $\wedge$ *x* $\in$ *path-image p*}
assumes *vts*!*0* $\in$ *frontier* (*convex hull* (*set vts*))
shows *measure lebesgue* (*path-inside p*) = *I* + *B*/*2* − *1*
⟨*proof*⟩

theorem *pick*:
fixes *p* :: *R-to-R2*
assumes *polygon p*
assumes *p* = *make-polygonal-path vts*
assumes *all-integral vts*
assumes *I* = *card* {*x*. *integral-vec x* $\wedge$ *x* $\in$ *path-inside p*}
assumes *B* = *card* {*x*. *integral-vec x* $\wedge$ *x* $\in$ *path-image p*}
shows *measure lebesgue* (*path-inside p*) = *I* + *B*/*2* − *1*
⟨*proof*⟩

end

References

[1] B. Grünbaum and G. C. Shephard. Pick's theorem. *The American Mathematical Monthly*, 100(2):150–161, 1993.

[2] J. Harrison. A formal proof of Pick's theorem. *Math. Struct. Comput. Sci.*, 21(4):715–729, 2011.