An Isabelle/HOL formalisation of Green's Theorem

Mohammad Abdulaziz and Lawrence C. Paulson

February 6, 2026

Abstract

We formalise a statement of Greens theorem—the first formalisation to our knowledge—in Isabelle/HOL. The theorem statement that we formalise is enough for most applications, especially in physics and engineering. Our formalisation is made possible by a novel proof that avoids the ubiquitous line integral cancellation argument. This eliminates the need to formalise orientations and region boundaries explicitly with respect to the outwards-pointing normal vector. Instead we appeal to a homological argument about equivalences between paths.

1 Acknowledgements

Paulson was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council at the University of Cambridge, UK.

theory *General-Utils*
  imports *HOL−Analysis.Analysis*
begin

lemma *lambda-skolem-gen*: $(\forall\, i.\ \exists f'::('a\ \hat{}\ 'n) \Rightarrow 'a.\ P\ i\ f') \longleftrightarrow$
  $(\exists f'::('a\ \hat{}\ 'n) \Rightarrow ('a\ \hat{}\ 'n).\ \forall\, i.\ P\ i\ ((\lambda x.\ (f'\ x)\ \$\ i)))$ (**is** *?lhs* $\longleftrightarrow$ *?rhs*)

⟨*proof*⟩

lemma *lambda-skolem-euclidean*: $(\forall\, i{\in}Basis.\ \exists f'::('a::\{euclidean\text{-}space\}{\Rightarrow}real).\ P\ i\ f') \longleftrightarrow$
  $(\exists f'::('a::euclidean\text{-}space{\Rightarrow}'b::euclidean\text{-}space).\ \forall\, i{\in}Basis.\ P\ i\ ((\lambda x.\ (f'\ x)\cdot i)))$ (**is** *?lhs* $\longleftrightarrow$ *?rhs*)

⟨*proof*⟩

lemma *lambda-skolem-euclidean-explicit*: $(\forall\, i{\in}Basis.\ \exists f'::('a::\{euclidean\text{-}space\}{\Rightarrow}real).\ P\ i\ f') \longleftrightarrow$
  $(\exists f'::('a::\{euclidean\text{-}space\}{\Rightarrow}'a).\ \forall\, i{\in}Basis.\ P\ i\ ((\lambda x.\ (f'\ x)\cdot i)))$ (**is** *?lhs* $\longleftrightarrow$ *?rhs*)

⟨proof⟩

lemma *indic-ident*:
  $\bigwedge$ (*f*::′*a* ⇒ *real*) *s*. (λ*x*. (*f x*) ∗ *indicator s x*) = (λ*x*. *if x* ∈ *s then f x else 0*)
⟨proof⟩

lemma *real-pair-basis*: *Basis* = {(*1*::*real*,*0*::*real*), (*0*::*real*,*1*::*real*)}
  ⟨proof⟩

lemma *real-singleton-in-borel*:
  shows {*a*::*real*} ∈ *sets borel*
  ⟨proof⟩

lemma *real-singleton-in-lborel*:
  shows {*a*::*real*} ∈ *sets lborel*
  ⟨proof⟩

lemma *cbox-diff*:
  shows {*0*::*real*..*1*} − {*0*,*1*} = *box 0 1*
  ⟨proof⟩

lemma *sum-bij*:
  assumes *bij F*
    ∀ *x*∈*s*. *f x* = *g* (*F x*)
  shows $\bigwedge$*t*. *F* ‘ *s* = *t* ⟹ *sum f s* = *sum g t*
  ⟨proof⟩

abbreviation *surj-on* **where**
  surj-on s f ≡ *s* ⊆ *range f*

lemma *surj-on-image-vimage-eq*: *surj-on s f* ⟹ *f* ‘ (*f* −‘ *s*) = *s*
  ⟨proof⟩

end
theory *Derivs*
  imports *General-Utils*
begin

lemma *field-simp-has-vector-derivative* [*derivative-intros*]:
  (*f has-field-derivative y*) *F* ⟹ (*f has-vector-derivative y*) *F*
  ⟨proof⟩

lemma *continuous-on-cases-empty* [*continuous-intros*]:
  ⟦*closed S*; *continuous-on S f*; $\bigwedge$*x*. ⟦*x* ∈ *S*; ¬ *P x*⟧ ⟹ *f x* = *g x*⟧ ⟹
    continuous-on S (λ*x*. *if P x then f x else g x*)
  ⟨proof⟩

lemma *inj-on-cases*:
  assumes *inj-on f* (*Collect P* ∩ *S*) *inj-on g* (*Collect* (*Not* ∘ *P*) ∩ *S*)
    f ‘ (*Collect P* ∩ *S*) ∩ *g* ‘ (*Collect* (*Not* ∘ *P*) ∩ *S*) = {}
  shows *inj-on* (λ*x*. *if P x then f x else g x*) *S*
  ⟨*proof*⟩

lemma *inj-on-arccos*: *S* ⊆ {−*1*..*1*} ⟹ *inj-on arccos S*
  ⟨*proof*⟩

lemma *has-vector-derivative-componentwise-within*:
  (*f has-vector-derivative f′*) (*at a within S*) ⟷
    (∀ *i* ∈ *Basis*. ((λ*x*. *f x* · *i*) *has-vector-derivative* (*f′* · *i*)) (*at a within S*))
  ⟨*proof*⟩

lemma *has-vector-derivative-pair-within*:
  fixes *f* :: *real* ⇒ ′*a*::*euclidean-space* **and** *g* :: *real* ⇒ ′*b*::*euclidean-space*
  assumes ⋀*u*. *u* ∈ *Basis* ⟹ ((λ*x*. *f x* · *u*) *has-vector-derivative f′* · *u*) (*at x within S*)
    ⋀*u*. *u* ∈ *Basis* ⟹ ((λ*x*. *g x* · *u*) *has-vector-derivative g′* · *u*) (*at x within S*)
  shows ((λ*x*. (*f x*, *g x*)) *has-vector-derivative* (*f′*,*g′*)) (*at x within S*)
  ⟨*proof*⟩

lemma *piecewise-C1-differentiable-const*:
  shows (λ*x*. *c*) *piecewise-C1-differentiable-on s*
  ⟨*proof*⟩

declare *piecewise-C1-differentiable-const* [*simp*, *derivative-intros*]
declare *piecewise-C1-differentiable-neg* [*simp*, *derivative-intros*]
declare *piecewise-C1-differentiable-add* [*simp*, *derivative-intros*]
declare *piecewise-C1-differentiable-diff* [*simp*, *derivative-intros*]

lemma *piecewise-C1-differentiable-on-ident* [*simp*, *derivative-intros*]:
  fixes *f* :: *real* ⇒ ′*a*::*real-normed-vector*
  shows (λ*x*. *x*) *piecewise-C1-differentiable-on S*
  ⟨*proof*⟩

lemma *piecewise-C1-differentiable-on-mult* [*simp*, *derivative-intros*]:
  fixes *f* :: *real* ⇒ ′*a*::*real-normed-algebra*
  assumes *f piecewise-C1-differentiable-on S g piecewise-C1-differentiable-on S*
  shows (λ*x*. *f x* ∗ *g x*) *piecewise-C1-differentiable-on S*
  ⟨*proof*⟩

lemma *C1-differentiable-on-cdiv* [*simp*, *derivative-intros*]:
  fixes *f* :: *real* ⇒ ′*a* :: *real-normed-field*
  shows *f C1-differentiable-on S* ⟹ (λ*x*. *f x* / *c*) *C1-differentiable-on S*
  ⟨*proof*⟩

lemma *piecewise-C1-differentiable-on-cdiv* [*simp*, *derivative-intros*]:
  fixes *f* :: *real* ⇒ *'a::real-normed-field*
  assumes *f piecewise-C1-differentiable-on S*
  shows (λ*x*. *f x* / *c*) *piecewise-C1-differentiable-on S*
  ⟨*proof*⟩

lemma *sqrt-C1-differentiable* [*simp*, *derivative-intros*]:
  assumes *f*: *f C1-differentiable-on S* **and** *fim*: *f* ‘ *S* ⊆ {*0*<..}
  shows (λ*x*. *sqrt* (*f x*)) *C1-differentiable-on S*
⟨*proof*⟩

lemma *sqrt-piecewise-C1-differentiable* [*simp*, *derivative-intros*]:
  assumes *f*: *f piecewise-C1-differentiable-on S* **and** *fim*: *f* ‘ *S* ⊆ {*0*<..}
  shows (λ*x*. *sqrt* (*f x*)) *piecewise-C1-differentiable-on S*
  ⟨*proof*⟩

lemma
  fixes *f* :: *real* ⇒ *'a::{banach,real-normed-field}*
  assumes *f*: *f C1-differentiable-on S*
  shows *sin-C1-differentiable* [*simp*, *derivative-intros*]: (λ*x*. *sin* (*f x*)) *C1-differentiable-on S*
  and *cos-C1-differentiable* [*simp*, *derivative-intros*]: (λ*x*. *cos* (*f x*)) *C1-differentiable-on S*
⟨*proof*⟩

lemma *has-derivative-abs*:
  fixes *a::real*
  assumes *a* ≠ *0*
  shows (*abs has-derivative* ((∗) (*sgn a*))) (*at a*)
⟨*proof*⟩

lemma *abs-C1-differentiable* [*simp*, *derivative-intros*]:
  fixes *f* :: *real* ⇒ *real*
  assumes *f*: *f C1-differentiable-on S* **and** *0* ∉ *f* ‘ *S*
  shows (λ*x*. *abs* (*f x*)) *C1-differentiable-on S*
⟨*proof*⟩

lemma *C1-differentiable-on-pair* [*simp*, *derivative-intros*]:
  fixes *f* :: *real* ⇒ *'a::euclidean-space* **and** *g* :: *real* ⇒ *'b::euclidean-space*
  assumes *f C1-differentiable-on S g C1-differentiable-on S*
  shows (λ*x*. (*f x*, *g x*)) *C1-differentiable-on S*
  ⟨*proof*⟩

lemma *piecewise-C1-differentiable-on-pair* [*simp*, *derivative-intros*]:
  fixes *f* :: *real* ⇒ *'a::euclidean-space* **and** *g* :: *real* ⇒ *'b::euclidean-space*
  assumes *f piecewise-C1-differentiable-on S g piecewise-C1-differentiable-on S*
  shows (λ*x*. (*f x*, *g x*)) *piecewise-C1-differentiable-on S*
  ⟨*proof*⟩

lemma *test2*:
  assumes *s*: $\bigwedge x.\ x \in \{0..1\} - s \Longrightarrow g$ *differentiable at* x
    and *fs*: *finite s* **and** *uv*: $u \in \{0..1\}$ $v \in \{0..1\}$ $u \leq v$
    and $x \in \{0..1\}$ $x \notin (\lambda t.\ (v-u) *_R t + u) -`\ s$
  shows *vector-derivative* $(\lambda x.\ g\ ((v-u) * x + u))$ $(at\ x\ within\ \{0..1\}) = (v-u)$ $*_R$ *vector-derivative* g $(at\ ((v-u) * x + u)\ within\{0..1\})$
⟨*proof*⟩

lemma *C1-differentiable-on-components*:
  assumes $\bigwedge i.\ i \in Basis \Longrightarrow (\lambda x.\ f\ x \cdot i)$ *C1-differentiable-on s*
  shows *f C1-differentiable-on s*
⟨*proof*⟩

lemma *piecewise-C1-differentiable-on-components*:
  assumes *finite t*
    $\bigwedge i.\ i \in Basis \Longrightarrow (\lambda x.\ f\ x \cdot i)$ *C1-differentiable-on* $s - t$
    $\bigwedge i.\ i \in Basis \Longrightarrow$ *continuous-on s* $(\lambda x.\ f\ x \cdot i)$
  shows *f piecewise-C1-differentiable-on s*
  ⟨*proof*⟩

lemma *all-components-smooth-one-pw-smooth-is-pw-smooth*:
  assumes $\bigwedge i.\ i \in Basis - \{j\} \Longrightarrow (\lambda x.\ f\ x \cdot i)$ *C1-differentiable-on s*
  assumes $(\lambda x.\ f\ x \cdot j)$ *piecewise-C1-differentiable-on s*
  shows *f piecewise-C1-differentiable-on s*
⟨*proof*⟩

lemma *derivative-component-fun-component*:
  fixes i::$'a$::*euclidean-space*
  assumes *f differentiable* $(at\ x)$
  shows $((vector\text{-}derivative\ f\ (at\ x)) \cdot i) = ((vector\text{-}derivative\ (\lambda x.\ (f\ x) \cdot i)\ (at\ x)))$
⟨*proof*⟩

lemma *gamma-deriv-at-within*:
  assumes *a-leq-b*: $a < b$ **and**
    x-within-bounds: $x \in \{a..b\}$ **and**
    gamma-differentiable: $\forall x \in \{a\ ..\ b\}.\ \gamma$ *differentiable at* x
  shows *vector-derivative* γ $(at\ x\ within\ \{a..b\}) =$ *vector-derivative* γ $(at\ x)$
  ⟨*proof*⟩

lemma *islimpt-diff-finite*:
  assumes *finite* $(t::'a::t1\text{-}space\ set)$
  shows x *islimpt* $s - t = x$ *islimpt* s
⟨*proof*⟩

lemma *ivl-limpt-diff*:
  assumes *finite s* $a < b$ $(x::real) \in \{a..b\} - s$
  shows x *islimpt* $\{a..b\} - s$

⟨*proof*⟩

lemma *ivl-closure-diff-del*:
assumes *finite s* $a < b$ $(x{::}real) \in \{a..b\} - s$
shows $x \in$ *closure* $((\{a..b\} - s) - \{x\})$
⟨*proof*⟩

lemma *ivl-not-trivial-limit-within*:
assumes *finite s*
$a < b$
$(x{::}real) \in \{a..b\} - s$
shows *at x within* $\{a..b\} - s \neq$ *bot*
⟨*proof*⟩

lemma *vector-derivative-at-within-non-trivial-limit*:
at x within s $\neq$ *bot* $\wedge$ (*f has-vector-derivative* f') (*at x*) $\Longrightarrow$
vector-derivative f (*at x within s*) $= f'$
⟨*proof*⟩

lemma *vector-derivative-at-within-ivl-diff*:
finite s $\wedge$ $a < b$ $\wedge$ $(x{::}real) \in \{a..b\} - s$ $\wedge$ (*f has-vector-derivative* f') (*at x*) $\Longrightarrow$
vector-derivative f (*at x within* $\{a..b\} - s$) $= f'$
⟨*proof*⟩

lemma *gamma-deriv-at-within-diff*:
assumes *a-leq-b*: $a < b$ **and**
x-within-bounds: $x \in \{a..b\} - s$ **and**
gamma-differentiable: $\forall x \in \{a\ ..\ b\} - s.\ \gamma$ *differentiable at x* **and**
s-subset: $s \subseteq \{a..b\}$ **and**
finite-s: *finite s*
shows *vector-derivative* γ (*at x within* $\{a..b\} - s$)
$=$ *vector-derivative* γ (*at x*)
⟨*proof*⟩

lemma *gamma-deriv-at-within-gen*:
assumes *a-leq-b*: $a < b$ **and**
x-within-bounds: $x \in s$ **and**
s-subset: $s \subseteq \{a..b\}$ **and**
gamma-differentiable: $\forall x \in s.\ \gamma$ *differentiable at x*
shows *vector-derivative* γ (*at x within* $(\{a..b\})$) $=$ *vector-derivative* γ (*at x*)
⟨*proof*⟩

lemma *derivative-component-fun-component-at-within-gen*:
assumes *gamma-differentiable*: $\forall x \in s.\ \gamma$ *differentiable at x* **and** *s-subset*: $s \subseteq \{0..1\}$
shows $\forall x \in s.$ *vector-derivative* $(\lambda x.\ \gamma\ x)$ (*at x within* $\{0..1\}$) $\cdot$ (i::$'a$:: *euclidean-space*)
$=$ *vector-derivative* $(\lambda x.\ \gamma\ x \cdot i)$ (*at x within* $\{0..1\}$)
⟨*proof*⟩

lemma *derivative-component-fun-component-at-within*:
  assumes *gamma-differentiable*: $\forall x \in \{0 .. 1\}$. γ *differentiable at x*
  shows $\forall x \in \{0..1\}$. *vector-derivative* ($\lambda x.$ γ *x*) (*at x within* {*0..1*}) $\cdot$ (*i*::'*a*:: *euclidean-space*)
  = *vector-derivative* ($\lambda x.$ γ *x* $\cdot$ *i*) (*at x within* {*0..1*})
⟨*proof*⟩

lemma *straight-path-diffrentiable-x*:
  fixes *b* :: *real* **and** *y1* ::*real*
  assumes *gamma-def*: $\gamma = (\lambda x.\ (b,\ y2 + y1 * x))$
  shows $\forall x.$ γ *differentiable at x*
  ⟨*proof*⟩

lemma *straight-path-diffrentiable-y*:
  fixes *b* :: *real* **and**
    y1 y2 ::*real*
  assumes *gamma-def*: $\gamma = (\lambda x.\ (y2 + y1 * x\ ,\ b))$
  shows $\forall x.$ γ *differentiable at x*
  ⟨*proof*⟩

lemma *piecewise-C1-differentiable-on-imp-continuous-on*:
  assumes *f piecewise-C1-differentiable-on s*
  shows *continuous-on s f*
  ⟨*proof*⟩

lemma *boring-lemma1*:
  fixes *f* :: *real*⇒*real*
  assumes (*f has-vector-derivative D*) (*at x*)
  shows (($\lambda x.$ (*f x*, *0*)) *has-vector-derivative* ((*D*,*0*::*real*))) (*at x*)
⟨*proof*⟩

lemma *boring-lemma2*:
  fixes *f* :: *real*⇒*real*
  assumes (*f has-vector-derivative D*) (*at x*)
  shows (($\lambda x.$ (*0*, *f x*)) *has-vector-derivative* (*0*, *D*)) (*at x*)
⟨*proof*⟩

lemma *pair-prod-smooth-pw-smooth*:
  assumes (*f*::*real*⇒*real*) *C1-differentiable-on s* (*g*::*real*⇒*real*) *piecewise-C1-differentiable-on s*
  shows ($\lambda x.$ (*f x*, *g x*)) *piecewise-C1-differentiable-on s*
⟨*proof*⟩

lemma *scale-shift-smooth*:
  shows ($\lambda x.$ $a + b * x$) *C1-differentiable-on s*
⟨*proof*⟩

lemma *open-diff*:

assumes *finite* (t::$'a$::*t1-space set*)
open (s::$'a$ *set*)
shows *open* (s $-t$)
⟨*proof*⟩

lemma *has-derivative-transform-within*:
assumes $0 < d$
and $x \in s$
and $\forall x'{\in}s.\ dist\ x'\ x < d \longrightarrow f\ x' = g\ x'$
and (f *has-derivative* f') (*at* x *within* s)
shows (g *has-derivative* f') (*at* x *within* s)
⟨*proof*⟩

lemma *has-derivative-transform-within-ivl*:
assumes (0::*real*) $< b$
and $\forall x{\in}\{a..b\} -s.\ f\ x = g\ x$
and $x \in \{a..b\} -s$
and (f *has-derivative* f') (*at* x *within* $\{a..b\} - s$)
shows (g *has-derivative* f') (*at* x *within* $\{a..b\} - s$)
⟨*proof*⟩

lemma *has-vector-derivative-transform-within-ivl*:
assumes (0::*real*) $< b$
and $\forall x{\in}\{a..b\} -s\ .\ f\ x = g\ x$
and $x \in \{a..b\} - s$
and (f *has-vector-derivative* f') (*at* x *within* $\{a..b\} - s$)
shows (g *has-vector-derivative* f') (*at* x *within* $\{a..b\} - s$)
⟨*proof*⟩

lemma *has-derivative-transform-at*:
assumes $0 < d$
and $\forall x'.\ dist\ x'\ x < d \longrightarrow f\ x' = g\ x'$
and (f *has-derivative* f') (*at* x)
shows (g *has-derivative* f') (*at* x)
⟨*proof*⟩

lemma *has-vector-derivative-transform-at*:
assumes $0 < d$
and $\forall x'.\ dist\ x'\ x < d \longrightarrow f\ x' = g\ x'$
and (f *has-vector-derivative* f') (*at* x)
shows (g *has-vector-derivative* f') (*at* x)
⟨*proof*⟩

lemma *C1-diff-components-2*:
assumes $b \in$ *Basis*
assumes f *C1-differentiable-on* s
shows ($\lambda x.\ f\ x \cdot b$) *C1-differentiable-on* s
⟨*proof*⟩

lemma *eq-smooth*:
  assumes $0 < d$
    $\forall x\in s.\ \forall y.\ dist\ x\ y < d \longrightarrow f\ y = g\ y$
    f C1-differentiable-on s
  shows *g C1-differentiable-on s*
⟨*proof*⟩

lemma *eq-pw-smooth*:
  assumes $0 < d$
    $\forall x\in s.\ \forall y.\ dist\ x\ y < d \longrightarrow f\ y = g\ y$
    $\forall x\in s.\ f\ x = g\ x$
    f piecewise-C1-differentiable-on s
  shows *g piecewise-C1-differentiable-on s*
⟨*proof*⟩

lemma *scale-piecewise-C1-differentiable-on*:
  assumes *f piecewise-C1-differentiable-on s*
  shows $(\lambda x.\ (c::real) * (f\ x))$ *piecewise-C1-differentiable-on s*
⟨*proof*⟩

lemma *eq-smooth-gen*:
  assumes *f C1-differentiable-on s*
    $\forall x.\ f\ x = g\ x$
  shows *g C1-differentiable-on s*
  ⟨*proof*⟩

lemma *subpath-compose*:
  shows $(subpath\ a\ b\ \gamma) = \gamma\ o\ (\lambda x.\ (b - a) * x + a)$
  ⟨*proof*⟩

lemma *subpath-smooth*:
  assumes γ *C1-differentiable-on* $\{0..1\}$ $0 \le a$ $a < b$ $b \le 1$
  shows $(subpath\ a\ b\ \gamma)$ *C1-differentiable-on* $\{0..1\}$
⟨*proof*⟩

lemma *has-vector-derivative-divide*[*derivative-intros*]:
  fixes $a :: 'a::real\text{-}normed\text{-}field$
  shows $(f\ has\text{-}vector\text{-}derivative\ x)\ F \Longrightarrow ((\lambda x.\ f\ x\ /\ a)\ has\text{-}vector\text{-}derivative\ (x\ /\ a))\ F$
  ⟨*proof*⟩

end
theory *Integrals*
  imports *HOL−Analysis.Analysis General-Utils*
begin

lemma *gauge-integral-Fubini-universe-x*:
  fixes $f :: ('a::euclidean\text{-}space * 'b::euclidean\text{-}space) \Rightarrow 'c::euclidean\text{-}space$
  assumes *fun-lesbegue-integrable*: *integrable lborel f* **and**

x-axis-integral-measurable: $(\lambda x.\ integral\ UNIV\ (\lambda y.\ f(x,\ y))) \in borel\text{-}measurable$ *lborel*
shows *integral UNIV f* = *integral UNIV* $(\lambda x.\ integral\ UNIV\ (\lambda y.\ f(x,y)))$
$(\lambda x.\ integral\ UNIV\ (\lambda y.\ f(x,y)))$ *integrable-on UNIV*
⟨*proof*⟩

lemma *gauge-integral-Fubini-universe-y*:
fixes *f* :: (*'a*::*euclidean-space* ∗ *'b*::*euclidean-space*) ⇒ *'c*::*euclidean-space*
assumes *fun-lesbegue-integrable*: *integrable lborel f* **and**
y-axis-integral-measurable: $(\lambda x.\ integral\ UNIV\ (\lambda y.\ f(y,\ x))) \in borel\text{-}measurable$ *lborel*
shows *integral UNIV f* = *integral UNIV* $(\lambda x.\ integral\ UNIV\ (\lambda y.\ f(y,\ x)))$
$(\lambda x.\ integral\ UNIV\ (\lambda y.\ f(y,\ x)))$ *integrable-on UNIV*
⟨*proof*⟩

lemma *gauge-integral-Fubini-curve-bounded-region-x*:
fixes *f g* :: (*'a*::*euclidean-space* ∗ *'b*::*euclidean-space*) ⇒ *'c*::*euclidean-space* **and**
g1 g2:: *'a* ⇒ *'b* **and**
s:: (*'a* ∗ *'b*) *set*
assumes *fun-lesbegue-integrable*: *integrable lborel f* **and**
x-axis-gauge-integrable: $\bigwedge x.\ (\lambda y.\ f(x,\ y))$ *integrable-on UNIV* **and**

x-axis-integral-measurable: $(\lambda x.\ integral\ UNIV\ (\lambda y.\ f(x,\ y))) \in borel\text{-}measurable$ *lborel* **and**
f-is-g-indicator: $f = (\lambda x.\ if\ x \in s\ then\ g\ x\ else\ 0)$ **and**
s-is-bounded-by-g1-and-g2: $s = \{(x,y).\ (\forall i \in Basis.\ a \cdot i \le x \cdot i \wedge x \cdot i \le b \cdot i) \wedge$
$(\forall i \in Basis.\ (g1\ x) \cdot i \le y \cdot i \wedge y \cdot i \le (g2\ x) \cdot i)\}$
shows *integral s g* = *integral* (*cbox a b*) $(\lambda x.\ integral\ (cbox\ (g1\ x)\ (g2\ x))\ (\lambda y.\ g(x,y)))$
⟨*proof*⟩

lemma *gauge-integral-Fubini-curve-bounded-region-y*:
fixes *f g* :: (*'a*::*euclidean-space* ∗ *'b*::*euclidean-space*) ⇒ *'c*::*euclidean-space* **and**
g1 g2:: *'b* ⇒ *'a* **and**
s:: (*'a* ∗ *'b*) *set*
assumes *fun-lesbegue-integrable*: *integrable lborel f* **and**
y-axis-gauge-integrable: $\bigwedge x.\ (\lambda y.\ f(y,\ x))$ *integrable-on UNIV* **and**

y-axis-integral-measurable: $(\lambda x.\ integral\ UNIV\ (\lambda y.\ f(y,\ x))) \in borel\text{-}measurable$ *lborel* **and**
f-is-g-indicator: $f = (\lambda x.\ if\ x \in s\ then\ g\ x\ else\ 0)$ **and**
s-is-bounded-by-g1-and-g2: $s = \{(y,\ x).\ (\forall i \in Basis.\ a \cdot i \le x \cdot i \wedge x \cdot i \le b \cdot i) \wedge$
$(\forall i \in Basis.\ (g1\ x) \cdot i \le y \cdot i \wedge y \cdot i \le (g2\ x) \cdot i)\}$
shows *integral s g* = *integral* (*cbox a b*) $(\lambda x.\ integral\ (cbox\ (g1\ x)\ (g2\ x))\ (\lambda y.\ g(y,\ x)))$
⟨*proof*⟩

lemma *gauge-integral-by-substitution*:
fixes f::(*real* ⇒ *real*) **and**
g::(*real* ⇒ *real*) **and**
g'::*real* ⇒ *real* **and**
a::*real* **and**
b::*real*
assumes *a-le-b*: $a \leq b$ **and**
ga-le-gb: $g\ a \leq g\ b$ **and**
g'*-derivative*: $\forall x \in \{a..b\}$. (g *has-vector-derivative* ($g'\ x$)) (*at* x *within* $\{a..b\}$)
and
g'*-continuous*: *continuous-on* $\{a..b\}$ g' **and**
f-continuous: *continuous-on* (g ‘ $\{a..b\}$) f
shows *integral* $\{g\ a..g\ b\}$ (f) = *integral* $\{a..b\}$ $(\lambda x.\ f(g\ x) * (g'\ x))$
⟨*proof*⟩

lemma
assumes $a <$ (b::*real*)
shows *frontier-ic*: *frontier* $\{a<..b\} = \{a,b\}$
and *frontier-ci*: *frontier* $\{a<..<b\} = \{a,b\}$
⟨*proof*⟩

lemma *ic-not-closed*:
assumes $a <$ (b::*real*)
shows ¬ *closed* $\{a<..b\}$
⟨*proof*⟩

lemma *closure-ic-union-ci*:
assumes $a <$ (b::*real*) $b < c$
shows *closure* $(\{a..<b\} \cup \{b<..c\}) = \{a\ ..\ c\}$
⟨*proof*⟩

lemma *interior-ic-ci-union*:
assumes $a <$ (b::*real*) $b < c$
shows $b \notin$ (*interior* $(\{a..<b\} \cup \{b<..c\})$)
⟨*proof*⟩

lemma *frontier-ic-union-ci*:
assumes $a <$ (b::*real*) $b < c$
shows $b \in$ *frontier* $(\{a..<b\} \cup \{b<..c\})$
⟨*proof*⟩

lemma *ic-union-ci-not-closed*:
assumes $a <$ (b::*real*) $b < c$
shows ¬ *closed* $(\{a..<b\} \cup \{b<..c\})$
⟨*proof*⟩

lemma *integrable-continuous-*:
fixes f :: $'b$::*euclidean-space* ⇒ $'a$::*banach*

assumes *continuous-on* (*cbox a b*) *f*
shows *f integrable-on cbox a b*
⟨*proof*⟩

lemma *removing-singletons-from-div*:
assumes $\forall t \in S.\ \exists c\ d{::}real.\ c < d \wedge \{c..d\} = t$
$\{x\} \cup \bigcup S = \{a..b\}$ $a < x$ $x < b$
finite S
shows $\exists t \in S.\ x \in t$
⟨*proof*⟩

lemma *remove-singleton-from-division-of*:
assumes *A division-of* $\{a{::}real..b\}$ $a < b$
assumes $x \in \{a..b\}$
shows $\exists c\ d.\ c < d \wedge \{c..d\} \in A \wedge x \in \{c..d\}$
⟨*proof*⟩

lemma *remove-singleton-from-tagged-division-of*:
assumes *A tagged-division-of* $\{a{::}real..b\}$ $a < b$
assumes $x \in \{a..b\}$
shows $\exists k\ c\ d.\ c < d \wedge (k, \{c..d\}) \in A \wedge x \in \{c..d\}$
⟨*proof*⟩

lemma *tagged-div-wo-singlestons*:
assumes *p tagged-division-of* $\{a{::}real..b\}$ $a < b$
shows $(p - \{xk.\ \exists x\ y.\ xk = (x,\{y\})\})$ *tagged-division-of cbox a b*
⟨*proof*⟩

lemma *tagged-div-wo-empty*:
assumes *p tagged-division-of* $\{a{::}real..b\}$ $a < b$
shows $(p - \{xk.\ \exists x.\ xk = (x,\{\})\})$ *tagged-division-of cbox a b*
⟨*proof*⟩

lemma *fine-diff*:
assumes γ *fine p*
shows γ *fine* $(p - s)$
⟨*proof*⟩

lemma *tagged-div-tage-notin-set*:
assumes *finite* (*s*::*real set*)
p tagged-division-of $\{a..b\}$
γ *fine p* $(\forall (x, K) \in p.\ \exists c\ d{::}real.\ c < d \wedge K = \{c..d\})$ *gauge* γ
shows $\exists p'\ \gamma'.\ p'$ *tagged-division-of* $\{a..b\} \wedge$
γ' *fine* $p' \wedge (\forall (x, K) \in p'.\ x \notin s) \wedge$ *gauge* γ'
⟨*proof*⟩

lemma *has-integral-bound-spike-finite*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*real-normed-vector*
assumes $0 \leq B$ **and** *finite S*

and f: (f *has-integral* i) (*cbox* a b)
and *leB*: $\bigwedge x.\ x \in$ *cbox* a $b - S \Longrightarrow$ *norm* (f x) $\leq B$
shows *norm* $i \leq B *$ *measure lborel* (*cbox* a b)
⟨*proof*⟩

lemma *has-integral-bound-*:
fixes f :: *real* ⇒ *'a*::*real-normed-vector*
assumes $a < b$
and $0 \leq B$
and f: (f *has-integral* i) (*cbox* a b)
and *finite* s
and $\forall x \in$(*cbox* a b)$-s$. *norm* (f x) $\leq B$
shows *norm* $i \leq B *$ *measure lborel* (*cbox* a b)
⟨*proof*⟩

corollary *has-integral-bound-real'*:
fixes f :: *real* ⇒ *'b*::*real-normed-vector*
assumes $0 \leq B$
and f: (f *has-integral* i) (*cbox* a b)
and *finite* s
and $\forall x \in$(*cbox* a b)$-s$. *norm* (f x) $\leq B$
shows *norm* $i \leq B *$ *measure lborel* $\{a..b\}$
⟨*proof*⟩

lemma *integral-has-vector-derivative-continuous-at'*:
fixes f :: *real* ⇒ *'a*::*banach*
assumes *finite* s
and f: f *integrable-on* $\{a..b\}$
and x: $x \in \{a..b\} - s$
and *fx*: *continuous* (*at* x *within* ($\{a..b\} - s$)) f
shows ((λu. *integral* $\{a..u\}$ f) *has-vector-derivative* f x) (*at* x *within* ($\{a..b\} - s$))
⟨*proof*⟩

lemma *at-within-closed-interval-finite*:
fixes x::*real*
assumes $a < x$ $x < b$ $x \notin S$ *finite* S
shows (*at* x *within* $\{a..b\} - S$) = *at* x
⟨*proof*⟩

lemma *fundamental-theorem-of-calculus-interior-stronger'*:
fixes f :: *real* ⇒ *'a*::*banach*
assumes *finite* S
and $a \leq b$ $\bigwedge x.\ x \in \{a <..< b\} - S \Longrightarrow$ (f *has-vector-derivative* $f'(x)$) (*at* x *within* $\{a..b\} - S$)
and *continuous-on* $\{a .. b\}$ f
shows (f' *has-integral* (f $b - f$ a)) $\{a .. b\}$
⟨*proof*⟩

lemma *has-integral-substitution-general-*:
fixes f :: *real* $\Rightarrow$ *'a::euclidean-space* **and** g :: *real* $\Rightarrow$ *real*
assumes *s*: *finite s* **and** *le*: $a \leq b$
and *subset*: g ` $\{a..b\} \subseteq \{c..d\}$
and *f*: *f integrable-on* $\{c..d\}$ *continuous-on* $(\{c..d\} - (g$ ` $s))$ f
and g : *continuous-on* $\{a..b\}$ g *inj-on* g $(\{a..b\} \cup s)$
and *deriv* [*derivative-intros*]:
$\bigwedge x.\ x \in \{a..b\} - s \Longrightarrow$ (*g has-field-derivative* g' x) (*at* x *within* $\{a..b\}$)
shows $((\lambda x.\ g'\ x *_R f\ (g\ x))$ *has-integral* (*integral* $\{g\ a..g\ b\}$ f $-$ *integral* $\{g\ b..g\ a\}$ f)) $\{a..b\}$
⟨*proof*⟩

lemma *has-integral-substitution-general--*:
fixes f :: *real* $\Rightarrow$ *'a::euclidean-space* **and** g :: *real* $\Rightarrow$ *real*
assumes *s*: *finite s* **and** *le*: $a \leq b$ **and** *s-subset*: $s \subseteq \{a..b\}$
and *subset*: g ` $\{a..b\} \subseteq \{c..d\}$
and *f*: *f integrable-on* $\{c..d\}$ *continuous-on* $(\{c..d\} - (g$ ` $s))$ f
and g : *continuous-on* $\{a..b\}$ g *inj-on* g $\{a..b\}$
and *deriv* [*derivative-intros*]:
$\bigwedge x.\ x \in \{a..b\} - s \Longrightarrow$ (*g has-field-derivative* g' x) (*at* x *within* $\{a..b\}$)
shows $((\lambda x.\ g'\ x *_R f\ (g\ x))$ *has-integral* (*integral* $\{g\ a..g\ b\}$ f $-$ *integral* $\{g\ b..g\ a\}$ f)) $\{a..b\}$
⟨*proof*⟩

lemma *has-integral-substitution-general-'*:
fixes f :: *real* $\Rightarrow$ *'a::euclidean-space* **and** g :: *real* $\Rightarrow$ *real*
assumes *s*: *finite s* **and** *le*: $a \leq b$ **and** s': *finite* s'
and *subset*: g ` $\{a..b\} \subseteq \{c..d\}$
and *f*: *f integrable-on* $\{c..d\}$ *continuous-on* $(\{c..d\} - s')$ f
and g : *continuous-on* $\{a..b\}$ g $\forall x{\in}s'.$ *finite* $(g$ -` $\{x\})$ *surj-on* $s'$ $g$ *inj-on* $g$ $(\{a..b\} \cup ((s \cup g$ -` $s')))$
and *deriv* [*derivative-intros*]:
$\bigwedge x.\ x \in \{a..b\} - s \Longrightarrow$ (*g has-field-derivative* g' x) (*at* x *within* $\{a..b\}$)
shows $((\lambda x.\ g'\ x *_R f\ (g\ x))$ *has-integral* (*integral* $\{g\ a..g\ b\}$ f $-$ *integral* $\{g\ b..g\ a\}$ f)) $\{a..b\}$
⟨*proof*⟩

end
theory *Paths*
imports *Derivs General-Utils Integrals*
begin

lemma *reverse-subpaths-join*:
shows *subpath 1 (1 / 2) p +++ subpath (1 / 2) 0 p = reversepath p*
⟨*proof*⟩

definition *line-integral*:: $('a::euclidean\text{-}space \Rightarrow 'a::euclidean\text{-}space) \Rightarrow (('a)\ set)$ $\Rightarrow (real \Rightarrow 'a) \Rightarrow real$ **where**
$line\text{-}integral\ F\ basis\ g \equiv integral\ \{0\ ..\ 1\}\ (\lambda x.\ \sum b \in basis.\ (F(g\ x) \cdot b) * (vector\text{-}derivative\ g\ (at\ x\ within\ \{0..1\}) \cdot b))$

definition *line-integral-exists* **where**
$line\text{-}integral\text{-}exists\ F\ basis\ \gamma \equiv (\lambda x.\ \sum b \in basis.\ F(\gamma\ x) \cdot b * (vector\text{-}derivative\ \gamma\ (at\ x\ within\ \{0..1\}) \cdot b))\ integrable\text{-}on\ \{0..1\}$

lemma *line-integral-on-pair-straight-path*:
fixes $F::('a::euclidean\text{-}space) \Rightarrow 'a$ **and** $g :: real \Rightarrow real$ **and** γ
assumes *gamma-const*: $\forall x.\ \gamma(x) \cdot i = a$
and *gamma-smooth*: $\forall x \in \{0\ ..\ 1\}.\ \gamma\ differentiable\ at\ x$
shows $(line\text{-}integral\ F\ \{i\}\ \gamma) = 0\ (line\text{-}integral\text{-}exists\ F\ \{i\}\ \gamma)$
⟨*proof*⟩

lemma *line-integral-on-pair-path-strong*:
fixes $F::('a::euclidean\text{-}space) \Rightarrow ('a)$ **and**
$g::real \Rightarrow 'a$ **and**
$\gamma::(real \Rightarrow 'a)$ **and**
$i::'a$
assumes *i-norm-1*: $norm\ i = 1$ **and**
g-orthogonal-to-i: $\forall x.\ g(x) \cdot i = 0$ **and**
gamma-is-in-terms-of-i: $\gamma = (\lambda x.\ f(x) *_R i + g(f(x)))$ **and**
gamma-smooth: $\gamma\ piecewise\text{-}C1\text{-}differentiable\text{-}on\ \{0\ ..\ 1\}$ **and**
g-continuous-on-f: $continuous\text{-}on\ (f\ `\ \{0..1\})\ g$ **and**
path-start-le-path-end: $(pathstart\ \gamma) \cdot i \leq (pathfinish\ \gamma) \cdot i$ **and**
field-i-comp-cont: $continuous\text{-}on\ (path\text{-}image\ \gamma)\ (\lambda x.\ F\ x \cdot i)$
shows $line\text{-}integral\ F\ \{i\}\ \gamma$
$= integral\ (cbox\ ((pathstart\ \gamma) \cdot i)\ ((pathfinish\ \gamma) \cdot i))\ (\lambda f\text{-}var.\ (F\ (f\text{-}var *_R i + g(f\text{-}var)) \cdot i))$
$line\text{-}integral\text{-}exists\ F\ \{i\}\ \gamma$
⟨*proof*⟩

lemma *line-integral-on-pair-path*:
fixes $F::('a::euclidean\text{-}space) \Rightarrow ('a)$ **and**
$g::real \Rightarrow 'a$ **and**
$\gamma::(real \Rightarrow 'a)$ **and**
$i::'a$
assumes *i-norm-1*: $norm\ i = 1$ **and**
g-orthogonal-to-i: $\forall x.\ g(x) \cdot i = 0$ **and**
gamma-is-in-terms-of-i: $\gamma = (\lambda x.\ f(x) *_R i + g(f(x)))$ **and**
gamma-smooth: $\gamma\ C1\text{-}differentiable\text{-}on\ \{0\ ..\ 1\}$ **and**
g-continuous-on-f: $continuous\text{-}on\ (f\ `\ \{0..1\})\ g$ **and**
path-start-le-path-end: $(pathstart\ \gamma) \cdot i \leq (pathfinish\ \gamma) \cdot i$ **and**
field-i-comp-cont: $continuous\text{-}on\ (path\text{-}image\ \gamma)\ (\lambda x.\ F\ x \cdot i)$
shows $(line\text{-}integral\ F\ \{i\}\ \gamma)$
$= integral\ (cbox\ ((pathstart\ \gamma) \cdot i)\ ((pathfinish\ \gamma) \cdot i))\ (\lambda f\text{-}var.\ (F\ (f\text{-}var *_R i + g(f\text{-}var)) \cdot i))$

⟨*proof*⟩

lemma *content-box-cases*:
  measure lborel (*box a b*) = (*if* ∀ *i*∈*Basis*. *a*•*i* ≤ *b*•*i then prod* (λ*i*. *b*•*i* − *a*•*i*) *Basis else 0*)
  ⟨*proof*⟩

lemma *content-box-cbox*:
  shows *measure lborel* (*box a b*) = *measure lborel* (*cbox a b*)
  ⟨*proof*⟩

lemma *content-eq-0*: *measure lborel* (*box a b*) = *0* ⟷ (∃ *i*∈*Basis*. *b*•*i* ≤ *a*•*i*)
  ⟨*proof*⟩

lemma *content-pos-lt-eq*: *0* < *measure lborel* (*cbox a* (*b*::′*a*::*euclidean-space*)) ⟷ (∀ *i*∈*Basis*. *a*•*i* < *b*•*i*)
  ⟨*proof*⟩

lemma *content-lt-nz*: *0* < *measure lborel* (*box a b*) ⟷ *measure lborel* (*box a b*) ≠ *0*
  ⟨*proof*⟩

lemma *content-subset*: *cbox a b* ⊆ *box c d* ⟹ *measure lborel* (*cbox a b*) ≤ *measure lborel* (*box c d*)
  ⟨*proof*⟩

lemma *sum-content-null*:
  assumes *measure lborel* (*box a b*) = *0*
    and *p tagged-division-of* (*box a b*)
  shows *sum* (λ(*x*,*k*). *measure lborel k* $*_R$ *f x*) *p* = (*0*::′*a*::*real-normed-vector*)
⟨*proof*⟩

lemma *has-integral-null* [*intro*]: *measure lborel*(*box a b*) = *0* ⟹ (*f has-integral 0*) (*box a b*)
  ⟨*proof*⟩

lemma *line-integral-distrib*:
  assumes *line-integral-exists f basis g1*
    line-integral-exists f basis g2
    valid-path g1 valid-path g2
  shows *line-integral f basis* (*g1* +++ *g2*) = *line-integral f basis g1* + *line-integral f basis g2*
    line-integral-exists f basis (*g1* +++ *g2*)
⟨*proof*⟩

lemma *line-integral-exists-joinD1*:
  assumes *line-integral-exists f basis* (*g1* +++ *g2*) *valid-path g1*
  shows *line-integral-exists f basis g1*

⟨*proof*⟩

lemma *line-integral-exists-joinD2*:
assumes *line-integral-exists f basis* (*g1* +++ *g2*) *valid-path g2*
shows *line-integral-exists f basis g2*
⟨*proof*⟩

lemma *has-line-integral-on-reverse-path*:
assumes *g*: *valid-path g* **and** *int*:
$((\lambda x. \sum b \in basis.\ F\ (g\ x) \cdot b * (vector\text{-}derivative\ g\ (at\ x\ within\ \{0..1\}) \cdot b))$ $has\text{-}integral\ c)\{0..1\}$
shows $((\lambda x. \sum b \in basis.\ F\ ((reversepath\ g)\ x) \cdot b * (vector\text{-}derivative\ (reversepath\ g)\ (at\ x\ within\ \{0..1\}) \cdot b))\ has\text{-}integral\ -c)\{0..1\}$
⟨*proof*⟩

lemma *line-integral-on-reverse-path*:
assumes *valid-path* γ *line-integral-exists F basis* γ
shows *line-integral F basis* $\gamma = -$ (*line-integral F basis* (*reversepath* γ))
line-integral-exists F basis (*reversepath* γ)
⟨*proof*⟩

lemma *line-integral-exists-on-degenerate-path*:
assumes *finite basis*
shows *line-integral-exists F basis* ($\lambda x.\ c$)
⟨*proof*⟩

lemma *degenerate-path-is-valid-path*: *valid-path* ($\lambda x.\ c$)
⟨*proof*⟩

lemma *line-integral-degenerate-path*:
assumes *finite basis*
shows *line-integral F basis* ($\lambda x.\ c$) = *0*
⟨*proof*⟩

definition *point-path* **where**
point-path $\gamma \equiv \exists\, c.\ \gamma = (\lambda x.\ c)$

lemma *line-integral-point-path*:
assumes *point-path* γ
assumes *finite basis*
shows *line-integral F basis* γ = *0*
⟨*proof*⟩

lemma *line-integral-exists-point-path*:
assumes *finite basis point-path* γ
shows *line-integral-exists F basis* γ
⟨*proof*⟩

lemma *line-integral-exists-subpath*:

assumes *f*: *line-integral-exists f basis g* **and** *g*: *valid-path g*
and *uv*: $u \in \{0..1\}$ $v \in \{0..1\}$ $u \le v$
shows (*line-integral-exists f basis* (*subpath u v g*))
⟨*proof*⟩

type-synonym *path* = *real* ⇒ (*real* ∗ *real*)
type-synonym *one-cube* = (*real* ⇒ (*real* ∗ *real*))
type-synonym *one-chain* = (*int* ∗ *path*) *set*
type-synonym *two-cube* = (*real* ∗ *real*) ⇒ (*real* ∗ *real*)
type-synonym *two-chain* = *two-cube set*

definition *one-chain-line-integral* :: ((*real* ∗ *real*) ⇒ (*real* ∗ *real*)) => ((*real*∗*real*) *set*) ⇒ *one-chain* ⇒ *real* **where**
one-chain-line-integral F b C ≡ ($\sum$ (*k*,*g*)∈*C*. *k* ∗ (*line-integral F b g*))

definition *boundary-chain* **where**
boundary-chain s ≡ (∀ (*k*, *γ*) ∈ *s*. $k = 1 \lor k = -1$)

fun *coeff-cube-to-path*::(*int* ∗ *one-cube*) ⇒ *path*
where *coeff-cube-to-path* (*k*, *γ*) = (*if k* = *1 then γ else* (*reversepath γ*))

fun *rec-join* :: (*int*∗*path*) *list* ⇒ *path* **where**
rec-join [] = (λ*x*.*0*) |
rec-join [*oneC*] = *coeff-cube-to-path oneC* |
rec-join (*oneC* # *xs*) = *coeff-cube-to-path oneC* +++ (*rec-join xs*)

fun *valid-chain-list* **where**
valid-chain-list [] = *True* |
valid-chain-list [*oneC*] = *True* |
valid-chain-list (*oneC*#*l*) = (*pathfinish* (*coeff-cube-to-path* (*oneC*)) = *pathstart* (*rec-join l*) ∧ *valid-chain-list l*)

lemma *joined-is-valid*:
assumes *boundary-chain*: *boundary-chain* (*set l*) **and**
valid-path: ⋀*k γ*. (*k*, *γ*) ∈ *set l* ⟹ *valid-path γ* **and**
valid-chain-list-ass: *valid-chain-list l*
shows *valid-path* (*rec-join l*)
⟨*proof*⟩

lemma *pathstart-rec-join-1*:
pathstart (*rec-join* ((*1*, *γ*) # *l*)) = *pathstart γ*
⟨*proof*⟩

lemma *pathstart-rec-join-2*:
pathstart (*rec-join* ((−*1*, *γ*) # *l*)) = *pathstart* (*reversepath γ*)
⟨*proof*⟩

lemma *pathstart-rec-join*:
  pathstart (*rec-join* ((*1*, γ) # *l*)) = *pathstart* γ
  pathstart (*rec-join* ((−*1*, γ) # *l*)) = *pathstart* (*reversepath* γ)
  ⟨*proof*⟩

lemma *line-integral-exists-on-rec-join*:
  assumes *boundary-chain*: *boundary-chain* (*set l*) **and**
    valid-chain-list: *valid-chain-list l* **and**
    valid-path: ⋀*k* γ. (*k*, γ) ∈ *set l* ⟹ *valid-path* γ **and**
    line-integral-exists: ∀ (*k*, γ) ∈ *set l*. *line-integral-exists F basis* γ
  shows *line-integral-exists F basis* (*rec-join l*)
  ⟨*proof*⟩

lemma *line-integral-exists-rec-join-cons*:
  assumes *line-integral-exists F basis* (*rec-join* ((*1*,γ) # *l*))
    (⋀*k*′ γ′. (*k*′, γ′) ∈ *set* ((*1*,γ) # *l*) ⟹ *valid-path* γ′)
    finite basis
  shows *line-integral-exists F basis* (γ +++ (*rec-join l*))
⟨*proof*⟩

lemma *line-integral-exists-rec-join-cons-2*:
  assumes *line-integral-exists F basis* (*rec-join* ((−*1*,γ) # *l*))
    (⋀*k*′ γ′. (*k*′, γ′) ∈ *set* ((*1*,γ) # *l*) ⟹ *valid-path* γ′)
    finite basis
  shows *line-integral-exists F basis* ((*reversepath* γ) +++ (*rec-join l*))
⟨*proof*⟩

lemma *line-integral-exists-on-rec-join*′:
  assumes *boundary-chain*: *boundary-chain* (*set l*) **and**
    valid-chain-list: *valid-chain-list l* **and**
    valid-path: ⋀*k* γ. (*k*, γ) ∈ *set l* ⟹ *valid-path* γ **and**
    line-integral-exists: *line-integral-exists F basis* (*rec-join l*) **and**
    finite-basis: *finite basis*
  shows ∀ (*k*, γ) ∈ *set l*. *line-integral-exists F basis* γ
  ⟨*proof*⟩

inductive *chain-subdiv-path*
  where *I*: *chain-subdiv-path* γ (*set l*) **if** *distinct l rec-join l* = γ *valid-chain-list l*

lemma *valid-path-equiv-valid-chain-list*:
  assumes *path-eq-chain*: *chain-subdiv-path* γ *one-chain*
    and *boundary-chain one-chain* ∀ (*k*, γ) ∈ *one-chain*. *valid-path* γ
  shows *valid-path* γ
⟨*proof*⟩

lemma *line-integral-rec-join-cons*:
  assumes *line-integral-exists F basis* γ
    line-integral-exists F basis (*rec-join* ((*l*)))

($\bigwedge k'\ \gamma'$. $(k', \gamma') \in$ *set* $((1,\gamma)\ \#\ l) \Longrightarrow$ *valid-path* γ')
finite basis
shows *line-integral F basis* (*rec-join* $((1,\gamma)\ \#\ l)$) = *line-integral F basis* (γ +++ (*rec-join l*))
⟨*proof*⟩

lemma *line-integral-rec-join-cons-2*:
assumes *line-integral-exists F basis* γ
line-integral-exists F basis (*rec-join* ((*l*)))
($\bigwedge k'\ \gamma'$. $(k', \gamma') \in$ *set* $((-1,\gamma)\ \#\ l) \Longrightarrow$ *valid-path* γ')
finite basis
shows *line-integral F basis* (*rec-join* $((-1,\gamma)\ \#\ l)$) = *line-integral F basis* ((*reversepath* γ) +++ (*rec-join l*))
⟨*proof*⟩

lemma *one-chain-line-integral-rec-join*:
assumes *l-props*: *set l* = *one-chain distinct l valid-chain-list l* **and**
boundary-chain: *boundary-chain one-chain* **and**
line-integral-exists: $\forall$ (*k*::*int*, γ) $\in$ *one-chain*. *line-integral-exists F basis* γ **and**
valid-path: $\forall$ (*k*::*int*, γ) $\in$ *one-chain*. *valid-path* γ **and**
finite-basis: *finite basis*
shows *line-integral F basis* (*rec-join l*) = *one-chain-line-integral F basis one-chain*
⟨*proof*⟩

lemma *line-integral-on-path-eq-line-integral-on-equiv-chain*:
assumes *path-eq-chain*: *chain-subdiv-path* γ *one-chain* **and**
boundary-chain: *boundary-chain one-chain* **and**
line-integral-exists: $\forall$ (*k*::*int*, γ) $\in$ *one-chain*. *line-integral-exists F basis* γ **and**
valid-path: $\forall$ (*k*::*int*, γ) $\in$ *one-chain*. *valid-path* γ **and**
finite-basis: *finite basis*
shows *one-chain-line-integral F basis one-chain* = *line-integral F basis* γ
line-integral-exists F basis γ
valid-path γ
⟨*proof*⟩

lemma *line-integral-on-path-eq-line-integral-on-equiv-chain'*:
assumes *path-eq-chain*: *chain-subdiv-path* γ *one-chain* **and**
boundary-chain: *boundary-chain one-chain* **and**
line-integral-exists: *line-integral-exists F basis* γ **and**
valid-path: $\forall$ (*k*, γ) $\in$ *one-chain*. *valid-path* γ **and**
finite-basis: *finite basis*
shows *one-chain-line-integral F basis one-chain* = *line-integral F basis* γ
$\forall$ (*k*, γ) $\in$ *one-chain*. *line-integral-exists F basis* γ
⟨*proof*⟩

definition *chain-subdiv-chain* **where**
chain-subdiv-chain one-chain1 subdiv
$\equiv \exists f$. ($\bigcup$(*f* ‘ *one-chain1*)) = *subdiv* $\wedge$
($\forall$ *c*$\in$*one-chain1*. *chain-subdiv-path* (*coeff-cube-to-path c*) (*f c*)) $\wedge$

pairwise (λ *p p′*. *f p* $\cap$ *f p′* = {}) *one-chain1* $\wedge$
($\forall$ *x* $\in$ *one-chain1*. *finite* (*f x*))

lemma *chain-subdiv-chain-character*:
shows *chain-subdiv-chain one-chain1 subdiv* $\longleftrightarrow$
($\exists$ *f*. $\bigcup$(*f* ‘ *one-chain1*) = *subdiv* $\wedge$
($\forall$ (*k*, γ)$\in$*one-chain1*.
if k = *1*
then chain-subdiv-path γ (*f* (*k*, γ))
else chain-subdiv-path (*reversepath* γ) (*f* (*k*, γ))) $\wedge$
($\forall$ *p*$\in$*one-chain1*.
$\forall$ *p′*$\in$*one-chain1*. *p* $\neq$ *p′* $\longrightarrow$ *f p* $\cap$ *f p′* = {}) $\wedge$
($\forall$ *x*$\in$*one-chain1*. *finite* (*f x*)))
⟨*proof*⟩

lemma *chain-subdiv-chain-imp-finite-subdiv*:
assumes *finite one-chain1*
chain-subdiv-chain one-chain1 subdiv
shows *finite subdiv*
⟨*proof*⟩

lemma *valid-subdiv-imp-valid-one-chain*:
assumes *chain1-eq-chain2*: *chain-subdiv-chain one-chain1 subdiv* **and**
boundary-chain1: *boundary-chain one-chain1* **and**
boundary-chain2: *boundary-chain subdiv* **and**
valid-path: $\forall$ (*k*, γ) $\in$ *subdiv*. *valid-path* γ
shows $\forall$ (*k*, γ) $\in$ *one-chain1*. *valid-path* γ
⟨*proof*⟩

lemma *one-chain-line-integral-eq-line-integral-on-sudivision*:
assumes *chain1-eq-chain2*: *chain-subdiv-chain one-chain1 subdiv* **and**
boundary-chain1: *boundary-chain one-chain1* **and**
boundary-chain2: *boundary-chain subdiv* **and**
line-integral-exists-on-chain2: $\forall$ (*k*, γ) $\in$ *subdiv*. *line-integral-exists F basis* γ
and
valid-path: $\forall$ (*k*, γ) $\in$ *subdiv*. *valid-path* γ **and**
finite-chain1: *finite one-chain1* **and**
finite-basis: *finite basis*
shows *one-chain-line-integral F basis one-chain1* = *one-chain-line-integral F basis subdiv*
$\forall$ (*k*, γ) $\in$ *one-chain1*. *line-integral-exists F basis* γ
⟨*proof*⟩

lemma *one-chain-line-integral-eq-line-integral-on-sudivision′*:
assumes *chain1-eq-chain2*: *chain-subdiv-chain one-chain1 subdiv* **and**
boundary-chain1: *boundary-chain one-chain1* **and**
boundary-chain2: *boundary-chain subdiv* **and**
line-integral-exists-on-chain1: $\forall$ (*k*, γ) $\in$ *one-chain1*. *line-integral-exists F basis*
γ **and**

valid-path: ∀ (*k*, *γ*) ∈ *subdiv*. *valid-path* *γ* **and**
finite-chain1: *finite one-chain1* **and**
finite-basis: *finite basis*
shows *one-chain-line-integral F basis one-chain1* = *one-chain-line-integral F basis subdiv*
∀ (*k*, *γ*) ∈ *subdiv*. *line-integral-exists F basis γ*
⟨*proof*⟩

lemma *line-integral-sum-gen*:
assumes *finite-basis*:
finite basis **and**
line-integral-exists:
line-integral-exists F basis1 γ
line-integral-exists F basis2 γ **and**
basis-partition:
basis1 ∪ *basis2* = *basis basis1* ∩ *basis2* = {}
shows *line-integral F basis γ* = (*line-integral F basis1 γ*) + (*line-integral F basis2 γ*)
line-integral-exists F basis γ
⟨*proof*⟩

definition *common-boundary-sudivision-exists* **where**
common-boundary-sudivision-exists one-chain1 one-chain2 ≡
∃ *subdiv*. *chain-subdiv-chain one-chain1 subdiv* ∧
chain-subdiv-chain one-chain2 subdiv ∧
(∀ (*k*, *γ*) ∈ *subdiv*. *valid-path γ*)∧
boundary-chain subdiv

lemma *common-boundary-sudivision-commutative*:
(*common-boundary-sudivision-exists one-chain1 one-chain2*) = (*common-boundary-sudivision-exists one-chain2 one-chain1*)
⟨*proof*⟩

lemma *common-subdivision-imp-eq-line-integral*:
assumes (*common-boundary-sudivision-exists one-chain1 one-chain2*)
boundary-chain one-chain1
boundary-chain one-chain2
∀ (*k*, *γ*)∈*one-chain1*. *line-integral-exists F basis γ*
finite one-chain1
finite one-chain2
finite basis
shows *one-chain-line-integral F basis one-chain1* = *one-chain-line-integral F basis one-chain2*
∀ (*k*, *γ*)∈*one-chain2*. *line-integral-exists F basis γ*
⟨*proof*⟩

definition *common-sudiv-exists* **where**
common-sudiv-exists one-chain1 one-chain2 ≡
∃ *subdiv ps1 ps2*. *chain-subdiv-chain* (*one-chain1* − *ps1*) *subdiv* ∧

chain-subdiv-chain (*one-chain2* − *ps2*) *subdiv* ∧
(∀ (*k*, *γ*) ∈ *subdiv*. *valid-path* *γ*) ∧
(*boundary-chain subdiv*) ∧
(∀ (*k*, *γ*) ∈ *ps1*. *point-path* *γ*) ∧
(∀ (*k*, *γ*) ∈ *ps2*. *point-path* *γ*)

lemma *common-sudiv-exists-comm*:
shows *common-sudiv-exists C1 C2* = *common-sudiv-exists C2 C1*
⟨*proof*⟩

lemma *line-integral-degenerate-chain*:
assumes (∀ (*k*, *γ*) ∈ *chain*. *point-path* *γ*)
assumes *finite basis*
shows *one-chain-line-integral F basis chain* = *0*
⟨*proof*⟩

lemma *gen-common-subdiv-imp-common-subdiv*:
shows (*common-sudiv-exists one-chain1 one-chain2*) = (∃ *ps1 ps2*. (*common-boundary-sudivision-exists* (*one-chain1* − *ps1*) (*one-chain2* − *ps2*)) ∧ (∀ (*k*, *γ*)∈*ps1*. *point-path* *γ*) ∧ (∀ (*k*, *γ*)∈*ps2*. *point-path* *γ*))
⟨*proof*⟩

lemma *common-subdiv-imp-gen-common-subdiv*:
assumes (*common-boundary-sudivision-exists one-chain1 one-chain2*)
shows (*common-sudiv-exists one-chain1 one-chain2*)
⟨*proof*⟩

lemma *one-chain-line-integral-point-paths*:
assumes *finite one-chain*
assumes *finite basis*
assumes (∀ (*k*, *γ*)∈*ps*. *point-path* *γ*)
shows *one-chain-line-integral F basis* (*one-chain* − *ps*) = *one-chain-line-integral F basis* (*one-chain*)
⟨*proof*⟩

lemma *boundary-chain-diff*:
assumes *boundary-chain one-chain*
shows *boundary-chain* (*one-chain* − *s*)
⟨*proof*⟩

lemma *gen-common-subdivision-imp-eq-line-integral*:
assumes (*common-sudiv-exists one-chain1 one-chain2*)
boundary-chain one-chain1
boundary-chain one-chain2
∀ (*k*, *γ*)∈*one-chain1*. *line-integral-exists F basis* *γ*
finite one-chain1
finite one-chain2
finite basis
shows *one-chain-line-integral F basis one-chain1* = *one-chain-line-integral F*

basis one-chain2
  $\forall (k, \gamma)\in$*one-chain2*. *line-integral-exists F basis* γ
⟨*proof*⟩

lemma *common-sudiv-exists-refl*:
  assumes *common-sudiv-exists C1 C2*
  shows *common-sudiv-exists C2 C1*
  ⟨*proof*⟩

lemma *chain-subdiv-path-singleton*:
  shows *chain-subdiv-path* γ $\{(1,\gamma)\}$
⟨*proof*⟩

lemma *chain-subdiv-path-singleton-reverse*:
  shows *chain-subdiv-path* (*reversepath* γ) $\{(-1,\gamma)\}$
⟨*proof*⟩

lemma *chain-subdiv-chain-refl*:
  assumes *boundary-chain C*
  shows *chain-subdiv-chain C C*
  ⟨*proof*⟩

definition *reparam-weak* **where**
  reparam-weak $\gamma 1$ $\gamma 2 \equiv \exists \varphi.\ (\forall x\in\{0..1\}.\ \gamma 1\ x = (\gamma 2\ o\ \varphi)\ x) \wedge \varphi$ *piecewise-C1-differentiable-on* $\{0..1\} \wedge \varphi(0) = 0 \wedge \varphi(1) = 1 \wedge \varphi$ ‘ $\{0..1\} = \{0..1\}$

definition *reparam* **where**
  reparam $\gamma 1$ $\gamma 2 \equiv \exists \varphi.\ (\forall x\in\{0..1\}.\ \gamma 1\ x = (\gamma 2\ o\ \varphi)\ x) \wedge \varphi$ *piecewise-C1-differentiable-on* $\{0..1\} \wedge \varphi(0) = 0 \wedge \varphi(1) = 1 \wedge$ *bij-betw* $\varphi\ \{0..1\}\ \{0..1\} \wedge \varphi$ −‘ $\{0..1\} \subseteq \{0..1\}$ $\wedge$ $(\forall x\in\{0..1\}.$ *finite* $(\varphi$ −‘ $\{x\}))$

lemma *reparam-weak-eq-refl*:
  shows *reparam-weak* $\gamma 1$ $\gamma 1$
  ⟨*proof*⟩

lemma *line-integral-exists-smooth-one-base*:
  assumes γ *C1-differentiable-on* $\{0..1\}$
    continuous-on (*path-image* γ) $(\lambda x.\ F\ x \cdot b)$
  shows *line-integral-exists F* $\{b\}$ γ
⟨*proof*⟩

lemma *contour-integral-primitive-lemma*:
  fixes f :: *complex* ⇒ *complex* **and** g :: *real* ⇒ *complex*
  assumes $a \le b$
    and $\bigwedge x.\ x \in s \Longrightarrow$ (f *has-field-derivative* $f'\ x$) (*at* x *within* s)
    and g *piecewise-differentiable-on* $\{a..b\}$ $\bigwedge x.\ x \in \{a..b\} \Longrightarrow g\ x \in s$
   shows $((\lambda x.\ f'(g\ x) *$ *vector-derivative* g (*at* x *within* $\{a..b\}$))

has-integral $(f(g\ b) - f(g\ a)))$ $\{a..b\}$
⟨*proof*⟩

lemma *line-integral-primitive-lemma*:
fixes f :: $'a$::{*euclidean-space*,*real-normed-field*} ⇒ $'a$::{*euclidean-space*,*real-normed-field*}
and
g :: *real* ⇒ $'a$
assumes $\bigwedge(a::'a).\ a \in s \Longrightarrow$ (f *has-field-derivative* $(f'\ a)$) (*at* a *within* s)
and g *piecewise-differentiable-on* $\{0::real..1\}$ $\bigwedge x.\ x \in \{0..1\} \Longrightarrow g\ x \in s$
and *base-vec* ∈ *Basis*
shows $((\lambda x.\ ((f'(g\ x)) * ($*vector-derivative* g (*at* x *within* $\{0..1\}$))) · *base-vec*)
has-integral $(((f(g\ 1))$ · *base-vec* − $(f(g\ 0))$ · *base-vec*))) $\{0..1\}$
⟨*proof*⟩

lemma *reparam-eq-line-integrals*:
assumes *reparam*: *reparam* γ*1* γ*2* **and**
pw-smooth: γ*2* *piecewise-C1-differentiable-on* $\{0..1\}$ **and**
cont: *continuous-on* (*path-image* γ*2*) ($\lambda x.\ F\ x \cdot b$) **and**
line-integral-ex: *line-integral-exists* F $\{b\}$ γ*2*
shows *line-integral* F $\{b\}$ γ*1* = *line-integral* F $\{b\}$ γ*2*
line-integral-exists F $\{b\}$ γ*1*
⟨*proof*⟩

lemma *reparam-weak-eq-line-integrals*:
assumes *reparam-weak* γ*1* γ*2*
γ*2* *C1-differentiable-on* $\{0..1\}$
continuous-on (*path-image* γ*2*) ($\lambda x.\ F\ x \cdot b$)
shows *line-integral* F $\{b\}$ γ*1* = *line-integral* F $\{b\}$ γ*2*
line-integral-exists F $\{b\}$ γ*1*
⟨*proof*⟩

lemma *line-integral-sum-basis*:
assumes *finite* (*basis*::($'a$::*euclidean-space*) *set*) ∀ *b*∈*basis*. *line-integral-exists* F
$\{b\}$ γ
shows *line-integral* F *basis* γ = ($\sum$ *b*∈*basis*. *line-integral* F $\{b\}$ γ)
line-integral-exists F *basis* γ
⟨*proof*⟩

lemma *reparam-weak-eq-line-integrals-basis*:
assumes *reparam-weak* γ*1* γ*2*
γ*2* *C1-differentiable-on* $\{0..1\}$
∀ *b*∈*basis*. *continuous-on* (*path-image* γ*2*) ($\lambda x.\ F\ x \cdot b$)
finite basis
shows *line-integral* F *basis* γ*1* = *line-integral* F *basis* γ*2*
line-integral-exists F *basis* γ*1*
⟨*proof*⟩

lemma *reparam-eq-line-integrals-basis*:
assumes *reparam* γ*1* γ*2*

γ2 piecewise-C1-differentiable-on {0..1}
∀ b∈basis. continuous-on (path-image γ2) (λx. F x · b)
finite basis
∀ b∈basis. line-integral-exists F {b} γ2
shows *line-integral F basis γ1 = line-integral F basis γ2*
line-integral-exists F basis γ1
⟨*proof*⟩

lemma *line-integral-exists-smooth*:
assumes *γ C1-differentiable-on {0..1}*
∀ (b::'a::euclidean-space) ∈basis. continuous-on (path-image γ) (λx. F x · b)
finite basis
shows *line-integral-exists F basis γ*
⟨*proof*⟩

lemma *smooth-path-imp-reverse*:
assumes *g C1-differentiable-on {0..1}*
shows *(reversepath g) C1-differentiable-on {0..1}*
⟨*proof*⟩

lemma *piecewise-smooth-path-imp-reverse*:
assumes *g piecewise-C1-differentiable-on {0..1}*
shows *(reversepath g) piecewise-C1-differentiable-on {0..1}*
⟨*proof*⟩

definition *chain-reparam-weak-chain* **where**
chain-reparam-weak-chain one-chain1 one-chain2 ≡
∃f. bij f ∧ f ' one-chain1 = one-chain2 ∧ (∀ (k,γ)∈one-chain1. if k = fst (f(k,γ)) then reparam-weak γ (snd (f(k,γ))) else reparam-weak γ (reversepath (snd (f(k,γ)))))

lemma *chain-reparam-weak-chain-line-integral*:
assumes *chain-reparam-weak-chain one-chain1 one-chain2*
∀ (k2,γ2)∈one-chain2. γ2 C1-differentiable-on {0..1}
∀ (k2,γ2)∈one-chain2.∀ b∈basis. continuous-on (path-image γ2) (λx. F x · b)
finite basis
and *bound1*: *boundary-chain one-chain1*
and *bound2*: *boundary-chain one-chain2*
shows *one-chain-line-integral F basis one-chain1 = one-chain-line-integral F basis one-chain2*
∀ (k, γ)∈one-chain1. line-integral-exists F basis γ
⟨*proof*⟩

definition *chain-reparam-chain* **where**
chain-reparam-chain one-chain1 one-chain2 ≡
∃f. bij f ∧ f ' one-chain1 = one-chain2 ∧ (∀ (k,γ)∈one-chain1. if k = fst (f(k,γ)) then reparam γ (snd (f(k,γ))) else reparam γ (reversepath (snd (f(k,γ)))))

definition *chain-reparam-weak-path*::((*real*) ⇒ (*real* ∗ *real*)) ⇒ ((*int* ∗ ((*real*) ⇒

(*real* ∗ *real*))) *set*) ⇒ *bool* **where**
chain-reparam-weak-path γ *one-chain*
≡ ∃ *l*. *set l* = *one-chain* ∧ *distinct l* ∧ *reparam* γ (*rec-join l*) ∧ *valid-chain-list l* ∧ *l* ≠ []

lemma *chain-reparam-chain-line-integral*:
assumes *chain-reparam-chain one-chain1 one-chain2*
∀ (*k2*,γ*2*)∈*one-chain2*. γ*2 piecewise-C1-differentiable-on* {*0..1*}
∀ (*k2*,γ*2*)∈*one-chain2*.∀ *b*∈*basis*. *continuous-on* (*path-image* γ*2*) (λ*x*. *F x* · *b*)
finite basis
and *bound1*: *boundary-chain one-chain1*
and *bound2*: *boundary-chain one-chain2*
and *line*: ∀ (*k2*,γ*2*)∈*one-chain2*. (∀ *b*∈*basis*. *line-integral-exists F* {*b*} γ*2*)
shows *one-chain-line-integral F basis one-chain1* = *one-chain-line-integral F basis one-chain2*
∀ (*k*, γ)∈*one-chain1*. *line-integral-exists F basis* γ
⟨*proof*⟩

lemma *path-image-rec-join*:
fixes γ::*real* ⇒ (*real* × *real*)
fixes *k*::*int*
fixes *l*
shows ⋀*k* γ. (*k*, γ) ∈ *set l* ⟹ *valid-chain-list l* ⟹ *path-image* γ ⊆ *path-image* (*rec-join l*)
⟨*proof*⟩

lemma *path-image-rec-join-2*:
fixes *l*
shows *l* ≠ [] ⟹ *valid-chain-list l* ⟹ *path-image* (*rec-join l*) ⊆ (⋃ (*k*, γ) ∈ *set l*. *path-image* γ)
⟨*proof*⟩

lemma *continuous-on-closed-UN*:
assumes *finite S*
shows ((⋀*s*. *s* ∈ *S* ⟹ *closed s*) ⟹ (⋀*s*. *s* ∈ *S* ⟹ *continuous-on s f*) ⟹ *continuous-on* (⋃ *S*) *f*)
⟨*proof*⟩

lemma *chain-reparam-weak-path-line-integral*:
assumes *path-eq-chain*: *chain-reparam-weak-path* γ *one-chain* **and**
boundary-chain: *boundary-chain one-chain* **and**
line-integral-exists: ∀ *b*∈*basis*. ∀ (*k*::*int*, γ) ∈ *one-chain*. *line-integral-exists F* {*b*} γ **and**
valid-path: ∀ (*k*::*int*, γ) ∈ *one-chain*. *valid-path* γ **and**
finite-basis: *finite basis* **and**
cont: ∀ *b*∈*basis*. ∀ (*k*,γ*2*)∈*one-chain*. *continuous-on* (*path-image* γ*2*) (λ*x*. *F x* · *b*) **and**
finite-one-chain: *finite one-chain*
shows *line-integral F basis* γ = *one-chain-line-integral F basis one-chain*

line-integral-exists F basis γ

⟨*proof*⟩

definition *chain-reparam-chain′* **where**
chain-reparam-chain′ one-chain1 subdiv
$\equiv \exists f.$ (($\bigcup$(*f* ‘ *one-chain1*)) = *subdiv*) $\wedge$
($\forall$ *cube* $\in$ *one-chain1*. *chain-reparam-weak-path* (*rec-join* [*cube*]) (*f cube*)) $\wedge$
($\forall$ *p*$\in$*one-chain1*. $\forall$ *p′*$\in$*one-chain1*. $p \neq p' \longrightarrow f\ p \cap f\ p' = \{\}$) $\wedge$
($\forall$ *x* $\in$ *one-chain1*. *finite* (*f x*))

lemma *chain-reparam-chain′-imp-finite-subdiv*:
assumes *finite one-chain1*
chain-reparam-chain′ one-chain1 subdiv
shows *finite subdiv*
⟨*proof*⟩

lemma *chain-reparam-chain′-line-integral*:
assumes *chain1-eq-chain2*: *chain-reparam-chain′ one-chain1 subdiv* **and**
boundary-chain1: *boundary-chain one-chain1* **and**
boundary-chain2: *boundary-chain subdiv* **and**
line-integral-exists-on-chain2: $\forall$ *b*$\in$*basis*. $\forall$ (*k*::*int*, γ) $\in$ *subdiv*. *line-integral-exists F* {*b*} γ **and**
valid-path: $\forall$ (*k*, γ) $\in$ *subdiv*. *valid-path* γ **and**
valid-path-2: $\forall$ (*k*, γ) $\in$ *one-chain1*. *valid-path* γ **and**
finite-chain1: *finite one-chain1* **and**
finite-basis: *finite basis* **and**
cont-field: $\forall$ *b*$\in$*basis*. $\forall$ (*k*, γ*2*)$\in$*subdiv*. *continuous-on* (*path-image* γ*2*) (λx. *F x* $\cdot$ *b*)
shows *one-chain-line-integral F basis one-chain1* = *one-chain-line-integral F basis subdiv*
$\forall$ (*k*, γ) $\in$ *one-chain1*. *line-integral-exists F basis* γ
⟨*proof*⟩

lemma *chain-reparam-chain′-line-integral-smooth-cubes*:
assumes *chain-reparam-chain′ one-chain1 one-chain2*
$\forall$ (*k2*,γ*2*)$\in$*one-chain2*. γ*2 C1-differentiable-on* {*0..1*}
$\forall$ *b*$\in$*basis*.$\forall$ (*k2*,γ*2*)$\in$*one-chain2*. *continuous-on* (*path-image* γ*2*) (λx. *F x* $\cdot$ *b*)
finite basis
finite one-chain1
boundary-chain one-chain1
boundary-chain one-chain2
$\forall$ (*k*,γ)$\in$*one-chain1*. *valid-path* γ
shows *one-chain-line-integral F basis one-chain1* = *one-chain-line-integral F basis one-chain2*
$\forall$ (*k*, γ)$\in$*one-chain1*. *line-integral-exists F basis* γ
⟨*proof*⟩

lemma *chain-subdiv-path-pathimg-subset*:
  assumes *chain-subdiv-path* γ *subdiv*
  shows $\forall (k',\gamma')\in$*subdiv*. (*path-image* γ') $\subseteq$ *path-image* γ
  ⟨*proof*⟩

lemma *reparam-path-image*:
  assumes *reparam* $\gamma 1$ $\gamma 2$
  shows *path-image* $\gamma 1$ = *path-image* $\gamma 2$
  ⟨*proof*⟩

lemma *chain-reparam-weak-path-pathimg-subset*:
  assumes *chain-reparam-weak-path* γ *subdiv*
  shows $\forall (k',\gamma')\in$*subdiv*. (*path-image* γ') $\subseteq$ *path-image* γ
  ⟨*proof*⟩

lemma *chain-subdiv-chain-pathimg-subset'*:
  assumes *chain-subdiv-chain one-chain subdiv*
  assumes $(k,\gamma)\in$ *subdiv*
  shows $\exists k'\ \gamma'.\ (k',\gamma')\in$ *one-chain* $\wedge$ *path-image* $\gamma \subseteq$ *path-image* γ'
  ⟨*proof*⟩

lemma *chain-subdiv-chain-pathimg-subset*:
  assumes *chain-subdiv-chain one-chain subdiv*
  shows $\bigcup$ (*path-image* ‘ $\{\gamma.\ \exists k.\ (k,\gamma)\in$ *subdiv*$\}$) $\subseteq \bigcup$ (*path-image* ‘ $\{\gamma.\ \exists k.\ (k,\gamma)\in$ *one-chain*$\}$)
  ⟨*proof*⟩

lemma *chain-reparam-chain'-pathimg-subset'*:
  assumes *chain-reparam-chain' one-chain subdiv*
  assumes $(k,\gamma)\in$ *subdiv*
  shows $\exists k'\ \gamma'.\ (k',\gamma')\in$ *one-chain* $\wedge$ *path-image* $\gamma \subseteq$ *path-image* γ'
  ⟨*proof*⟩

definition *common-reparam-exists*:: (*int* × (*real* ⇒ *real* × *real*)) *set* ⇒ (*int* × (*real* ⇒ *real* × *real*)) *set* ⇒ *bool* **where**
    common-reparam-exists one-chain1 one-chain2 ≡
    (∃ *subdiv ps1 ps2*.
    chain-reparam-chain' (*one-chain1* − *ps1*) *subdiv* ∧
    chain-reparam-chain' (*one-chain2* − *ps2*) *subdiv* ∧
    (∀ (*k*, γ)∈*subdiv*. γ *C1-differentiable-on* {*0..1*}) ∧
    boundary-chain subdiv ∧
    (∀ (*k*, γ)∈*ps1*. *point-path* γ) ∧
    (∀ (*k*, γ)∈*ps2*. *point-path* γ))

lemma *common-reparam-exists-imp-eq-line-integral*:
  assumes *finite-basis*: *finite basis* **and**
    finite one-chain1
    finite one-chain2
    boundary-chain (*one-chain1*::(*int* × (*real* ⇒ *real* × *real*)) *set*)

boundary-chain (*one-chain2*::(*int* × (*real* ⇒ *real* × *real*)) *set*)
∀ (*k2*, *γ2*)∈*one-chain2*. ∀ *b*∈*basis*. *continuous-on* (*path-image* *γ2*) (λ*x*. *F x* · *b*)
(*common-reparam-exists one-chain1 one-chain2*)
(∀ (*k*, *γ*)∈*one-chain1*. *valid-path* *γ*)
(∀ (*k*, *γ*)∈*one-chain2*. *valid-path* *γ*)
shows *one-chain-line-integral F basis one-chain1* = *one-chain-line-integral F basis one-chain2*
∀ (*k*, *γ*)∈*one-chain1*. *line-integral-exists F basis γ*
⟨*proof*⟩

definition *subcube* :: *real* ⇒ *real* ⇒ (*int* × (*real* ⇒ *real* × *real*)) ⇒ (*int* × (*real* ⇒ *real* × *real*)) **where**
subcube a b cube = (*fst cube*, *subpath a b* (*snd cube*))

lemma *subcube-valid-path*:
assumes *valid-path* (*snd cube*) *a* ∈ {*0*..*1*} *b* ∈ {*0*..*1*}
shows *valid-path* (*snd* (*subcube a b cube*))
⟨*proof*⟩

end
theory *Green*
imports *Paths Derivs Integrals General-Utils*
begin

lemma *frontier-Un-subset-Un-frontier*:
frontier (*s* ∪ *t*) ⊆ (*frontier s*) ∪ (*frontier t*)
⟨*proof*⟩

definition *has-partial-derivative*:: ((′*a*::*euclidean-space*) ⇒ ′*b*::*euclidean-space*) ⇒ ′*a* ⇒ (′*a* ⇒ ′*b*) ⇒ (′*a*) ⇒ *bool* **where**
has-partial-derivative F base-vec F′ a
≡ ((λ*x*::′*a*::*euclidean-space*. *F*((*a* − ((*a* · *base-vec*) $*_R$ *base-vec*)) + (*x* · *base-vec*) $*_R$ *base-vec*))
has-derivative F′) (*at a*)

definition *has-partial-vector-derivative*:: ((′*a*::*euclidean-space*) ⇒ ′*b*::*euclidean-space*) ⇒ ′*a* ⇒ (′*b*) ⇒ (′*a*) ⇒ *bool* **where**
has-partial-vector-derivative F base-vec F′ a
≡ ((λ*x*. *F*((*a* − ((*a* · *base-vec*) $*_R$ *base-vec*)) + *x* $*_R$ *base-vec*))
has-vector-derivative F′) (*at* (*a* · *base-vec*))

definition *partially-vector-differentiable* **where**
partially-vector-differentiable F base-vec p ≡ (∃ *F′*. *has-partial-vector-derivative F base-vec F′ p*)

definition *partial-vector-derivative*:: ((′*a*::*euclidean-space*) ⇒ ′*b*::*euclidean-space*) ⇒ ′*a* ⇒ ′*a* ⇒ ′*b* **where**
partial-vector-derivative F base-vec a

$\equiv$ (*vector-derivative* (λx. *F*((*a* $-$ ((*a* $\cdot$ *base-vec*) $*_R$ *base-vec*)) + *x* $*_R$ *base-vec*)) (*at* (*a* $\cdot$ *base-vec*)))

lemma *partial-vector-derivative-works*:
assumes *partially-vector-differentiable F base-vec a*
shows *has-partial-vector-derivative F base-vec* (*partial-vector-derivative F base-vec a*) *a*
⟨*proof*⟩

lemma *fundamental-theorem-of-calculus-partial-vector*:
fixes *a b*:: *real* **and**
F:: ('*a*::*euclidean-space* $\Rightarrow$ '*b*::*euclidean-space*) **and**
i:: '*a* **and**
j:: '*b* **and**
F-j-i:: ('*a*::*euclidean-space* $\Rightarrow$ *real*)
assumes *a-leq-b*: $a \leq b$ **and**
Base-vecs: $i \in$ *Basis* $j \in$ *Basis* **and**
no-i-component: $c \cdot i = 0$ **and**
has-partial-deriv: $\forall p \in D$. *has-partial-vector-derivative* (λx. (*F x*) $\cdot$ *j*) *i* (*F-j-i p*) *p* **and**
domain-subset-of-D: $\{x *_R i + c \mid x.\ a \leq x \wedge x \leq b\} \subseteq D$
shows ((λx. *F-j-i*($x *_R i + c$)) *has-integral*
F($b *_R i + c$) $\cdot$ *j* $-$ *F*($a *_R i + c$) $\cdot$ *j*) (*cbox a b*)
⟨*proof*⟩

lemma *fundamental-theorem-of-calculus-partial-vector-gen*:
fixes *k1 k2*:: *real* **and**
F:: ('*a*::*euclidean-space* $\Rightarrow$ '*b*::*euclidean-space*) **and**
i:: '*a* **and**
F-i:: ('*a*::*euclidean-space* $\Rightarrow$ '*b*)
assumes *a-leq-b*: $k1 \leq k2$ **and**
unit-len: $i \cdot i = 1$ **and**
no-i-component: $c \cdot i = 0$ **and**
has-partial-deriv: $\forall p \in D$. *has-partial-vector-derivative F i* (*F-i p*) *p* **and**
domain-subset-of-D: $\{v.\ \exists x.\ k1 \leq x \wedge x \leq k2 \wedge v = x *_R i + c\} \subseteq D$
shows ((λx. *F-i*($x *_R i + c$)) *has-integral*
F($k2 *_R i + c$) $-$ *F*($k1 *_R i + c$)) (*cbox k1 k2*)
⟨*proof*⟩

lemma *add-scale-img*:
assumes $a < b$ **shows** (λx::*real*. $a + (b - a) * x$) ‘ {*0* .. *1*} = {*a* .. *b*}
⟨*proof*⟩

lemma *add-scale-img'*:
assumes $a \leq b$
shows (λx::*real*. $a + (b - a) * x$) ‘ {*0* .. *1*} = {*a* .. *b*}
⟨*proof*⟩

definition *analytically-valid*:: '*a*::*euclidean-space set* $\Rightarrow$ ('*a* $\Rightarrow$ '*b*::{*euclidean-space*,*times*,*zero-neq-one*})

$\Rightarrow$ *'a* $\Rightarrow$ *bool* **where**
analytically-valid s F i $\equiv$
($\forall$ *a* $\in$ *s. partially-vector-differentiable F i a*) $\wedge$
continuous-on s F $\wedge$ — TODO: should we replace this with saying that *F* is partially diffrerentiable on *Dy*,
— i.e. there is a partial derivative on every dimension
integrable lborel (λ*p.* (*partial-vector-derivative F i*) *p* $*$ *indicator s p*) $\wedge$
(λ*x. integral UNIV* (λ*y.* (*partial-vector-derivative F i* (*y* $*_R$ *i* + *x* $*_R$ ($\sum$ *b* $\in$(*Basis* $-$ $\{i\}$). *b*)))
$*$ (*indicator s* (*y* $*_R$ *i* + *x* $*_R$ ($\sum$ *b* $\in$ *Basis* $-$ $\{i\}$. *b*))))) $\in$ *borel-measurable lborel*

lemma *analytically-valid-imp-part-deriv-integrable-on*:
assumes *analytically-valid* (*s*::(*real*$*$*real*) *set*) (*f*::(*real*$*$*real*)$\Rightarrow$ *real*) *i*
shows (*partial-vector-derivative f i*) *integrable-on s*
$\langle$*proof*$\rangle$

definition *typeII-twoCube* :: ((*real* $*$ *real*) $\Rightarrow$ (*real* $*$ *real*)) $\Rightarrow$ *bool* **where**
typeII-twoCube twoC
$\equiv$ $\exists$ *a b g1 g2*. *a* < *b* $\wedge$ ($\forall$ *x* $\in$ $\{a..b\}$. *g2 x* $\leq$ *g1 x*) $\wedge$
twoC = (λ(*y*, *x*). ((*1* $-$ *y*) $*$ (*g2* ((*1*$-$*x*)$*$*a* + *x*$*$*b*)) + *y* $*$ (*g1* ((*1*$-$*x*)$*$*a* + *x*$*$*b*)),
(*1*$-$*x*)$*$*a* + *x*$*$*b*)) $\wedge$
g1 piecewise-C1-differentiable-on $\{a .. b\}$ $\wedge$
g2 piecewise-C1-differentiable-on $\{a .. b\}$

abbreviation *unit-cube* **where** *unit-cube* $\equiv$ *cbox* (*0*,*0*) (*1*::*real*,*1*::*real*)

definition *cubeImage*:: *two-cube* $\Rightarrow$ ((*real*$*$*real*) *set*) **where**
cubeImage twoC $\equiv$ (*twoC* ' *unit-cube*)

lemma *typeII-twoCubeImg*:
assumes *typeII-twoCube twoC*
shows $\exists$ *a b g1 g2*. *a* < *b* $\wedge$ ($\forall$ *x* $\in$ $\{a .. b\}$. *g2 x* $\leq$ *g1 x*) $\wedge$
cubeImage twoC = $\{(y,x).\ x \in \{a..b\} \wedge y \in \{g2\ x\ ..\ g1\ x\}\}$
$\wedge$ *twoC* = (λ(*y*, *x*). ((*1* $-$ *y*) $*$ *g2* ((*1* $-$ *x*) $*$ *a* + *x* $*$ *b*) + *y* $*$ *g1* ((*1* $-$ *x*) $*$ *a* + *x* $*$ *b*), (*1* $-$ *x*) $*$ *a* + *x* $*$ *b*))
$\wedge$ *g1 piecewise-C1-differentiable-on* $\{a .. b\}$ $\wedge$ *g2 piecewise-C1-differentiable-on* $\{a .. b\}$
$\langle$*proof*$\rangle$

definition *horizontal-boundary* :: *two-cube* $\Rightarrow$ *one-chain* **where**
horizontal-boundary twoC $\equiv$ $\{(1,\ (\lambda x.\ twoC(x,0))),\ (-1,\ (\lambda x.\ twoC(x,1)))\}$

definition *vertical-boundary* :: *two-cube* $\Rightarrow$ *one-chain* **where**

vertical-boundary twoC ≡ $\{(-1, (\lambda y.\ twoC(0,y))), (1, (\lambda y.\ twoC(1,y)))\}$

definition *boundary* :: *two-cube* ⇒ *one-chain* **where**
boundary twoC ≡ *horizontal-boundary twoC* ∪ *vertical-boundary twoC*

definition *valid-two-cube* **where**
valid-two-cube twoC ≡ *card* (*boundary twoC*) = *4*

definition *two-chain-integral*:: *two-chain* ⇒ ((*real*∗*real*)⇒(*real*)) ⇒ *real* **where**
two-chain-integral twoChain F ≡ $\sum$ *C*∈*twoChain*. (*integral* (*cubeImage C*) *F*)

definition *valid-two-chain* **where**
valid-two-chain twoChain ≡ (∀ *twoCube* ∈ *twoChain*. *valid-two-cube twoCube*) ∧ *pairwise* (λ*c1 c2*. ((*boundary c1*) ∩ (*boundary c2*)) = {}) *twoChain* ∧ *inj-on cubeImage twoChain*

definition *two-chain-boundary*:: *two-chain* ⇒ *one-chain* **where**
two-chain-boundary twoChain == $\bigcup$(*boundary* ‘ *twoChain*)

definition *gen-division* **where**
gen-division s S ≡ (*finite S* ∧ ($\bigcup$*S* = *s*) ∧ *pairwise* (λ*X Y*. *negligible* (*X* ∩ *Y*)) *S*)

definition *two-chain-horizontal-boundary*:: *two-chain* ⇒ *one-chain* **where**
two-chain-horizontal-boundary twoChain ≡ $\bigcup$(*horizontal-boundary* ‘ *twoChain*)

definition *two-chain-vertical-boundary*:: *two-chain* ⇒ *one-chain* **where**
two-chain-vertical-boundary twoChain ≡ $\bigcup$(*vertical-boundary* ‘ *twoChain*)

definition *only-horizontal-division* **where**
only-horizontal-division one-chain two-chain
  ≡ ∃ $\mathcal{H}$ $\mathcal{V}$. *finite* $\mathcal{H}$ ∧ *finite* $\mathcal{V}$ ∧
    (∀ (*k*,γ) ∈ $\mathcal{H}$.
      (∃ (*k*′, γ′) ∈ *two-chain-horizontal-boundary two-chain*.
        (∃ *a* ∈ {*0*..*1*}. ∃ *b* ∈ {*0*..*1*}. *a* ≤ *b* ∧ *subpath a b* γ′ = γ))) ∧
    (*common-sudiv-exists* (*two-chain-vertical-boundary two-chain*) $\mathcal{V}$
    ∨ *common-reparam-exists* $\mathcal{V}$ (*two-chain-vertical-boundary two-chain*)) ∧
    boundary-chain $\mathcal{V}$ ∧
    one-chain = $\mathcal{H}$ ∪ $\mathcal{V}$ ∧ (∀ (*k*,γ)∈$\mathcal{V}$. *valid-path* γ)

lemma *sum-zero-set*:
assumes ∀ *x* ∈ *s*. *f x* = *0* *finite s finite t*
shows *sum f* (*s* ∪ *t*) = *sum f t*
⟨*proof*⟩

abbreviation *valid-typeII-division s twoChain* ≡ ((∀ *twoCube* ∈ *twoChain*. *typeII-twoCube twoCube*) ∧

(*gen-division s* (*cubeImage* ‘ *twoChain*)) ∧
(*valid-two-chain twoChain*))

lemma *two-chain-vertical-boundary-is-boundary-chain*:
shows *boundary-chain* (*two-chain-vertical-boundary twoChain*)
⟨*proof*⟩

lemma *two-chain-horizontal-boundary-is-boundary-chain*:
shows *boundary-chain* (*two-chain-horizontal-boundary twoChain*)
⟨*proof*⟩

definition *typeI-twoCube* :: *two-cube* ⇒ *bool* **where**
typeI-twoCube (*twoC*::*two-cube*)
≡ ∃ *a b g1 g2*. *a* < *b* ∧ (∀ *x* ∈ {*a*..*b*}. *g2 x* ≤ *g1 x*) ∧
twoC = (λ(*x*,*y*). ((*1*−*x*)∗*a* + *x*∗*b*,
(*1* − *y*) ∗ (*g2* ((*1*−*x*)∗*a* + *x*∗*b*)) + *y* ∗ (*g1* ((*1*−*x*)∗*a* + *x*∗*b*)))) ∧
g1 piecewise-C1-differentiable-on {*a*..*b*} ∧
g2 piecewise-C1-differentiable-on {*a*..*b*}

lemma *typeI-twoCubeImg*:
assumes *typeI-twoCube twoC*
shows ∃ *a b g1 g2*. *a* < *b* ∧ (∀ *x* ∈ {*a* .. *b*}. *g2 x* ≤ *g1 x*) ∧
cubeImage twoC = {(*x*,*y*). *x* ∈ {*a*..*b*} ∧ *y* ∈ {*g2 x* .. *g1 x*}} ∧
twoC = (λ(*x*, *y*). ((*1* − *x*) ∗ *a* + *x* ∗ *b*, (*1* − *y*) ∗ *g2* ((*1* − *x*) ∗ *a* + *x* ∗ *b*) + *y* ∗ *g1* ((*1* − *x*) ∗ *a* + *x* ∗ *b*))) ∧
g1 piecewise-C1-differentiable-on {*a* .. *b*} ∧ *g2 piecewise-C1-differentiable-on* {*a* .. *b*}
⟨*proof*⟩

lemma *typeI-cube-explicit-spec*:
assumes *typeI-twoCube twoC*
shows ∃ *a b g1 g2*. *a* < *b* ∧ (∀ *x* ∈ {*a* .. *b*}. *g2 x* ≤ *g1 x*) ∧
cubeImage twoC = {(*x*,*y*). *x* ∈ {*a*..*b*} ∧ *y* ∈ {*g2 x* .. *g1 x*}}
∧ *twoC* = (λ(*x*, *y*). ((*1* − *x*) ∗ *a* + *x* ∗ *b*, (*1* − *y*) ∗ *g2* ((*1* − *x*) ∗ *a* + *x* ∗ *b*) + *y* ∗ *g1* ((*1* − *x*) ∗ *a* + *x* ∗ *b*)))
∧ *g1 piecewise-C1-differentiable-on* {*a* .. *b*} ∧ *g2 piecewise-C1-differentiable-on* {*a* .. *b*}
∧ (λ*x*. *twoC*(*x*, *0*)) = (λ*x*. (*a* + (*b* − *a*) ∗ *x*, *g2* (*a* + (*b* − *a*) ∗ *x*)))
∧ (λ*y*. *twoC*(*1*, *y*)) = (λ*x*. (*b*, *g2 b* + *x* $*_R$ (*g1 b* − *g2 b*)))
∧ (λ*x*. *twoC*(*x*, *1*)) = (λ*x*. (*a* + (*b* − *a*) ∗ *x*, *g1* (*a* + (*b* − *a*) ∗ *x*)))
∧ (λ*y*. *twoC*(*0*, *y*)) = (λ*x*. (*a*, *g2 a* + *x* $*_R$ (*g1 a* − *g2 a*)))
⟨*proof*⟩

lemma *typeI-twoCube-smooth-edges*:
assumes *typeI-twoCube twoC*
(*k*,γ) ∈ *boundary twoC*

shows γ *piecewise-C1-differentiable-on* $\{0..1\}$
⟨*proof*⟩

lemma *two-chain-integral-eq-integral-divisable*:
assumes *f-integrable*: $\forall$ *twoCube* $\in$ *twoChain*. *F integrable-on cubeImage twoCube*
and
gen-division: *gen-division s* (*cubeImage* ‘ *twoChain*) **and**
valid-two-chain: *valid-two-chain twoChain*
shows *integral s F* = *two-chain-integral twoChain F*
⟨*proof*⟩

definition *only-vertical-division* **where**
only-vertical-division one-chain two-chain ≡
$\exists\,\mathcal{V}\ \mathcal{H}$. *finite* $\mathcal{H}$ $\wedge$ *finite* $\mathcal{V}$ $\wedge$
($\forall\,(k,\gamma) \in \mathcal{V}$.
($\exists\,(k',\gamma') \in$ *two-chain-vertical-boundary two-chain*.
($\exists\, a \in \{0..1\}$. $\exists\, b \in \{0..1\}$. $a \leq b \wedge$ *subpath a b* $\gamma' = \gamma$))) $\wedge$
(*common-sudiv-exists* (*two-chain-horizontal-boundary two-chain*) $\mathcal{H}$
$\vee$ *common-reparam-exists* $\mathcal{H}$ (*two-chain-horizontal-boundary two-chain*))
$\wedge$
boundary-chain $\mathcal{H}$ $\wedge$ *one-chain* = $\mathcal{V} \cup \mathcal{H}$ $\wedge$
($\forall\,(k,\gamma)\in\mathcal{H}$. *valid-path* γ)

abbreviation *valid-typeI-division s twoChain*
≡ ($\forall$ *twoCube* $\in$ *twoChain*. *typeI-twoCube twoCube*) $\wedge$
gen-division s (*cubeImage* ‘ *twoChain*) $\wedge$ *valid-two-chain twoChain*

lemma *field-cont-on-typeI-region-cont-on-edges*:
assumes *typeI-twoC*: *typeI-twoCube twoC*
and *field-cont*: *continuous-on* (*cubeImage twoC*) *F*
and *member-of-boundary*: $(k,\gamma) \in$ *boundary twoC*
shows *continuous-on* (γ ‘ $\{0 .. 1\}$) *F*
⟨*proof*⟩

lemma *typeII-cube-explicit-spec*:
assumes *typeII-twoCube twoC*
shows $\exists\, a\ b\ g1\ g2.\ a < b \wedge (\forall\, x \in \{a .. b\}.\ g2\ x \leq g1\ x) \wedge$
cubeImage twoC = $\{(y, x).\ x \in \{a..b\} \wedge y \in \{g2\ x .. g1\ x\}\}$
$\wedge$ *twoC* = $(\lambda(y, x).\ ((1 - y) * g2\ ((1 - x) * a + x * b) + y * g1$
$((1 - x) * a + x * b), (1 - x) * a + x * b))$
$\wedge$ *g1 piecewise-C1-differentiable-on* $\{a .. b\}$ $\wedge$ *g2 piecewise-C1-differentiable-on*
$\{a .. b\}$
$\wedge\ (\lambda x.\ twoC(0, x)) = (\lambda x.\ (g2\ (a + (b - a) * x), a + (b - a) * x))$
$\wedge\ (\lambda y.\ twoC(y, 1)) = (\lambda x.\ (g2\ b + x *_R (g1\ b - g2\ b), b))$
$\wedge\ (\lambda x.\ twoC(1, x)) = (\lambda x.\ (g1\ (a + (b - a) * x), a + (b - a) * x))$
$\wedge\ (\lambda y.\ twoC(y, 0)) = (\lambda x.\ (g2\ a + x *_R (g1\ a - g2\ a), a))$
⟨*proof*⟩

lemma *typeII-twoCube-smooth-edges*:
assumes *typeII-twoCube twoC* $(k,\gamma) \in$ *boundary twoC*
shows γ *piecewise-C1-differentiable-on* $\{0..1\}$
⟨*proof*⟩

lemma *field-cont-on-typeII-region-cont-on-edges*:
assumes *typeII-twoC*:
typeII-twoCube twoC **and**
field-cont:
continuous-on (*cubeImage twoC*) F **and**
member-of-boundary:
$(k,\gamma) \in$ *boundary twoC*
shows *continuous-on* $(\gamma$ ‘ $\{0 .. 1\})$ F
⟨*proof*⟩

lemma *two-cube-boundary-is-boundary*: *boundary-chain* (*boundary* C)
⟨*proof*⟩

lemma *common-boundary-subdiv-exists-refl*:
assumes $\forall (k,\gamma)\in$*boundary twoC*. *valid-path* γ
shows *common-boundary-sudivision-exists* (*boundary twoC*) (*boundary twoC*)
⟨*proof*⟩

lemma *common-boundary-subdiv-exists-refl'*:
assumes $\forall (k,\gamma)\in C$. *valid-path* γ
boundary-chain (C::(*int* $\times$ (*real* $\Rightarrow$ *real* $\times$ *real*)) *set*)
shows *common-boundary-sudivision-exists* (C) (C)
⟨*proof*⟩

lemma *gen-common-boundary-subdiv-exists-refl-twochain-boundary*:
assumes $\forall (k,\gamma)\in C$. *valid-path* γ
boundary-chain (C::(*int* $\times$ (*real* $\Rightarrow$ *real* $\times$ *real*)) *set*)
shows *common-sudiv-exists* (C) (C)
⟨*proof*⟩

lemma *two-chain-boundary-is-boundary-chain*:
shows *boundary-chain* (*two-chain-boundary twoChain*)
⟨*proof*⟩

lemma *typeI-edges-are-valid-paths*:
assumes *typeI-twoCube twoC* $(k,\gamma) \in$ *boundary twoC*
shows *valid-path* γ
⟨*proof*⟩

lemma *typeII-edges-are-valid-paths*:
assumes *typeII-twoCube twoC* $(k,\gamma) \in$ *boundary twoC*
shows *valid-path* γ
⟨*proof*⟩

lemma *finite-two-chain-vertical-boundary*:
  assumes *finite two-chain*
  shows *finite* (*two-chain-vertical-boundary two-chain*)
  ⟨*proof*⟩

lemma *finite-two-chain-horizontal-boundary*:
  assumes *finite two-chain*
  shows *finite* (*two-chain-horizontal-boundary two-chain*)
  ⟨*proof*⟩

locale *R2* =
  fixes *i j*
  assumes *i-is-x-axis*: *i* = (*1*::*real*,*0*::*real*) **and**
    j-is-y-axis: *j* = (*0*::*real*, *1*::*real*)
begin

lemma *analytically-valid-y*:
  assumes *analytically-valid s F i*
  shows (λ*x*. *integral UNIV* (λ*y*. (*partial-vector-derivative F i*) (*y*, *x*) ∗ (*indicator s* (*y*, *x*)))) ∈ *borel-measurable lborel*
⟨*proof*⟩

lemma *analytically-valid-x*:
  assumes *analytically-valid s F j*
  shows (λ*x*. *integral UNIV* (λ*y*. ((*partial-vector-derivative F j*) (*x*, *y*)) ∗ (*indicator s* (*x*, *y*)))) ∈ *borel-measurable lborel*
⟨*proof*⟩

lemma *Greens-thm-type-I*:
  fixes *F*:: ((*real*∗*real*) ⇒ (*real* ∗ *real*)) **and**
    gamma1 gamma2 gamma3 gamma4 :: (*real* ⇒ (*real* ∗ *real*)) **and**
    a:: *real* **and** *b*:: *real* **and**
    g1:: (*real* ⇒ *real*) **and** *g2*:: (*real* ⇒ *real*)
  assumes *Dy-def*: *Dy-pair* = {(*x*::*real*,*y*) . *x* ∈ *cbox a b* ∧ *y* ∈ *cbox* (*g2 x*) (*g1 x*)} **and**
    gamma1-def: *gamma1* = (λ*x*. (*a* + (*b* − *a*) ∗ *x*, *g2*(*a* + (*b* − *a*) ∗ *x*))) **and**
    gamma1-smooth: *gamma1 piecewise-C1-differentiable-on* {*0*..*1*} **and**
    gamma2-def: *gamma2* = (λ*x*. (*b*, *g2*(*b*) + *x* $*_R$ (*g1*(*b*) − *g2*(*b*)))) **and**
    gamma3-def: *gamma3* = (λ*x*. (*a* + (*b* − *a*) ∗ *x*, *g1*(*a* + (*b* − *a*) ∗ *x*))) **and**
    gamma3-smooth: *gamma3 piecewise-C1-differentiable-on* {*0*..*1*} **and**
    gamma4-def: *gamma4* = (λ*x*. (*a*, *g2*(*a*) + *x* $*_R$ (*g1*(*a*) − *g2*(*a*)))) **and**
    F-i-analytically-valid: *analytically-valid Dy-pair* (λ*p*. *F*(*p*) · *i*) *j* **and**
    g2-leq-g1: ∀ *x* ∈ *cbox a b*. (*g2 x*) ≤ (*g1 x*) **and**
    a-lt-b: *a* < *b*
  shows (*line-integral F* {*i*} *gamma1*) +
      (*line-integral F* {*i*} *gamma2*) −
      (*line-integral F* {*i*} *gamma3*) −
      (*line-integral F* {*i*} *gamma4*)
          = (*integral Dy-pair* (λ*a*. − (*partial-vector-derivative* (λ*p*. *F*(*p*) · *i*) *j*

a)))
  line-integral-exists F {*i*} *gamma4*
  line-integral-exists F {*i*} *gamma3*
  line-integral-exists F {*i*} *gamma2*
  line-integral-exists F {*i*} *gamma1*
⟨*proof*⟩

theorem *Greens-thm-type-II*:
 fixes *F*:: ((*real*∗*real*) ⇒ (*real* ∗ *real*)) **and**
  gamma4 gamma3 gamma2 gamma1 :: (*real* ⇒ (*real* ∗ *real*)) **and**
  a:: *real* **and** *b*:: *real* **and**
  g1:: (*real* ⇒ *real*) **and** *g2*:: (*real* ⇒ *real*)
 assumes *Dx-def*: *Dx-pair* = {(*x*::*real*,*y*) . *y* ∈ *cbox a b* ∧ *x* ∈ *cbox* (*g2 y*) (*g1 y*)}
and
  gamma4-def: *gamma4* = (λ*x*. (*g2*(*a* + (*b* − *a*) ∗ *x*), *a* + (*b* − *a*) ∗ *x*)) **and**
  gamma4-smooth: *gamma4 piecewise-C1-differentiable-on* {*0*..*1*} **and**
  gamma3-def: *gamma3* = (λ*x*. (*g2*(*b*) + *x* $*_R$ (*g1*(*b*) − *g2*(*b*)), *b*)) **and**
  gamma2-def: *gamma2* = (λ*x*. (*g1*(*a* + (*b* − *a*) ∗ *x*), *a* + (*b* − *a*) ∗ *x*)) **and**
  gamma2-smooth: *gamma2 piecewise-C1-differentiable-on* {*0*..*1*} **and**
  gamma1-def: *gamma1* = (λ*x*. (*g2*(*a*) + *x* $*_R$ (*g1*(*a*) − *g2*(*a*)), *a*)) **and**
  F-j-analytically-valid: *analytically-valid Dx-pair* (λ*p*. *F*(*p*) · *j*) *i* **and**
  g2-leq-g1: ∀ *x* ∈ *cbox a b*. (*g2 x*) ≤ (*g1 x*) **and**
  a-lt-b: *a* < *b*
 shows −(*line-integral F* {*j*} *gamma4*) −
   (*line-integral F* {*j*} *gamma3*) +
   (*line-integral F* {*j*} *gamma2*) +
   (*line-integral F* {*j*} *gamma1*)
     = (*integral Dx-pair* (λ*a*. (*partial-vector-derivative* (λ*a*. (*F a*) · *j*) *i*
a)))
  line-integral-exists F {*j*} *gamma4*
  line-integral-exists F {*j*} *gamma3*
  line-integral-exists F {*j*} *gamma2*
  line-integral-exists F {*j*} *gamma1*
⟨*proof*⟩

end

locale *green-typeII-cube* = *R2* +
 fixes *twoC F*
 assumes
  two-cube: *typeII-twoCube twoC* **and**
  valid-two-cube: *valid-two-cube twoC* **and**
  f-analytically-valid: *analytically-valid* (*cubeImage twoC*) (λ*x*. (*F x*) · *j*) *i*
begin

lemma *GreenThm-typeII-twoCube*:
 shows *integral* (*cubeImage twoC*) (λ*a*. *partial-vector-derivative* (λ*x*. (*F x*) · *j*) *i*
a) = *one-chain-line-integral F* {*j*} (*boundary twoC*)
  ∀ (*k*,γ) ∈ *boundary twoC*. *line-integral-exists F* {*j*} γ

⟨*proof*⟩

lemma *line-integral-exists-on-typeII-Cube-boundaries′*:
assumes $(k,\gamma) \in$ *boundary twoC*
shows *line-integral-exists F* $\{j\}$ γ
⟨*proof*⟩

end

locale *green-typeII-chain* = *R2* +
fixes *F two-chain s*
assumes *valid-typeII-div*: *valid-typeII-division s two-chain* **and**
F-anal-valid: $\forall$ *twoC* $\in$ *two-chain. analytically-valid* (*cubeImage twoC*) ($\lambda x.$ $(F\ x) \cdot j)$ i
begin

lemma *two-chain-valid-valid-cubes*: $\forall$ *two-cube* $\in$ *two-chain. valid-two-cube two-cube*
⟨*proof*⟩

lemma *typeII-chain-line-integral-exists-boundary′*:
shows $\forall$ $(k,\gamma) \in$ *two-chain-vertical-boundary two-chain. line-integral-exists F* $\{j\}$ γ
⟨*proof*⟩

lemma *typeII-chain-line-integral-exists-boundary′′*:
$\forall$ $(k,\gamma) \in$ *two-chain-horizontal-boundary two-chain. line-integral-exists F* $\{j\}$ γ
⟨*proof*⟩

lemma *typeII-cube-line-integral-exists-boundary*:
$\forall$ $(k,\gamma) \in$ *two-chain-boundary two-chain. line-integral-exists F* $\{j\}$ γ
⟨*proof*⟩

lemma *type-II-chain-horiz-bound-valid*:
$\forall$ $(k,\gamma) \in$ *two-chain-horizontal-boundary two-chain. valid-path* γ
⟨*proof*⟩

lemma *type-II-chain-vert-bound-valid*:
$\forall$ $(k,\gamma) \in$ *two-chain-vertical-boundary two-chain. valid-path* γ
⟨*proof*⟩

lemma *members-of-only-horiz-div-line-integrable′*:
assumes *only-horizontal-division one-chain two-chain*
(*k*::*int*, γ)$\in$*one-chain*
(*k*::*int*, γ)$\in$*one-chain*
finite two-chain
$\forall$ *two-cube* $\in$ *two-chain. valid-two-cube two-cube*
shows *line-integral-exists F* $\{j\}$ γ
⟨*proof*⟩

lemma *GreenThm-typeII-twoChain*:
shows *two-chain-integral two-chain* (*partial-vector-derivative* ($\lambda a.$ (F a) $\cdot$ j) i) = *one-chain-line-integral* F $\{j\}$ (*two-chain-boundary two-chain*)
⟨*proof*⟩

lemma *GreenThm-typeII-divisible*:
assumes
gen-division: *gen-division* s (*cubeImage* ‘ *two-chain*)
shows *integral* s (*partial-vector-derivative* ($\lambda x.$ (F x) $\cdot$ j) i) = *one-chain-line-integral* F $\{j\}$ (*two-chain-boundary two-chain*)
⟨*proof*⟩

lemma *GreenThm-typeII-divisible-region-boundary-gen*:
assumes *only-horizontal-division*: *only-horizontal-division* γ *two-chain*
shows *integral* s (*partial-vector-derivative* ($\lambda x.$ (F x) $\cdot$ j) i) = *one-chain-line-integral* F $\{j\}$ γ
⟨*proof*⟩

lemma *GreenThm-typeII-divisible-region-boundary*:
assumes
two-cubes-trace-vertical-boundaries:
two-chain-vertical-boundary two-chain $\subseteq$ γ **and**
boundary-of-region-is-subset-of-partition-boundary:
γ $\subseteq$ *two-chain-boundary two-chain*
shows *integral* s (*partial-vector-derivative* ($\lambda x.$ (F x) $\cdot$ j) i) = *one-chain-line-integral* F $\{j\}$ γ
⟨*proof*⟩

end

locale *green-typeI-cube* = *R2* +
fixes *twoC F*
assumes
two-cube: *typeI-twoCube twoC* **and**
valid-two-cube: *valid-two-cube twoC* **and**
f-analytically-valid: *analytically-valid* (*cubeImage twoC*) ($\lambda x.$ (F x) $\cdot$ i) j
begin

lemma *GreenThm-typeI-twoCube*:
shows *integral* (*cubeImage twoC*) ($\lambda a.$ − *partial-vector-derivative* ($\lambda p.$ F p $\cdot$ i) j a) = *one-chain-line-integral* F $\{i\}$ (*boundary twoC*)
$\forall$ (k,γ) $\in$ *boundary twoC*. *line-integral-exists* F $\{i\}$ γ
⟨*proof*⟩

lemma *line-integral-exists-on-typeI-Cube-boundaries′*:
assumes (k,γ) $\in$ *boundary twoC*
shows *line-integral-exists* F $\{i\}$ γ
⟨*proof*⟩

end

locale *green-typeI-chain* = *R2* +
fixes *F two-chain s*
assumes *valid-typeI-div*: *valid-typeI-division s two-chain* **and**
F-anal-valid: ∀ *twoC* ∈ *two-chain*. *analytically-valid* (*cubeImage twoC*) (λ*x*. (*F x*) · *i*) *j*
begin

lemma *two-chain-valid-valid-cubes*: ∀ *two-cube* ∈ *two-chain*. *valid-two-cube two-cube*
⟨*proof*⟩

lemma *typeI-cube-line-integral-exists-boundary′*:
assumes ∀ *two-cube* ∈ *two-chain*. *typeI-twoCube two-cube*
assumes ∀ *twoC* ∈ *two-chain*. *analytically-valid* (*cubeImage twoC*) (λ*x*. (*F x*) · *i*) *j*
assumes ∀ *two-cube* ∈ *two-chain*. *valid-two-cube two-cube*
shows ∀ (*k*,γ) ∈ *two-chain-vertical-boundary two-chain*. *line-integral-exists F* {*i*} γ
⟨*proof*⟩

lemma *typeI-cube-line-integral-exists-boundary″*:
∀ (*k*,γ) ∈ *two-chain-horizontal-boundary two-chain*. *line-integral-exists F* {*i*} γ
⟨*proof*⟩

lemma *typeI-cube-line-integral-exists-boundary*:
∀ (*k*,γ) ∈ *two-chain-boundary two-chain*. *line-integral-exists F* {*i*} γ
⟨*proof*⟩

lemma *type-I-chain-horiz-bound-valid*:
∀ (*k*,γ) ∈ *two-chain-horizontal-boundary two-chain*. *valid-path* γ
⟨*proof*⟩

lemma *type-I-chain-vert-bound-valid*:
assumes ∀ *two-cube* ∈ *two-chain*. *typeI-twoCube two-cube*
shows ∀ (*k*,γ) ∈ *two-chain-vertical-boundary two-chain*. *valid-path* γ
⟨*proof*⟩

lemma *members-of-only-vertical-div-line-integrable′*:
assumes *only-vertical-division one-chain two-chain*
(*k*::*int*, γ)∈*one-chain*
(*k*::*int*, γ)∈*one-chain*
finite two-chain
shows *line-integral-exists F* {*i*} γ
⟨*proof*⟩

lemma *GreenThm-typeI-two-chain*:
two-chain-integral two-chain (λ*a*. − *partial-vector-derivative* (λ*x*. (*F x*) · *i*) *j a*) = *one-chain-line-integral F* {*i*} (*two-chain-boundary two-chain*)

⟨*proof*⟩

lemma *GreenThm-typeI-divisible*:
assumes *gen-division*: *gen-division s* (*cubeImage ‘ two-chain*)
shows *integral s* ($\lambda x.$ − *partial-vector-derivative* ($\lambda a.$ $F(a) \cdot i$) *j x*) = *one-chain-line-integral* *F* {*i*} (*two-chain-boundary two-chain*)
⟨*proof*⟩

lemma *GreenThm-typeI-divisible-region-boundary*:
assumes
gen-division: *gen-division s* (*cubeImage ‘ two-chain*) **and**
two-cubes-trace-horizontal-boundaries:
two-chain-horizontal-boundary two-chain ⊆ γ **and**
boundary-of-region-is-subset-of-partition-boundary:
γ ⊆ *two-chain-boundary two-chain*
shows *integral s* ($\lambda x.$ − *partial-vector-derivative* ($\lambda a.$ $F(a) \cdot i$) *j x*) = *one-chain-line-integral* *F* {*i*} γ
⟨*proof*⟩

lemma *GreenThm-typeI-divisible-region-boundary-gen*:
assumes *valid-typeI-div*: *valid-typeI-division s two-chain* **and**
f-analytically-valid: ∀ *twoC* ∈ *two-chain*. *analytically-valid* (*cubeImage twoC*) ($\lambda a.$ $F(a) \cdot i$) *j* **and**
only-vertical-division:
only-vertical-division γ *two-chain*
shows *integral s* ($\lambda x.$ − *partial-vector-derivative* ($\lambda a.$ $F(a) \cdot i$) *j x*) = *one-chain-line-integral* *F* {*i*} γ
⟨*proof*⟩

end

locale *green-typeI-typeII-chain* = *R2*: *R2 i j* + *T1*: *green-typeI-chain i j F two-chain-typeI* + *T2*: *green-typeII-chain i j F two-chain-typeII* **for** *i j F two-chain-typeI two-chain-typeII*
begin

lemma *GreenThm-typeI-typeII-divisible-region-boundary*:
assumes
gen-divisions: *gen-division s* (*cubeImage ‘ two-chain-typeI*)
gen-division s (*cubeImage ‘ two-chain-typeII*) **and**
typeI-two-cubes-trace-horizontal-boundaries:
two-chain-horizontal-boundary two-chain-typeI ⊆ γ **and**
typeII-two-cubes-trace-vertical-boundaries:
two-chain-vertical-boundary two-chain-typeII ⊆ γ **and**
boundary-of-region-is-subset-of-partition-boundaries:
γ ⊆ *two-chain-boundary two-chain-typeI*
γ ⊆ *two-chain-boundary two-chain-typeII*
shows *integral s* ($\lambda x.$ *partial-vector-derivative* ($\lambda a.$ *F a* · *j*) *i x* − *partial-vector-derivative* ($\lambda a.$ *F a* · *i*) *j x*)
= *one-chain-line-integral F* {*i*, *j*} γ

⟨*proof*⟩

lemma *GreenThm-typeI-typeII-divisible-region′*:
assumes
only-vertical-division:
only-vertical-division one-chain-typeI two-chain-typeI
boundary-chain one-chain-typeI **and**
only-horizontal-division:
only-horizontal-division one-chain-typeII two-chain-typeII
boundary-chain one-chain-typeII **and**
typeI-and-typII-one-chains-have-gen-common-subdiv:
common-sudiv-exists one-chain-typeI one-chain-typeII
shows *integral s* (λ*x*. *partial-vector-derivative* (λ*x*. (*F x*) · *j*) *i x* − *partial-vector-derivative* (λ*x*. (*F x*) · *i*) *j x*) = *one-chain-line-integral F* {*i*, *j*} *one-chain-typeI*
integral s (λ*x*. *partial-vector-derivative* (λ*x*. (*F x*) · *j*) *i x* − *partial-vector-derivative* (λ*x*. (*F x*) · *i*) *j x*) = *one-chain-line-integral F* {*i*, *j*} *one-chain-typeII*
⟨*proof*⟩

lemma *GreenThm-typeI-typeII-divisible-region*:
assumes *only-vertical-division*:
only-vertical-division one-chain-typeI two-chain-typeI
boundary-chain one-chain-typeI **and**
only-horizontal-division:
only-horizontal-division one-chain-typeII two-chain-typeII
boundary-chain one-chain-typeII **and**
typeI-and-typII-one-chains-have-common-subdiv:
common-boundary-sudivision-exists one-chain-typeI one-chain-typeII
shows *integral s* (λ*x*. *partial-vector-derivative* (λ*x*. (*F x*) · *j*) *i x* − *partial-vector-derivative* (λ*x*. (*F x*) · *i*) *j x*) = *one-chain-line-integral F* {*i*, *j*} *one-chain-typeI*
integral s (λ*x*. *partial-vector-derivative* (λ*x*. (*F x*) · *j*) *i x* − *partial-vector-derivative* (λ*x*. (*F x*) · *i*) *j x*) = *one-chain-line-integral F* {*i*, *j*} *one-chain-typeII*
⟨*proof*⟩

lemma *GreenThm-typeI-typeII-divisible-region-finite-holes*:
assumes *valid-cube-boundary*: ∀ (*k*,γ)∈*boundary C*. *valid-path* γ **and**
only-vertical-division:
only-vertical-division (*boundary C*) *two-chain-typeI* **and**
only-horizontal-division:
only-horizontal-division (*boundary C*) *two-chain-typeII* **and**
s-is-oneCube: *s* = *cubeImage C*
shows *integral* (*cubeImage C*) (λ*x*. *partial-vector-derivative* (λ*x*. *F x* · *j*) *i x* − *partial-vector-derivative* (λ*x*. *F x* · *i*) *j x*) =
one-chain-line-integral F {*i*, *j*} (*boundary C*)
⟨*proof*⟩

lemma *GreenThm-typeI-typeII-divisible-region-equivallent-boundary*:
assumes
gen-divisions: *gen-division s* (*cubeImage* ‘ *two-chain-typeI*)
gen-division s (*cubeImage* ‘ *two-chain-typeII*) **and**

typeI-two-cubes-trace-horizontal-boundaries:
two-chain-horizontal-boundary two-chain-typeI ⊆ *one-chain-typeI* **and**
typeII-two-cubes-trace-vertical-boundaries:
two-chain-vertical-boundary two-chain-typeII ⊆ *one-chain-typeII* **and**
boundary-of-region-is-subset-of-partition-boundaries:
one-chain-typeI ⊆ *two-chain-boundary two-chain-typeI*
one-chain-typeII ⊆ *two-chain-boundary two-chain-typeII* **and**
typeI-and-typII-one-chains-have-common-subdiv:
common-boundary-sudivision-exists one-chain-typeI one-chain-typeII
shows *integral s* (λ*x*. *partial-vector-derivative* (λ*x*. (*F x*) · *j*) *i x* − *partial-vector-derivative* (λ*x*. (*F x*) · *i*) *j x*) = *one-chain-line-integral F* {*i*, *j*} *one-chain-typeI*
integral s (λ*x*. *partial-vector-derivative* (λ*x*. (*F x*) · *j*) *i x* − *partial-vector-derivative* (λ*x*. (*F x*) · *i*) *j x*) = *one-chain-line-integral F* {*i*, *j*} *one-chain-typeII*
⟨*proof*⟩

end
end
theory *SymmetricR2Shapes*
imports *Green*
begin

context *R2*
begin

lemma *valid-path-valid-swap*:
assumes *valid-path* (λ*x*::*real*. ((*f x*)::*real*, (*g x*)::*real*))
shows *valid-path* (*prod.swap o* (λ*x*. (*f x*, *g x*)))
⟨*proof*⟩

lemma *pair-fun-components*: *C* = (λ*x*. (*C x* · *i*, *C x* · *j*))
⟨*proof*⟩

lemma *swap-pair-fun*: (λ*y*. *prod.swap* (*C* (*y*, *0*))) = (λ*x*. (*C* (*x*, *0*) · *j*, *C* (*x*, *0*) · *i*))
⟨*proof*⟩

lemma *swap-pair-fun'*: (λ*y*. *prod.swap* (*C* (*y*, *1*))) = (λ*x*. (*C* (*x*, *1*) · *j*, *C* (*x*, *1*) · *i*))
⟨*proof*⟩

lemma *swap-pair-fun''*: (λ*y*. *prod.swap* (*C* (*0*, *y*))) = (λ*x*. (*C* (*0*,*x*) · *j*, *C* (*0*,*x*) · *i*))
⟨*proof*⟩

lemma *swap-pair-fun'''*: (λ*y*. *prod.swap* (*C* (*1*, *y*))) = (λ*x*. (*C* (*1*,*x*) · *j*, *C* (*1*,*x*) · *i*))
⟨*proof*⟩

lemma *swap-valid-boundaries*:

assumes ∀ $(k,\gamma)\in$*boundary C. valid-path* γ
assumes $(k,\gamma)\in$*boundary* (*prod.swap o C o prod.swap*)
shows *valid-path* γ
⟨*proof*⟩

lemma *prod-comp-eq*:
assumes f = *prod.swap o g*
shows *prod.swap o f* = g
⟨*proof*⟩

lemma *swap-typeI-is-typeII*:
assumes *typeI-twoCube C*
shows *typeII-twoCube* (*prod.swap o C o prod.swap*)
⟨*proof*⟩

lemma *valid-cube-valid-swap*:
assumes *valid-two-cube C*
shows *valid-two-cube* (*prod.swap o C o prod.swap*)
⟨*proof*⟩

lemma *twoChainVertDiv-of-itself*:
assumes *finite C*
∀ $(k, \gamma)\in$(*two-chain-boundary C*). *valid-path* γ
shows *only-vertical-division* (*two-chain-boundary C*) *C*
⟨*proof*⟩

end

definition *x-coord* **where** *x-coord* ≡ (λt::*real*. $t - 1/2$)

lemma *x-coord-smooth*: *x-coord C1-differentiable-on* {*a..b*}
⟨*proof*⟩

lemma *x-coord-bounds*:
assumes (*0*::*real*) $\leq x$ $x \leq 1$
shows $-1/2 \leq$ *x-coord x* ∧ *x-coord x* $\leq 1/2$
⟨*proof*⟩

lemma *x-coord-img*: *x-coord* ‘ {(*0*::*real*)..*1*} = {−*1/2* .. *1/2*}
⟨*proof*⟩

lemma *x-coord-back-img*: *finite* ({*0..1*} ∩ *x-coord* −‘ {*x*::*real*})
⟨*proof*⟩

abbreviation *rot-x t1 t2* ≡ (*if* (*t1* − *1/2*) ≤ *0 then* (*2* ∗ *t2* − *1*) ∗ *t1* + *1/2* ::*real else 2* ∗ *t2* − *2* ∗ *t1* ∗ *t2* +*t1* −*1/2*::*real*)

lemma *rot-x-ivl*:
assumes $0 \leq x$
$x \leq 1$
$0 \leq y$
$y \leq 1$
shows $0 \leq$ *rot-x x y* $\wedge$ *rot-x x y* ≤ 1
⟨*proof*⟩

end

2 The Circle Example

theory *CircExample*
imports *Green SymmetricR2Shapes*

begin

locale *circle* = *R2* +
fixes *d*::*real*
assumes *d-gt-0*: $0 < d$
begin

definition *circle-y* **where**
circle-y t = *sqrt* $(1/4 - t * t)$

definition *circle-cube* **where**
circle-cube = $(\lambda(x,y).\ ((x - 1/2) * d, (2 * y - 1) * d *$ *sqrt* $(1/4 - (x - 1/2)*(x - 1/2))))$

lemma *circle-cube-nice*:
shows *circle-cube* = $(\lambda(x,y).\ (d *$ *x-coord* $x, (2 * y - 1) * d *$ *circle-y* (*x-coord* x)))
⟨*proof*⟩

definition *rot-circle-cube* **where**
rot-circle-cube = *prod.swap* ∘ (*circle-cube*) ∘ *prod.swap*

abbreviation *rot-y t1 t2* ≡ ((*t1*−1/2)/(2∗ *circle-y* (*x-coord* (*rot-x t1 t2*))) +1/2::*real*)

definition *x-coord-inv* (*x*::*real*) = (1/2) + *x*

lemma *x-coord-inv-1*: *x-coord-inv* (*x-coord* (*x*::*real*)) = *x*
⟨*proof*⟩

lemma *x-coord-inv-2*: *x-coord* (*x-coord-inv* (*x*::*real*)) = *x*
⟨*proof*⟩

definition *circle-y-inv* = *circle-y*

abbreviation *rot-x''* (*x*::*real*) (*y*::*real*) ≡ (*x-coord-inv* ((*2* ∗ *y* − *1*) ∗ *circle-y* (*x-coord x*)))

lemma *circle-y-bounds*:
assumes $-1/2 \leq (x::real) \wedge x \leq 1/2$
shows $0 \leq$ *circle-y x circle-y x* $\leq 1/2$
⟨*proof*⟩

lemma *circle-y-x-coord-bounds*:
assumes $0 \leq (x::real)\ x \leq 1$
shows $0 \leq$ *circle-y* (*x-coord x*) ∧ *circle-y* (*x-coord x*) $\leq 1/2$
⟨*proof*⟩

lemma *rot-x-ivl*:
assumes $(0::real) \leq x\ x \leq 1\ \ 0 \leq y\ y \leq 1$
shows $0 \leq$ *rot-x''* *x y* ∧ *rot-x''* *x y* ≤ 1
⟨*proof*⟩

abbreviation *rot-y''* (*x*::*real*) (*y*::*real*) ≡ (*x-coord x*)/(*2*∗ (*circle-y* (*x-coord* (*rot-x''* *x y*)))) + *1/2*

lemma *rot-y-ivl*:
assumes $(0::real) \leq x\ x \leq 1\ 0 \leq y\ y \leq 1$
shows $0 \leq$ *rot-y''* *x y* ∧ *rot-y''* *x y* ≤ 1
⟨*proof*⟩

lemma *circle-eq-rot-circle*:
assumes $0 \leq x\ x \leq 1\ 0 \leq y\ y \leq 1$
shows (*circle-cube* (*x*, *y*)) = (*rot-circle-cube* (*rot-y''* *x y*, *rot-x''* *x y*))
⟨*proof*⟩

lemma *rot-circle-eq-circle*:
assumes $0 \leq x\ x \leq 1\ 0 \leq y\ y \leq 1$
shows (*rot-circle-cube* (*x*, *y*)) = (*circle-cube* (*rot-x''* *y x*, *rot-y''* *y x*))
⟨*proof*⟩

lemma *rot-img-eq*:
assumes $0 < d$
shows (*cubeImage* (*circle-cube*)) = (*cubeImage* (*rot-circle-cube*))
⟨*proof*⟩

lemma *rot-circle-div-circle*:
assumes $0 < (d::real)$
shows *gen-division* (*cubeImage circle-cube*) (*cubeImage* ‘ {*rot-circle-cube*})
⟨*proof*⟩

lemma *circle-cube-boundary-valid*:
assumes (*k*,*γ*)∈*boundary circle-cube*
shows *valid-path γ*

⟨*proof*⟩

lemma *rot-circle-cube-boundary-valid*:
assumes $(k,\gamma) \in$ *boundary rot-circle-cube*
shows *valid-path* γ
⟨*proof*⟩

lemma *diff-divide-cancel*:
fixes *z::real* **shows** $z \neq 0 \Longrightarrow (a * z - a * (b * z)) / z = (a - a * b)$
⟨*proof*⟩

lemma *circle-cube-is-type-I*:
assumes $0 < d$
shows *typeI-twoCube circle-cube*
⟨*proof*⟩

lemma *rot-circle-cube-is-type-II*:
shows *typeII-twoCube rot-circle-cube*
⟨*proof*⟩

definition *circle-bot-edge* **where**
circle-bot-edge = (*1::int*, $\lambda t.$ (*x-coord t* $*$ *d*, $-$ *d* $*$ *circle-y* (*x-coord t*)))

definition *circle-top-edge* **where**
circle-top-edge = ($-$ *1::int*, $\lambda t.$ (*x-coord t* $*$ *d*, *d* $*$ *circle-y* (*x-coord t*)))

definition *circle-right-edge* **where**
circle-right-edge = (*1::int*, $\lambda y.$ (*d/2*, *0*))

definition *circle-left-edge* **where**
circle-left-edge = ($-$ *1::int*, $\lambda y.$ ($-$ (*d/2*), *0*))

lemma *circle-cube-boundary-explicit*:
boundary circle-cube = {*circle-left-edge*,*circle-right-edge*,*circle-bot-edge*,*circle-top-edge*}
⟨*proof*⟩

definition *rot-circle-right-edge* **where**
rot-circle-right-edge = (*1::int*, $\lambda t.$ (*d* $*$ *circle-y* (*x-coord t*), *x-coord t* $*$ *d*))

definition *rot-circle-left-edge* **where**
rot-circle-left-edge = ($-$ *1::int*, $\lambda t.$ ($-$ *d* $*$ *circle-y* (*x-coord t*), *x-coord t* $*$ *d*))

definition *rot-circle-top-edge* **where**
rot-circle-top-edge = ($-$ *1::int*, $\lambda y.$ (*0*, *d/2*))

definition *rot-circle-bot-edge* **where**
rot-circle-bot-edge = (*1::int*, $\lambda y.$ (*0*, $-$ (*d/2*)))

lemma *rot-circle-cube-boundary-explicit*:

boundary (*rot-circle-cube*) =
{*rot-circle-top-edge*,*rot-circle-bot-edge*,*rot-circle-right-edge*,*rot-circle-left-edge*}
⟨*proof*⟩

lemma *rot-circle-cube-vertical-boundary-explicit*:
vertical-boundary rot-circle-cube = {*rot-circle-right-edge*,*rot-circle-left-edge*}
⟨*proof*⟩

lemma *circ-left-edge-neq-top*:
$(- 1::int, \lambda y::real. (- (d/2), 0)) \neq (- 1, \lambda x. ((x - 1/2) * d, d * sqrt (1/4 - (x - 1/2) * (x - 1/2))))$
⟨*proof*⟩

lemma *circle-cube-valid-two-cube*: *valid-two-cube* (*circle-cube*)
⟨*proof*⟩

lemma *rot-circle-cube-valid-two-cube*:
shows *valid-two-cube rot-circle-cube*
⟨*proof*⟩

definition *circle-arc-0* **where** $circle\text{-}arc\text{-}0 = (1, \lambda t::real. (0,0))$

lemma *circle-top-bot-edges-neq′* [*simp*]:
shows *circle-top-edge* ≠ *circle-bot-edge*
⟨*proof*⟩

lemma *rot-circle-top-left-edges-neq* [*simp*]: *rot-circle-top-edge* ≠ *rot-circle-left-edge*
⟨*proof*⟩

lemma *rot-circle-bot-left-edges-neq* [*simp*]: *rot-circle-bot-edge* ≠ *rot-circle-left-edge*
⟨*proof*⟩

lemma *rot-circle-top-right-edges-neq* [*simp*]: *rot-circle-top-edge* ≠ *rot-circle-right-edge*
⟨*proof*⟩

lemma *rot-circle-bot-right-edges-neq* [*simp*]: *rot-circle-bot-edge* ≠ *rot-circle-right-edge*
⟨*proof*⟩

lemma *rot-circle-right-top-edges-neq′* [*simp*]: *rot-circle-right-edge* ≠ *rot-circle-left-edge*
⟨*proof*⟩

lemma *rot-circle-left-bot-edges-neq* [*simp*]: *rot-circle-left-edge* ≠ *rot-circle-top-edge*
⟨*proof*⟩

lemma *circle-right-top-edges-neq* [*simp*]: *circle-right-edge* ≠ *circle-top-edge*
⟨*proof*⟩

lemma *circle-left-bot-edges-neq* [*simp*]: *circle-left-edge* ≠ *circle-bot-edge*
⟨*proof*⟩

lemma *circle-left-top-edges-neq* [*simp*]: *circle-left-edge* $\neq$ *circle-top-edge*
⟨*proof*⟩

lemma *circle-right-bot-edges-neq* [*simp*]: *circle-right-edge* $\neq$ *circle-bot-edge*
⟨*proof*⟩

definition *circle-polar* **where**
circle-polar t = ((*d*/*2*) ∗ *cos* (*2* ∗ *pi* ∗ *t*), (*d*/*2*) ∗ *sin* (*2* ∗ *pi* ∗ *t*))

lemma *circle-polar-smooth*: (*circle-polar*) *C1-differentiable-on* {*0*..*1*}
⟨*proof*⟩

abbreviation *custom-arccos* ≡ (λ*x*. (*if* −*1* ≤ *x* ∧ *x* ≤ *1 then arccos x else* (*if x* < −*1 then* −*x* + *pi else 1* −*x*)))

lemma *cont-custom-arccos*:
assumes *S* ⊆ {−*1*..*1*}
shows *continuous-on S custom-arccos*
⟨*proof*⟩

lemma *custom-arccos-has-deriv*:
assumes − *1* < *x x* < *1*
shows *DERIV custom-arccos x* :> *inverse* (− *sqrt* (*1* − x^2))
⟨*proof*⟩

declare
custom-arccos-has-deriv[*THEN DERIV-chain2*, *derivative-intros*]
custom-arccos-has-deriv[*THEN DERIV-chain2*, *unfolded has-field-derivative-def*, *derivative-intros*]

lemma *circle-boundary-reparams*:
shows *rot-circ-left-edge-reparam-polar-circ-split*:
reparam (*rec-join* [(*rot-circle-left-edge*)]) (*rec-join* [(*subcube* (*1*/*4*) (*1*/*2*) (*1*, *circle-polar*)), (*subcube* (*1*/*2*) (*3*/*4*) (*1*, *circle-polar*))])
(**is** *?P1*)
and *circ-top-edge-reparam-polar-circ-split*:
reparam (*rec-join* [(*circle-top-edge*)]) (*rec-join* [(*subcube 0* (*1*/*4*) (*1*, *circle-polar*)), (*subcube* (*1*/*4*) (*1*/*2*) (*1*, *circle-polar*))])
(**is** *?P2*)
and *circ-bot-edge-reparam-polar-circ-split*:
reparam (*rec-join* [(*circle-bot-edge*)]) (*rec-join* [(*subcube* (*1*/*2*) (*3*/*4*) (*1*, *circle-polar*)), (*subcube* (*3*/*4*) *1* (*1*, *circle-polar*))])
(**is** *?P3*)
and *rot-circ-right-edge-reparam-polar-circ-split*:
reparam (*rec-join* [(*rot-circle-right-edge*)]) (*rec-join* [(*subcube* (*3*/*4*) *1* (*1*, *circle-polar*)), (*subcube 0* (*1*/*4*) (*1*, *circle-polar*))])
(**is** *?P4*)
⟨*proof*⟩

definition *circle-cube-boundary-to-polarcircle* **where**
circle-cube-boundary-to-polarcircle $\gamma \equiv$
if (γ = (*circle-top-edge*)) *then*
{*subcube 0* (*1/4*) (*1*, *circle-polar*), *subcube* (*1/4*) (*1/2*) (*1*, *circle-polar*)}
else if (γ = (*circle-bot-edge*)) *then*
{*subcube* (*1/2*) (*3/4*) (*1*, *circle-polar*), *subcube* (*3/4*) *1* (*1*, *circle-polar*)}
else {}

definition *rot-circle-cube-boundary-to-polarcircle* **where**
rot-circle-cube-boundary-to-polarcircle $\gamma \equiv$
if (γ = (*rot-circle-left-edge*)) *then*
{*subcube* (*1/4*) (*1/2*) (*1*, *circle-polar*), *subcube* (*1/2*) (*3/4*) (*1*, *circle-polar*)}
else if (γ = (*rot-circle-right-edge*)) *then*
{*subcube* (*3/4*) *1* (*1*, *circle-polar*), *subcube 0* (*1/4*) (*1*, *circle-polar*)}
else {}

lemma *circle-arcs-neq*:
assumes $0 \leq k$ $k \leq 1$ $0 \leq n$ $n \leq 1$ $n < k$ $k + n < 1$
shows *subcube k m* (*1*, *circle-polar*) $\neq$ *subcube n q* (*1*, *circle-polar*)
⟨*proof*⟩

lemma *circle-arcs-neq-2*:
assumes $0 \leq k$ $k \leq 1$ $0 \leq n$ $n \leq 1$ $n < k$ $0 < n$ **and** *kn12*: $1/2 < k + n$ **and** $k + n < 3/2$
shows *subcube k m* (*1*, *circle-polar*) $\neq$ *subcube n q* (*1*, *circle-polar*)
⟨*proof*⟩

lemma *circle-cube-is-only-horizontal-div-of-rot*:
shows *only-horizontal-division* (*boundary* (*circle-cube*)) {*rot-circle-cube*}
⟨*proof*⟩

lemma *GreenThm-cirlce*:
assumes ∀ *twoC*∈{*circle-cube*}. *analytically-valid* (*cubeImage twoC*) (λ*x*. *F x* · *i*) *j*
∀ *twoC*∈{*rot-circle-cube*}. *analytically-valid* (*cubeImage twoC*) (λ*x*. *F x* · *j*) *i*
shows *integral* (*cubeImage* (*circle-cube*)) (λ*x*. *partial-vector-derivative* (λ*x*. *F x* · *j*) *i x* − *partial-vector-derivative* (λ*x*. *F x* · *i*) *j x*) =
one-chain-line-integral F {*i*, *j*} (*boundary* (*circle-cube*))
⟨*proof*⟩
end
end

3 The Diamond Example

theory *DiamExample*
imports *Green SymmetricR2Shapes*

begin

lemma *abs-if′*:
fixes a :: $'a$ :: {*abs-if*,*ordered-ab-group-add*}
shows $|a| = ($*if* $a \leq 0$ *then* $- a$ *else* $a)$
⟨*proof*⟩

locale *diamond* = *R2* +
fixes *d*::*real*
assumes *d-gt-0*: $0 < d$
begin

definition *diamond-y-gen* :: *real* ⇒ *real* **where**
diamond-y-gen ≡λ*t*. $1/2 - |t|$

definition *diamond-cube-gen*:: ((*real* ∗ *real*) ⇒ (*real* ∗ *real*)) **where**
diamond-cube-gen ≡ (λ(*x*,*y*). (*d* ∗ *x-coord x*, (*2* ∗ *y* − *1*) ∗ (*d* ∗ *diamond-y-gen* (*x-coord x*))))

lemma *diamond-y-gen-valid*:
assumes $a \leq 0$ $0 \leq b$
shows *diamond-y-gen piecewise-C1-differentiable-on* {*a*..*b*}
⟨*proof*⟩

lemma *diamond-cube-gen-boundary-valid*:
assumes (*k*,γ)∈*boundary* (*diamond-cube-gen*)
shows *valid-path* γ
⟨*proof*⟩

definition *diamond-x* **where**
diamond-x ≡ λ*t*. (*t* − *1/2*) ∗ *d*

definition *diamond-y* **where**
diamond-y ≡ λ*t*. *d/2* − |*t*|

definition *diamond-cube* **where**
diamond-cube = (λ(*x*,*y*). (*diamond-x x*, (*2* ∗ *y* − *1*) ∗ (*diamond-y* (*diamond-x x*))))

definition *rot-diamond-cube* **where**
rot-diamond-cube = *prod.swap o* (*diamond-cube*) *o prod.swap*

lemma *diamond-eq-characterisations*:
shows *diamond-cube* (*x*,*y*)= *diamond-cube-gen* (*x*,*y*)
⟨*proof*⟩

lemma *diamond-eq-characterisations-fun*: *diamond-cube* = *diamond-cube-gen*
⟨*proof*⟩

lemma *diamond-y-valid*:
shows *diamond-y piecewise-C1-differentiable-on* $\{-d/2..d/2\}$ (**is** *?P*)
($\lambda x.$ *diamond-y x*) *piecewise-C1-differentiable-on* $\{-d/2..d/2\}$ (**is** *?Q*)
⟨*proof*⟩

lemma *diamond-cube-boundary-valid*:
assumes $(k,\gamma) \in$ *boundary* (*diamond-cube*)
shows *valid-path* γ
⟨*proof*⟩

lemma *diamond-cube-is-type-I*:
shows *typeI-twoCube* (*diamond-cube*)
⟨*proof*⟩

lemma *diamond-cube-valid-two-cube*:
shows *valid-two-cube* (*diamond-cube*)
⟨*proof*⟩

lemma *rot-diamond-cube-boundary-valid*:
assumes $(k,\gamma)\in$*boundary* (*rot-diamond-cube*)
shows *valid-path* γ
⟨*proof*⟩

lemma *rot-diamond-cube-is-type-II*:
shows *typeII-twoCube* (*rot-diamond-cube*)
⟨*proof*⟩

lemma *rot-diamond-cube-valid-two-cube*: *valid-two-cube* (*rot-diamond-cube*)
⟨*proof*⟩

definition *diamond-top-edges* **where**
diamond-top-edges = (− *1*::*int*, $\lambda x.$ (*diamond-x x*, *diamond-y* (*diamond-x x*)))

definition *diamond-bot-edges* **where**
diamond-bot-edges = (*1*::*int*, $\lambda x.$ (*diamond-x x*, − *diamond-y* (*diamond-x x*)))

lemma *diamond-cube-boundary-explicit*:
boundary (*diamond-cube*) =
{*diamond-top-edges*,
diamond-bot-edges,
(− *1*::*int*, $\lambda y.$ (*diamond-x 0*, $(2 * y - 1) *$ *diamond-y* (*diamond-x 0*))),
(*1*::*int*, $\lambda y.$ (*diamond-x 1*, $(2 * y - 1) *$ *diamond-y* (*diamond-x 1*)))}
⟨*proof*⟩

definition *diamond-top-left-edge* **where**
diamond-top-left-edge = (− *1*::*int*, ($\lambda x.$ (*diamond-x* $(1/2 * x)$, (*diamond-x* $(1/2 * x)$) + $d/2$)))

definition *diamond-top-right-edge* **where**

diamond-top-right-edge = (− *1*::*int*, (λ*x*. (*diamond-x* (*1*/*2* ∗ *x* + *1*/*2*), (−(*diamond-x* (*1*/*2* ∗ *x* + *1*/*2*)) + *d*/*2*))))

definition *diamond-bot-left-edge* **where**
diamond-bot-left-edge = (*1*::*int*, (λ*x*. (*diamond-x* (*1*/*2* ∗ *x*), − (*diamond-x* (*1*/*2* ∗ *x*) + *d*/*2*))))

definition *diamond-bot-right-edge* **where**
diamond-bot-right-edge = (*1*::*int*, (λ*x*. (*diamond-x* (*1*/*2* ∗ *x* + *1*/*2*), − (−(*diamond-x* (*1*/*2* ∗ *x* + *1*/*2*)) + *d*/*2*))))

lemma *diamond-edges-are-valid*:
valid-path (*snd* (*diamond-top-left-edge*))
valid-path (*snd* (*diamond-top-right-edge*))
valid-path (*snd* (*diamond-bot-left-edge*))
valid-path (*snd* (*diamond-bot-right-edge*))
⟨*proof*⟩

definition *diamond-cube-boundary-to-subdiv* **where**
diamond-cube-boundary-to-subdiv (*gamma*::(*int* × (*real* ⇒ *real* × *real*))) ≡
if (*gamma* = *diamond-top-edges*) *then*
{*diamond-top-left-edge*, *diamond-top-right-edge*}
else if (*gamma* = *diamond-bot-edges*) *then*
{*diamond-bot-left-edge*, *diamond-bot-right-edge*}
else {}

lemma *rot-diam-edge-1*:
(*1*::*int*, λ*x*::*real*. ((*x*::*real*) ∗ (*2* ∗ *diamond-y* (*diamond-x 0*)) − *1* ∗ *diamond-y* (*diamond-x 0*), *diamond-x 0*)) =
(*1*, λ*x*. (*x* ∗ (*2* ∗ *diamond-y* (*diamond-x 0*)) − (*diamond-y* (*diamond-x 0*)), *diamond-x 0*))
⟨*proof*⟩

definition *diamond-left-edges* **where**
diamond-left-edges = (− *1*, λ*y*. (− *diamond-y* (*diamond-x y*), *diamond-x y*))

definition *diamond-right-edges* **where**
diamond-right-edges = (*1*, λ*y*. (*diamond-y* (*diamond-x y*), *diamond-x y*))

lemma *rot-diamond-cube-boundary-explicit*:
boundary (*rot-diamond-cube*) = {(*1*::*int*, λ*x*::*real*. ((*2* ∗ *x* − *1*) ∗ *diamond-y* (*diamond-x 0*), *diamond-x 0*)),
(− *1*, λ*x*. ((*2* ∗ *x* − *1*) ∗ *diamond-y* (*diamond-x 1*), *diamond-x 1*)),
diamond-left-edges, *diamond-right-edges*}
⟨*proof*⟩

lemma *rot-diamond-cube-vertical-boundary-explicit*:
vertical-boundary (*rot-diamond-cube*) = {*diamond-left-edges*, *diamond-right-edges*}

⟨*proof*⟩

definition *rot-diamond-cube-boundary-to-subdiv* **where**
rot-diamond-cube-boundary-to-subdiv (*gamma*::(*int* × (*real* ⇒ *real* × *real*))) ≡
if (*gamma* = *diamond-left-edges*) *then* {*diamond-bot-left-edge* , *diamond-top-left-edge*}
else if (*gamma* = *diamond-right-edges*) *then* {*diamond-bot-right-edge*, *diamond-top-right-edge*}
else {}

definition *diamond-boundaries-reparam-map* **where**
diamond-boundaries-reparam-map ≡ *id*

lemma *diamond-boundaries-reparam-map-bij*:
bij (*diamond-boundaries-reparam-map*)
⟨*proof*⟩

lemma *diamond-bot-edges-neq-diamond-top-edges*:
diamond-bot-edges ≠ *diamond-top-edges*
⟨*proof*⟩

lemma *diamond-top-left-edge-neq-diamond-top-right-edge*:
diamond-top-left-edge ≠ *diamond-top-right-edge*
⟨*proof*⟩

lemma *neqs1*:
shows (*λx*. (*diamond-x x*, *diamond-y* (*diamond-x x*))) ≠ (*λx*. (*diamond-x x*, − *diamond-y* (*diamond-x x*)))
and (*λy*. (− *diamond-y* (*diamond-x y*), *diamond-x y*)) ≠ (*λy*. (*diamond-y* (*diamond-x y*), *diamond-x y*))
and (*λx*. (*diamond-x*(*x*/*2* + *1*/*2*), *diamond-x*(*x*/*2* + *1*/*2*) − *d*/*2*)) ≠ (*λx*. (*diamond-x*(*x*/*2*), − *diamond-x*(*x*/*2*) − *d*/*2*))
and (*λx*. (*diamond-x*(*x*/*2* + *1*/*2*), *d*/*2* − *diamond-x*(*x*/*2* + *1*/*2*))) ≠ (*λx*. (*diamond-x*(*x*/*2*), *diamond-x*(*x*/*2*) + *d*/*2*))
and (*λx*. (*diamond-x*(*x*/*2*), − *diamond-x*(*x*/*2*) − *d*/*2*)) ≠ (*λx*. (*diamond-x*(*x*/*2* + *1*/*2*), *diamond-x*(*x*/*2* + *1*/*2*) − *d*/*2*))
and (*λx*. (*diamond-x*(*x*/*2*), *diamond-x*(*x*/*2*) + *d*/*2*)) ≠ (*λx*. (*diamond-x*(*x*/*2* + *1*/*2*), *d*/*2* − *diamond-x*(*x*/*2* + *1*/*2*)))
⟨*proof*⟩

lemma *neqs2*:
shows (*λx*. (*diamond-x x*, *diamond-y* (*diamond-x x*))) ≠ (*λx*. ((*2* ∗ *x* − *1*) ∗ *diamond-y* (*diamond-x 1*), *diamond-x 1*))
and (*λx*. (*diamond-x x*, − *diamond-y* (*diamond-x x*))) ≠ (*λx*. ((*2* ∗ *x* − *1*) ∗ *diamond-y* (*diamond-x 0*), *diamond-x 0*))
⟨*proof*⟩

lemma *diamond-cube-is-only-horizontal-div-of-rot*:
shows *only-horizontal-division* (*boundary* (*diamond-cube*)) {*rot-diamond-cube*}
⟨*proof*⟩

abbreviation *rot-y t1 t2* ≡ (*t1* − *1/2*) / (*2*∗ *diamond-y-gen* (*x-coord* (*rot-x t1 t2*))) + *1/2*

lemma *rot-y-ivl*:
assumes (*0*::*real*) ≤ *x x* ≤ *1 0* ≤ *y y* ≤ *1*
shows *0* ≤ *rot-y x y* ∧ *rot-y x y* ≤ *1*
⟨*proof*⟩

lemma *diamond-gen-eq-rot-diamond*:
assumes*0* ≤ *x x* ≤ *1 0* ≤ *y y* ≤ *1*
shows (*diamond-cube-gen* (*x*, *y*)) = (*rot-diamond-cube* (*rot-y x y*, *rot-x x y*))
⟨*proof*⟩

lemma *rot-diamond-eq-diamond-gen*:
assumes *0* ≤ *x x* ≤ *1 0* ≤ *y y* ≤ *1*
shows *rot-diamond-cube* (*x*, *y*) = *diamond-cube-gen* (*rot-x y x*, *rot-y y x*)
⟨*proof*⟩

lemma *rot-img-eq*: *cubeImage* (*diamond-cube-gen*) = *cubeImage* (*rot-diamond-cube*)
⟨*proof*⟩

lemma *rot-diamond-gen-div-diamond-gen*:
shows *gen-division* (*cubeImage* (*diamond-cube-gen*)) (*cubeImage* ‘ {*rot-diamond-cube*})
⟨*proof*⟩

lemma *rot-diamond-gen-div-diamond*:
shows *gen-division* (*cubeImage* (*diamond-cube*)) (*cubeImage* ‘ {*rot-diamond-cube*})
⟨*proof*⟩

lemma *GreenThm-diamond*:
assumes *analytically-valid* (*cubeImage* (*diamond-cube*)) (λ*x*. *F x* · *i*) *j*
analytically-valid (*cubeImage* (*diamond-cube*)) (λ*x*. *F x* · *j*) *i*
shows *integral* (*cubeImage* (*diamond-cube*)) (λ*x*. *partial-vector-derivative* (λ*x*. *F x* · *j*) *i x* − *partial-vector-derivative* (λ*x*. *F x* · *i*) *j x*) =
one-chain-line-integral F {*i*, *j*} (*boundary* (*diamond-cube*))
⟨*proof*⟩
end
end