

An Isabelle/HOL formalisation of Green's Theorem

Mohammad Abdulaziz and Lawrence C. Paulson

June 24, 2019

Abstract

We formalise a statement of Greens theoremthe first formalisation to our knowlegein 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.

theory *General-Utils*

imports *HOL-Analysis.Analysis*

begin

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

proof –

{ **assume** *H*: *?rhs*

then have *?lhs* **by** *auto* }

moreover

{ **assume** *H*: *?lhs*

then obtain *f''* **where** $f''::\forall i. P\ i\ (f''\ i)$ **unfolding** *choice-iff*
 by *metis*

let *?x* = $(\lambda x. (\chi\ i. (f''\ i)\ x))$

 { **fix** *i*

from *f''* **have** $P\ i\ (f''\ i)$ **by** *metis*

then have $P\ i\ (\lambda x. (?x\ x)\ \$\ i)$ **by** *auto*

 }

hence $\forall i. P\ i\ (\lambda x. (?x\ x)\ \$\ i)$ **by** *metis*

hence *?rhs* **by** *metis* }

ultimately show *?thesis* **by** *metis*

qed

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

proof –

- { **assume** $H: ?rhs$
- then have** $?lhs$ **by** *auto* }
- moreover**
- { **assume** $H: ?lhs$
- then obtain** f'' **where** $f'': \forall i::('b::euclidean-space) \in Basis. P\ i\ (f''\ i)$ **unfolding** *choice-iff*
- by** *metis*
- let** $?x = (\lambda x. (\sum i \in Basis. ((f''\ i)\ x) *_{\mathbb{R}} i))$
- { **fix** $i::'b::euclidean-space$
- assume** $ass: i \in Basis$
- then have** $P\ i\ (f''\ i)$
- using** f''
- by** *metis*
- then have** $P\ i\ (\lambda x. (?x\ x) \cdot i)$ **using** ass **by** *auto*
- }
- hence** $*$: $\forall i \in Basis. P\ i\ (\lambda x. (?x\ x) \cdot i)$ **by** *auto*
- then have** $?rhs$
- apply** *auto*
- proof**
- let** $?f'6 = ?x$
- show** $\forall i \in Basis. P\ i\ (\lambda x. ?f'6\ x \cdot i)$ **using** $*$ **by** *auto*
- qed**}
- ultimately show** $?thesis$ **by** *metis*
- qed**

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

proof –

- { **assume** $H: ?rhs$
- then have** $?lhs$ **by** *auto* }
- moreover**
- { **assume** $H: ?lhs$
- then obtain** f'' **where** $f'': \forall i::('a::euclidean-space) \in Basis. P\ i\ (f''\ i)$ **unfolding** *choice-iff*
- by** *metis*
- let** $?x = (\lambda x. (\sum i \in Basis. ((f''\ i)\ x) *_{\mathbb{R}} i))$
- { **fix** $i::'a::euclidean-space$
- assume** $ass: i \in Basis$
- then have** $P\ i\ (f''\ i)$
- using** f''
- by** *metis*
- then have** $P\ i\ (\lambda x. (?x\ x) \cdot i)$ **using** ass **by** *auto*
- }
- hence** $*$: $\forall i \in Basis. P\ i\ (\lambda x. (?x\ x) \cdot i)$ **by** *auto*

```

then have ?rhs
  apply auto
proof
  let ?f'6 = ?x
  show  $\forall i \in \text{Basis}. P i (\lambda x. ?f'6 x \cdot i)$  using * by auto
qed}
ultimately show ?thesis by metis
qed

```

```

lemma indic-ident:
 $\bigwedge (f :: 'a \Rightarrow \text{real}) s. (\lambda x. (f x) * \text{indicator } s x) = (\lambda x. \text{if } x \in s \text{ then } f x \text{ else } 0)$ 
proof
  fix f :: 'a  $\Rightarrow$  real
  fix s :: 'a set
  fix x :: 'a
  show  $f x * \text{indicator } s x = (\text{if } x \in s \text{ then } f x \text{ else } 0)$ 
    by (simp add: indicator-def)
qed

```

```

lemma real-pair-basis:  $\text{Basis} = \{(1::\text{real}, 0::\text{real}), (0::\text{real}, 1::\text{real})\}$ 
  by (simp add: Basis-prod-def insert-commute)

```

```

lemma real-singleton-in-borel:
  shows  $\{a::\text{real}\} \in \text{sets borel}$ 
  using Borel-Space.cbox-borel[of a a]
  apply auto
  done

```

```

lemma real-singleton-in-lborel:
  shows  $\{a::\text{real}\} \in \text{sets lborel}$ 
  using real-singleton-in-borel
  apply auto
  done

```

```

lemma cbox-diff:
  shows  $\{0::\text{real}..1\} - \{0,1\} = \text{box } 0 \ 1$ 
  by (auto simp add: cbox-def)

```

```

lemma sum-bij:
  assumes bij F
     $\forall x \in s. f x = g (F x)$ 
  shows  $\bigwedge t. F ' s = t \implies \text{sum } f s = \text{sum } g t$ 
  by (metis assms bij-betw-def bij-betw-subset subset-UNIV sum.reindex-cong)

```

```

abbreviation surj-on where
  surj-on  $s f \equiv s \subseteq \text{range } f$ 

```

lemma *surj-on-image-vimage-eq*: $\text{surj-on } s f \implies f' (f -' s) = s$
by *fastforce*

end

theory *Derivs*

imports *General-Utills*

begin

lemma *field-simp-has-vector-derivative* [*derivative-intros*]:
 $(f \text{ has-field-derivative } y) F \implies (f \text{ has-vector-derivative } y) F$
by (*simp add: has-field-derivative-iff-has-vector-derivative*)

lemma *continuous-on-cases-empty* [*continuous-intros*]:
 $\llbracket \text{closed } S; \text{ continuous-on } S f; \bigwedge x. \llbracket x \in S; \neg P x \rrbracket \implies f x = g x \rrbracket \implies$
 $\text{continuous-on } S (\lambda x. \text{ if } P x \text{ then } f x \text{ else } g x)$
using *continuous-on-cases [of - {}]* **by** *force*

lemma *inj-on-cases*:
assumes *inj-on* f (*Collect* $P \cap S$) *inj-on* g (*Collect* (*Not* $\circ P$) $\cap S$)
 $f' (\text{Collect } P \cap S) \cap g' (\text{Collect } (\text{Not } \circ P) \cap S) = \{\}$
shows *inj-on* $(\lambda x. \text{ if } P x \text{ then } f x \text{ else } g x)$ S
using *assms* **by** (*force simp: inj-on-def*)

lemma *inj-on-arccos*: $S \subseteq \{-1..1\} \implies \text{inj-on } \arccos S$
by (*metis atLeastAtMost-iff cos-arccos inj-onI subsetCE*)

lemma *has-vector-derivative-componentwise-within*:
 $(f \text{ has-vector-derivative } f') (\text{at } a \text{ within } S) \iff$
 $(\forall i \in \text{Basis}. ((\lambda x. f x \cdot i) \text{ has-vector-derivative } (f' \cdot i)) (\text{at } a \text{ within } S))$
apply (*simp add: has-vector-derivative-def*)
apply (*subst has-derivative-componentwise-within*)
apply *simp*
done

lemma *has-vector-derivative-pair-within*:
fixes $f :: \text{real} \Rightarrow 'a::\text{euclidean-space}$ **and** $g :: \text{real} \Rightarrow 'b::\text{euclidean-space}$
assumes $\bigwedge u. u \in \text{Basis} \implies ((\lambda x. f x \cdot u) \text{ has-vector-derivative } f' \cdot u)$ (*at x within S*)
 $\bigwedge u. u \in \text{Basis} \implies ((\lambda x. g x \cdot u) \text{ has-vector-derivative } g' \cdot u)$ (*at x within S*)
shows $((\lambda x. (f x, g x)) \text{ has-vector-derivative } (f', g'))$ (*at x within S*)
apply (*subst has-vector-derivative-componentwise-within*)
apply (*auto simp: assms Basis-prod-def*)
done

lemma *piecewise-C1-differentiable-const*:
shows $(\lambda x. c)$ *piecewise-C1-differentiable-on* s
using *continuous-on-const*
by (*auto simp add: piecewise-C1-differentiable-on-def*)

```

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  $f :: \text{real} \Rightarrow 'a::\text{real-normed-vector}$ 
  shows  $(\lambda x. x)$  piecewise-C1-differentiable-on  $S$ 
  unfolding piecewise-C1-differentiable-on-def using C1-differentiable-on-ident
  by (blast intro: continuous-on-id C1-differentiable-on-ident)

```

```

lemma piecewise-C1-differentiable-on-mult [simp, derivative-intros]:
  fixes  $f :: \text{real} \Rightarrow 'a::\text{real-normed-algebra}$ 
  assumes  $f$  piecewise-C1-differentiable-on  $S$   $g$  piecewise-C1-differentiable-on  $S$ 
  shows  $(\lambda x. f\ x * g\ x)$  piecewise-C1-differentiable-on  $S$ 
  using assms
  unfolding piecewise-C1-differentiable-on-def
  apply safe
  apply (blast intro: continuous-intros)
  apply (rename-tac A B)
  apply (rule-tac x=A  $\cup$  B in exI)
  apply (auto intro: C1-differentiable-on-mult C1-differentiable-on-subset)
  done

```

```

lemma C1-differentiable-on-cdiv [simp, derivative-intros]:
  fixes  $f :: \text{real} \Rightarrow 'a :: \text{real-normed-field}$ 
  shows  $f$  C1-differentiable-on  $S \implies (\lambda x. f\ x / c)$  C1-differentiable-on  $S$ 
  by (simp add: divide-inverse)

```

```

lemma piecewise-C1-differentiable-on-cdiv [simp, derivative-intros]:
  fixes  $f :: \text{real} \Rightarrow 'a::\text{real-normed-field}$ 
  assumes  $f$  piecewise-C1-differentiable-on  $S$ 
  shows  $(\lambda x. f\ x / c)$  piecewise-C1-differentiable-on  $S$ 
  by (simp add: divide-inverse piecewise-C1-differentiable-const piecewise-C1-differentiable-on-mult
assms)

```

```

lemma sqrt-C1-differentiable [simp, derivative-intros]:
  assumes  $f: f$  C1-differentiable-on  $S$  and  $\text{fim}: f' S \subseteq \{0<..\}$ 
  shows  $(\lambda x. \text{sqrt}\ (f\ x))$  C1-differentiable-on  $S$ 
proof –
  have contf: continuous-on  $S$   $f$ 
  by (simp add: C1-differentiable-imp-continuous-on  $f$ )
  show ?thesis
  using assms
  unfolding C1-differentiable-on-def has-field-derivative-iff-has-vector-derivative
  [symmetric]

```

by (fastforce intro!: contf continuous-intros derivative-intros)
qed

lemma *sqrt-piecewise-C1-differentiable* [simp, derivative-intros]:
assumes $f: f$ *piecewise-C1-differentiable-on* S **and** $fim: f ' S \subseteq \{0<..\}$
shows $(\lambda x. \text{sqrt } (f x))$ *piecewise-C1-differentiable-on* S
using *assms*
unfolding *piecewise-C1-differentiable-on-def*
by (fastforce intro!: continuous-intros derivative-intros)

lemma
fixes $f :: \text{real} \Rightarrow 'a::\{\text{banach,real-normed-field}\}$
assumes $f: f$ *C1-differentiable-on* S
shows *sin-C1-differentiable* [simp, derivative-intros]: $(\lambda x. \text{sin } (f x))$ *C1-differentiable-on* S
and *cos-C1-differentiable* [simp, derivative-intros]: $(\lambda x. \text{cos } (f x))$ *C1-differentiable-on* S
proof –
have *contf: continuous-on* S f
by (simp add: *C1-differentiable-imp-continuous-on* f)
note $df\text{-sin} = \text{field-vector-diff-chain-at}$ [where $g=\text{sin}$, *unfolded o-def*]
note $df\text{-cos} = \text{field-vector-diff-chain-at}$ [where $g=\text{cos}$, *unfolded o-def*]
show $(\lambda x. \text{sin } (f x))$ *C1-differentiable-on* S $(\lambda x. \text{cos } (f x))$ *C1-differentiable-on* S
using *assms*
unfolding *C1-differentiable-on-def has-field-derivative-iff-has-vector-derivative*
[*symmetric*]
apply *auto*
by (rule contf continuous-intros $df\text{-sin}$ $df\text{-cos}$ derivative-intros *exI conjI ballI* |
force)
qed

lemma *has-derivative-abs*:
fixes $a::\text{real}$
assumes $a \neq 0$
shows $(\text{abs has-derivative } ((*) (\text{sgn } a)))$ (at a)
proof –
have [simp]: *norm = abs*
using *real-norm-def* **by** *force*
show ?thesis
using *has-derivative-norm* [where $'a=\text{real}$, *simplified*] *assms*
by (simp add: *mult-commute-abs*)
qed

lemma *abs-C1-differentiable* [simp, derivative-intros]:
fixes $f :: \text{real} \Rightarrow \text{real}$
assumes $f: f$ *C1-differentiable-on* S **and** $0 \notin f ' S$
shows $(\lambda x. \text{abs } (f x))$ *C1-differentiable-on* S
proof –

```

have contf: continuous-on S f
  by (simp add: C1-differentiable-imp-continuous-on f)
note df = DERIV-chain [where f=abs and g=f, unfolded o-def]
show ?thesis
  using assms
  unfolding C1-differentiable-on-def has-field-derivative-iff-has-vector-derivative
  [symmetric]
  apply clarify
  apply (rule df exI conjI ballI)+
  apply (force simp: has-field-derivative-def intro: has-derivative-abs continuous-intros
contf)+
  done
qed

```

```

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 ( $\lambda x. (f x, g x)$ ) C1-differentiable-on S
  using assms unfolding C1-differentiable-on-def
  apply safe
  apply (rename-tac A B)
  apply (intro exI ballI conjI)
  apply (rule-tac f'=A x and g'=B x in has-vector-derivative-pair-within)
  using has-vector-derivative-componentwise-within
  by (blast intro: continuous-on-Pair)+

```

```

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 ( $\lambda x. (f x, g x)$ ) piecewise-C1-differentiable-on S
  using assms unfolding piecewise-C1-differentiable-on-def
  by (blast intro!: continuous-intros C1-differentiable-on-pair del: continuous-on-discrete
intro: C1-differentiable-on-subset)

```

```

lemma test2:
  assumes s:  $\bigwedge x. x \in \{0..1\} - s \implies g$  differentiable at x
  and fs: finite s and uv:  $u \in \{0..1\} v \in \{0..1\} u \leq v$ 
  and  $x \in \{0..1\} x \notin (\lambda t. (v-u) *_{\mathbb{R}} t + u) - 's$ 
  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 -
  have i:(g has-vector-derivative vector-derivative g (at ((v - u) * x + u))) (at
((v-u) * x + u))
  using assms s [of (v - u) * x + u] uv mult-left-le [of x v-u]
  by (auto simp: vector-derivative-works)
  have ii:( $\lambda x. g ((v - u) * x + u)$ ) has-vector-derivative (v - u) *R vector-derivative
g (at ((v - u) * x + u))) (at x)
  by (intro vector-diff-chain-at [simplified o-def] derivative-eq-intros | simp add:
i)+

```

have 0 : $0 \leq (v - u) * x + u$
using *assms uv by auto*
have 1 : $(v - u) * x + u \leq 1$
using *assms uv*
by *simp (metis add.commute atLeastAtMost-iff atLeastatMost-empty-iff diff-ge-0-iff-ge-empty-iff le-diff-eq mult-left-le)*
have iii : *vector-derivative* g (at $((v - u) * x + u)$ within $\{0..1\}$) = *vector-derivative* g (at $((v - u) * x + u)$)
using *Derivative.vector-derivative-at-within-ivl[OF i, of 0 1, OF 0 1]*
by *auto*
have iv : *vector-derivative* $(\lambda x. g ((v - u) * x + u))$ (at x within $\{0..1\}$) = $(v - u) *_R$ *vector-derivative* g (at $((v - u) * x + u)$)
using *Derivative.vector-derivative-at-within-ivl[OF ii, of 0 1] assms*
by *auto*
show *?thesis*
using iii iv **by** *auto*
qed

lemma *C1-differentiable-on-components*:
assumes $\bigwedge i. i \in \text{Basis} \implies (\lambda x. f x \cdot i)$ *C1-differentiable-on* s
shows f *C1-differentiable-on* s
proof (*clarsimp simp add: C1-differentiable-on-def has-vector-derivative-def*)
have $*$: $\forall f i x. x *_R (f \cdot i) = (x *_R f) \cdot i$ **by** *auto*
have $\exists f'. \forall i \in \text{Basis}. \forall x \in s. ((\lambda x. f x \cdot i)$ *has-derivative* $(\lambda z. z *_R f' x \cdot i))$ (at x) \wedge *continuous-on* s f'
using *assms lambda-skolem-euclidean[of $\lambda i D. (\forall x \in s. ((\lambda x. f x \cdot i)$ *has-derivative* $(\lambda z. z *_R D x))$ (at x)) \wedge *continuous-on* s D]*
apply (*simp only: C1-differentiable-on-def has-vector-derivative-def **)
using *continuous-on-componentwise[of s]*
by *metis*
then obtain f' **where** $f': \forall i \in \text{Basis}. \forall x \in s. ((\lambda x. f x \cdot i)$ *has-derivative* $(\lambda z. z *_R f' x \cdot i))$ (at x) \wedge *continuous-on* s f'
by *auto*
then have 0 : $(\forall x \in s. (f$ *has-derivative* $(\lambda z. z *_R f' x))$ (at x)) \wedge *continuous-on* s f'
using *f' has-derivative-componentwise-within[of f , where $S = \text{UNIV}$]*
by *auto*
then show $\exists D. (\forall x \in s. (f$ *has-derivative* $(\lambda z. z *_R D x))$ (at x)) \wedge *continuous-on* s D **by** *metis*
qed

lemma *piecewise-C1-differentiable-on-components*:
assumes *finite* t
 $\bigwedge i. i \in \text{Basis} \implies (\lambda x. f x \cdot i)$ *C1-differentiable-on* $s - t$
 $\bigwedge i. i \in \text{Basis} \implies$ *continuous-on* s $(\lambda x. f x \cdot i)$
shows f *piecewise-C1-differentiable-on* s
using *C1-differentiable-on-components assms continuous-on-componentwise piecewise-C1-differentiable-on-de*
by *blast*

lemma *all-components-smooth-one-pw-smooth-is-pw-smooth*:
assumes $\bigwedge i. i \in \text{Basis} - \{j\} \implies (\lambda x. f x \cdot i)$ *C1-differentiable-on s*
assumes $(\lambda x. f x \cdot j)$ *piecewise-C1-differentiable-on s*
shows f *piecewise-C1-differentiable-on s*
proof –
have *is-cont*: $\forall i \in \text{Basis}. \text{continuous-on } s (\lambda x. f x \cdot i)$
using *assms C1-differentiable-imp-continuous-on piecewise-C1-differentiable-on-def*
by *fastforce*
obtain t **where** $t: (\text{finite } t \wedge (\lambda x. f x \cdot j) \text{ C1-differentiable-on } s - t)$ **using**
assms(2) piecewise-C1-differentiable-on-def **by** *auto*
show *?thesis*
using *piecewise-C1-differentiable-on-components[where ?f = f]*
using *assms(2) piecewise-C1-differentiable-on-def*
C1-differentiable-on-subset[OF assms(1) Diff-subset, where ?B1 = t] *t is-cont*
by *fastforce*
qed

lemma *derivative-component-fun-component*:
fixes $i::'a::\text{euclidean-space}$
assumes f *differentiable (at x)*
shows $((\text{vector-derivative } f \text{ (at } x)) \cdot i) = ((\text{vector-derivative } (\lambda x. (f x) \cdot i) \text{ (at } x)))$
proof –
have $(\lambda x. f x \cdot i)$ *has-vector-derivative* $\text{vector-derivative } f \text{ (at } x) \cdot i \text{ (at } x)$
using *assms and bounded-linear.has-vector-derivative[of (\lambda x. x \cdot i) f (vector-derivative f (at x) (at x)) and*
bounded-linear-inner-left[of i] and vector-derivative-works[of f (at x)]
by *blast*
then show $((\text{vector-derivative } f \text{ (at } x)) \cdot i) = ((\text{vector-derivative } (\lambda x. (f x) \cdot i) \text{ (at } x)))$
using *vector-derivative-works[of (\lambda x. (f x) \cdot i) (at x)] and*
differentiableI-vector[of (\lambda x. (f x) \cdot i) (vector-derivative f (at x) \cdot i) (at x)]
and
Derivative.vector-derivative-at
by *force*
qed

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 $\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{a..b\}) = \text{vector-derivative } \gamma \text{ (at } x)$
using *Derivative.vector-derivative-at-within-ivl[of \gamma vector-derivative \gamma (at x) x a b]*
gamma-differentiable x-within-bounds a-leq-b
by *(auto simp add: vector-derivative-works)*

lemma *islimpt-diff-finite*:
assumes *finite (t::'a::t1-space set)*

shows $x \text{ islimpt } s - t = x \text{ islimpt } s$
proof –
 have $iii: s - t = s - (t \cap s)$ **by** *auto*
 have $(t \cap s) \subseteq s$ **by** *auto*
 have $ii: \text{finite } (t \cap s)$ **using** *assms(1)* **by** *auto*
 have $i: (t \cap s) \cup (s - (t \cap s)) = (s)$
 using *assms* **by** *auto*
 then have $x \text{ islimpt } s - (t \cap s) = x \text{ islimpt } s$
 using *islimpt-Un-finite[OF ii, where ?t = s - (t \cap s)] i assms(1)*
 by *force*
 then show *?thesis* **using** *iii* **by** *auto*
qed

lemma *ivl-limpt-diff*:
 assumes *finite s a < b (x::real) ∈ {a..b} - s*
 shows $x \text{ islimpt } \{a..b\} - s$
proof –
 have $x \text{ islimpt } \{a..b\}$
proof (*cases x ∈ {a,b}*)
 have $i: \text{finite } \{a,b\}$ **and** $ii: \{a, b\} \cup \{a <..< b\} = \{a..b\}$ **using** *assms* **by** *auto*
 assume $x \in \{a,b\}$
 then show *?thesis*
 using *islimpt-Un-finite[OF i, where ?t = {a <..< b}]*
 by (*metis assms(2) empty-iff ii insert-iff islimpt-greaterThanLessThan1 islimpt-greaterThanLessThan2*)
next
 assume $x \notin \{a,b\}$
 then show $x \text{ islimpt } \{a..b\}$ **using** *assms* **by** *auto*
qed
 then show $x \text{ islimpt } \{a..b\} - s$ **using** *islimpt-diff-finite[OF assms(1)] assms*
 by *fastforce*
qed

lemma *ivl-closure-diff-del*:
 assumes *finite s a < b (x::real) ∈ {a..b} - s*
 shows $x \in \text{closure } ((\{a..b\} - s) - \{x\})$
using *ivl-limpt-diff islimpt-in-closure assms* **by** *blast*

lemma *ivl-not-trivial-limit-within*:
 assumes *finite s*
 $a < b$
 $(x::real) \in \{a..b\} - s$
 shows $\text{at } x \text{ within } \{a..b\} - s \neq \text{bot}$
using *assms ivl-closure-diff-del not-trivial-limit-within*
by *blast*

lemma *vector-derivative-at-within-non-trivial-limit*:
 at x within $s \neq \text{bot} \wedge (f \text{ has-vector-derivative } f') \text{ (at } x) \implies$
 $\text{vector-derivative } f \text{ (at } x \text{ within } s) = f'$

using *has-vector-derivative-at-within* *vector-derivative-within* **by** *fastforce*

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*) \implies
vector-derivative *f* (at *x* within {*a..b*} - *s*) = *f'*

using *vector-derivative-at-within-non-trivial-limit* *ivl-not-trivial-limit-within* **by** *fastforce*

lemma *gamma-deriv-at-within-diff*:

assumes *a-leq-b*: *a < b* **and**

x-within-bounds: *x* \in {*a..b*} - *s* **and**

gamma-differentiable: $\forall x \in \{a .. b\} - s$. γ *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*)

using *vector-derivative-at-within-ivl-diff* [*of s a b x* γ *vector-derivative* γ (at *x*)]
gamma-differentiable

x-within-bounds *a-leq-b* *s-subset* *finite-s*

by (*auto simp add: vector-derivative-works*)

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: $\forall x \in s$. γ *differentiable* at *x*

shows *vector-derivative* γ (at *x* within ({*a..b*})) = *vector-derivative* γ (at *x*)

using *Derivative.vector-derivative-at-within-ivl* [*of* γ *vector-derivative* γ (at *x*) *x*
a b]

gamma-differentiable *x-within-bounds* *a-leq-b* *s-subset*

by (*auto simp add: vector-derivative-works*)

lemma *derivative-component-fun-component-at-within-gen*:

assumes *gamma-differentiable*: $\forall x \in s$. γ *differentiable* at *x* **and** *s-subset*: *s* \subseteq
{*0..1*}

shows $\forall x \in s$. *vector-derivative* (λx . γx) (at *x* within {*0..1*}) \cdot (*i::'a*: *eclidean-space*)
= *vector-derivative* (λx . $\gamma x \cdot i$) (at *x* within {*0..1*})

proof -

have *gamma-i-component-smooth*:

$\forall x \in s$. (λx . $\gamma x \cdot i$) *differentiable* at *x*

using *gamma-differentiable*

by *auto*

show $\forall x \in s$. *vector-derivative* (λx . γx) (at *x* within {*0..1*}) \cdot *i*

= *vector-derivative* (λx . $\gamma x \cdot i$) (at *x* within {*0..1*})

proof

fix *x::real*

assume *x-within-bounds*: *x* \in *s*

have *gamma-deriv-at-within*:

vector-derivative (λx . γx) (at *x* within {*0..1*}) = *vector-derivative* (λx . γ

x) (at x)
using *gamma-deriv-at-within-gen*[of 0 1] *x-within-bounds*
gamma-differentiable s-subset
by (*auto simp add: vector-derivative-works*)
then have *gamma-component-deriv-at-within*:
vector-derivative ($\lambda x. \gamma x \cdot i$) (at x)
 $=$ *vector-derivative* ($\lambda x. \gamma x \cdot i$) (at x within {0..1})
using *gamma-deriv-at-within-gen*[of 0 1, **where** $\gamma = (\lambda x. \gamma x \cdot i)$] *x-within-bounds*
gamma-i-component-smooth s-subset
by (*auto simp add: vector-derivative-works*)
have *gamma-component-deriv-eq-gamma-deriv-component*:
vector-derivative ($\lambda x. \gamma x \cdot i$) (at x) $=$ *vector-derivative* ($\lambda x. \gamma x$) (at x) $\cdot i$
using *derivative-component-fun-component*[of $\gamma x i$] *gamma-differentiable*
x-within-bounds
by auto
show *vector-derivative* γ (at x within {0..1}) $\cdot i =$ *vector-derivative* ($\lambda x. \gamma x$
 $\cdot i$) (at x within {0..1})
using *gamma-component-deriv-eq-gamma-deriv-component* *gamma-component-deriv-at-within*
gamma-deriv-at-within
by auto
qed
qed

lemma *derivative-component-fun-component-at-within*:

assumes *gamma-differentiable*: $\forall x \in \{0 .. 1\}. \gamma$ *differentiable at* x
shows $\forall x \in \{0..1\}. \text{vector-derivative } (\lambda x. \gamma x) \text{ (at } x \text{ within } \{0..1\}) \cdot (i::'a::$
euclidean-space)
 $=$ *vector-derivative* ($\lambda x. \gamma x \cdot i$) (at x within {0..1})

proof –

have *gamma-i-component-smooth*:
 $\forall x \in \{0 .. 1\}. (\lambda x. \gamma x \cdot i)$ *differentiable at* x
using *gamma-differentiable* **by auto**
show $\forall x \in \{0..1\}. \text{vector-derivative } (\lambda x. \gamma x) \text{ (at } x \text{ within } \{0..1\}) \cdot i$
 $=$ *vector-derivative* ($\lambda x. \gamma x \cdot i$) (at x within {0..1})

proof

fix $x::\text{real}$
assume *x-within-bounds*: $x \in \{0..1\}$
have *gamma-deriv-at-within*:
vector-derivative ($\lambda x. \gamma x$) (at x within {0..1}) $=$ *vector-derivative* ($\lambda x. \gamma x$)
(at x)
using *gamma-deriv-at-within*[of 0 1] *x-within-bounds*
gamma-differentiable
by (*auto simp add: vector-derivative-works*)
have *gamma-component-deriv-at-within*:
vector-derivative ($\lambda x. \gamma x \cdot i$) (at x) $=$ *vector-derivative* ($\lambda x. \gamma x \cdot i$) (at x
within {0..1})
using *Derivative.vector-derivative-at-within-ivl*[of ($\lambda x. (\gamma x) \cdot i$) *vector-derivative*
($\lambda x. (\gamma x) \cdot i$) (at x) x 0 1]
has-vector-derivative-at-within[of ($\lambda x. \gamma x \cdot i$) *vector-derivative* ($\lambda x. \gamma x \cdot$

```

i) (at x) x {0..1}
  gamma-i-component-smooth x-within-bounds
  by (simp add: vector-derivative-works)
  have gamma-component-deriv-eq-gamma-deriv-component:
    vector-derivative (λx. γ x · i) (at x) = vector-derivative (λx. γ x) (at x) · i
  using derivative-component-fun-component[of γ x i] gamma-differentiable
x-within-bounds
  by auto
  show vector-derivative γ (at x within {0..1}) · i = vector-derivative (λx. γ x
· i) (at x within {0..1})
  using gamma-component-deriv-eq-gamma-deriv-component gamma-component-deriv-at-within
gamma-deriv-at-within
  by auto
qed
qed

```

```

lemma straight-path-differentiable-x:
  fixes b :: real and y1 :: real
  assumes gamma-def: γ = (λx. (b, y2 + y1 * x))
  shows ∀x. γ differentiable at x
  unfolding gamma-def differentiable-def
  by (fast intro!: derivative-intros)

```

```

lemma straight-path-differentiable-y:
  fixes b :: real and
  y1 y2 :: real
  assumes gamma-def: γ = (λx. (y2 + y1 * x, b))
  shows ∀x. γ differentiable at x
  unfolding gamma-def differentiable-def
  by (fast intro!: derivative-intros)

```

```

lemma piecewise-C1-differentiable-on-imp-continuous-on:
  assumes f piecewise-C1-differentiable-on s
  shows continuous-on s f
  using assms
  by (auto simp add: piecewise-C1-differentiable-on-def)

```

```

lemma boring-lemma1:
  fixes f :: real ⇒ real
  assumes (f has-vector-derivative D) (at x)
  shows ((λx. (f x, 0)) has-vector-derivative ((D, 0 :: real))) (at x)
proof -
  have *: ((λx. (f x) *R (1, 0)) has-vector-derivative (D *R (1, 0))) (at x)
  using bounded-linear.has-vector-derivative[OF Real-Vector-Spaces.bounded-linear-scaleR-left
assms(1),
of (1, 0)] by auto
  have ((λx. (f x) *R (1, 0)) has-vector-derivative (D, 0)) (at x)
proof -
  have (D, 0 :: 'a) = D *R (1, 0)

```

```

    by simp
  then show ?thesis
    by (metis (no-types) *)
qed
then show ?thesis by auto
qed

```

lemma boring-lemma2:

```

  fixes f :: real ⇒ real
  assumes (f has-vector-derivative D) (at x)
  shows ((λx. (0, f x)) has-vector-derivative (0, D)) (at x)
proof -
  have *: ((λx. (f x) *R (0,1)) has-vector-derivative (D *R (0,1))) (at x)
    using bounded-linear.has-vector-derivative[OF Real-Vector-Spaces.bounded-linear-scaleR-left
  assms(1),
    of (0,1)] by auto
  then have ((λx. (f x) *R (0,1)) has-vector-derivative ((0,D))) (at x)
    using scaleR-Pair Real-Vector-Spaces.real-scaleR-def
  proof -
    have (0::'b, D) = D *R (0, 1)
      by auto
    then show ?thesis
      by (metis (no-types) *)
  qed
  then show ?thesis by auto
qed

```

lemma pair-prod-smooth-pw-smooth:

```

  assumes (f::real ⇒ real) C1-differentiable-on s (g::real ⇒ real) piecewise-C1-differentiable-on
  s
  shows (λx. (f x, g x)) piecewise-C1-differentiable-on s
proof -
  have f-cont: continuous-on s f
    using assms(1) C1-differentiable-imp-continuous-on
    by fastforce
  have g-cont: continuous-on s g
    using assms(2) by (auto simp add: piecewise-C1-differentiable-on-def)
  obtain t where t:(finite t ∧ g C1-differentiable-on s - t) using assms(2)
  piecewise-C1-differentiable-on-def by auto
  show ?thesis
    using piecewise-C1-differentiable-on-components[where ?f = (λx. (f x, g x))]
    apply (simp add: real-pair-basis)
    using assms(2) piecewise-C1-differentiable-on-def
    C1-differentiable-on-subset[OF assms(1) Diff-subset, where ?B1 = t] t
    f-cont g-cont
    by fastforce
qed

```

lemma scale-shift-smooth:

shows $(\lambda x. a + b * x)$ *C1-differentiable-on s*
proof –
show $(\lambda x. a + b * x)$ *C1-differentiable-on s*
using *Cauchy-Integral-Theorem.C1-differentiable-on-mult*
Cauchy-Integral-Theorem.C1-differentiable-on-add
Cauchy-Integral-Theorem.C1-differentiable-on-const
Cauchy-Integral-Theorem.C1-differentiable-on-ident
by *auto*
qed

lemma *open-diff*:
assumes *finite (t::'a::t1-space set)*
open (s::'a set)
shows *open (s - t)*
using *assms*
proof(*induction t*)
show *open s* \implies *open (s - {})* **by** *auto*
next
fix *x::'a::t1-space*
fix *F::'a::t1-space set*
assume *step: finite F x \notin F open s*
then have *i: (s - insert x F) = (s - F) - {x}* **by** *auto*
assume *ind-hyp: (open s \implies open (s - F))*
show *open (s - insert x F)*
apply (*simp only: i*)
using *open-delete[of s - F] ind-hyp[OF step(3)]* **by** *auto*
qed

lemma *has-derivative-transform-within*:
assumes $0 < d$
and $x \in s$
and $\forall x' \in s. \text{dist } x' x < d \implies f x' = g x'$
and (*f has-derivative f'*) (*at x within s*)
shows (*g has-derivative f'*) (*at x within s*)
using *assms*
unfolding *has-derivative-within*
by (*force simp add: intro: Lim-transform-within*)

lemma *has-derivative-transform-within-ivl*:
assumes $(0::\text{real}) < b$
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*)
using *has-derivative-transform-within[of b x {a..b} - s] assms*
by *auto*

lemma *has-vector-derivative-transform-within-ivl*:
assumes $(0::\text{real}) < b$

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)
using *assms has-derivative-transform-within-ivl*
apply (*auto simp add: has-vector-derivative-def*)
by *blast*

lemma *has-derivative-transform-at:*

assumes $0 < d$
and $\forall x'. \text{dist } x' x < d \longrightarrow f x' = g x'$
and (*f has-derivative f'*) (at x)
shows (*g has-derivative f'*) (at x)
using *has-derivative-transform-within [of d x UNIV f g f'] assms*
by *simp*

lemma *has-vector-derivative-transform-at:*

assumes $0 < d$
and $\forall x'. \text{dist } x' x < d \longrightarrow f x' = g x'$
and (*f has-vector-derivative f'*) (at x)
shows (*g has-vector-derivative f'*) (at x)
using *assms*
unfolding *has-vector-derivative-def*
by (*rule has-derivative-transform-at*)

lemma *C1-diff-components-2:*

assumes $b \in \text{Basis}$
assumes *f C1-differentiable-on s*
shows $(\lambda x. f x \cdot b)$ *C1-differentiable-on s*
proof –
obtain *D* **where** $D: (\forall x \in s. (f \text{ has-derivative } (\lambda z. z *_R D x)) (at x))$ *continuous-on s D*
using *assms(2)* **by** (*fastforce simp add: C1-differentiable-on-def has-vector-derivative-def*)
show *?thesis*
proof (*simp add: C1-differentiable-on-def has-vector-derivative-def, intro exI conjI*)
show *continuous-on s* $(\lambda x. D x \cdot b)$ **using** *D continuous-on-componentwise assms(1)* **by** *fastforce*
show $(\forall x \in s. ((\lambda x. f x \cdot b) \text{ has-derivative } (\lambda y. y * (\lambda x. D x \cdot b) x)) (at x))$
using *has-derivative-inner-left D(1)* **by** *fastforce*
qed
qed

lemma *eq-smooth:*

assumes $0 < d$
 $\forall x \in s. \forall y. \text{dist } x y < d \longrightarrow f y = g y$
f C1-differentiable-on s
shows *g C1-differentiable-on s*
proof (*simp add: C1-differentiable-on-def*)

obtain D **where** D : $(\forall x \in s. (f \text{ has-vector-derivative } D x) (at x)) \wedge \text{continuous-on } s D$
using $assms$ **by** $(\text{auto simp add: } C1\text{-differentiable-on-def})$
then have f : $(\forall x \in s. (g \text{ has-vector-derivative } D x) (at x))$
using $assms(1-2)$
by $(metis \text{dist-commute has-vector-derivative-transform-at})$
have $(\forall x \in s. (g \text{ has-vector-derivative } D x) (at x)) \wedge \text{continuous-on } s D$ **using** D
by $auto$
then show $\exists D. (\forall x \in s. (g \text{ has-vector-derivative } D x) (at x)) \wedge \text{continuous-on } s D$ **by** $metis$
qed

lemma $eq-pw\text{-smooth}$:

assumes $0 < d$
 $\forall x \in s. \forall y. \text{dist } x y < d \longrightarrow f y = g y$
 $\forall x \in s. f x = g x$
 $f \text{ piecewise-}C1\text{-differentiable-on } s$
shows $g \text{ piecewise-}C1\text{-differentiable-on } s$
proof $(\text{simp add: piecewise-}C1\text{-differentiable-on-def})$
have $g\text{-cont}$: $\text{continuous-on } s g$ **using** $assms \text{piecewise-}C1\text{-differentiable-const}$
by $(\text{simp add: piecewise-}C1\text{-differentiable-on-def})$
obtain t **where** t : $\text{finite } t \wedge f \text{ } C1\text{-differentiable-on } s - t$
using $assms$ **by** $(\text{auto simp add: piecewise-}C1\text{-differentiable-on-def})$
then have $g \text{ } C1\text{-differentiable-on } s - t$ **using** $assms \text{eq-smooth}$ **by** $blast$
then show $\text{continuous-on } s g \wedge (\exists t. \text{finite } t \wedge g \text{ } C1\text{-differentiable-on } s - t)$
using $t \text{ } g\text{-cont}$ **by** $metis$
qed

lemma $scale\text{-piecewise-}C1\text{-differentiable-on}$:

assumes $f \text{ piecewise-}C1\text{-differentiable-on } s$
shows $(\lambda x. (c::\text{real}) * (f x)) \text{ piecewise-}C1\text{-differentiable-on } s$
proof $(\text{simp add: piecewise-}C1\text{-differentiable-on-def, intro conjI})$
show $\text{continuous-on } s (\lambda x. c * f x)$
using $assms \text{continuous-on-mult-left}$
by $(\text{auto simp add: piecewise-}C1\text{-differentiable-on-def})$
show $\exists t. \text{finite } t \wedge (\lambda x. c * f x) \text{ } C1\text{-differentiable-on } s - t$
using $assms \text{continuous-on-mult-left}$
by $(\text{auto simp add: piecewise-}C1\text{-differentiable-on-def})$
qed

lemma $eq\text{-smooth-gen}$:

assumes $f \text{ } C1\text{-differentiable-on } s$
 $\forall x. f x = g x$
shows $g \text{ } C1\text{-differentiable-on } s$
using $assms$ **unfolding** $C1\text{-differentiable-on-def}$
by $(metis (\text{no-types, lifting}) \text{has-vector-derivative-weaken UNIV-I top-greatest})$

lemma $subpath\text{-compose}$:

shows $(\text{subpath } a b \gamma) = \gamma o (\lambda x. (b - a) * x + a)$

```

by (auto simp add: subpath-def)

lemma subpath-smooth:
  assumes  $\gamma$  C1-differentiable-on  $\{0..1\}$   $0 \leq a < b \leq 1$ 
  shows (subpath a b  $\gamma$ ) C1-differentiable-on  $\{0..1\}$ 
proof -
  have  $\gamma$  C1-differentiable-on  $\{a..b\}$ 
  apply (rule C1-differentiable-on-subset)
  using assms by auto
  then have  $\gamma$  C1-differentiable-on  $(\lambda x. (b - a) * x + a)$  '  $\{0..1\}$ 
  using  $\langle a < b \rangle$  closed-segment-eq-real-ivl closed-segment-real-eq by auto
  moreover have finite  $(\{0..1\} \cap (\lambda x. (b - a) * x + a)^{-1} \{x\})$  for  $x$ 
  proof -
    have  $(\lambda x. (b - a) * x + a)^{-1} \{x\} = \{(x - a) / (b - a)\}$ 
    using assms by (auto simp add: divide-simps)
    then show ?thesis
    by auto
  qed
  ultimately show ?thesis
  by (force simp add: subpath-compose intro: C1-differentiable-compose derivative-intros)
qed

lemma has-vector-derivative-divide[derivative-intros]:
  fixes  $a :: 'a :: \text{real-normed-field}$ 
  shows (f has-vector-derivative  $x$ )  $F \implies ((\lambda x. f x / a)$  has-vector-derivative  $(x / a)$ )  $F$ 
  unfolding divide-inverse by (fact has-vector-derivative-mult-left)

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

lemma gauge-integral-Fubini-universe-x:
  fixes  $f :: ('a :: \text{euclidean-space} * 'b :: \text{euclidean-space}) \Rightarrow 'c :: \text{euclidean-space}$ 
  assumes fun-lesbegue-integrable: integrable lborel f and
  x-axis-integral-measurable:  $(\lambda x. \text{integral UNIV } (\lambda y. f(x, y))) \in \text{borel-measurable lborel}$ 
  shows  $\text{integral UNIV } f = \text{integral UNIV } (\lambda x. \text{integral UNIV } (\lambda y. f(x, y)))$ 
   $(\lambda x. \text{integral UNIV } (\lambda y. f(x, y)))$  integrable-on UNIV
proof -
  have f-is-measurable:  $f \in \text{borel-measurable lborel}$ 
  using fun-lesbegue-integrable and borel-measurable-integrable
  by auto
  have fun-lborel-prod-integrable:
  integrable  $(\text{lborel} \otimes_M \text{lborel}) f$ 
  using fun-lesbegue-integrable
  by (simp add: lborel-prod)
  then have region-integral-is-one-twoD-integral:

```

```

    LBINT x. LBINT y. f (x, y) = integralL (lborel  $\otimes_M$  lborel) f
  using lborel-pair.integral-fst'
  by auto
  then have AE-one-D-integrals-eq: AE x in lborel. LBINT y. f (x, y) = integral
  UNIV ( $\lambda y. f(x,y)$ )
  proof -
    have AE x in lborel. integrable lborel ( $\lambda y. f(x,y)$ )
      using lborel-pair.AE-integrable-fst' and fun-lborel-prod-integrable
      by blast
    then show ?thesis
      using integral-lborel and always-eventually
      and AE-mp
      by fastforce
  qed
  have one-D-integral-measurable:
    ( $\lambda x. LBINT y. f (x, y)$ )  $\in$  borel-measurable lborel
    using f-is-measurable and lborel.borel-measurable-lebesgue-integral
    by auto
  then have second-lesbegue-integral-eq:
    LBINT x. LBINT y. f (x, y) = LBINT x. (integral UNIV ( $\lambda y. f(x,y)$ ))
    using x-axis-integral-measurable and integral-cong-AE and AE-one-D-integrals-eq
    by blast
  have integrable lborel ( $\lambda x. LBINT y. f (x, y)$ )
    using fun-lborel-prod-integrable and lborel-pair.integrable-fst'
    by auto
  then have oneD-gauge-integral-lesbegue-integrable:
    integrable lborel ( $\lambda x. integral UNIV (\lambda y. f(x,y))$ )
    using x-axis-integral-measurable and AE-one-D-integrals-eq and integrable-cong-AE-imp
    by blast
  then show one-D-gauge-integral-integrable:
    ( $\lambda x. integral UNIV (\lambda y. f(x,y))$ ) integrable-on UNIV
    using integrable-on-lborel
    by auto
  have LBINT x. (integral UNIV ( $\lambda y. f(x,y)$ )) = integral UNIV ( $\lambda x. integral$ 
  UNIV ( $\lambda y. f(x, y)$ ))
    using integral-lborel oneD-gauge-integral-lesbegue-integrable
    by fastforce
  then have twoD-lesbeuge-eq-twoD-gauge:
    LBINT x. LBINT y. f (x, y) = integral UNIV ( $\lambda x. integral UNIV (\lambda y. f(x,$ 
  y)))
    using second-lesbegue-integral-eq
    by auto
  then show integral UNIV f = integral UNIV ( $\lambda x. integral UNIV (\lambda y. f(x,y))$ )
    using fun-lesbegue-integrable and integral-lborel and region-integral-is-one-twoD-integral
    by (metis lborel-prod)
  qed

lemma gauge-integral-Fubini-universe-y:
  fixes f :: ('a::euclidean-space * 'b::euclidean-space)  $\Rightarrow$  'c::euclidean-space

```

assumes *fun-lesbegue-integrable*: *integrable lborel f* **and**
y-axis-integral-measurable: $(\lambda x. \text{integral UNIV } (\lambda y. f(y, x))) \in \text{borel-measurable lborel}$
shows $\text{integral UNIV } f = \text{integral UNIV } (\lambda x. \text{integral UNIV } (\lambda y. f(y, x)))$
 $(\lambda x. \text{integral UNIV } (\lambda y. f(y, x))) \text{integrable-on UNIV}$
proof –
have *f-is-measurable*: $f \in \text{borel-measurable lborel}$
using *fun-lesbegue-integrable* **and** *borel-measurable-integrable*
by *auto*
have *fun-lborel-prod-integrable*:
 $\text{integrable } (\text{lborel } \otimes_M \text{lborel}) f$
using *fun-lesbegue-integrable*
by (*simp add: lborel-prod*)
then have *region-integral-is-one-twoD-integral*:
 $\text{LBINT } x. \text{LBINT } y. f(y, x) = \text{integral}^L (\text{lborel } \otimes_M \text{lborel}) f$
using *lborel-pair.integral-fst'*
f-is-measurable lborel-pair.integrable-product-swap lborel-pair.integral-fst lborel-pair.integral-product-swap
lborel-prod
by *force*
then have *AE-one-D-integrals-eq*: *AE x in lborel. LBINT y. f(y, x) = integral UNIV (λy. f(y,x))*
proof –
have *AE x in lborel. integrable lborel (λy. f(y,x))*
using *lborel-pair.AE-integrable-fst'* **and** *fun-lborel-prod-integrable*
lborel-pair.AE-integrable-fst lborel-pair.integrable-product-swap
by *blast*
then show *?thesis*
using *integral-lborel* **and** *always-eventually*
and *AE-mp*
by *fastforce*
qed
have *one-D-integral-measurable*:
 $(\lambda x. \text{LBINT } y. f(y, x)) \in \text{borel-measurable lborel}$
using *f-is-measurable* **and** *lborel.borel-measurable-lebesgue-integral*
by *auto*
then have *second-lesbegue-integral-eq*:
 $\text{LBINT } x. \text{LBINT } y. f(y, x) = \text{LBINT } x. (\text{integral UNIV } (\lambda y. f(y, x)))$
using *y-axis-integral-measurable* **and** *integral-cong-AE* **and** *AE-one-D-integrals-eq*
by *blast*
have *integrable lborel (λx. LBINT y. f(y, x))*
using *fun-lborel-prod-integrable* **and** *lborel-pair.integrable-fst'*
by (*simp add: lborel-pair.integrable-fst lborel-pair.integrable-product-swap*)
then have *oneD-gauge-integral-lesbegue-integrable*:
 $\text{integrable lborel } (\lambda x. \text{integral UNIV } (\lambda y. f(y, x)))$
using *y-axis-integral-measurable* **and** *AE-one-D-integrals-eq* **and** *integrable-cong-AE-imp*
by *blast*
then show *one-D-gauge-integral-integrable*:
 $(\lambda x. \text{integral UNIV } (\lambda y. f(y, x))) \text{integrable-on UNIV}$
using *integrable-on-lborel*

by auto
have $LBINT\ x. (integral\ UNIV\ (\lambda y. f(y, x))) = integral\ UNIV\ (\lambda x. integral\ UNIV\ (\lambda y. f(y, x)))$
using *integral-lborel oneD-gauge-integral-lesbegue-integrable*
by fastforce
then have *twoD-lesbeuge-eq-twoD-gauge:*
 $LBINT\ x. LBINT\ y. f(y, x) = integral\ UNIV\ (\lambda x. integral\ UNIV\ (\lambda y. f(y, x)))$
using *second-lesbegue-integral-eq*
by auto
then show $integral\ UNIV\ f = integral\ UNIV\ (\lambda x. integral\ UNIV\ (\lambda y. f(y, x)))$
using *fun-lesbegue-integrable and integral-lborel and region-integral-is-one-twoD-integral*
by (*metis lborel-prod*)
qed

lemma *gauge-integral-Fubini-curve-bounded-region-x:*

fixes $f\ g :: ('a::euclidean-space * 'b::euclidean-space) \Rightarrow 'c::euclidean-space$ **and**
 $g1\ g2 :: 'a \Rightarrow 'b$ **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: $(\lambda x. integral\ UNIV\ (\lambda y. f(x, y))) \in borel-measurable$
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: $s = \{(x, y). (\forall i \in Basis. a \cdot i \leq x \cdot i \wedge x \cdot i \leq b \cdot i) \wedge$

$(\forall i \in Basis. (g1\ x) \cdot i \leq y \cdot i \wedge y \cdot i \leq (g2\ x) \cdot i)\}$

shows $integral\ s\ g = integral\ (cbox\ a\ b)\ (\lambda x. integral\ (cbox\ (g1\ x)\ (g2\ x))\ (\lambda y. g(x, y)))$

proof –

have *two-D-integral-to-one-D: $integral\ UNIV\ f = integral\ UNIV\ (\lambda x. integral\ UNIV\ (\lambda y. f(x, y)))$*

using *gauge-integral-Fubini-universe-x and fun-lesbegue-integrable and x-axis-integral-measurable*
by auto

have *one-d-integral-integrable: $(\lambda x. integral\ UNIV\ (\lambda y. f(x, y)))$ integrable-on UNIV*

using *gauge-integral-Fubini-universe-x(2) and assms*

by blast

have *case-x-in-range:*

$\forall x \in cbox\ a\ b. integral\ (cbox\ (g1\ x)\ (g2\ x))\ (\lambda y. g(x, y)) = integral\ UNIV\ (\lambda y. f(x, y))$

proof

fix $x :: 'a$

assume *within-range: $x \in (cbox\ a\ b)$*

let *?f-one-D-spec = $(\lambda y. if\ y \in (cbox\ (g1\ x)\ (g2\ x))\ then\ g(x, y)\ else\ 0)$*

have *f-one-D-region: $(\lambda y. f(x, y)) = (\lambda y. if\ y \in cbox\ (g1\ x)\ (g2\ x)\ then\ g(x, y)\ else\ 0)$*

proof

```

fix y::'b
show f (x, y) = (if y ∈ (cbox (g1 x) (g2 x)) then g (x, y) else 0)
  apply (simp add: f-is-g-indicator s-is-bounded-by-g1-and-g2)
  using within-range
  apply (simp add: cbox-def)
  by smt
qed
have zero-out-of-bound: ∀ y. y ∉ cbox (g1 x) (g2 x) → f (x,y) = 0
  using f-is-g-indicator and s-is-bounded-by-g1-and-g2
  by (auto simp add: cbox-def)
have (λy. f(x,y)) integrable-on cbox (g1 x) (g2 x)
proof -
  have ?f-one-D-spec integrable-on UNIV
    using f-one-D-region and x-axis-gauge-integrable
    by metis
  then have ?f-one-D-spec integrable-on cbox(g1 x) (g2 x)
    using integrable-on-subcbox
    by blast
  then show ?thesis using f-one-D-region by auto
qed
then have f-integrale-x: ((λy. f(x,y)) has-integral (integral (cbox (g1 x) (g2
x)) (λy. f(x,y)))) (cbox (g1 x) (g2 x))
  using integrable-integral and within-range and x-axis-gauge-integrable
  by auto
have integral (cbox (g1 x) (g2 x)) (λy. f (x, y)) = integral UNIV (λy. f (x,
y))
  using has-integral-on-superset[OF f-integrale-x - Set.subset-UNIV] zero-out-of-bound
  by (simp add: integral-unique)
then have ((λy. f(x, y)) has-integral integral UNIV (λy. f (x, y))) (cbox (g1
x) (g2 x))
  using f-integrale-x
  by simp
then have ((λy. g(x, y)) has-integral integral UNIV (λy. f (x, y))) (cbox (g1
x)(g2 x))
  using Henstock-Kurzweil-Integration.has-integral-restrict [OF subset-refl ]
and
  f-one-D-region
  by (smt has-integral-eq)
then show integral (cbox (g1 x) (g2 x)) (λy. g (x, y)) = integral UNIV (λy.
f (x, y))
  by auto
qed
have case-x-not-in-range:
  ∀ x. x ∉ cbox a b → integral UNIV (λy. f(x,y)) = 0
proof
fix x::'a
have x ∉ (cbox a b) → (∀ y. f(x,y) = 0)
  apply (simp add: s-is-bounded-by-g1-and-g2 f-is-g-indicator cbox-def)
  by auto

```

```

then show  $x \notin \text{cbox } a \ b \longrightarrow \text{integral UNIV } (\lambda y. f(x, y)) = 0$ 
  by (simp)
qed
have RHS:  $\text{integral UNIV } (\lambda x. \text{integral UNIV } (\lambda y. f(x, y))) = \text{integral } (\text{cbox } a \ b)$ 
 $(\lambda x. \text{integral } (\text{cbox } (g1 \ x) \ (g2 \ x)) (\lambda y. g(x, y)))$ 
proof –
  let ?first-integral =  $(\lambda x. \text{integral } (\text{cbox } (g1 \ x) \ (g2 \ x)) (\lambda y. g(x, y)))$ 
  let ?x-integral-cases =  $(\lambda x. \text{if } x \in \text{cbox } a \ b \ \text{then } \text{?first-integral } x \ \text{else } 0)$ 
  have x-integral-cases-integral:  $(\lambda x. \text{integral UNIV } (\lambda y. f(x, y))) = \text{?x-integral-cases}$ 
    using case-x-in-range and case-x-not-in-range
    by auto
  have  $((\lambda x. \text{integral UNIV } (\lambda y. f(x, y))) \text{ has-integral } (\text{integral UNIV } f)) \text{ UNIV}$ 
    using two-D-integral-to-one-D
    one-d-integral-integrable
    by auto
  then have  $(\text{?x-integral-cases} \text{ has-integral } (\text{integral UNIV } f)) \text{ UNIV}$ 
    using x-integral-cases-integral by auto
  then have  $(\text{?first-integral} \text{ has-integral } (\text{integral UNIV } f)) (\text{cbox } a \ b)$ 
    using has-integral-restrict-UNIV[of cbox a b ?first-integral integral UNIV f]
    by auto
  then show ?thesis
    using two-D-integral-to-one-D
    by (simp add: integral-unique)
qed
have f-integrable:f integrable-on UNIV
  using fun-lesbegue-integrable and integrable-on-lborel
  by auto
then have LHS:  $\text{integral UNIV } f = \text{integral } s \ g$ 
  apply (simp add: f-is-g-indicator)
  using integrable-restrict-UNIV
    integral-restrict-UNIV
  by auto
then show ?thesis
  using RHS and two-D-integral-to-one-D
  by auto
qed

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

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

```

($\forall i \in \text{Basis}. (g1\ x) \cdot i \leq y \cdot i \wedge y \cdot i \leq$
 $(g2\ x) \cdot i$)

shows $\text{integral } s\ g = \text{integral } (\text{cbox } a\ b) (\lambda x. \text{integral } (\text{cbox } (g1\ x)\ (g2\ x)) (\lambda y. g(y, x)))$

proof –

have *two-D-integral-to-one-D: integral UNIV f = integral UNIV* ($\lambda x. \text{integral UNIV } (\lambda y. f(y, x))$)

using *gauge-integral-Fubini-universe-y and fun-lesbegue-integrable and y-axis-integral-measurable*
by *auto*

have *one-d-integral-integrable: ($\lambda x. \text{integral UNIV } (\lambda y. f(y, x))$) integrable-on UNIV*

using *gauge-integral-Fubini-universe-y(2) and assms*
by *blast*

have *case-y-in-range:*
 $\forall x \in \text{cbox } a\ b. \text{integral } (\text{cbox } (g1\ x)\ (g2\ x)) (\lambda y. g(y, x)) = \text{integral UNIV } (\lambda y. f(y, x))$

proof

fix $x::'b$

assume *within-range: $x \in (\text{cbox } a\ b)$*

let $?f\text{-one-D-spec} = (\lambda y. \text{if } y \in (\text{cbox } (g1\ x)\ (g2\ x)) \text{ then } g(y, x) \text{ else } 0)$

have *f-one-D-region: ($\lambda y. f(y, x) = (\lambda y. \text{if } y \in \text{cbox } (g1\ x)\ (g2\ x) \text{ then } g(y, x) \text{ else } 0)$*

proof

fix $y::'a$

show $f(y, x) = (\text{if } y \in (\text{cbox } (g1\ x)\ (g2\ x)) \text{ then } g(y, x) \text{ else } 0)$

apply (*simp add: f-is-g-indicator s-is-bounded-by-g1-and-g2*)

using *within-range*

apply (*simp add: cbox-def*)

by *smt*

qed

have *zero-out-of-bound: $\forall y. y \notin \text{cbox } (g1\ x)\ (g2\ x) \longrightarrow f(y, x) = 0$*

using *f-is-g-indicator and s-is-bounded-by-g1-and-g2*
by (*auto simp add: cbox-def*)

have ($\lambda y. f(y, x)$) *integrable-on cbox (g1 x) (g2 x)*

proof –

have *?f-one-D-spec integrable-on UNIV*

using *f-one-D-region and y-axis-gauge-integrable*
by *metis*

then have *?f-one-D-spec integrable-on cbox(g1 x) (g2 x)*

using *integrable-on-subcbox*
by *blast*

then show *?thesis using f-one-D-region by auto*

qed

then have *f-integrale-y: (($\lambda y. f(y, x)$) has-integral (integral (cbox (g1 x) (g2 x)) ($\lambda y. f(y, x)$))) (cbox (g1 x) (g2 x))*

using *integrable-integral and within-range and y-axis-gauge-integrable*
by *auto*

have $\text{integral } (\text{cbox } (g1\ x)\ (g2\ x)) (\lambda y. f(y, x)) = \text{integral UNIV } (\lambda y. f(y, x))$


```

using has-integral-on-superset[OF f-integrale-y - Set.subset-UNIV] zero-out-of-bound
by (simp add: integral-unique)
then have  $((\lambda y. f(y, x)) \text{ has-integral integral UNIV } (\lambda y. f(y, x)))$  (cbox (g1
x) (g2 x))
using f-integrale-y
by simp
then have  $((\lambda y. g(y, x)) \text{ has-integral integral UNIV } (\lambda y. f(y, x)))$  (cbox (g1
x)(g2 x))
using Henstock-Kurzweil-Integration.has-integral-restrict [OF subset-refl ]
and
f-one-D-region
by (smt has-integral-eq)
then show  $\text{integral (cbox (g1 x) (g2 x)) } (\lambda y. g(y, x)) = \text{integral UNIV } (\lambda y.$ 
 $f(y, x))$ 
by auto
qed
have case-y-not-in-range:
 $\forall x. x \notin \text{cbox } a \ b \longrightarrow \text{integral UNIV } (\lambda y. f(y, x)) = 0$ 
proof
fix x::'b
have  $x \notin \text{cbox } a \ b \longrightarrow (\forall y. f(y, x) = 0)$ 
apply (simp add: s-is-bounded-by-g1-and-g2 f-is-g-indicator cbox-def)
by auto
then show  $x \notin \text{cbox } a \ b \longrightarrow \text{integral UNIV } (\lambda y. f(y, x)) = 0$ 
by (simp)
qed
have RHS: integral UNIV  $(\lambda x. \text{integral UNIV } (\lambda y. f(y, x))) = \text{integral (cbox } a$ 
 $b) (\lambda x. \text{integral (cbox (g1 x) (g2 x)) } (\lambda y. g(y, x)))$ 
proof –
let ?first-integral  $= (\lambda x. \text{integral (cbox (g1 x) (g2 x)) } (\lambda y. g(y, x)))$ 
let ?x-integral-cases  $= (\lambda x. \text{if } x \in \text{cbox } a \ b \text{ then } ?\text{first-integral } x \text{ else } 0)$ 
have y-integral-cases-integral:  $(\lambda x. \text{integral UNIV } (\lambda y. f(y, x))) = ?\text{x-integral-cases}$ 
using case-y-in-range and case-y-not-in-range
by auto
have  $(\lambda x. \text{integral UNIV } (\lambda y. f(y, x))) \text{ has-integral (integral UNIV } f) \text{ UNIV}$ 
using two-D-integral-to-one-D
one-d-integral-integrable
by auto
then have  $(?\text{x-integral-cases} \text{ has-integral (integral UNIV } f) \text{ UNIV}$ 
using y-integral-cases-integral by auto
then have  $(?\text{first-integral} \text{ has-integral (integral UNIV } f) \text{ (cbox } a \ b)$ 
using has-integral-restrict-UNIV[of cbox a b ?first-integral integral UNIV f]
by auto
then show ?thesis
using two-D-integral-to-one-D
by (simp add: integral-unique)
qed
have f-integrable:f integrable-on UNIV
using fun-lesbeque-integrable and integrable-on-lborel

```

```

    by auto
  then have LHS: integral UNIV f = integral s g
    apply (simp add: f-is-g-indicator)
    using integrable-restrict-UNIV
      integral-restrict-UNIV
    by auto
  then show ?thesis
    using RHS and two-D-integral-to-one-D
    by auto
qed

lemma gauge-integral-by-substitution:
  fixes f::(real  $\Rightarrow$  real) and
    g::(real  $\Rightarrow$  real) and
    g'::real  $\Rightarrow$  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:  $\forall x \in \{a..b\}. (g \text{ has-vector-derivative } (g' x)) \text{ (at } x \text{ within } \{a..b\})$ 
  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 -
    have  $\forall x \in \{a..b\}. (g \text{ has-real-derivative } (g' x)) \text{ (at } x \text{ within } \{a..b\})$ 
      using has-field-derivative-iff-has-vector-derivative[of g] and g'-derivative
      by auto
    then have 2: interval-lebesgue-integral lborel (ereal (a)) (ereal (b)) ( $\lambda x. g' x *_R$ 
      f (g x))
      = interval-lebesgue-integral lborel (ereal (g a)) (ereal (g b)) f
      using interval-integral-substitution-finite[of a b g g' f] and g'-continuous and
      a-le-b and f-continuous
      by auto
    have g-continuous: continuous-on {a .. b} g
      using Derivative.differentiable-imp-continuous-on
      apply (simp add: differentiable-on-def differentiable-def)
      by (metis continuous-on-vector-derivative g'-derivative)
    have set-integrable lborel {a .. b} ( $\lambda x. g' x *_R f (g x)$ )
    proof -
      have continuous-on {a .. b} ( $\lambda x. g' x *_R f (g x)$ )
    proof -
      have continuous-on {a .. b} ( $\lambda x. f (g x)$ )
    proof -
      show ?thesis
        using Topological-Spaces.continuous-on-compose f-continuous g-continuous
        by auto
    qed
  then show ?thesis

```

```

    using Limits.continuous-on-mult g'-continuous
    by auto
  qed
  then show ?thesis
    using borel-integrable-atLeastAtMost' by blast
  qed
  then have 0: interval-lebesgue-integral lborel (ereal (a)) (ereal (b)) ( $\lambda x. g' x *_{\mathbb{R}} f (g x)$ )
    = integral {a .. b} ( $\lambda x. g' x *_{\mathbb{R}} f (g x)$ )
    using a-le-b and interval-integral-eq-integral
    by (metis (no-types))
  have set-integrable lborel {g a .. g b} f
  proof -
    have continuous-on {g a .. g b} f
    proof -
      have {g a .. g b}  $\subseteq$  g ` {a .. b}
      using g-continuous
      by (metis a-le-b atLeastAtMost-iff atLeastAtMost-subset-iff continuous-image-closed-interval
imageI order-refl)
    then show continuous-on {g a .. g b} f
      using f-continuous continuous-on-subset
      by blast
    qed
  then show ?thesis
    using borel-integrable-atLeastAtMost'
    by blast
  qed
  then have 1: interval-lebesgue-integral lborel (ereal (g a)) (ereal (g b)) f
    = integral {g a .. g b} f
    using ga-le-gb and interval-integral-eq-integral
    by (metis (no-types))
  then show ?thesis
    using 0 and 1 and 2
    by (metis (no-types, lifting) Henstock-Kurzweil-Integration.integral-cong mult.commute
real-scaleR-def)
  qed

```

```

lemma frontier-ic:
  assumes a < (b::real)
  shows frontier {a<..b} = {a,b}
  apply(simp add: frontier-def)
  using assms
  by auto

```

```

lemma frontier-ci:
  assumes a < (b::real)
  shows frontier {a<..} = {a,b}
  apply(simp add: frontier-def)
  using assms

```

by *auto*

lemma *ic-not-closed*:

assumes $a < (b::real)$

shows $\neg \text{closed } \{a <..b\}$

using *assms frontier-subset-eq frontier-ic greaterThanAtMost-iff* **by** *blast*

lemma *closure-ic-union-ci*:

assumes $a < (b::real)$ $b < c$

shows $\text{closure } (\{a..<b\} \cup \{b<..c\}) = \{a .. c\}$

using *frontier-ic[OF assms(1)] frontier-ci[OF assms(2)] closure-Un assms*

apply(*simp add: frontier-def*)

by *auto*

lemma *interior-ic-ci-union*:

assumes $a < (b::real)$ $b < c$

shows $b \notin (\text{interior } (\{a..<b\} \cup \{b<..c\}))$

proof –

have $b \notin (\{a..<b\} \cup \{b<..c\})$ **by** *auto*

then show *?thesis*

using *interior-subset* **by** *blast*

qed

lemma *frontier-ic-union-ci*:

assumes $a < (b::real)$ $b < c$

shows $b \in \text{frontier } (\{a..<b\} \cup \{b<..c\})$

using *closure-ic-union-ci assms interior-ic-ci-union*

by(*simp add: frontier-def*)

lemma *ic-union-ci-not-closed*:

assumes $a < (b::real)$ $b < c$

shows $\neg \text{closed } (\{a..<b\} \cup \{b<..c\})$

proof –

have $b \notin (\{a..<b\} \cup \{b<..c\})$ **by** *auto*

then show *?thesis*

using *assms frontier-subset-eq frontier-ic-union-ci[OF assms]*

by (*auto simp only: subset-iff*)

qed

lemma *integrable-continuous*:

fixes $f :: 'b::euclidean-space \Rightarrow 'a::banach$

assumes *continuous-on* (*cbox* a b) f

shows f *integrable-on* *cbox* a b

by (*simp add: assms integrable-continuous*)

lemma *removing-singletons-from-div*:

assumes $\forall t \in S. \exists c d :: real. c < d \wedge \{c..d\} = t$

$\{x\} \cup \bigcup S = \{a..b\}$ $a < x < b$

finite S

shows $\exists t \in S. x \in t$
proof(*rule ccontr*)
assume $\neg(\exists t \in S. x \in t)$
then have $\forall t \in S. x \notin t$ **by** *auto*
then have $x \notin \bigcup S$ **by** *auto*
then have $i: \bigcup S = \{a..b\} - \{x\}$ **using** *assms (2)* **by** *auto*
have $x \in \{a..b\}$ **using** *assms* **by** *auto*
then have $\{a..b\} - \{x\} = \{a..<x\} \cup \{x<..b\}$ **by** *auto*
then have $0: \bigcup S = \{a..<x\} \cup \{x<..b\}$ **using** i **by** *auto*
have $1: \text{closed}(\bigcup S)$
apply(*rule closed-Union*)
proof-
show *finite S*
using *assms* **by** *auto*
show $\forall T \in S. \text{closed } T$ **using** *assms* **by** *auto*
qed
show *False* **using** $0\ 1$ *ic-union-ci-not-closed* *assms* **by** *auto*
qed

lemma *remove-singleton-from-division-of*:
assumes A *division-of* $\{a..b\}$ $a < b$
assumes $x \in \{a..b\}$
shows $\exists c\ d. c < d \wedge \{c..d\} \in A \wedge x \in \{c..d\}$
proof -
from *assms* **have** $x \text{ islimpt } \{a..b\}$
by (*intro connected-imp-perfect*) *auto*
also have $\{a..b\} = \{x. \{x..x\} \in A\} \cup (\{a..b\} - \{x. \{x..x\} \in A\})$
using *assms* **by** *auto*
also have $x \text{ islimpt } \dots \iff x \text{ islimpt } \{a..b\} - \{x. \{x..x\} \in A\}$
proof (*intro islimpt-Un-finite*)
have $\{x. \{x..x\} \in A\} \subseteq \text{Inf } 'A$
proof *safe*
fix x **assume** $\{x..x\} \in A$
thus $x \in \text{Inf } 'A$
by (*auto intro!*: *bexI[of - {x}] simp: image-iff*)
qed
moreover from *assms* **have** *finite A* **by** (*auto simp: division-of-def*)
hence *finite (Inf 'A)* **by** *auto*
ultimately show *finite* $\{x. \{x..x\} \in A\}$ **by** (*rule finite-subset*)
qed
also have $\{a..b\} = \bigcup A$
using *assms* **by** (*auto simp: division-of-def*)
finally have $x \text{ islimpt } \bigcup (A - \text{range } (\lambda x. \{x..x\}))$
by (*rule islimpt-subset*) *auto*
moreover have *closed* $(\bigcup (A - \text{range } (\lambda x. \{x..x\})))$
using *assms* **by** (*intro closed-Union*) *auto*
ultimately have $x \in (\bigcup (A - \text{range } (\lambda x. \{x..x\})))$
by (*auto simp: closed-limpt*)
then obtain X **where** $x \in X \wedge X \in A \wedge X \notin \text{range } (\lambda x. \{x..x\})$

by *blast*
moreover from *division-ofD(2)[OF assms(1) this(2)] division-ofD(3)[OF assms(1) this(2)]*
division-ofD(4)[OF assms(1) this(2)]
obtain $c\ d$ **where** $X = \text{cbox } c\ d\ X \subseteq \{a..b\}\ X \neq \{\}$ **by** *blast*
ultimately have $c \leq d$ **by** *auto*
have $c \neq d$
proof
assume $c = d$
with $\langle X = \text{cbox } c\ d \rangle$ **have** $X = \{c..c\}$ **by** *auto*
hence $X \in \text{range } (\lambda x. \{x..x\})$ **by** *blast*
with $\langle X \notin \text{range } (\lambda x. \{x..x\}) \rangle$ **show** *False* **by** *contradiction*
qed
with $\langle c \leq d \rangle$ **have** $c < d$ **by** *simp*
with $\langle X = \text{cbox } c\ d \rangle$ **and** $\langle x \in X \rangle$ **and** $\langle X \in A \rangle$ **show** *?thesis*
by *auto*
qed

lemma *remove-singleton-from-tagged-division-of:*
assumes A *tagged-division-of* $\{a::\text{real}..b\}$ $a < b$
assumes $x \in \{a..b\}$
shows $\exists k\ c\ d. c < d \wedge (k, \{c..d\}) \in A \wedge x \in \{c..d\}$
using *remove-singleton-from-division-of[OF division-of-tagged-division[OF assms(1)] assms(2)]*

using *assms(3)* **by** *fastforce*

lemma *tagged-div-wo-singletons:*
assumes p *tagged-division-of* $\{a::\text{real}..b\}$ $a < b$
shows $(p - \{xk. \exists x\ y. xk = (x, \{y\})\})$ *tagged-division-of* $\text{cbox } a\ b$
using *remove-singleton-from-tagged-division-of[OF assms] assms*
apply *(auto simp add: tagged-division-of-def tagged-partial-division-of-def)*
apply *blast*
apply *blast*
apply *blast*
by *fastforce*

lemma *tagged-div-wo-empty:*
assumes p *tagged-division-of* $\{a::\text{real}..b\}$ $a < b$
shows $(p - \{xk. \exists x. xk = (x, \{\})\})$ *tagged-division-of* $\text{cbox } a\ b$
using *remove-singleton-from-tagged-division-of[OF assms] assms*
apply *(auto simp add: tagged-division-of-def tagged-partial-division-of-def)*
apply *blast*
apply *blast*
apply *blast*
by *fastforce*

lemma *fine-diff:*
assumes γ *fine* p

```

shows  $\gamma$  fine ( $p - s$ )
apply (auto simp add: fine-def)
using assms by auto

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

proof –
have ( $\forall (x::\text{real}, K) \in p. \exists x'. x' \notin s \wedge x' \in \text{interior } K$ )
proof –
  {fix  $x::\text{real}$ 
   fix  $K$ 
   assume ass: ( $x::\text{real}, K$ )  $\in p$ 
   have ( $\forall (x, K) \in p. \text{infinite } (\text{interior } K)$ )
     using assms(4) infinite-Ioo interior-atLeastAtMost-real
     by (smt split-beta)
   then have  $i$ : infinite (interior  $K$ ) using ass by auto
   have  $\exists x'. x' \notin s \wedge x' \in \text{interior } K$ 
     using infinite-imp-nonempty[OF Diff-infinite-finite[OF assms(1)  $i$ ]] by
auto}
  then show ?thesis by auto
qed
then obtain  $f$  where  $f$ : ( $\forall (x::\text{real}, K) \in p. (f(x, K)) \notin s \wedge (f(x, K)) \in \text{interior } K$ )
  using choice-iff[where  $?Q = \lambda(x, K) x'. (x::\text{real}, K) \in p \longrightarrow x' \notin s \wedge x' \in \text{interior } K$ ]
  apply (auto simp add: case-prod-beta)
  by metis
have  $f'$ : ( $\forall (x::\text{real}, K) \in p. (f(x, K)) \notin s \wedge (f(x, K)) \in K$ )
  using  $f$  interior-subset
  by (auto simp add: case-prod-beta subset-iff)
let  $?p' = \{m. (\exists xK. m = ((f xK), \text{snd } xK) \wedge xK \in p)\}$ 
have  $0$ : ( $\forall (x, K) \in ?p'. x \notin s$ )
  using  $f$ 
  by (auto simp add: case-prod-beta)
have  $i$ : finite  $\{(f(a, b), b) \mid a b. (a, b) \in p\}$ 
proof –
  have  $a$ :  $\{(f(a, b), b) \mid a b. (a, b) \in p\} = (\% (a, b). (f(a, b), b)) ' p$  by auto
  have  $b$ : finite  $p$  using assms(2) by auto
  show ?thesis using  $a b$  by auto
qed
have  $1$ :  $?p'$  tagged-division-of  $\{a..b\}$ 
  using assms(2)  $f'$ 
apply (auto simp add: tagged-division-of-def tagged-partial-division-of-def case-prod-beta)
  apply (metis i)
  apply blast

```

```

apply blast
by fastforce

have f-inj: inj-on f p
apply(simp add: inj-on-def)
proof -
  {fix x y
   assume x ∈ p y ∈ p
   f x = f y
   then have x = y
   using f
   tagged-division-ofD(5)[OF assms(2)]

   by (smt case-prodE disjoint-insert(2) mk-disjoint-insert)}note * = this
show ∀ x ∈ p. ∀ y ∈ p. f x = f y → x = y using * by auto
qed
let ?γ' = λx. if (∃ xK ∈ p. f xK = x) then (γ o fst o the-inv-into p f) x else
γ x
have 2: ?γ' fine ?p' using assms(3)
apply(auto simp add: fine-def case-prod-beta the-inv-into-f-f[OF f-inj])
by force
have 3: gauge ?γ'
using assms(5) assms(3) f'
apply(auto simp add: fine-def gauge-def case-prod-beta the-inv-into-f-f[OF
f-inj])
by force
have ?p' tagged-division-of {a..b} ∧ ?γ' fine ?p' ∧ (∀ (x, K) ∈ ?p'. x ∉ s) ∧ gauge
?γ'
using 0 1 2 3 by auto
then show ?thesis by smt
qed

lemma has-integral-bound-spike-finite:
fixes f :: 'a::euclidean-space ⇒ 'b::real-normed-vector
assumes 0 ≤ B and finite S
and f: (f has-integral i) (cbox a b)
and leB: ∧x. x ∈ cbox a b - S ⇒ norm (f x) ≤ B
shows norm i ≤ B * content (cbox a b)
proof -
define g where g ≡ (λx. if x ∈ S then 0 else f x)
then have ∧x. x ∈ cbox a b - S ⇒ norm (g x) ≤ B
using leB by simp
moreover have (g has-integral i) (cbox a b)
using has-integral-spike-finite [OF ⟨finite S⟩ - f]
by (simp add: g-def)
ultimately show ?thesis
by (simp add: ⟨0 ≤ B⟩ g-def has-integral-bound)
qed

```


lemma *has-integral-bound-*:
fixes $f :: \text{real} \Rightarrow 'a::\text{real-normed-vector}$
assumes $a < b$
and $0 \leq B$
and $f: (f \text{ has-integral } i) (cbox \ a \ b)$
and *finite* s
and $\forall x \in (cbox \ a \ b) - s. \text{norm } (f \ x) \leq B$
shows $\text{norm } i \leq B * \text{content } (cbox \ a \ b)$
using *has-integral-bound-spike-finite* **assms** **by** *blast*

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

by (*metis* *assms*(1) *assms*(3) *assms*(4) *box-real*(2) *f has-integral-bound-spike-finite*)

lemma *integral-has-vector-derivative-continuous-at'*:
fixes $f :: \text{real} \Rightarrow 'a::\text{banach}$
assumes *finite* s
and $f: f \text{ integrable-on } \{a..b\}$
and $x: x \in \{a..b\} - s$
and $f_x: \text{continuous } (\text{at } x \text{ within } (\{a..b\} - s)) \ f$
shows $((\lambda u. \text{integral } \{a..u\} \ f) \text{ has-vector-derivative } f \ x) (\text{at } x \text{ within } (\{a..b\} - s))$

proof –
let $?I = \lambda a \ b. \text{integral } \{a..b\} \ f$
{ **fix** $e::\text{real}$
assume $e > 0$
obtain d **where** $d > 0$ **and** $d: \bigwedge x'. \llbracket x' \in \{a..b\} - s; |x' - x| < d \rrbracket \implies \text{norm}(f \ x' - f \ x) \leq e$
using $\langle e > 0 \rangle \ f_x$ **by** (*auto simp: continuous-within-eps-delta dist-norm less-imp-le*)
have $\text{norm } (\text{integral } \{a..y\} \ f - \text{integral } \{a..x\} \ f - (y-x) *_R f \ x) \leq e * |y - x|$
if $y: y \in \{a..b\} - s$ **and** $yx: |y - x| < d$ **for** y
proof (*cases* $y < x$)
case *False*
have $f \text{ integrable-on } \{a..y\}$
using $f \ y$ **by** (*simp add: integrable-subinterval-real*)
then **have** *Idiff*: $?I \ a \ y - ?I \ a \ x = ?I \ x \ y$
using *False* x **by** (*simp add: algebra-simps integral-combine*)
have $f_{ux-int}: ((\lambda u. f \ u - f \ x) \text{ has-integral } \text{integral } \{x..y\} \ f - (y-x) *_R f \ x)$
 $\{x..y\}$
apply (*rule has-integral-diff*)
using $x \ y$ **apply** (*auto intro: integrable-integral [OF integrable-subinterval-real [OF f]]*)

```

using has-integral-const-real [of f x x y] False
apply (simp add: )
done
show ?thesis
using False
apply (simp add: abs-eq-content del: content-real-if measure-lborel-Icc)
apply (rule has-integral-bound-real'[where  $f=(\lambda u. f u - f x)$ ])
using  $yx$  False  $d$   $x$   $y$   $\langle e>0$  apply (auto simp add: Idiff fux-int)
proof-
  let  $?M48 = mset-set$   $s$ 
  show  $\bigwedge z. y - x < d \implies (\bigwedge x'. a \leq x' \wedge x' \leq b \wedge x' \notin s \implies |x' - x| < d \implies norm (f x' - f x) \leq e) \implies 0 < e \implies z \notin ?M48 \implies a \leq x \implies x \notin s \implies y \leq b \implies y \notin s \implies x \leq z \implies z \leq y \implies norm (f z - f x) \leq e$ 
    using assms by auto
  qed
next
case True
have  $f$  integrable-on  $\{a..x\}$ 
  using  $f$   $x$  by (simp add: integrable-subinterval-real)
then have Idiff:  $?I a x - ?I a y = ?I y x$ 
  using True  $x$   $y$  by (simp add: algebra-simps integral-combine)
have  $fux-int$ :  $((\lambda u. f u - f x)$  has-integral integral  $\{y..x\}$   $f - (x - y) *_{\mathbb{R}} f$ 
 $x$ )  $\{y..x\}$ 
  apply (rule has-integral-diff)
using  $x$   $y$  apply (auto intro: integrable-integral [OF integrable-subinterval-real [OF f]])
  using has-integral-const-real [of f x y x] True
  apply (simp add: )
  done
have  $norm$  (integral  $\{a..x\}$   $f - \text{integral } \{a..y\} f - (x - y) *_{\mathbb{R}} f$   $x$ )  $\leq e * |y - x|$ 
  using True
  apply (simp add: abs-eq-content del: content-real-if measure-lborel-Icc)
  apply (rule has-integral-bound-real'[where  $f=(\lambda u. f u - f x)$ ])
  using  $yx$  True  $d$   $x$   $y$   $\langle e>0$  apply (auto simp add: Idiff fux-int)
proof-
  let  $?M44 = mset-set$   $s$ 
  show  $\bigwedge xa. x - y < d \implies y < x \implies (\bigwedge x'. a \leq x' \wedge x' \leq b \wedge x' \notin s \implies |x' - x| < d \implies norm (f x' - f x) \leq e) \implies 0 < e \implies xa \notin ?M44 \implies x \leq b \implies x \notin s \implies a \leq y \implies y \notin s \implies y \leq xa \implies xa \leq x \implies norm (f xa - f x) \leq e$ 
    using assms by auto
  qed
then show ?thesis
  by (simp add: algebra-simps norm-minus-commute)
qed
then have  $\exists d>0. \forall y \in \{a..b\} - s. |y - x| < d \implies norm$  (integral  $\{a..y\}$   $f - \text{integral } \{a..x\} f - (y-x) *_{\mathbb{R}} f$   $x$ )  $\leq e * |y - x|$ 
  using  $\langle d>0$  by blast

```

```

}
then show ?thesis
by (simp add: has-vector-derivative-def has-derivative-within-alt bounded-linear-scaleR-left)
qed

```

```

lemma integral-has-vector-derivative':
  fixes f :: real  $\Rightarrow$  'a::banach
  assumes finite s
    f integrable-on {a..b}
    x  $\in$  {a..b} - s
    continuous (at x within {a..b} - s) f
  shows (( $\lambda u$ . integral {a .. u} f) has-vector-derivative f(x)) (at x within {a .. b}
- s)
  apply (rule integral-has-vector-derivative-continuous-at')
  using assms
  apply (auto simp: continuous-on-eq-continuous-within)
  done

```

```

lemma fundamental-theorem-of-calculus-interior-stronger:
  fixes f :: real  $\Rightarrow$  'a::banach
  assumes finite S
    and a  $\leq$  b  $\wedge$  x. x  $\in$  {a <..< b} - S  $\implies$  (f has-vector-derivative f'(x)) (at x)
    and continuous-on {a .. b} f
  shows (f' has-integral (f b - f a)) {a .. b}
  using assms
proof (induction arbitrary: a b)
  case empty
  then show ?case
  using fundamental-theorem-of-calculus-interior by force
next
  case (insert x S)
  show ?case
  proof (cases x  $\in$  {a <..< b})
    case False then show ?thesis
    using insert by blast
  next
    case True then have a < x x < b
    by auto
    have (f' has-integral f x - f a) {a..x}
    apply (rule insert)
    using (a < x) (x < b) insert.prem1 continuous-on-subset by force+
    moreover have (f' has-integral f b - f x) {x..b}
    apply (rule insert)
    using (x < b) (a < x) insert.prem2 continuous-on-subset by force+
    ultimately show ?thesis
    by (meson finite-insert fundamental-theorem-of-calculus-interior-strong in-
sert.hyps(1) insert.prem1 insert.prem2 insert.prem3)
  qed
qed

```

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 –
have $interior\ (\{a..b\} - S) = \{a <..<b\} - S$
using $\langle finite\ S \rangle$
by $(simp\ add:\ interior-diff\ finite-imp-closed)$
then show *?thesis*
using *at-within-interior assms* **by** *fastforce*
qed

lemma *at-within-cbox-finite*:
assumes $x \in cbox\ a\ b$ $x \notin S$ *finite S*
shows $(at\ x\ within\ cbox\ a\ b - S) = at\ x$
proof –
have $interior\ (cbox\ a\ b - S) = cbox\ a\ b - S$
using $\langle finite\ S \rangle$ **by** $(simp\ add:\ interior-diff\ finite-imp-closed)$
then show *?thesis*
using *at-within-interior assms* **by** *fastforce*
qed

lemma *fundamental-theorem-of-calculus-interior-stronger'*:
fixes $f :: real \Rightarrow 'a::banach$
assumes *finite S*
and $a \leq b$ $\wedge x. x \in \{a <..<b\} - S \implies (f\ has-vector-derivative\ f'(x))$ $(at\ x\ within\ \{a..b\} - S)$
and *continuous-on {a .. b} f*
shows $(f'\ has-integral\ (f\ b - f\ a))\ \{a .. b\}$
using *assms fundamental-theorem-of-calculus-interior-strong at-within-cbox-finite*

by $(metis\ DiffD1\ DiffD2\ interior-atLeastAtMost-real\ interior-cbox\ interval-cbox)$

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 $deriv\ [derivative-intros]:$
 $\wedge x. x \in \{a..b\} - s \implies (g\ has-field-derivative\ g'\ x)$ $(at\ x\ within\ \{a..b\})$
shows $((\lambda x. g'\ x *_{\mathbb{R}} f\ (g\ x))\ has-integral\ (integral\ \{g\ a..g\ b\} f - integral\ \{g\ b..g\ a\} f))\ \{a..b\}$
proof –
let $?F = \lambda x. integral\ \{c..g\ x\} f$
have *cont-int: continuous-on {a..b} ?F*
by $(rule\ continuous-on-compose2[OF\ -\ g(1)\ subset])\ indefinite-integral-continuous-1$

```

    f)+
  have deriv:  $\bigwedge x. x \in \{a..b\} - s \implies (((\lambda x. \text{integral } \{c..x\} f) \circ g) \text{ has-vector-derivative } g' x *_{\mathbb{R}} f (g x))$ 
    (at x within ( $\{a..b\} - s$ ))
  apply (rule has-vector-derivative-eq-rhs)
  apply (rule vector-diff-chain-within)
  apply (subst has-field-derivative-iff-has-vector-derivative [symmetric])
proof-
  fix x::real
  assume ass:  $x \in \{a..b\} - s$ 
  let ?f'3 =  $g' x$ 
  have i:  $\{a..b\} - s \subseteq \{a..b\}$  by auto
  have ii: ( $g$  has-vector-derivative  $g' x$ ) (at x within  $\{a..b\}$ ) using deriv[OF ass]
    by (simp only: has-field-derivative-iff-has-vector-derivative)
  show ( $g$  has-real-derivative ?f'3) (at x within  $\{a..b\} - s$ )
    using has-vector-derivative-within-subset[OF ii i]
    by (simp only: has-field-derivative-iff-has-vector-derivative)
next
  let ?g'3 =  $f \circ g$ 
  show  $\bigwedge x. x \in \{a..b\} - s \implies ((\lambda x. \text{integral } \{c..x\} f) \text{ has-vector-derivative } ?g'3 x)$  (at  $(g x)$  within  $g' \{a..b\} - s$ )
proof-
  fix x::real
  assume ass:  $x \in \{a..b\} - s$ 
  have finite ( $g' s$ ) using s by auto
  then have i: ( $\lambda x. \text{integral } \{c..x\} f$ ) has-vector-derivative  $f(g x)$  (at  $(g x)$  within  $\{c..d\} - g' s$ )
    apply (rule integral-has-vector-derivative')
  proof-
    show  $f$  integrable-on  $\{c..d\}$  using f by auto
    show  $g x \in \{c..d\} - g' s$  using ass subset
  by (smt Diff-iff Un-upper1 Un-upper2 g(2) imageE image-subset-iff inj-onD subsetCE)
  show continuous (at  $(g x)$  within  $\{c..d\} - g' s$ ) f
    using  $\langle g x \in \{c..d\} - g' s \rangle$  continuous-on-eq-continuous-within f(2) by blast
  qed
  have ii:  $g' (\{a..b\} - s) \subseteq (\{c..d\} - g' s)$ 
    using subset g(2)
  by (smt Diff-subset Un-Diff Un-commute Un-upper2 inj-on-image-set-diff subset-trans sup.order-iff)
  then show  $((\lambda x. \text{integral } \{c..x\} f) \text{ has-vector-derivative } ?g'3 x)$  (at  $(g x)$  within  $g' (\{a..b\} - s)$ )
    by (smt Diff-subset has-vector-derivative-weaken Un-upper1 Un-upper2  $\langle \text{finite } (g' s) \rangle$  ass comp-def continuous-on-eq-continuous-within f(1) f(2) g(2))

```

image-diff-subset image-subset-iff inj-on-image-set-diff integral-has-vector-derivative-continuous-at' subset-trans)

```

qed
  show  $\bigwedge x. x \in \{a..b\} - s \implies g' x *_R ?g' \exists x = g' x *_R f (g x)$  by auto
qed
have deriv: (?F has-vector-derivative  $g' x *_R f (g x)$ )
  (at x within  $\{a..b\} - s$ ) if  $x \in \{a < .. < b\} - (s)$  for  $x$ 
  using deriv[of x] that by (simp add: at-within-Icc-at o-def)
have ( $\lambda x. g' x *_R f (g x)$ ) has-integral ( $?F b - ?F a$ )  $\{a..b\}$ 
  using cont-int
  using fundamental-theorem-of-calculus-interior-stronger'[OF s le deriv]
  by blast
also
from subset have  $g x \in \{c..d\}$  if  $x \in \{a..b\}$  for  $x$  using that by blast
from this[of a] this[of b] le have  $cd: c \leq g a \wedge g b \leq d \wedge c \leq g b \wedge g a \leq d$  by auto
have integral  $\{c..g b\} f - \text{integral } \{c..g a\} f = \text{integral } \{g a..g b\} f - \text{integral } \{g b..g a\} f$ 
proof cases
  assume  $g a \leq g b$ 
  note  $le = le$  this
  from cd have integral  $\{c..g a\} f + \text{integral } \{g a..g b\} f = \text{integral } \{c..g b\} f$ 
  using integral-combine f(1) le
  by (smt atLeastatMost-subset-iff integrable-subinterval-real)
  with le show ?thesis
  by (cases  $g a = g b$ ) (simp-all add: algebra-simps)
next
  assume less:  $\neg g a \leq g b$ 
  then have  $g a \geq g b$  by simp
  note  $le = le$  this
  from cd have integral  $\{c..g b\} f + \text{integral } \{g b..g a\} f = \text{integral } \{c..g a\} f$ 
  using integral-combine f(1) le
  by (smt atLeastatMost-subset-iff integrable-subinterval-real)
  with less show ?thesis
  by (simp-all add: algebra-simps)
qed
finally show ?thesis .
qed

```

lemma has-integral-substitution-general-:

```

fixes  $f :: \text{real} \Rightarrow 'a::\text{euclidean-space}$  and  $g :: \text{real} \Rightarrow \text{real}$ 
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]:
   $\bigwedge x. x \in \{a..b\} - s \implies (g \text{ has-field-derivative } g' x) (\text{at } x \text{ within } \{a..b\})$ 
shows ( $\lambda x. g' x *_R f (g x)$ ) has-integral (integral  $\{g a..g b\} f - \text{integral } \{g b..g a\} f$ )  $\{a..b\}$ 

```

using *s-subset has-integral-substitution-general*-[OF *s le subset f g(1) - deriv*]
by (*simp add: g(2) sup-absorb1*)

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 $\forall x \in s'. \text{finite } (g -' \{x\}) \text{ surj-on } s' g \text{ inj-on } g$*
({a..b} $\cup ((s \cup g -' s'))$)

and *deriv [derivative-intros]:*

$\bigwedge x. x \in \{a..b\} - s \implies (g \text{ has-field-derivative } g' x) \text{ (at } x \text{ within } \{a..b\})$

shows *(($\lambda x. g' x *_{\mathbb{R}} f (g x)$) has-integral (integral {g a..g b} f - integral {g b..g a} f)) {a..b}*

proof -

have *a: g -' s' = $\bigcup \{t. \exists x. t = g -' \{x\} \wedge x \in s'\}$*

using *s s' by blast*

have *finite ($\bigcup \{t. \exists x. t = g -' \{x\} \wedge x \in s'\}$) using s'*

by (*metis (no-types, lifting) $\langle g -' s' = \bigcup \{g -' \{x\} \mid x. x \in s'\} \rangle$ finite-UN-I*
g(2) vimage-eq-UN)

then have *0: finite (s \cup (g -' s'))*

using *a s*

by *simp*

have *1: continuous-on ({c..d} - g ' $(s \cup g -' s')$) f*

using *f(2) surj-on-image-vimage-eq*

by (*metis Diff-mono Un-upper2 continuous-on-subset equalityE g(3) image-Un*)

have *2: ($\bigwedge x. x \in \{a..b\} - (s \cup g -' s') \implies (g \text{ has-real-derivative } g' x) \text{ (at } x$*
within } \{a..b\}))

using *deriv by auto*

show *?thesis using has-integral-substitution-general*-[OF *0 assms(2) subset f(1)*
1 g(1) g(4) 2]

by *auto*

qed

end

theory *Paths*

imports *Derivs General-Utills Integrals*

begin

lemma *reverse-subpaths-join:*

shows *subpath 1 (1 / 2) p +++ subpath (1 / 2) 0 p = reversepath p*

using *reversepath-subpath join-subpaths-middle pathfinish-subpath pathstart-subpath*
reversepath-joinpaths

by (*metis (no-types, lifting)*)

definition *line-integral:: ('a::euclidean-space \Rightarrow 'a::euclidean-space) \Rightarrow (('a) set)*

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

definition line-integral-exists **where**

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

lemma line-integral-on-pair-straight-path:

fixes $F :: ('a :: \text{euclidean-space}) \Rightarrow 'a$ **and** $g :: \text{real} \Rightarrow \text{real}$ **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-integral F $\{i\}$ γ) = 0 (line-integral-exists F $\{i\}$ γ)

proof (simp add: line-integral-def)

have *: $F(\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i) = 0$

if $0 \leq x \wedge x \leq 1$ **for** x

proof -

have (($\lambda x. \gamma(x) \cdot i$) has-vector-derivative 0) (at x)

using vector-derivative-const-at[of a x] **and** gamma-const

by auto

then have (vector-derivative γ (at x) $\cdot i$) = 0

using derivative-component-fun-component[of γ x i]

and gamma-smooth **and** that

by (simp add: vector-derivative-at)

then have (vector-derivative γ (at x within $\{0 .. 1\}$) $\cdot i$) = 0

using has-vector-derivative-at-within vector-derivative-at-within-ivl that

by (metis atLeastAtMost-iff gamma-smooth vector-derivative-works zero-less-one)

then show ?thesis

by auto

qed

then have (($\lambda x. F(\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i)$)
has-integral 0) $\{0..1\}$)

using has-integral-is-0[of $\{0 .. 1\}$ ($\lambda x. F(\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i)$)]

by auto

then have (($\lambda x. F(\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i)$)
integrable-on $\{0..1\}$)

by auto

then show line-integral-exists F $\{i\}$ γ **by** (auto simp add: line-integral-exists-def)

show integral $\{0..1\}$ ($\lambda x. F(\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i)$) = 0

using * has-integral-is-0[of $\{0 .. 1\}$ ($\lambda x. F(\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i)$)]

by auto

qed

lemma line-integral-on-pair-path-strong:

fixes $F :: ('a :: \text{euclidean-space}) \Rightarrow ('a)$ **and**

$g :: \text{real} \Rightarrow 'a$ **and**


```

   $\gamma :: (\text{real} \Rightarrow 'a)$  and
   $i :: 'a$ 
assumes i-norm-1:  $\text{norm } i = 1$  and
  g-orthogonal-to-i:  $\forall x. g(x) \cdot i = 0$  and
  gamma-is-in-terms-of-i:  $\gamma = (\lambda x. f(x) *_{\mathbb{R}} i + g(f(x)))$  and
  gamma-smooth:  $\gamma$  piecewise-C1-differentiable-on  $\{0 .. 1\}$  and
  g-continuous-on-f: continuous-on  $(f \text{ ' } \{0..1\})$   $g$  and
  path-start-le-path-end:  $(\text{pathstart } \gamma) \cdot i \leq (\text{pathfinish } \gamma) \cdot i$  and
  field-i-comp-cont: continuous-on  $(\text{path-image } \gamma)$   $(\lambda x. F x \cdot i)$ 
shows line-integral  $F \{i\} \gamma$ 
  = integral  $(\text{cbox } ((\text{pathstart } \gamma) \cdot i) ((\text{pathfinish } \gamma) \cdot i))$   $(\lambda f\text{-var}. (F (f\text{-var}$ 
 $*_{\mathbb{R}} i + g(f\text{-var})) \cdot i))$ 
  line-integral-exists  $F \{i\} \gamma$ 
proof (simp add: line-integral-def)
  obtain  $s$  where gamma-differentiable:  $\text{finite } s (\forall x \in \{0 .. 1\} - s. \gamma \text{ differentiable at } x)$ 
  using gamma-smooth
  by (auto simp add: C1-differentiable-on-eq piecewise-C1-differentiable-on-def)
  then have gamma-i-component-smooth:  $\forall x \in \{0 .. 1\} - s. (\lambda x. \gamma x \cdot i)$  differentiable at  $x$ 
  by auto
  have field-cont-on-path: continuous-on  $((\lambda x. \gamma x \cdot i) \text{ ' } \{0..1\})$   $(\lambda f\text{-var}. F (f\text{-var}$ 
 $*_{\mathbb{R}} i + g f\text{-var}) \cdot i)$ 
  proof -
  have  $0: (\lambda x. \gamma x \cdot i) = f$ 
  proof
  fix  $x$ 
  show  $\gamma x \cdot i = f x$ 
  using g-orthogonal-to-i i-norm-1
  by (simp only: gamma-is-in-terms-of-i real-inner-class.inner-add-left g-orthogonal-to-i inner-scaleR-left inner-same-Basis norm-eq-1)
  qed
  show ?thesis
  unfolding  $0$ 
  apply (rule continuous-on-compose2 [of -  $(\lambda x. F(x) \cdot i)$   $f \text{ ' } \{0..1\}$   $(\lambda x. x$ 
 $*_{\mathbb{R}} i + g x)$ ])
  field-i-comp-cont g-continuous-on-f field-i-comp-cont continuous-intros)+
  by (auto simp add: gamma-is-in-terms-of-i path-image-def)
  qed
  have path-start-le-path-end':  $\gamma 0 \cdot i \leq \gamma 1 \cdot i$  using path-start-le-path-end by
(auto simp add: pathstart-def pathfinish-def)
  have gamm-cont: continuous-on  $\{0..1\}$   $(\lambda a. \gamma a \cdot i)$ 
  apply(rule continuous-on-inner)
  using gamma-smooth
  apply (simp add: piecewise-C1-differentiable-on-def)
  using continuous-on-const by auto
  then obtain  $c d$  where cd:  $c \leq d (\lambda a. \gamma a \cdot i) \text{ ' } \{0..1\} = \{c..d\}$ 
  by (meson continuous-image-closed-interval zero-le-one)
  then have subset-cd:  $(\lambda a. \gamma a \cdot i) \text{ ' } \{0..1\} \subseteq \{c..d\}$  by auto

```

have *field-cont-on-path-cd*:
continuous-on {c..d} ($\lambda f\text{-var}. F (f\text{-var} *_R i + g f\text{-var}) \cdot i$)
using *field-cont-on-path cd* **by** *auto*
have *path-vector-deriv-line-integrals*:
 $\forall x \in \{0..1\} - s. ((\lambda x. \gamma x \cdot i) \text{ has-vector-derivative}$
 $\text{vector-derivative } (\lambda x. \gamma x \cdot i) \text{ (at } x))$
 $\text{(at } x))$
using *gamma-i-component-smooth* **and** *derivative-component-fun-component*
and
vector-derivative-works
by *blast*
then have $\forall x \in \{0..1\} - s. ((\lambda x. \gamma x \cdot i) \text{ has-vector-derivative}$
 $\text{vector-derivative } (\lambda x. \gamma x \cdot i) \text{ (at } x \text{ within}$
 $\{\{0..1\}\}))$
 $\text{(at } x \text{ within } \{\{0..1\}\}))$
using *has-vector-derivative-at-within* *vector-derivative-at-within-ivl*
by *fastforce*
then have *has-int*: $((\lambda x. \text{vector-derivative } (\lambda x. \gamma x \cdot i) \text{ (at } x \text{ within } \{0..1\}) *_R$
 $(F ((\gamma x \cdot i) *_R i + g (\gamma x \cdot i)) \cdot i)) \text{ has-integral}$
 $\text{integral } \{\gamma 0 \cdot i.. \gamma 1 \cdot i\} (\lambda f\text{-var}. F (f\text{-var} *_R i + g f\text{-var}) \cdot i)) \{0..1\}$
using *has-integral-substitution-strong*[*OF gamma-differentiable(1) rel-simps(44)*
path-start-le-path-end' subset-cd field-cont-on-path-cd gamm-cont,
of ($\lambda x. \text{vector-derivative } (\lambda x. \gamma(x) \cdot i) \text{ (at } x \text{ within } \{\{0..1\}\}))$]
gamma-is-in-terms-of-i
by (*auto simp only: has-field-derivative-iff-has-vector-derivative*)
then have *has-int'*: $((\lambda x. (F(\gamma(x)) \cdot i) * (\text{vector-derivative } (\lambda x. \gamma(x) \cdot i) \text{ (at } x$
 $\text{within } \{\{0..1\}\}))) \text{ has-integral}$
 $\text{integral } \{((\text{pathstart } \gamma) \cdot i)..((\text{pathfinish } \gamma) \cdot i)\} (\lambda f\text{-var}. F (f\text{-var} *_R i +$
 $g f\text{-var}) \cdot i)) \{0..1\}$
using *gamma-is-in-terms-of-i i-norm-1*
apply (*auto simp add: pathstart-def pathfinish-def*)
apply (*simp only: real-inner-class.inner-add-left inner-not-same-Basis g-orthogonal-to-i*
inner-scaleR-left norm-eq-1)
by (*auto simp add: algebra-simps*)
have *substitute*:
 $\text{integral } (\{((\text{pathstart } \gamma) \cdot i)..((\text{pathfinish } \gamma) \cdot i)\}) (\lambda f\text{-var}. (F (f\text{-var} *_R i +$
 $g(f\text{-var}) \cdot i))$
 $= \text{integral } (\{0..1\}) (\lambda x. (F(\gamma(x)) \cdot i) * (\text{vector-derivative } (\lambda x. \gamma(x) \cdot$
 $i) \text{ (at } x \text{ within } \{\{0..1\}\})))$
using *gamma-is-in-terms-of-i integral-unique*[*OF has-int*] *i-norm-1*
apply (*auto simp add: pathstart-def pathfinish-def*)
apply (*simp only: real-inner-class.inner-add-left inner-not-same-Basis g-orthogonal-to-i*
inner-scaleR-left norm-eq-1)
by (*auto simp add: algebra-simps*)
have *comp-in-eq-comp-out*: $\forall x \in \{0..1\} - s.$
 $\text{vector-derivative } (\lambda x. \gamma(x) \cdot i) \text{ (at } x \text{ within } \{0..1\})$
 $= (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\})) \cdot i$
proof
fix *x*:: *real*

assume $ass: x \in \{0..1\} - s$
then have $x\text{-bounds}: x \in \{0..1\}$ **by** *auto*
have γ *differentiable at x* **using** ass *gamma-differentiable* **by** *auto*
then have *dotprod-in-is-out*:

$$\text{vector-derivative } (\lambda x. \gamma(x) \cdot i) \text{ (at } x) = (\text{vector-derivative } \gamma \text{ (at } x)) \cdot i$$
using *derivative-component-fun-component*
by *force*
then have $0: (\text{vector-derivative } \gamma \text{ (at } x)) \cdot i = (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\})) \cdot i$
proof –
have $(\gamma \text{ has-vector-derivative } \text{vector-derivative } \gamma \text{ (at } x)) \text{ (at } x)$
using $\langle \gamma \text{ differentiable at } x \rangle$ *vector-derivative-works* **by** *blast*
moreover have $0 \leq x \wedge x \leq 1$
using $x\text{-bounds}$ **by** *presburger*
ultimately show *?thesis*
by (*simp add: vector-derivative-at-within-ivl*)
qed
have $1: \text{vector-derivative } (\lambda x. \gamma(x) \cdot i) \text{ (at } x) = \text{vector-derivative } (\lambda x. \gamma(x) \cdot i) \text{ (at } x \text{ within } \{0..1\})$
using *path-vector-deriv-line-integrals* **and** *vector-derivative-at-within-ivl* **and** $x\text{-bounds}$
by (*metis ass atLeastAtMost-iff zero-less-one*)
show $\text{vector-derivative } (\lambda x. \gamma x \cdot i) \text{ (at } x \text{ within } \{0..1\}) = \text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i$
using 0 **and** 1 **and** *dotprod-in-is-out*
by *auto*
qed
show $\text{integral } \{0..1\} (\lambda x. F (\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i)) =$

$$\text{integral } \{\text{pathstart } \gamma \cdot i.. \text{pathfinish } \gamma \cdot i\} (\lambda f\text{-var. } F (f\text{-var} *_{\mathbb{R}} i + g f\text{-var}) \cdot i)$$
using *substitute* **and** *comp-in-eq-comp-out* **and** *negligible-finite*
 $\text{Henstock-Kurzweil-Integration.integral-spike}$

$$[\text{of } s \{0..1\} (\lambda x. F (\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i))$$

$$(\lambda x. F (\gamma x) \cdot i * \text{vector-derivative } (\lambda x. \gamma x \cdot i) \text{ (at } x \text{ within } \{0..1\}))]$$
by (*metis (no-types, lifting) gamma-differentiable(1)*)
have $((\lambda x. F (\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i)) \text{ has-integral } \text{integral } \{\text{pathstart } \gamma \cdot i.. \text{pathfinish } \gamma \cdot i\} (\lambda f\text{-var. } F (f\text{-var} *_{\mathbb{R}} i + g f\text{-var}) \cdot i)) \{0..1\}$
using *has-int'* **and** *comp-in-eq-comp-out* **and** *negligible-finite*
 $\text{Henstock-Kurzweil-Integration.has-integral-spike}$

$$[\text{of } s \{0..1\} (\lambda x. F (\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i))$$

$$(\lambda x. F (\gamma x) \cdot i * \text{vector-derivative } (\lambda x. \gamma x \cdot i) \text{ (at } x \text{ within } \{0..1\}))$$

$$\text{integral } \{\text{pathstart } \gamma \cdot i.. \text{pathfinish } \gamma \cdot i\} (\lambda f\text{-var. } F (f\text{-var} *_{\mathbb{R}} i + g f\text{-var}) \cdot i)]$$
by (*metis (no-types, lifting) gamma-differentiable(1)*)
then have $(\lambda x. F (\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i))$ *integrable-on* $\{0..1\}$

using *integrable-on-def* **by** *auto*
then show *line-integral-exists* $F \{i\} \gamma$
by (*auto simp add: line-integral-exists-def*)
qed

lemma *line-integral-on-pair-path:*

fixes $F::('a::\text{euclidean-space}) \Rightarrow ('a)$ **and**
 $g::\text{real} \Rightarrow 'a$ **and**
 $\gamma::(\text{real} \Rightarrow 'a)$ **and**
 $i::'a$
assumes *i-norm-1*: $\text{norm } i = 1$ **and**
g-orthogonal-to-i: $\forall x. g(x) \cdot i = 0$ **and**
gamma-is-in-terms-of-i: $\gamma = (\lambda x. f(x) *_{\mathbb{R}} i + g(f(x)))$ **and**
gamma-smooth: γ *C1-differentiable-on* $\{0 .. 1\}$ **and**
g-continuous-on-f: *continuous-on* $(f \text{ ' } \{0..1\})$ g **and**
path-start-le-path-end: $(\text{pathstart } \gamma) \cdot i \leq (\text{pathfinish } \gamma) \cdot i$ **and**
field-i-comp-cont: *continuous-on* $(\text{path-image } \gamma)$ $(\lambda x. F x \cdot i)$
shows (*line-integral* $F \{i\} \gamma$)
 $= \text{integral } (\text{cbox } ((\text{pathstart } \gamma) \cdot i) ((\text{pathfinish } \gamma) \cdot i)) (\lambda f\text{-var}. (F$
 $(f\text{-var} *_{\mathbb{R}} i + g(f\text{-var})) \cdot i))$
proof (*simp add: line-integral-def*)
have *gamma-differentiable*: $\forall x \in \{0 .. 1\}. \gamma$ *differentiable at* x
using *gamma-smooth C1-differentiable-on-eq* **by** *blast*
then have *gamma-i-component-smooth*:
 $\forall x \in \{0 .. 1\}. (\lambda x. \gamma x \cdot i)$ *differentiable at* x
by *auto*
have *vec-deriv-i-comp-cont*:
continuous-on $\{0..1\}$ $(\lambda x. \text{vector-derivative } (\lambda x. \gamma x \cdot i) (\text{at } x \text{ within } \{0..1\}))$
proof –
have *continuous-on* $\{0..1\}$ $(\lambda x. \text{vector-derivative } (\lambda x. \gamma x) (\text{at } x \text{ within } \{0..1\}))$
using *gamma-smooth C1-differentiable-on-eq*
by (*smt C1-differentiable-on-def atLeastAtMost-iff continuous-on-eq vector-derivative-at-within-ivl*)
then have *deriv-comp-cont*:
 $\text{continuous-on } \{0..1\} (\lambda x. \text{vector-derivative } (\lambda x. \gamma x) (\text{at } x \text{ within } \{0..1\})) \cdot$
 i
by (*simp add: continuous-intros*)
show *?thesis*
using *derivative-component-fun-component-at-within*[*OF gamma-differentiable,*
of i]
 continuous-on-eq [*OF deriv-comp-cont, of* $(\lambda x. \text{vector-derivative } (\lambda x. \gamma x \cdot$
 $i) (\text{at } x \text{ within } \{0..1\}))$]
by *fastforce*
qed
have *field-cont-on-path*:
continuous-on $((\lambda x. \gamma x \cdot i) \text{ ' } \{0..1\}) (\lambda f\text{-var}. F (f\text{-var} *_{\mathbb{R}} i + g f\text{-var}) \cdot i)$
proof –
have $0: (\lambda x. \gamma x \cdot i) = f$
proof
fix x

```

show  $\gamma x \cdot i = f x$ 
  using g-orthogonal-to-i i-norm-1
  by (simp only: gamma-is-in-terms-of-i real-inner-class.inner-add-left g-orthogonal-to-i
inner-scaleR-left inner-same-Basis norm-eq-1)
qed
show ?thesis
  unfolding 0
  apply (rule continuous-on-compose2 [of - ( $\lambda x. F(x) \cdot i$ ) f ' { 0..1} ( $\lambda x. x$ 
*R i + g x)]
    field-i-comp-cont g-continuous-on-f field-i-comp-cont continuous-intros)+
  by (auto simp add: gamma-is-in-terms-of-i path-image-def)
qed
have path-vector-deriv-line-integrals:
   $\forall x \in \{0..1\}. ((\lambda x. \gamma x \cdot i) \text{ has-vector-derivative}$ 
     $\text{vector-derivative } (\lambda x. \gamma x \cdot i) \text{ (at } x))$ 
     $\text{(at } x))$ 
  using gamma-i-component-smooth and derivative-component-fun-component
and
  vector-derivative-works
  by blast
then have  $\forall x \in \{0..1\}. ((\lambda x. \gamma x \cdot i) \text{ has-vector-derivative}$ 
   $\text{vector-derivative } (\lambda x. \gamma x \cdot i) \text{ (at } x \text{ within } \{0..1\}))$ 
   $\text{(at } x \text{ within } \{0..1\}))$ 
  using has-vector-derivative-at-within vector-derivative-at-within-ivl
  by fastforce
then have substitute:
   $\text{integral (cbox } ((\text{pathstart } \gamma) \cdot i) ((\text{pathfinish } \gamma) \cdot i)) (\lambda f\text{-var. } (F (f\text{-var} *_{\mathbb{R}} i +$ 
   $g(f\text{-var})) \cdot i))$ 
   $= \text{integral (cbox } 0 \ 1) (\lambda x. (F(\gamma(x)) \cdot i) * (\text{vector-derivative } (\lambda x. \gamma(x)$ 
   $\cdot i) \text{ (at } x \text{ within } \{0..1\})))$ 
  using gauge-integral-by-substitution
   $[\text{of } 0 \ 1 (\lambda x. (\gamma x) \cdot i)$ 
   $(\lambda x. \text{vector-derivative } (\lambda x. \gamma(x) \cdot i) \text{ (at } x \text{ within } \{0..1\}))$ 
   $(\lambda f\text{-var. } (F (f\text{-var} *_{\mathbb{R}} i + g(f\text{-var})) \cdot i))]$  and
  path-start-le-path-end and vec-deriv-i-comp-cont and field-cont-on-path and
  gamma-is-in-terms-of-i i-norm-1
  apply (auto simp add: pathstart-def pathfinish-def)
apply (simp only: real-inner-class.inner-add-left inner-not-same-Basis g-orthogonal-to-i
inner-scaleR-left norm-eq-1)
  by (auto)

have comp-in-eq-comp-out:  $\forall x \in \{0..1\}.$ 
   $(\text{vector-derivative } (\lambda x. \gamma(x) \cdot i) \text{ (at } x \text{ within } \{0..1\}))$ 
   $= (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\})) \cdot i$ 
proof
  fix x:: real
  assume x-bounds:  $x \in \{0..1\}$ 
  then have  $\gamma$  differentiable at x using gamma-differentiable by auto
  then have dotprod-in-is-out:

```

$\text{vector-derivative } (\lambda x. \gamma(x) \cdot i) \text{ (at } x)$
 $= (\text{vector-derivative } \gamma \text{ (at } x)) \cdot i$
using *derivative-component-fun-component*
by force
then have $0: (\text{vector-derivative } \gamma \text{ (at } x)) \cdot i$
 $= (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\})) \cdot i$
using *has-vector-derivative-at-within and x-bounds and vector-derivative-at-within-ivl*
by (*smt atLeastAtMost-iff gamma-differentiable inner-commute vector-derivative-works*)
have $1: \text{vector-derivative } (\lambda x. \gamma(x) \cdot i) \text{ (at } x) = \text{vector-derivative } (\lambda x. \gamma(x) \cdot$
 $i) \text{ (at } x \text{ within } \{0..1\})$
using *path-vector-deriv-line-integrals and vector-derivative-at-within-ivl and*
x-bounds
by fastforce
show $\text{vector-derivative } (\lambda x. \gamma x \cdot i) \text{ (at } x \text{ within } \{0..1\}) = \text{vector-derivative } \gamma$
 $\text{ (at } x \text{ within } \{0..1\}) \cdot i$
using 0 **and** 1 **and** *dotprod-in-is-out*
by auto
qed
show $\text{integral } \{0..1\} (\lambda x. F (\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\})$
 $\cdot i)) =$
 $\text{integral } \{\text{pathstart } \gamma \cdot i.. \text{pathfinish } \gamma \cdot i\} (\lambda f\text{-var}. F (f\text{-var} *_{\mathbb{R}} i +$
 $g f\text{-var}) \cdot i)$
using *substitute and comp-in-eq-comp-out and*
Henstock-Kurzweil-Integration.integral-cong
 $[\text{of } \{0..1\} (\lambda x. F (\gamma x) \cdot i * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot i))$
 $(\lambda x. F (\gamma x) \cdot i * \text{vector-derivative } (\lambda x. \gamma x \cdot i) \text{ (at } x \text{ within } \{0..1\}))]$
by auto
qed

lemma *content-box-cases*:

$\text{content } (\text{box } a \ b) = (\text{if } \forall i \in \text{Basis}. a \cdot i \leq b \cdot i \text{ then } \text{prod } (\lambda i. b \cdot i - a \cdot i) \ \text{Basis} \text{ else } 0)$
by (*simp add: measure-lborel-box-eq inner-diff*)

lemma *content-box-cbox*:

shows $\text{content } (\text{box } a \ b) = \text{content } (\text{cbox } a \ b)$
by (*simp add: content-box-cases content-cbox-cases*)

lemma *content-eq-0*: $\text{content } (\text{box } a \ b) = 0 \iff (\exists i \in \text{Basis}. b \cdot i \leq a \cdot i)$

by (*auto simp: content-box-cases not-le intro: less-imp-le antisym eq-refl*)

lemma *content-pos-lt-eq*: $0 < \text{content } (\text{cbox } a \ (b::'a::\text{euclidean-space})) \iff (\forall i \in \text{Basis}. a \cdot i < b \cdot i)$

by (*auto simp add: content-cbox-cases less-le prod-nonneg*)

lemma *content-lt-nz*: $0 < \text{content } (\text{box } a \ b) \iff \text{content } (\text{box } a \ b) \neq 0$

using *Paths.content-eq-0 zero-less-measure-iff* **by blast**

lemma *content-subset*: $\text{cbox } a \ b \subseteq \text{box } c \ d \implies \text{content } (\text{cbox } a \ b) \leq \text{content } (\text{box } c \ d)$

c d)
unfolding *measure-def*
by (*intro enn2real-mono emeasure-mono*) (*auto simp: emeasure-lborel-cbox-eq emeasure-lborel-box-eq*)

lemma *sum-content-null*:

assumes *content (box a b) = 0*
and *p tagged-division-of (box a b)*
shows *sum (λ(x,k). content k *_R f x) p = (0::'a::real-normed-vector)*
proof (*rule sum.neutral, rule*)
fix *y*
assume *y: y ∈ p*
obtain *x k where xk: y = (x, k)*
using *surj-pair[of y] by blast*
note *assm = tagged-division-ofD(3-4)[OF assms(2) y[unfolding xk]]*
from *this(2) obtain c d where k: k = cbox c d by blast*
have *(λ(x, k). content k *_R f x) y = content k *_R f x*
unfolding *xk by auto*
also have *... = 0*
using *content-subset[OF assm(1)[unfolding k]] content-pos-le[of cbox c d]*
unfolding *assms(1) k*
by *auto*
finally show *(λ(x, k). content k *_R f x) y = 0 .*
qed

lemma *has-integral-null* [*intro*]: *content(box a b) = 0 ⇒ (f has-integral 0) (box a b)*
by (*simp add: content-box-cbox content-eq-0-interior*)

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 –
obtain *s1 s2*
where *s1: finite s1 ∀ x∈{0..1} – s1. g1 differentiable at x*
and *s2: finite s2 ∀ x∈{0..1} – s2. g2 differentiable at x*
using *assms*
by (*auto simp: valid-path-def piecewise-C1-differentiable-on-def C1-differentiable-on-eq*)
obtain *i1 i2*
where *1: ((λx. ∑ b∈basis. f (g1 x) · b * (vector-derivative g1 (at x within {0..1}) · b)) has-integral i1) {0..1}*
and *2: ((λx. ∑ b∈basis. f (g2 x) · b * (vector-derivative g2 (at x within {0..1}) · b)) has-integral i2) {0..1}*
using *assms*

```

  by (auto simp: line-integral-exists-def)
  have i1: (( $\lambda x. 2 * (\sum_{b \in \text{basis}} f (g1 (2 * x)) \cdot b * (\text{vector-derivative } g1 \text{ (at } (2 * x) \text{ within } \{0..1\}) \cdot b)))$ ) has-integral i1) {0..1/2}
  and i2: (( $\lambda x. 2 * (\sum_{b \in \text{basis}} f (g2 (2 * x - 1)) \cdot b * (\text{vector-derivative } g2 \text{ (at } ((2 * x) - 1) \text{ within } \{0..1\}) \cdot b)))$ ) has-integral i2) {1/2..1}
  using has-integral-affinity01 [OF 1, where m= 2 and c=0, THEN has-integral-cmul
  [where c=2]]
  has-integral-affinity01 [OF 2, where m= 2 and c=-1, THEN has-integral-cmul
  [where c=2]]
  by (simp-all only: image-affinity-atLeastAtMost-div-diff, simp-all add: scaleR-conv-of-real
  mult-ac)
  have g1: [ $0 \leq z; z * 2 < 1; z * 2 \notin s1$ ]  $\implies$ 
    vector-derivative ( $\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1)$ ) (at
  z within {0..1}) =
    2 *R vector-derivative g1 (at (z*2) within {0..1}) for z
  proof -
    have i: [ $0 \leq z; z * 2 < 1; z * 2 \notin s1$ ]  $\implies$ 
      vector-derivative ( $\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x -$ 
    1)) (at z within {0..1}) = 2 *R vector-derivative g1 (at (z * 2)) for z
    proof -
      have g1-at-z: [ $0 \leq z; z * 2 < 1; z * 2 \notin s1$ ]  $\implies$ 
        (( $\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1)$ )
      has-vector-derivative
        2 *R vector-derivative g1 (at (z*2))) (at z) for z
      apply (rule has-vector-derivative-transform-at [of |z - 1/2| - ( $\lambda x. g1(2 * x)$ )])
      apply (simp-all add: dist-real-def abs-if split: if-split-asm)
      apply (rule vector-diff-chain-at [of  $\lambda x. 2 * x - 2 - g1$ , simplified o-def])
      apply (simp add: has-vector-derivative-def has-derivative-def bounded-linear-mult-left)
      using s1
      apply (auto simp: algebra-simps vector-derivative-works)
      done
      assume ass:  $0 \leq z \wedge z * 2 < 1 \wedge z * 2 \notin s1$ 
      then have z-ge:  $z \leq 1$  by auto
      show vector-derivative ( $\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x -$ 
    1)) (at z within {0..1}) = 2 *R vector-derivative g1 (at (z * 2))
      using Derivative.vector-derivative-at-within-ivl[OF g1-at-z[OF ass] ass(1)
    z-ge]
      by auto
    qed
    assume ass:  $0 \leq z \wedge z * 2 < 1 \wedge z * 2 \notin s1$ 
    then have (g1 has-vector-derivative ((vector-derivative g1 (at (z*2)))) (at
    (z*2)))
      using s1 by (auto simp: algebra-simps vector-derivative-works)
    then have ii: (vector-derivative g1 (at (z*2) within {0..1})) = (vector-derivative
    g1 (at (z*2)))
      using Derivative.vector-derivative-at-within-ivl ass
      by force
    show vector-derivative ( $\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1)$ )
    (at z within {0..1}) = 2 *R vector-derivative g1 (at (z * 2) within {0..1})

```



```

    using i[OF ass] ii
    by auto
  qed
  have g2:  $\llbracket 1 < z * 2; z \leq 1; z * 2 - 1 \notin s2 \rrbracket \implies$ 
    vector-derivative  $(\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1))$  (at
  z within  $\{0..1\}$ ) =
    2 *R vector-derivative g2 (at  $(z * 2 - 1)$  within  $\{0..1\}$ ) for z      proof
  -
    have i: $\llbracket 1 < z * 2; z \leq 1; z * 2 - 1 \notin s2 \rrbracket \implies$ 
      vector-derivative  $(\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x -$ 
    1)) (at z within  $\{0..1\}$ )
      = 2 *R vector-derivative g2 (at  $(z * 2 - 1)$ ) for z
    proof-
      have g2-at-z: $\llbracket 1 < z * 2; z \leq 1; z * 2 - 1 \notin s2 \rrbracket \implies$ 
         $((\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1))$ 
      has-vector-derivative 2 *R vector-derivative g2 (at  $(z * 2 - 1)$ )) (at z) for z
        apply (rule has-vector-derivative-transform-at [of  $|z - 1/2| - (\lambda x. g2 (2 * x$ 
      - 1))])
        apply (simp-all add: dist-real-def abs-if split: if-split-asm)
        apply (rule vector-diff-chain-at [of  $\lambda x. 2 * x - 1$  2 - g2, simplified o-def])
        apply (simp add: has-vector-derivative-def has-derivative-def bounded-linear-mult-left)
        using s2
        apply (auto simp: algebra-simps vector-derivative-works)
        done
      assume ass:  $1 < z * 2$   $z \leq 1$   $z * 2 - 1 \notin s2$ 
      then have z-le:  $z \leq 1$  by auto
      have z-ge:  $0 \leq z$  using ass by auto
      show vector-derivative  $(\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x -$ 
    1)) (at z within  $\{0..1\}$ )
        = 2 *R vector-derivative g2 (at  $(z * 2 - 1)$ )
        using Derivative.vector-derivative-at-within-ivl[OF g2-at-z[OF ass] z-ge z-le]
        by auto
    qed
    assume ass:  $1 < z * 2$   $z \leq 1$   $z * 2 - 1 \notin s2$ 
    then have (g2 has-vector-derivative ((vector-derivative g2 (at  $(z * 2 - 1)$ ))))
  (at  $(z * 2 - 1)$ )
      using s2 by (auto simp: algebra-simps vector-derivative-works)
    then have ii: (vector-derivative g2 (at  $(z * 2 - 1)$  within  $\{0..1\}$ )) = (vector-derivative
  g2 (at  $(z * 2 - 1)$ ))
      using Derivative.vector-derivative-at-within-ivl ass
      by force
    show vector-derivative  $(\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1))$ 
  (at z within  $\{0..1\}$ ) = 2 *R vector-derivative g2 (at  $(z * 2 - 1)$  within  $\{0..1\}$ )
      using i[OF ass] ii
      by auto
    qed
  have lem1:  $((\lambda x. \sum_{b \in \text{basis}} f ((g1 +++ g2) x) \cdot b * (\text{vector-derivative } (g1 +++ g2)$ 
  (at z within  $\{0..1\}$ ) \cdot b)) \text{has-integral } i1  $\{0..1/2\}$ 
    apply (rule has-integral-spike-finite [OF - - i1, of insert  $(1/2)$   $((*)2 - 's1)$ ])

```

```

using s1
  apply (force intro: finite-vimageI [where h = (*)2] inj-onI)
  apply (clarsimp simp add: joinpaths-def scaleR-conv-of-real mult-ac g1)
  by (simp add: sum-distrib-left)
  moreover have lem2: (( $\lambda x. \sum b \in \text{basis}. f ((g1+++g2) x) \cdot b * (\text{vector-derivative } (g1+++g2) (\text{at } x \text{ within } \{0..1\}) \cdot b)) \text{ has-integral } i2) \{1/2..1\}$ )
  apply (rule has-integral-spike-finite [OF - - i2, of insert (1/2) (( $\lambda x. 2*x-1$ ) - 's2)])
  using s2
  apply (force intro: finite-vimageI [where h =  $\lambda x. 2*x-1$ ] inj-onI)
  apply (clarsimp simp add: joinpaths-def scaleR-conv-of-real mult-ac g2)
  by (simp add: sum-distrib-left)
ultimately
show line-integral f basis (g1 +++ g2) = line-integral f basis g1 + line-integral f basis g2
  apply (simp add: line-integral-def)
  apply (rule integral-unique [OF has-integral-combine [where c = 1/2]])
  using 1 2 integral-unique apply auto
done
show line-integral-exists f basis (g1 +++ g2)
proof (simp add: line-integral-exists-def integrable-on-def)
  have (1::real)  $\leq 1 * 2 \wedge (0::real) \leq 1 / 2$ 
  by simp
  then show  $\exists r. ((\lambda r. \sum a \in \text{basis}. f ((g1+++g2) r) \cdot a * (\text{vector-derivative } (g1+++g2) (\text{at } r \text{ within } \{0..1\}) \cdot a)) \text{ has-integral } r) \{0..1\}$ 
  using has-integral-combine [where c = 1/2] 1 2 divide-le-eq-numeral1(1) lem1
  lem2 by blast
qed
qed

```

```

lemma line-integral-exists-joinD1:
  assumes line-integral-exists f basis (g1 +++ g2) valid-path g1
  shows line-integral-exists f basis g1
proof -
  obtain s1
  where s1: finite s1  $\forall x \in \{0..1\} - s1. g1$  differentiable at x
  using assms by (auto simp: valid-path-def piecewise-C1-differentiable-on-def C1-differentiable-on-eq)
  have ( $\lambda x. \sum b \in \text{basis}. f ((g1+++g2) (x/2)) \cdot b * (\text{vector-derivative } (g1+++g2) (\text{at } (x/2) \text{ within } \{0..1\}) \cdot b)) \text{ integrable-on } \{0..1\}$ 
  using assms
  apply (auto simp: line-integral-exists-def)
  apply (drule integrable-on-subbox [where a=0 and b=1/2])
  apply (auto intro: integrable-affinity [of - 0 1/2::real 1/2 0, simplified])
  done
  then have *: ( $\lambda x. \sum b \in \text{basis}. (f ((g1+++g2) (x/2)) \cdot b) / 2$ ) * (vector-derivative (g1+++g2) (at (x/2) within {0..1}) \cdot b) integrable-on {0..1}
  by (auto simp: Groups-Big.sum-distrib-left dest: integrable-cmul [where c=1/2] simp: scaleR-conv-of-real)

```

```

have g1:  $\llbracket 0 \leq z; z * 2 < 1; z * 2 \notin s1 \rrbracket \implies$ 
  vector-derivative  $(\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1))$  (at
z within  $\{0..1\}$ ) =
  2 *R vector-derivative g1 (at (z * 2) within  $\{0..1\}$ ) for z
proof -
  have i:  $\llbracket 0 \leq z; z * 2 < 1; z * 2 \notin s1 \rrbracket \implies$ 
    vector-derivative  $(\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x -$ 
1)) (at z within  $\{0..1\}$ ) = 2 *R vector-derivative g1 (at (z * 2)) for z
  proof -
    have g1-at-z:  $\llbracket 0 \leq z; z * 2 < 1; z * 2 \notin s1 \rrbracket \implies$ 
       $((\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1))$ 
has-vector-derivative
      2 *R vector-derivative g1 (at (z * 2))) (at z) for z
    apply (rule has-vector-derivative-transform-at [of  $|z - 1/2| - (\lambda x. g1(2 * x))$ ])
      apply (simp-all add: dist-real-def abs-if split: if-split-asm)
      apply (rule vector-diff-chain-at [of  $\lambda x. 2 * x - g1$ , simplified o-def])
      apply (simp add: has-vector-derivative-def has-derivative-def bounded-linear-mult-left)
      using s1
      apply (auto simp: algebra-simps vector-derivative-works)
      done
    assume ass:  $0 \leq z \wedge z * 2 < 1 \wedge z * 2 \notin s1$ 
    then have z-ge:  $z \leq 1$  by auto
    show vector-derivative  $(\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x -$ 
1)) (at z within  $\{0..1\}$ ) = 2 *R vector-derivative g1 (at (z * 2))
      using Derivative.vector-derivative-at-within-ivl[OF g1-at-z[OF ass] ass(1)
z-ge]
      by auto
    qed
    assume ass:  $0 \leq z \wedge z * 2 < 1 \wedge z * 2 \notin s1$ 
    then have (g1 has-vector-derivative ((vector-derivative g1 (at (z * 2)))) (at
(z * 2)))
      using s1 by (auto simp: algebra-simps vector-derivative-works)
    then have ii: (vector-derivative g1 (at (z * 2) within  $\{0..1\}$ )) = (vector-derivative
g1 (at (z * 2)))
      using Derivative.vector-derivative-at-within-ivl ass by force
    show vector-derivative  $(\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1))$ 
(at z within  $\{0..1\}$ ) = 2 *R vector-derivative g1 (at (z * 2) within  $\{0..1\}$ )
      using i[OF ass] ii by auto
    qed
show ?thesis
  using s1
  apply (auto simp: line-integral-exists-def)
  apply (rule integrable-spike-finite [of  $\{0,1\} \cup s1$ , OF - - *])
  apply (auto simp: joinpaths-def scaleR-conv-of-real g1)
  done
qed

```

lemma line-integral-exists-joinD2:

assumes line-integral-exists f basis (g1 +++ g2) valid-path g2

shows *line-integral-exists f basis g2*

proof –

obtain *s2*

where *s2*: *finite s2* $\forall x \in \{0..1\}$ – *s2.g2 differentiable at x*

using *assms* **by** (*auto simp: valid-path-def piecewise-C1-differentiable-on-def C1-differentiable-on-eq*)

have $(\lambda x. \sum b \in \text{basis}. f ((g1 \text{ +++ } g2) (x/2 + 1/2)) \cdot b * (\text{vector-derivative } (g1 \text{ +++ } g2) (\text{at } (x/2 + 1/2) \text{ within } \{0..1\}) \cdot b)) \text{ integrable-on } \{0..1\}$

using *assms*

apply (*auto simp: line-integral-exists-def*)

apply (*drule integrable-on-subcbox [where a=1/2 and b=1], auto*)

apply (*drule integrable-affinity [of - 1/2::real 1 1/2 1/2, simplified]*)

apply (*simp add: image-affinity-atLeastAtMost-diff*)

done

then **have** $*(\lambda x. \sum b \in \text{basis}. ((f ((g1 \text{ +++ } g2) (x/2 + 1/2)) \cdot b) / 2) * (\text{vector-derivative } (g1 \text{ +++ } g2) (\text{at } (x/2 + 1/2) \text{ within } \{0..1\}) \cdot b)) \text{ integrable-on } \{0..1\}$

by (*auto simp: Groups-Big.sum-distrib-left dest: integrable-cmul [where c=1/2] simp: scaleR-conv-of-real*)

have *g2*: $\llbracket 1 < z * 2; z \leq 1; z * 2 - 1 \notin s2 \rrbracket \implies$
 $\text{vector-derivative } (\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1)) (\text{at } z \text{ within } \{0..1\}) =$
 $2 *_{\mathbb{R}} \text{vector-derivative } g2 (\text{at } (z * 2 - 1) \text{ within } \{0..1\})$ **for** *z* **proof**

–

have *i*: $\llbracket 1 < z * 2; z \leq 1; z * 2 - 1 \notin s2 \rrbracket \implies$
 $\text{vector-derivative } (\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1)) (\text{at } z \text{ within } \{0..1\}) =$
 $2 *_{\mathbb{R}} \text{vector-derivative } g2 (\text{at } (z * 2 - 1) \text{ within } \{0..1\})$ **for** *z*

proof–

have *g2-at-z*: $\llbracket 1 < z * 2; z \leq 1; z * 2 - 1 \notin s2 \rrbracket \implies$
 $((\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1))) (\text{at } z)$ **for** *z*

has-vector-derivative $2 *_{\mathbb{R}} \text{vector-derivative } g2 (\text{at } (z * 2 - 1) \text{ within } \{0..1\})$ **for** *z*

apply (*rule has-vector-derivative-transform-at [of |z - 1/2| - (\lambda x. g2 (2 * x - 1))]*)

apply (*simp-all add: dist-real-def abs-if split: if-split-asm*)

apply (*rule vector-diff-chain-at [of \lambda x. 2 * x - 1 2 - g2, simplified o-def]*)

apply (*simp add: has-vector-derivative-def has-derivative-def bounded-linear-mult-left*)

using *s2*

apply (*auto simp: algebra-simps vector-derivative-works*)

done

assume *ass*: $1 < z * 2$ $z \leq 1$ $z * 2 - 1 \notin s2$

then **have** *z-le*: $z \leq 1$ **by** *auto*

have *z-ge*: $0 \leq z$ **using** *ass* **by** *auto*

show *vector-derivative* $(\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1)) (\text{at } z \text{ within } \{0..1\}) =$
 $2 *_{\mathbb{R}} \text{vector-derivative } g2 (\text{at } (z * 2 - 1) \text{ within } \{0..1\})$

using *Derivative.vector-derivative-at-within-ivl [OF g2-at-z [OF ass] z-ge z-le]*

by *auto*

qed

```

    assume ass:  $1 < z * 2 \leq 1$   $z * 2 - 1 \notin s2$ 
    then have (g2 has-vector-derivative ((vector-derivative g2 (at (z*2 - 1))))
(at (z*2 - 1))
    using s2 by (auto simp: algebra-simps vector-derivative-works)
    then have ii: (vector-derivative g2 (at (z*2 - 1) within {0..1})) = (vector-derivative
g2 (at (z*2 - 1)))
    using Derivative.vector-derivative-at-within-ivl ass
    by force
    show vector-derivative ( $\lambda x. \text{if } x * 2 \leq 1 \text{ then } g1 (2 * x) \text{ else } g2 (2 * x - 1)$ )
(at z within {0..1}) =  $2 *_{\mathbb{R}}$  vector-derivative g2 (at (z * 2 - 1) within {0..1})
    using i[OF ass] ii
    by auto
  qed
  show ?thesis
  using s2
  apply (auto simp: line-integral-exists-def)
  apply (rule integrable-spike-finite [of {0,1}  $\cup$  s2, OF - - *])
  apply (auto simp: joinpaths-def scaleR-conv-of-real g2)
  done
qed

```

lemma *has-line-integral-on-reverse-path*:

```

  assumes g: valid-path g and int:
    (( $\lambda x. \sum_{b \in \text{basis}. F} (g \ x) \cdot b * (\text{vector-derivative } g \text{ (at } x \text{ within } \{0..1\}) \cdot b)$ )
has-integral c){0..1}
  shows (( $\lambda x. \sum_{b \in \text{basis}. F} ((\text{reversepath } g) \ x) \cdot b * (\text{vector-derivative } (\text{reversepath }
g) \text{ (at } x \text{ within } \{0..1\}) \cdot b)$ ) has-integral  $-c$ ){0..1}
  proof -
    { fix s x
      assume xs: g C1-differentiable-on ( $\{0..1\} - s$ )  $x \notin (-) 1$   $s \leq x \leq 1$ 
      have vector-derivative ( $\lambda x. g (1 - x)$ ) (at x within {0..1}) =
        - vector-derivative g (at (1 - x) within {0..1})
      proof -
        obtain f' where f': (g has-vector-derivative f') (at (1 - x))
        using xs
        by (force simp: has-vector-derivative-def C1-differentiable-on-def)
        have (g o ( $\lambda x. 1 - x$ )) has-vector-derivative  $-1 *_{\mathbb{R}}$  f' (at x)
        apply (rule vector-diff-chain-within)
        apply (intro vector-diff-chain-within derivative-eq-intros | simp)+
        apply (rule has-vector-derivative-at-within [OF f'])
        done
        then have mf': (( $\lambda x. g (1 - x)$ ) has-vector-derivative  $-f'$ ) (at x)
        by (simp add: o-def)
        show ?thesis
        using xs
        by (auto simp: vector-derivative-at-within-ivl [OF mf'] vector-derivative-at-within-ivl
[OF f'])
      qed
    }
  } note * = this

```

obtain S **where** *continuous-on* $\{0..1\}$ g *finite* S g *C1-differentiable-on* $\{0..1\}$
 – S
using g
by (*auto simp: valid-path-def piecewise-C1-differentiable-on-def*)
then show *?thesis*
using *has-integral-affinity01* [*OF int, where* $m = -1$ **and** $c = 1$]
unfolding *reversepath-def*
by (*rule-tac* $S = (\lambda x. 1 - x)$ ‘ S **in** *has-integral-spike-finite*) (*auto simp: **
has-integral-neg Groups-Big.sum-negf)
qed

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 –
obtain i **where**
 $0: ((\lambda x. \sum b \in \text{basis}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot b)) \text{ has-integral } i) \{0..1\}$
using *assms unfolding integrable-on-def line-integral-exists-def* **by** *auto*
then have $1: ((\lambda x. \sum b \in \text{basis}. F ((\text{reversepath } \gamma) x) \cdot b * (\text{vector-derivative } (\text{reversepath } \gamma) \text{ (at } x \text{ within } \{0..1\}) \cdot b)) \text{ has-integral } -i) \{0..1\}$
using *has-line-integral-on-reverse-path assms*
by *auto*
then have *rev-line-integral:line-integral* F *basis* (*reversepath* γ) = $-i$
using *line-integral-def Henstock-Kurzweil-Integration.integral-unique*
by (*metis (no-types)*)
have *line-integral: line-integral* F *basis* $\gamma = i$
using *line-integral-def 0 Henstock-Kurzweil-Integration.integral-unique*
by *blast*
show *line-integral* F *basis* $\gamma = -$ (*line-integral* F *basis* (*reversepath* γ))
using *line-integral rev-line-integral*
by *auto*
show *line-integral-exists* F *basis* (*reversepath* γ)
using 1 *line-integral-exists-def*
by *auto*
qed

lemma *line-integral-exists-on-degenerate-path:*

assumes *finite basis*
shows *line-integral-exists* F *basis* $(\lambda x. c)$
proof –
have *every-component-integrable:*
 $\forall b \in \text{basis}. (\lambda x. F ((\lambda x. c) x) \cdot b * (\text{vector-derivative } (\lambda x. c) \text{ (at } x \text{ within } \{0..1\}) \cdot b)) \text{ integrable-on } \{0..1\}$
proof
fix b
assume *b-in-basis: b* \in *basis*
have *cont-field-zero-one: continuous-on* $\{0..1\}$ $(\lambda x. F ((\lambda x. c) x) \cdot b)$

using *continuous-on-const* **by** *fastforce*
have *cont-path-zero-one*:
continuous-on {0..1} (λx. (vector-derivative (λx. c) (at x within {0..1}))) · b
proof –
have $((\text{vector-derivative } (\lambda x. c) \text{ (at } x \text{ within } \{0..1\})) \cdot b) = 0 \text{ if } x \in \{0..1\}$
for x
proof –
have $\text{vector-derivative } (\lambda x. c) \text{ (at } x \text{ within } \{0..1\}) = 0$
using *that gamma-deriv-at-within[of 0 1] differentiable-const vector-derivative-const-at*
by *fastforce*
then show $\text{vector-derivative } (\lambda x. c) \text{ (at } x \text{ within } \{0..1\}) \cdot b = 0$
by *auto*
qed
then show *continuous-on {0..1} (λx. (vector-derivative (λx. c) (at x within {0..1}))) · b*
using *continuous-on-const[of {0..1} 0] continuous-on-eq[of {0..1} λx. 0*
(λx. (vector-derivative (λx. c) (at x within {0..1}))) · b]
by *auto*
qed
show $(\lambda x. F(c) \cdot b * (\text{vector-derivative } (\lambda x. c) \text{ (at } x \text{ within } \{0..1\}) \cdot b))$
integrable-on {0..1}
using *cont-field-zero-one cont-path-zero-one continuous-on-mult integrable-continuous-real*
by *blast*
qed
have *integrable-sum'*: $\bigwedge t f s. \text{finite } t \implies$
 $\forall a \in t. f a \text{ integrable-on } s \implies (\lambda x. \sum a \in t. f a x) \text{ integrable-on } s$
using *integrable-sum* **by** *metis*
have *field-integrable-on-basis*:
 $(\lambda x. \sum b \in \text{basis}. F(c) \cdot b * (\text{vector-derivative } (\lambda x. c) \text{ (at } x \text{ within } \{0..1\}) \cdot b))$
integrable-on {0..1}
using *integrable-sum'[OF assms(1) every-component-integrable]*
by *auto*
then show *?thesis*
using *line-integral-exists-def* **by** *auto*
qed

lemma *degenerate-path-is-valid-path*: *valid-path (λx. c)*
by *(auto simp add: valid-path-def piecewise-C1-differentiable-on-def continuous-on-const)*

lemma *line-integral-degenerate-path*:
assumes *finite basis*
shows *line-integral F basis (λx. c) = 0*
proof *(simp add: line-integral-def)*
have $((\text{vector-derivative } (\lambda x. c) \text{ (at } x \text{ within } \{0..1\})) \cdot b) = 0 \text{ if } x \in \{0..1\}$ **for**
 $x b$
proof –
have $\text{vector-derivative } (\lambda x. c) \text{ (at } x \text{ within } \{0..1\}) = 0$
using *that gamma-deriv-at-within[of 0 1] differentiable-const vector-derivative-const-at*
by *fastforce*

then show $\text{vector-derivative } (\lambda x. c) \text{ (at } x \text{ within } \{0..1\}) \cdot b = 0$
by auto
qed
then have $0: \bigwedge x. x \in \{0..1\} \implies (\lambda x. \sum b \in \text{basis}. F c \cdot b * (\text{vector-derivative } (\lambda x. c) \text{ (at } x \text{ within } \{0..1\}) \cdot b)) x = (\lambda x. 0) x$
by auto
then show $\text{integral } \{0..1\} (\lambda x. \sum b \in \text{basis}. F c \cdot b * (\text{vector-derivative } (\lambda x. c) \text{ (at } x \text{ within } \{0..1\}) \cdot b)) = 0$
using $\text{integral-cong[of } \{0..1\}, OF 0]$ **integral-0 by auto**
qed

definition point-path where
 $\text{point-path } \gamma \equiv \exists c. \gamma = (\lambda x. c)$

lemma line-integral-point-path:
assumes $\text{point-path } \gamma$
assumes finite basis
shows $\text{line-integral } F \text{ basis } \gamma = 0$
using $\text{assms(1) point-path-def line-integral-degenerate-path[OF assms(2)]}$
by force

lemma line-integral-exists-point-path:
assumes $\text{finite basis point-path } \gamma$
shows $\text{line-integral-exists } F \text{ basis } \gamma$
using assms
apply $(\text{simp add: point-path-def})$
using $\text{line-integral-exists-on-degenerate-path by auto}$

lemma line-integral-exists-subpath:
assumes $f: \text{line-integral-exists } f \text{ basis } g$ **and** $g: \text{valid-path } g$
and $uv: u \in \{0..1\} v \in \{0..1\} u \leq v$
shows $(\text{line-integral-exists } f \text{ basis } (\text{subpath } u v g))$
proof $(\text{cases } v=u)$
case $tr: \text{True}$
have $\text{zero: } (\sum b \in \text{basis}. f (g u) \cdot b * (\text{vector-derivative } (\lambda x. g u) \text{ (at } x \text{ within } \{0..1\}) \cdot b)) = 0$ **if** $x \in \{0..1\}$ **for** $x::\text{real}$
proof $-$
have $(\text{vector-derivative } (\lambda x. g u) \text{ (at } x \text{ within } \{0..1\})) = 0$
using $\text{Deriv.has-vector-derivative-const that Derivative.vector-derivative-at-within-ivl}$
by fastforce
then show $(\sum b \in \text{basis}. f (g u) \cdot b * (\text{vector-derivative } (\lambda x. g u) \text{ (at } x \text{ within } \{0..1\}) \cdot b)) = 0$
by auto
qed
then have $((\lambda x. \sum b \in \text{basis}. f (g u) \cdot b * (\text{vector-derivative } (\lambda x. g u) \text{ (at } x \text{ within } \{0..1\}) \cdot b)) \text{ has-integral } 0) \{0..1\}$
by $(\text{meson has-integral-is-0})$
then show $?thesis$
using $f \text{ tr by (auto simp add: line-integral-def line-integral-exists-def subpath-def)}$

next
case *False*
obtain *s* **where** $s: \bigwedge x. x \in \{0..1\} - s \implies g$ *differentiable at x* **and** *fs: finite s*
using *g unfolding piecewise-C1-differentiable-on-def C1-differentiable-on-eq valid-path-def* **by** *blast*
have $*$: $((\lambda x. \sum_{b \in \text{basis}} f (g ((v - u) * x + u)) \cdot b * (\text{vector-derivative } g \text{ (at } ((v - u) * x + u) \text{ within } \{0..1\}) \cdot b))$
 $\text{has-integral } (1 / (v - u)) * \text{integral } \{u..v\} (\lambda x. \sum_{b \in \text{basis}} f (g (x)) \cdot b * (\text{vector-derivative } g \text{ (at } x \text{ within } \{0..1\}) \cdot b)))$
 $\{0..1\}$
using *f uv*
apply (*simp add: line-integral-exists-def subpath-def*)
apply (*drule integrable-on-subcbox [where a=u and b=v, simplified]*)
apply (*simp-all add: has-integral-integral*)
apply (*drule has-integral-affinity [where m=v-u and c=u, simplified]*)
apply (*simp-all add: False image-affinity-atLeastAtMost-div-diff scaleR-conv-of-real*)
apply (*simp add: divide-simps False*)
done
have $vd: \bigwedge x. x \in \{0..1\} \implies$
 $x \notin (\lambda t. (v - u) *_R t + u) - 's \implies$
 $\text{vector-derivative } (\lambda x. g ((v - u) * x + u)) \text{ (at } x \text{ within } \{0..1\}) = (v - u)$
 $*_R \text{vector-derivative } g \text{ (at } ((v - u) * x + u) \text{ within } \{0..1\})$
using *test2[OF s fs uv]*
by *auto*
have $arg: \bigwedge x. (\sum_{n \in \text{basis}} (v - u) * (f (g ((v - u) * x + u)) \cdot n) * (\text{vector-derivative } g \text{ (at } ((v - u) * x + u) \text{ within } \{0..1\}) \cdot n))$
 $= (\sum_{b \in \text{basis}} f (g ((v - u) * x + u)) \cdot b * (v - u) * (\text{vector-derivative } g \text{ (at } ((v - u) * x + u) \text{ within } \{0..1\}) \cdot b))$
by (*simp add: mult.commute*)
have $((\lambda x. \sum_{b \in \text{basis}} f (g ((v - u) * x + u)) \cdot b * (\text{vector-derivative } (\lambda x. g$
 $((v - u) * x + u)) \text{ (at } x \text{ within } \{0..1\}) \cdot b)) \text{has-integral}$
 $(\text{integral } \{u..v\} (\lambda x. \sum_{b \in \text{basis}} f (g (x)) \cdot b * (\text{vector-derivative } g \text{ (at } x$
 $\text{within } \{0..1\}) \cdot b)))) \{0..1\}$
apply (*cut-tac Henstock-Kurzweil-Integration.has-integral-mult-right [OF *,*
where $c = v - u]$)
using *fs assms*
apply (*simp add: False subpath-def line-integral-exists-def*)
apply (*rule-tac S = (\lambda t. ((v - u) *_R t + u)) - 's in has-integral-spike-finite*)
apply (*auto simp: inj-on-def False vd finite-vimageI scaleR-conv-of-real*
Groups-Big.sum-distrib-left
mult.assoc[symmetric] arg)
done
then show (*line-integral-exists f basis (subpath u v g)*)
by (*auto simp add: line-integral-exists-def subpath-def integrable-on-def*)
qed

type-synonym $\text{path} = \text{real} \Rightarrow (\text{real} * \text{real})$

type-synonym *one-cube* = (real \Rightarrow (real * real))
type-synonym *one-chain* = (int * path) set
type-synonym *two-cube* = (real * real) \Rightarrow (real * real)
type-synonym *two-chain* = *two-cube* set

definition *one-chain-line-integral* :: ((real * real) \Rightarrow (real * real)) \Rightarrow ((real*real) set) \Rightarrow *one-chain* \Rightarrow real **where**
one-chain-line-integral F b C \equiv ($\sum (k,g) \in C. k * (\text{line-integral } F \text{ b } g)$)

definition *boundary-chain* **where**
boundary-chain s \equiv ($\forall (k, \gamma) \in s. k = 1 \vee k = -1$)

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

fun *rec-join* :: (int*path) list \Rightarrow path **where**
rec-join [] = ($\lambda x. 0$) |
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) \wedge *valid-chain-list* l

lemma *joined-is-valid*:

assumes *boundary-chain*: *boundary-chain* (set l) **and**
valid-path: $\bigwedge k \gamma. (k, \gamma) \in \text{set } l \implies \text{valid-path } \gamma$ **and**
valid-chain-list-ass: *valid-chain-list* l
shows *valid-path* (rec-join l)
using *assms*

proof(*induction* l)

case Nil

then show ?case

using *C1-differentiable-imp-piecewise*[OF *C1-differentiable-on-const*[of 0 {0..1}]]
by (*auto simp add: valid-path-def*)

next

case (Cons a l)

have *: *valid-path* (rec-join ((k::int, γ) # l))

if *boundary-chain* (set (l))

($\bigwedge k' \gamma'. (k', \gamma') \in \text{set } l \implies \text{valid-path } \gamma'$)

valid-chain-list l

valid-path (rec-join l)

($\bigwedge k' \gamma'. (k', \gamma') \in \text{set } ((k, \gamma) \# l) \implies \text{valid-path } \gamma'$)

valid-chain-list ((k, γ) # l)

boundary-chain (set ((k, γ) # l)) **for** k γ l

proof (*cases* l = [])

```

    case True
    with that show valid-path (rec-join ((k, γ) # l))
    by auto
next
case False
then obtain h l' where l-empty: l = h#l'
  by (meson rec-join.elims)
then show valid-path (rec-join ((k, γ) # l))
proof (simp, intro conjI impI)
  assume k-eq-1: k = 1
  have 0:valid-path γ
    using that by auto
  have 1:pathfinish γ = pathstart (rec-join (h#l'))
    using that(6) k-eq-1 l-empty by auto
  show valid-path (γ +++ rec-join (h#l'))
    using 0 1 valid-path-join that(4) l-empty by auto
next
assume k ≠ 1
then have k-eq-neg-1: k = -1
  using that(7)
  by (auto simp add: boundary-chain-def)
have valid-path γ
  using that by auto
then have 0: valid-path (reversepath γ)
  using Cauchy-Integral-Theorem.valid-path-imp-reverse
  by auto
have 1: pathfinish (reversepath γ) = pathstart (rec-join (h#l'))
  using that(6) k-eq-neg-1 l-empty by auto
show valid-path ((reversepath γ) +++ rec-join (h#l'))
  using 0 1 Cauchy-Integral-Theorem.valid-path-join that(4) l-empty by blast
qed
qed
have ∀ ps. valid-chain-list ps ∨ (∃ i f p psa. ps = (i, f) # p # psa ∧ ((i = 1 ∧
pathfinish f ≠ pathstart (rec-join (p # psa)) ∨ i ≠ 1 ∧ pathfinish (reversepath f)
≠ pathstart (rec-join (p # psa))) ∨ ¬ valid-chain-list (p # psa)))
  by (smt coeff-cube-to-path.elims valid-chain-list.elims(3))
moreover have boundary-chain (set l)
  by (meson Cons.premis(1) boundary-chain-def set-subset-Cons subset-eq)
ultimately show ?case
  using * Cons by (metis (no-types) list.set-intros(2) prod.collapse valid-chain-list.simps(3))
qed

lemma pathstart-rec-join-1:
  pathstart (rec-join ((1, γ) # l)) = pathstart γ
proof (cases l = [])
case True
  then show pathstart (rec-join ((1, γ) # l)) = pathstart γ
  by simp
next

```

case *False*
then obtain $h\ l'$ **where** $l = h\#l'$
 by (*meson rec-join.elims*)
then show $\text{pathstart } (\text{rec-join } ((1, \gamma) \# l)) = \text{pathstart } \gamma$
 by *simp*
qed

lemma *pathstart-rec-join-2*:
 $\text{pathstart } (\text{rec-join } ((-1, \gamma) \# l)) = \text{pathstart } (\text{reversepath } \gamma)$
proof *cases*
 assume $l = []$
then show $\text{pathstart } (\text{rec-join } ((-1, \gamma) \# l)) = \text{pathstart } (\text{reversepath } \gamma)$
 by *simp*
next
 assume $l \neq []$
then obtain $h\ l'$ **where** $l = h\#l'$
 by (*meson rec-join.elims*)
then show $\text{pathstart } (\text{rec-join } ((-1, \gamma) \# l)) = \text{pathstart } (\text{reversepath } \gamma)$
 by *simp*
qed

lemma *pathstart-rec-join*:
 $\text{pathstart } (\text{rec-join } ((1, \gamma) \# l)) = \text{pathstart } \gamma$
 $\text{pathstart } (\text{rec-join } ((-1, \gamma) \# l)) = \text{pathstart } (\text{reversepath } \gamma)$
using *pathstart-rec-join-1 pathstart-rec-join-2*
by *auto*

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: $\bigwedge k\ \gamma. (k, \gamma) \in \text{set } l \implies \text{valid-path } \gamma$ and
line-integral-exists: $\forall (k, \gamma) \in \text{set } l. \text{line-integral-exists } F \text{ basis } \gamma$
shows *line-integral-exists F basis (rec-join l)*
using *assms*
proof (*induction l*)
case *Nil*
then show *?case*
proof (*simp add: line-integral-exists-def*)
have $\forall x. (\text{vector-derivative } (\lambda x. 0) \text{ (at } x)) = 0$
using *Derivative.vector-derivative-const-at*
by *auto*
then have $\forall x. ((\lambda x. 0) \text{ has-vector-derivative } 0) \text{ (at } x)$
using *Derivative.vector-derivative-const-at*
by *auto*
then have $\forall x. ((\lambda x. 0) \text{ has-vector-derivative } 0) \text{ (at } x \text{ within } \{0..1\})$
using *Derivative.vector-derivative-const-at*
by *auto*
then have $0: \forall x \in \{0..1\}. (\text{vector-derivative } (\lambda x. 0) \text{ (at } x \text{ within } \{0..1\})) = 0$
by (*simp add: gamma-deriv-at-within*)

```

have ( $\forall x \in \{0..1\}. (\sum_{b \in \text{basis}} F 0 \cdot b * (\text{vector-derivative } (\lambda x. 0) (\text{at } x \text{ within } \{0..1\}) \cdot b)) = 0)$ )
  by (simp add: 0)
then have ( $(\lambda x. \sum_{b \in \text{basis}} F 0 \cdot b * (\text{vector-derivative } (\lambda x. 0) (\text{at } x \text{ within } \{0..1\}) \cdot b)) \text{ has-integral } 0 \{0..1\}$ )
  by (meson has-integral-is-0)
then show ( $\lambda x. \sum_{b \in \text{basis}} F 0 \cdot b * (\text{vector-derivative } (\lambda x. 0) (\text{at } x \text{ within } \{0..1\}) \cdot b)$ ) integrable-on  $\{0..1\}$ 
  by auto
qed
next
case (Cons a l)
obtain  $k \ \gamma$  where aeq: a = (k,  $\gamma$ )
  by force
show ?case
  unfolding aeq
proof cases
  assume l-empty: l = []
  then show line-integral-exists F basis (rec-join ((k,  $\gamma$ ) # l))
    using Cons.prem1 aeq line-integral-on-reverse-path(2) by fastforce
next
  assume  $l \neq []$ 
  then obtain  $h \ l'$  where l-nempty: l = h # l'
    by (meson rec-join.elims)
  show line-integral-exists F basis (rec-join ((k,  $\gamma$ ) # l))
  proof (auto simp add: l-nempty)
    assume k-eq-1: k = 1
    have 0: line-integral-exists F basis  $\gamma$ 
      using Cons.prem4 aeq by auto
    have 1: line-integral-exists F basis (rec-join l)
    by (metis (mono-tags) Cons boundary-chain-def list.set-intros(2) valid-chain-list.elims(3) valid-chain-list.simps(3))
    have 2: valid-path  $\gamma$ 
      using Cons aeq by auto
    have 3: valid-path (rec-join l)
    by (metis (no-types) Cons.prem1 boundary-chain-def joined-is-valid l-nempty set-subset-Cons subsetCE valid-chain-list.simps(3))
    show line-integral-exists F basis ( $\gamma$  +++ rec-join (h # l'))
      using line-integral-distrib(2)[OF 0 1 2 3] asms l-nempty by auto
next
    assume  $k \neq 1$ 
    then have k-eq-neg-1: k = -1
      using Cons aeq by (simp add: boundary-chain-def)
    have gamma-valid: valid-path  $\gamma$ 
      using Cons aeq by auto
    then have 2: valid-path (reversepath  $\gamma$ )
      using Cauchy-Integral-Theorem.valid-path-imp-reverse by auto
    have line-integral-exists F basis  $\gamma$ 
      using Cons aeq by auto

```

```

then have 0: line-integral-exists F basis (reversepath  $\gamma$ )
  using line-integral-on-reverse-path(2) gamma-valid
  by auto
have 1: line-integral-exists F basis (rec-join l)
  using Cons aeq
  by (metis (mono-tags) boundary-chain-def insert-iff list.set(2) list.set-intros(2)
valid-chain-list.elims(3) valid-chain-list.simps(3))
  have 3:valid-path (rec-join l)
  by (metis (no-types) Cons.prem(1) Cons.prem(2) Cons.prem(3) boundary-chain-def
joined-is-valid l-nempty set-subset-Cons subsetCE valid-chain-list.simps(3))
  show line-integral-exists F basis ((reversepath  $\gamma$ ) +++ rec-join (h#l'))
  using line-integral-distrib(2)[OF 0 1 2 3] assms l-nempty
  by auto
qed
qed
qed

```

```

lemma line-integral-exists-rec-join-cons:
  assumes line-integral-exists F basis (rec-join ((1, $\gamma$ ) # l))
    ( $\bigwedge k' \gamma'. (k', \gamma') \in \text{set } ((1, \gamma) \# l) \implies \text{valid-path } \gamma'$ )
    finite basis
  shows line-integral-exists F basis ( $\gamma$  +++ (rec-join l))
proof cases
  assume l-empty: l = []
  show line-integral-exists F basis ( $\gamma$  +++ rec-join l)
  using assms(2) line-integral-distrib(2)[OF assms(1) line-integral-exists-on-degenerate-path[OF
assms(3)], of 0]
  using degenerate-path-is-valid-path
  by (fastforce simp add: l-empty)
next
  assume l  $\neq$  []
  then obtain h l' where l = h#l'
  by (meson rec-join.elims)
  then show line-integral-exists F basis ( $\gamma$  +++ rec-join l)
  using assms by auto
qed

```

```

lemma line-integral-exists-rec-join-cons-2:
  assumes line-integral-exists F basis (rec-join ((-1, $\gamma$ ) # l))
    ( $\bigwedge k' \gamma'. (k', \gamma') \in \text{set } ((1, \gamma) \# l) \implies \text{valid-path } \gamma'$ )
    finite basis
  shows line-integral-exists F basis ((reversepath  $\gamma$ ) +++ (rec-join l))
proof cases
  assume l-empty: l = []
  show line-integral-exists F basis ((reversepath  $\gamma$ ) +++ rec-join l)
  using assms(2) line-integral-distrib(2)[OF assms(1) line-integral-exists-on-degenerate-path[OF
assms(3)], of 0]
  using degenerate-path-is-valid-path
  by (auto simp add: l-empty)

```

```

next
  assume  $l \neq []$ 
  then obtain  $h\ l'$  where  $l = h\#l'$ 
    by (meson rec-join.elims)
  with assms show line-integral-exists  $F$  basis ((reversepath  $\gamma$ ) +++ rec-join  $l$ )
    using assms by auto
qed

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:  $\bigwedge k\ \gamma. (k, \gamma) \in \text{set } l \implies \text{valid-path } \gamma$  and
    line-integral-exists: line-integral-exists  $F$  basis (rec-join  $l$ ) and
    finite-basis: finite basis
  shows  $\forall (k, \gamma) \in \text{set } l. \text{line-integral-exists } F \text{ basis } \gamma$ 
  using assms
proof (induction  $l$ )
  case Nil
  show ?case
    by (simp add: line-integral-exists-def)
next
  case ass: (Cons  $a\ l$ )
  obtain  $k\ \gamma$  where  $k\text{-gamma}: a = (k, \gamma)$ 
    by fastforce
  show ?case
    apply (auto simp add:  $k\text{-gamma}$ )
  proof -
    show line-integral-exists  $F$  basis  $\gamma$ 
    proof (cases  $k = 1$ )
      assume  $k\text{-eq-1}: k = 1$ 
      have 0: line-integral-exists  $F$  basis ( $\gamma$  +++ (rec-join  $l$ ))
        using line-integral-exists-rec-join-cons  $k\text{-eq-1}\ k\text{-gamma}\ \text{ass}(4)\ \text{ass}(5)\ \text{ass}(6)$ 
        by auto
      have 2: valid-path  $\gamma$ 
        using ass  $k\text{-gamma}$  by auto
      show line-integral-exists  $F$  basis  $\gamma$ 
        using line-integral-exists-joinD1[OF 0 2]
        by auto
    next
      assume  $k \neq 1$ 
      then have  $k\text{-eq-neg-1}: k = -1$ 
        using ass  $k\text{-gamma}$ 
        by (simp add: boundary-chain-def)
      have 0: line-integral-exists  $F$  basis ((reversepath  $\gamma$ ) +++ (rec-join  $l$ ))
        using line-integral-exists-rec-join-cons-2[OF ]  $k\text{-eq-neg-1}\ k\text{-gamma}\ \text{ass}(4)$ 
        ass(5) ass(6)
        by fastforce
      have  $\text{gamma-valid}: \text{valid-path } \gamma$ 

```

```

    using ass k-gamma by auto
  then have 2: valid-path (reversepath  $\gamma$ )
    using Cauchy-Integral-Theorem.valid-path-imp-reverse by auto
  have line-integral-exists F basis (reversepath  $\gamma$ )
    using line-integral-exists-joinD1 [OF 0 2] by auto
  then show line-integral-exists F basis ( $\gamma$ )
    using line-integral-on-reverse-path(2) [OF 2] reversepath-reversepath
    by auto
qed
next
have 0: boundary-chain (set l)
  using ass(2)
  by (auto simp add: boundary-chain-def)
have 1: valid-chain-list l
  using ass(3)
  apply (auto simp add: k-gamma)
  by (metis valid-chain-list.elims(3) valid-chain-list.simps(3))
have 2:  $(\bigwedge k \gamma. (k, \gamma) \in \text{set } l) \implies \text{valid-path } \gamma$ 
  using ass(4) by auto
have 3: valid-path (rec-join l)
  using joined-is-valid [OF 0] 1 2 by auto
have 4: line-integral-exists F basis (rec-join l)
proof(cases k = 1)
  assume k-eq-1: k = 1
  have 0: line-integral-exists F basis ( $\gamma$  +++ (rec-join l))
    using line-integral-exists-rec-join-cons k-eq-1 k-gamma ass(4) ass(5) ass(6)
  by auto
  show line-integral-exists F basis (rec-join l)
    using line-integral-exists-joinD2 [OF 0 3] by auto
next
  assume k  $\neq$  1
  then have k-eq-neg-1: k = -1
    using ass k-gamma
    by (simp add: boundary-chain-def)
  have 0: line-integral-exists F basis ((reversepath  $\gamma$ ) +++ (rec-join l))
    using line-integral-exists-rec-join-cons-2 [OF ] k-eq-neg-1 k-gamma ass(4)
ass(5) ass(6)
    by fastforce
  show line-integral-exists F basis (rec-join l)
    using line-integral-exists-joinD2 [OF 0 3]
    by auto
qed
show  $\bigwedge a b. (a, b) \in \text{set } l \implies \text{line-integral-exists F basis } b$ 
  using 0 1 2 3 4 ass(1) [OF 0 1 2] ass(6)
  by fastforce
qed
qed
inductive chain-subdiv-path

```


where I : *chain-subdiv-path* γ (set l) **if** *distinct* l *rec-join* $l = \gamma$ *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* $\forall (k, \gamma) \in \text{one-chain. valid-path } \gamma$

shows *valid-path* γ

proof –

obtain l **where** *l-props*: set $l = \text{one-chain distinct } l \text{ rec-join } l = \gamma \text{ valid-chain-list } l$

using *chain-subdiv-path.cases path-eq-chain* **by** *force*

show *valid-path* γ

using *joined-is-valid assms l-props* **by** *blast*

qed

lemma *line-integral-rec-join-cons*:

assumes *line-integral-exists F basis* γ

line-integral-exists F basis (rec-join ((l)))

$(\bigwedge k' \gamma'. (k', \gamma') \in \text{set } ((1, \gamma) \# l) \implies \text{valid-path } \gamma')$

finite basis

shows *line-integral F basis (rec-join (($1, \gamma$) # l)) = line-integral F basis (γ +++ (rec-join l))*

proof *cases*

assume *l-empty*: $l = []$

show *line-integral F basis (rec-join (($1, \gamma$) # l)) = line-integral F basis (γ +++ (rec-join l))*

using *assms line-integral-distrib(1)[OF assms(1) line-integral-exists-on-degenerate-path[OF assms(4)], of 0]*

apply (*auto simp add: l-empty*)

using *degenerate-path-is-valid-path line-integral-degenerate-path*

by *fastforce*

next

assume $l \neq []$

then obtain $h l'$ **where** *l-nempty*: $l = h \# l'$

by (*meson rec-join.elims*)

show *line-integral F basis (rec-join (($1, \gamma$) # l)) = line-integral F basis (γ +++ (rec-join l))*

using *assms* **by** (*auto simp add: l-nempty*)

qed

lemma *line-integral-rec-join-cons-2*:

assumes *line-integral-exists F basis* γ

line-integral-exists F basis (rec-join ((l)))

$(\bigwedge k' \gamma'. (k', \gamma') \in \text{set } ((-1, \gamma) \# l) \implies \text{valid-path } \gamma')$

finite basis

shows *line-integral F basis (rec-join (($-1, \gamma$) # l)) = line-integral F basis ((reversepath γ) +++ (rec-join l))*

proof *cases*

assume *l-empty*: $l = []$

have 0 : *line-integral-exists F basis (reversepath γ)*

```

    using assms line-integral-on-reverse-path(2) by fastforce
  have 1: valid-path (reversepath  $\gamma$ )
    using assms by fastforce
  show line-integral F basis (rec-join ((-1, $\gamma$ ) # l)) = line-integral F basis ((reversepath
 $\gamma$ ) +++ (rec-join l))
    using assms line-integral-distrib(1)[OF 0 line-integral-exists-on-degenerate-path[OF
assms(4)], of 0]
    apply (auto simp add: l-empty)
    using degenerate-path-is-valid-path line-integral-degenerate-path
    by fastforce
next
  assume l  $\neq$  []
  then obtain h l' where l-nempty: l = h#l'
    by (meson rec-join.elims)
  show line-integral F basis (rec-join ((-1, $\gamma$ ) # l)) = line-integral F basis ((reversepath
 $\gamma$ ) +++ (rec-join l))
    using assms by (auto simp add: l-nempty)
qed

```

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:  $\forall (k::int, \gamma) \in \text{one-chain. line-integral-exists F basis } \gamma$  and
    valid-path:  $\forall (k::int, \gamma) \in \text{one-chain. valid-path } \gamma$  and
    finite-basis: finite basis
  shows line-integral F basis (rec-join l) = one-chain-line-integral F basis one-chain
  proof -
    have 0: sum-list (map ( $\lambda(k::int, \gamma). (k::int) * (\text{line-integral F basis } \gamma)$ ) l) =
one-chain-line-integral F basis one-chain
      unfolding one-chain-line-integral-def
      using l-props Groups-List.comm-monoid-add-class.sum.distinct-set-conv-list[OF
l-props(2), of ( $\lambda(k, \gamma). (k::int) * (\text{line-integral F basis } \gamma)$ )]
      by auto
    have valid-chain-list l  $\implies$ 
      boundary-chain (set l)  $\implies$ 
      ( $\forall (k::int, \gamma) \in \text{set l. line-integral-exists F basis } \gamma$ )  $\implies$ 
      ( $\forall (k::int, \gamma) \in \text{set l. valid-path } \gamma$ )  $\implies$ 
      line-integral F basis (rec-join l) = sum-list (map ( $\lambda(k::int, \gamma). k *
(\text{line-integral F basis } \gamma)$ ) l)
    proof (induction l)
      case Nil
      show ?case
        unfolding line-integral-def boundary-chain-def
        apply (auto)
      proof
        have  $\forall x. (\text{vector-derivative } (\lambda x. 0) (at x)) = 0$ 
          using Derivative.vector-derivative-const-at
          by auto
        then have  $\forall x. ((\lambda x. 0) \text{ has-vector-derivative } 0) (at x)$ 

```

```

    using Derivative.vector-derivative-const-at
  by auto
  then have  $\forall x. ((\lambda x. 0) \text{ has-vector-derivative } 0) \text{ (at } x \text{ within } \{0..1\})$ 
    using Derivative.vector-derivative-const-at
  by auto
  then have  $0: \forall x \in \{0..1\}. (\text{vector-derivative } (\lambda x. 0) \text{ (at } x \text{ within } \{0..1\})) = 0$ 
    by (metis (no-types) box-real(2) vector-derivative-within-cbox zero-less-one)
  have  $(\forall x \in \{0..1\}. (\sum_{b \in \text{basis}} F 0 \cdot b * (\text{vector-derivative } (\lambda x. 0) \text{ (at } x \text{ within } \{0..1\}) \cdot b)) = 0)$ 
    by (simp add: 0)
  then show  $((\lambda x. \sum_{b \in \text{basis}} F 0 \cdot b * (\text{vector-derivative } (\lambda x. 0) \text{ (at } x \text{ within } \{0..1\}) \cdot b)) \text{ has-integral } 0) \{0..1\}$ 
    by (meson has-integral-is-0)
  qed
next
case ass: (Cons a l)
obtain  $k::\text{int}$  and
 $\gamma::\text{one-cube}$  where props:  $a = (k, \gamma)$ 
proof
  let  $?k2 = \text{fst } a$ 
  let  $? \gamma 2 = \text{snd } a$ 
  show  $a = (?k2, ? \gamma 2)$ 
  by auto
qed
have line-integral-exists  $F$  basis (rec-join (a # l))
  using line-integral-exists-on-rec-join[OF ass(3) ass(2)] ass(5) ass(4)
  by blast
have boundary-chain (set l)
  by (meson ass.prem(2) boundary-chain-def list.set-intros(2))
have val-l:  $\bigwedge f i. (i, f) \in \text{set } l \implies \text{valid-path } f$ 
  using ass.prem(4) by fastforce
have vcl-l: valid-chain-list l
  by (metis (no-types) ass.prem(1) valid-chain-list.elims(3) valid-chain-list.simps(3))
have line-integral-exists-on-joined:
  line-integral-exists  $F$  basis (rec-join l)
  by (metis (boundary-chain (set l)) (line-integral-exists  $F$  basis (rec-join (a # l))) emptyE val-l vcl-l joined-is-valid line-integral-exists-joinD2 line-integral-exists-on-rec-join list.set(1) neq-Nil-conv rec-join.simps(3))
have valid-path (rec-join (a # l))
  using joined-is-valid ass(5) ass(3) ass(2) by blast
then have joined-is-valid: valid-path (rec-join l)
  using (boundary-chain (set l)) val-l vcl-l joined-is-valid by blast
show ?case
proof (clarsimp, cases)
  assume k-eq-1:  $(k::\text{int}) = 1$ 
  have line-integral-exists-on-gamma: line-integral-exists  $F$  basis  $\gamma$ 
    using ass props by auto
  have gamma-is-valid: valid-path  $\gamma$ 
    using ass props by auto

```

```

have line-int-rw: line-integral F basis (rec-join ((k,  $\gamma$ ) # l)) = line-integral
F basis ( $\gamma$  +++ rec-join l)
proof –
  have gam-int: line-integral-exists F basis  $\gamma$  using ass props by auto
  have rec-join-int: line-integral-exists F basis (rec-join l)
    using line-integral-exists-on-rec-join
    using line-integral-exists-on-joined by blast
  show ?thesis
    using line-integral-rec-join-cons[OF gam-int rec-join-int] ass k-eq-1
finite-basis props by force
qed
show line-integral F basis (rec-join (a # l)) =
  (case a of (x,  $\gamma$ )  $\Rightarrow$  real-of-int x * line-integral F basis  $\gamma$ ) + ( $\sum$  (x,  $\gamma$ ) $\leftarrow$ l.
real-of-int x * line-integral F basis  $\gamma$ )
  apply (simp add: props line-int-rw)
  using line-integral-distrib[OF line-integral-exists-on-gamma line-integral-exists-on-joined
gamma-is-valid joined-is-valid]
  ass k-eq-1 vcl-l
  by (auto simp: boundary-chain-def props)
next
assume k  $\neq$  1
then have k-eq-neg-1: k = -1
  using ass props
  by (auto simp add: boundary-chain-def)
have line-integral-exists-on-gamma:
  line-integral-exists F basis (reversepath  $\gamma$ )
  using line-integral-on-reverse-path ass props
  by auto
have gamma-is-valid: valid-path (reversepath  $\gamma$ )
  using Cauchy-Integral-Theorem.valid-path-imp-reverse ass props by auto
have line-int-rw: line-integral F basis (rec-join ((k,  $\gamma$ ) # l)) = line-integral
F basis ((reversepath  $\gamma$ ) +++ rec-join l)
proof –
  have gam-int: line-integral-exists F basis  $\gamma$  using ass props by auto
  have rec-join-int: line-integral-exists F basis (rec-join l)
    using line-integral-exists-on-rec-join
    using line-integral-exists-on-joined by blast
  show ?thesis
    using line-integral-rec-join-cons-2[OF gam-int rec-join-int]
    using ass k-eq-neg-1
    using finite-basis props by blast
qed
show line-integral F basis (rec-join (a # l)) =
  (case a of (x,  $\gamma$ )  $\Rightarrow$  real-of-int x * line-integral F basis  $\gamma$ ) + ( $\sum$  (x,  $\gamma$ ) $\leftarrow$ l.
real-of-int x * line-integral F basis  $\gamma$ )
  apply (simp add: props line-int-rw)
  using line-integral-distrib[OF line-integral-exists-on-gamma line-integral-exists-on-joined
gamma-is-valid joined-is-valid]
  props ass line-integral-on-reverse-path(1)[of  $\gamma$  F basis] k-eq-neg-1

```

```

    using ⟨boundary-chain (set l)⟩ vcl-l by auto
  qed
  qed
  then have 1:line-integral F basis (rec-join l) = sum-list (map (λ(k::int, γ). k *
(line-integral F basis γ)) l)
    using l-props assms by auto
  then show ?thesis
    using 0 1 by auto
  qed

```

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: ∀ (k::int, γ) ∈ one-chain. line-integral-exists F basis γ and
    valid-path: ∀ (k::int, γ) ∈ 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 –
  obtain l where l-props: set l = one-chain distinct l rec-join l = γ valid-chain-list
  l
    using chain-subdiv-path.cases path-eq-chain by force
  show line-integral-exists F basis γ
    using line-integral-exists-on-rec-join assms l-props
    by blast
  show valid-path γ
    using joined-is-valid assms l-props
    by blast
  have line-integral F basis (rec-join l) = one-chain-line-integral F basis one-chain
    using one-chain-line-integral-rec-join l-props assms by auto
  then show one-chain-line-integral F basis one-chain = line-integral F basis γ
    using l-props
    by auto
  qed

```

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: ∀ (k, γ) ∈ one-chain. valid-path γ and
    finite-basis: finite basis
  shows one-chain-line-integral F basis one-chain = line-integral F basis γ
    ∀ (k, γ) ∈ one-chain. line-integral-exists F basis γ
  proof –
  obtain l where l-props: set l = one-chain distinct l rec-join l = γ valid-chain-list
  l
    using chain-subdiv-path.cases path-eq-chain by force
  show 0: ∀ (k, γ) ∈ one-chain. line-integral-exists F basis γ

```

using *line-integral-exists-on-rec-join'* *assms l-props*
by *blast*
show *one-chain-line-integral F basis one-chain = line-integral F basis γ*
using *line-integral-on-path-eq-line-integral-on-equiv-chain(1)[OF assms(1) assms(2)*
0 assms(4) assms(5)] by auto
qed

definition *chain-subdiv-chain where*

chain-subdiv-chain one-chain1 subdiv
 $\equiv \exists f. (\bigcup (f \text{ ' one-chain1})) = \text{subdiv} \wedge$
 $(\forall c \in \text{one-chain1}. \text{chain-subdiv-path } (\text{coeff-cube-to-path } c) (f \ c)) \wedge$
 $\text{pairwise } (\lambda p \ p'. f \ p \cap f \ p' = \{\}) \text{ one-chain1} \wedge$
 $(\forall x \in \text{one-chain1}. \text{finite } (f \ x))$

lemma *chain-subdiv-chain-character:*

shows *chain-subdiv-chain one-chain1 subdiv \longleftrightarrow*
 $(\exists f. \bigcup (f \text{ ' one-chain1}) = \text{subdiv} \wedge$
 $(\forall (k, \gamma) \in \text{one-chain1}.$
 $\text{if } k = 1$
 $\text{then chain-subdiv-path } \gamma (f \ (k, \gamma))$
 $\text{else chain-subdiv-path } (\text{reversepath } \gamma) (f \ (k, \gamma))) \wedge$
 $(\forall p \in \text{one-chain1}.$
 $\forall p' \in \text{one-chain1}. p \neq p' \longrightarrow f \ p \cap f \ p' = \{\}) \wedge$
 $(\forall x \in \text{one-chain1}. \text{finite } (f \ x)))$

unfolding *chain-subdiv-chain-def*

by (*safe; intro exI conjI iffI; fastforce simp add: pairwise-def*)

lemma *chain-subdiv-chain-imp-finite-subdiv:*

assumes *finite one-chain1*
chain-subdiv-chain one-chain1 subdiv
shows *finite subdiv*
using *assms by(auto simp add: chain-subdiv-chain-def)*

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: $\forall (k, \gamma) \in \text{subdiv}. \text{valid-path } \gamma$
shows $\forall (k, \gamma) \in \text{one-chain1}. \text{valid-path } \gamma$

proof –

obtain *f where f-props:*

$((\bigcup (f \text{ ' one-chain1})) = \text{subdiv})$
 $(\forall (k, \gamma) \in \text{one-chain1}. \text{if } k = 1 \text{ then chain-subdiv-path } \gamma (f \ (k, \gamma)) \text{ else chain-subdiv-path}$
 $(\text{reversepath } \gamma) (f \ (k, \gamma)))$
 $(\forall p \in \text{one-chain1}. \forall p' \in \text{one-chain1}. p \neq p' \longrightarrow f \ p \cap f \ p' = \{\})$

using *chain1-eq-chain2 chain-subdiv-chain-character by auto*

have $\bigwedge k \ \gamma. (k, \gamma) \in \text{one-chain1} \implies \text{valid-path } \gamma$

proof –

fix *k γ*

```

assume ass:  $(k, \gamma) \in \text{one-chain1}$ 
show valid-path  $\gamma$ 
proof (cases  $k = 1$ )
  assume k-eq-1:  $k = 1$ 
  then have i:chain-subdiv-path  $\gamma$  ( $f(k, \gamma)$ )
    using f-props(2) ass by auto
  have ii:boundary-chain ( $f(k, \gamma)$ )
    using f-props(1) ass assms
    apply (simp add: boundary-chain-def)
    by blast
  have iv:  $\forall (k, \gamma) \in f(k, \gamma). \text{valid-path } \gamma$ 
    using f-props(1) ass assms
    by blast
  show ?thesis
    using valid-path-equiv-valid-chain-list[OF i ii iv]
    by auto
next
  assume  $k \neq 1$ 
  then have k-eq-neg1:  $k = -1$ 
    using ass boundary-chain1
    by (auto simp add: boundary-chain-def)
  then have i:chain-subdiv-path (reversepath  $\gamma$ ) ( $f(k, \gamma)$ )
    using f-props(2) ass using  $\langle k \neq 1 \rangle$  by auto
  have ii:boundary-chain ( $f(k, \gamma)$ )
    using f-props(1) ass assms by (auto simp add: boundary-chain-def)
  have iv:  $\forall (k, \gamma) \in f(k, \gamma). \text{valid-path } \gamma$ 
    using f-props(1) ass assms
    by blast
  have valid-path (reversepath  $\gamma$ )
    using valid-path-equiv-valid-chain-list[OF i ii iv]
    by auto
  then show ?thesis
    using reversepath-reversepath Cauchy-Integral-Theorem.valid-path-imp-reverse
    by force
qed
qed
then show valid-path1:  $\forall (k, \gamma) \in \text{one-chain1}. \text{valid-path } \gamma$ 
  by auto
qed

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:  $\forall (k, \gamma) \in \text{subdiv}. \text{line-integral-exists } F \text{ basis } \gamma$ 
and
  valid-path:  $\forall (k, \gamma) \in \text{subdiv}. \text{valid-path } \gamma$  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*
 $\forall (k, \gamma) \in \text{one-chain1}. \text{line-integral-exists } F \text{ basis } \gamma$
proof –
obtain *f* **where** *f*-*props*:
 $((\bigcup (f \text{ ' one-chain1})) = \text{subdiv})$
 $(\forall (k, \gamma) \in \text{one-chain1}. \text{if } k = 1 \text{ then } \text{chain-subdiv-path } \gamma (f(k, \gamma)) \text{ else } \text{chain-subdiv-path } (\text{reversepath } \gamma) (f(k, \gamma)))$
 $(\forall p \in \text{one-chain1}. \forall p' \in \text{one-chain1}. p \neq p' \longrightarrow f p \cap f p' = \{\})$
 $(\forall x \in \text{one-chain1}. \text{finite } (f x))$
using *chain1-eq-chain2 chain-subdiv-chain-character* **by** (*auto simp add: pairwise-def chain-subdiv-chain-def*)
then have *0*: *one-chain-line-integral F basis subdiv = one-chain-line-integral F basis* $(\bigcup (f \text{ ' one-chain1}))$
by *auto*
have *finite-chain2*: *finite subdiv*
using *finite-chain1 f-props(1) f-props(4)*
apply (*simp add: image-def*)
using *f-props(1)* **by** *auto*
have $\bigwedge k \gamma. (k, \gamma) \in \text{one-chain1} \implies \text{line-integral-exists } F \text{ basis } \gamma$
proof–
fix *k* γ
assume *ass*: $(k, \gamma) \in \text{one-chain1}$
show *line-integral-exists F basis* γ
proof (*cases k = 1*)
assume *k-eq-1*: $k = 1$
then have *i*: *chain-subdiv-path* $\gamma (f(k, \gamma))$
using *f-props(2) ass* **by** *auto*
have *ii*: *boundary-chain* $(f(k, \gamma))$
using *f-props(1) ass assms* **by** (*auto simp add: boundary-chain-def*)
have *iii*: $\forall (k, \gamma) \in f (k, \gamma). \text{line-integral-exists } F \text{ basis } \gamma$
using *f-props(1) ass assms*
by *blast*
have *iv*: $\forall (k, \gamma) \in f (k, \gamma). \text{valid-path } \gamma$
using *f-props(1) ass assms*
by *blast*
show *?thesis*
using *line-integral-on-path-eq-line-integral-on-equiv-chain(2)[OF i ii iii iv finite-basis]*
by *auto*
next
assume $k \neq 1$
then have *k-eq-neg1*: $k = -1$
using *ass boundary-chain1*
by (*auto simp add: boundary-chain-def*)
then have *i*: *chain-subdiv-path* $(\text{reversepath } \gamma) (f(k, \gamma))$
using *f-props(2) ass* **by** *auto*
have *ii*: *boundary-chain* $(f(k, \gamma))$
using *f-props(1) ass assms* **by** (*auto simp add: boundary-chain-def*)


```

have iii:  $\forall (k, \gamma) \in f (k, \gamma). \text{line-integral-exists } F \text{ basis } \gamma$ 
  using f-props(1) ass assms
  by blast
have iv:  $\forall (k, \gamma) \in f (k, \gamma). \text{valid-path } \gamma$ 
  using f-props(1) ass assms
  by blast
have x:  $\text{line-integral-exists } F \text{ basis } (\text{reversepath } \gamma)$ 
  using line-integral-on-path-eq-line-integral-on-equiv-chain(2)[OF i ii iii iv
finite-basis]
  by auto
have valid-path (reversepath  $\gamma$ )
  using line-integral-on-path-eq-line-integral-on-equiv-chain(3)[OF i ii iii iv
finite-basis]
  by auto
then show ?thesis
  using line-integral-on-reverse-path(2) reversepath-reversepath x
  by fastforce
qed
qed
then show line-integral-exists-on-chain1:  $\forall (k, \gamma) \in \text{one-chain1}. \text{line-integral-exists}$ 
F basis  $\gamma$ 
  by auto
have 1:  $\text{one-chain-line-integral } F \text{ basis } (\bigcup (f \text{ ' one-chain1})) = \text{one-chain-line-integral}$ 
F basis one-chain1
proof –
  have 0:  $\text{one-chain-line-integral } F \text{ basis } (\bigcup (f \text{ ' one-chain1})) =$ 
 $(\sum \text{one-chain} \in (f \text{ ' one-chain1}). \text{one-chain-line-integral } F$ 
basis one-chain)
proof –
  have finite:  $\forall \text{chain} \in (f \text{ ' one-chain1}). \text{finite chain}$ 
  using f-props(1) finite-chain2
  by (meson Sup-upper finite-subset)
have disj:  $\forall A \in f \text{ ' one-chain1}. \forall B \in f \text{ ' one-chain1}. A \neq B \longrightarrow A \cap B = \{\}$ 
  by (metis (no-types, hide-lams) f-props(3) image-iff)
show  $\text{one-chain-line-integral } F \text{ basis } (\bigcup (f \text{ ' one-chain1})) =$ 
 $(\sum \text{one-chain} \in (f \text{ ' one-chain1}). \text{one-chain-line-integral}$ 
F basis one-chain)
  using Groups-Big.comm-monoid-add-class.sum.Union-disjoint[OF finite
disj]
   $\text{one-chain-line-integral-def}$ 
  by auto
qed
have 1:  $(\sum \text{one-chain} \in (f \text{ ' one-chain1}). \text{one-chain-line-integral } F \text{ basis one-chain})$ 
=
 $\text{one-chain-line-integral } F \text{ basis one-chain1}$ 
proof –
have  $(\sum \text{one-chain} \in (f \text{ ' one-chain1}). \text{one-chain-line-integral } F \text{ basis one-chain})$ 
=
 $(\sum (k, \gamma) \in \text{one-chain1}. k * (\text{line-integral } F \text{ basis } \gamma))$ 

```

```

proof –
  have  $i: (\sum \text{one-chain} \in (f \text{ ‘ } (\text{one-chain1} - \{p. f p = \{\}\})). \text{one-chain-line-integral}$ 
F basis one-chain) =
     $(\sum (k, \gamma) \in \text{one-chain1} - \{p. f p = \{\}\}). k * (\text{line-integral}$ 
F basis  $\gamma))$ 
  proof –
    have  $\text{inj-on } f (\text{one-chain1} - \{p. f p = \{\}\})$ 
    unfolding  $\text{inj-on-def}$  using  $f\text{-props}(3)$  by  $\text{blast}$ 
    then have  $0: (\sum \text{one-chain} \in (f \text{ ‘ } (\text{one-chain1} - \{p. f p = \{\}\})).$ 
one-chain-line-integral F basis one-chain)
      =  $(\sum (k, \gamma) \in (\text{one-chain1} - \{p. f p$ 
      =  $\{\}\}). \text{one-chain-line-integral } F \text{ basis } (f (k, \gamma)))$ 
    using  $\text{Groups-Big.comm-monoid-add-class.sum.reindex}$ 
    by  $\text{auto}$ 
    have  $\bigwedge k \gamma. (k, \gamma) \in (\text{one-chain1} - \{p. f p = \{\}\}) \implies$ 
       $\text{one-chain-line-integral } F \text{ basis } (f(k, \gamma)) = k * (\text{line-integral } F$ 
basis  $\gamma)$ 
    proof –
      fix  $k \gamma$ 
      assume  $\text{ass}: (k, \gamma) \in (\text{one-chain1} - \{p. f p = \{\}\})$ 
      have  $\text{bchain}: \text{boundary-chain } (f(k, \gamma))$ 
        using  $f\text{-props}(1)$   $\text{boundary-chain2 ass}$ 
        by  $(\text{auto simp add: boundary-chain-def})$ 
      have  $\text{wexist}: \forall (k, \gamma) \in (f(k, \gamma)). \text{line-integral-exists } F \text{ basis } \gamma$ 
        using  $f\text{-props}(1)$   $\text{line-integral-exists-on-chain2 ass}$ 
        by  $\text{blast}$ 
      have  $\text{vpath}: \forall (k, \gamma) \in (f(k, \gamma)). \text{valid-path } \gamma$ 
        using  $f\text{-props}(1)$   $\text{assms ass}$ 
        by  $\text{blast}$ 
      show  $\text{one-chain-line-integral } F \text{ basis } (f (k, \gamma)) = k * \text{line-integral } F \text{ basis}$ 
γ
    proof( $\text{cases } k = 1$ )
      assume  $k\text{-eq-1}: k = 1$ 
      have  $\text{one-chain-line-integral } F \text{ basis } (f (k, \gamma)) = \text{line-integral } F \text{ basis } \gamma$ 
      using  $f\text{-props}(2)$   $k\text{-eq-1}$   $\text{line-integral-on-path-eq-line-integral-on-equiv-chain}$ 
bchain wexist vpath ass finite-basis
      by  $\text{auto}$ 
      then show  $\text{one-chain-line-integral } F \text{ basis } (f (k, \gamma)) = k * \text{line-integral}$ 
F basis  $\gamma$ 
        using  $k\text{-eq-1}$  by  $\text{auto}$ 
      next
      assume  $k \neq 1$ 
      then have  $k\text{-eq-neg1}: k = -1$ 
        using  $\text{ass}$   $\text{boundary-chain1}$ 
        by  $(\text{auto simp add: boundary-chain-def})$ 
      have  $\text{one-chain-line-integral } F \text{ basis } (f (k, \gamma)) = \text{line-integral } F \text{ basis}$ 
(reversepath  $\gamma)$ 
        using  $f\text{-props}(2)$   $k\text{-eq-neg1}$   $\text{line-integral-on-path-eq-line-integral-on-equiv-chain}$ 
bchain wexist vpath ass finite-basis

```

```

      by auto
    then have one-chain-line-integral F basis (f (k, γ)) = - (line-integral
F basis γ)
      using line-integral-on-reverse-path(1) ass line-integral-exists-on-chain1
      valid-subdiv-imp-valid-one-chain[OF chain1-eq-chain2 boundary-chain1
boundary-chain2 valid-path]
      by force
    then show one-chain-line-integral F basis (f (k, γ)) = k * line-integral
F basis γ
      using k-eq-neg1 by auto
    qed
  qed
  then have (∑ (k, γ) ∈ (one-chain1 - {p. f p = {}}). one-chain-line-integral
F basis (f (k, γ)))
    = (∑ (k, γ) ∈ (one-chain1 - {p. f p = {}}). k * (line-integral F
basis γ))
    by (auto intro!: Finite-Cartesian-Product.sum-cong-aux)
    then show (∑ one-chain ∈ (f ' (one-chain1 - {p. f p = {}})).
one-chain-line-integral F basis one-chain) =
      (∑ (k, γ) ∈ (one-chain1 - {p. f p =
{}}). k * (line-integral F basis γ))
      using 0 by auto
    qed
  have ∧ (k::int) γ. (k, γ) ∈ one-chain1 ⇒ (f (k, γ) = { }) ⇒ (k, γ) ∈
{(k', γ'). k' * (line-integral F basis γ') = 0}
  proof-
    fix k::int
    fix γ::one-cube
    assume ass:(k, γ) ∈ one-chain1
      (f (k, γ) = { })
    then have zero-line-integral:one-chain-line-integral F basis (f (k, γ)) = 0
      using one-chain-line-integral-def
      by auto
    have bchain: boundary-chain (f(k,γ))
      using f-props(1) boundary-chain2 ass
      by (auto simp add: boundary-chain-def)
    have wexist: ∀ (k, γ) ∈ (f(k,γ)). line-integral-exists F basis γ
      using f-props(1) line-integral-exists-on-chain2 ass
      by blast
    have vpath: ∀ (k, γ) ∈ (f(k, γ)). valid-path γ
      using f-props(1) assms ass by blast
    have one-chain-line-integral F basis (f (k, γ)) = k * line-integral F basis
γ
  proof(cases k = 1)
    assume k-eq-1: k = 1
      have one-chain-line-integral F basis (f (k, γ)) = line-integral F basis γ
      using f-props(2) k-eq-1 line-integral-on-path-eq-line-integral-on-equiv-chain
bchain wexist vpath ass finite-basis
      by auto

```

```

then show one-chain-line-integral F basis (f (k, γ)) = k * line-integral
F basis γ
  using k-eq-1
  by auto
next
  assume k ≠ 1
  then have k-eq-neg1: k = -1
  using ass boundary-chain1
  by (auto simp add: boundary-chain-def)
  have one-chain-line-integral F basis (f (k, γ)) = line-integral F basis
(reversepath γ)
  using f-props(2) k-eq-neg1 line-integral-on-path-eq-line-integral-on-equiv-chain
bchain wexist vpath ass finite-basis
  by auto
  then have one-chain-line-integral F basis (f (k, γ)) = - (line-integral
F basis γ)
  using line-integral-on-reverse-path(1) ass line-integral-exists-on-chain1
valid-subdiv-imp-valid-one-chain[OF chain1-eq-chain2 boundary-chain1
boundary-chain2 valid-path]
  by force
  then show one-chain-line-integral F basis (f (k::int, γ)) = k * line-integral
F basis γ
  using k-eq-neg1 by auto
  qed
  then show (k, γ) ∈ {(k'::int, γ'). k' * line-integral F basis γ' = 0}
  using zero-line-integral by auto
  qed
  then have ii:(∑ one-chain ∈ (f ' (one-chain1 - {p. f p = {}})). one-chain-line-integral
F basis one-chain) =
(∑ one-chain ∈ (f ' (one-chain1)).
one-chain-line-integral F basis one-chain)
  proof -
  have  $\bigwedge one-chain. one-chain \in (f ' (one-chain1)) - (f ' (one-chain1 - \{p. f p = \{\}\})) \implies$ 
one-chain-line-integral F basis
one-chain = 0
  proof -
  fix one-chain
  assume one-chain ∈ (f ' (one-chain1)) - (f ' (one-chain1 - {p. f p =
{}}))
  then show one-chain-line-integral F basis one-chain = 0
  by (auto simp add: one-chain-line-integral-def)
  qed
  then have 0:(∑ one-chain ∈ f ' (one-chain1) - (f ' (one-chain1 - {p. f
p = {}})). one-chain-line-integral F basis one-chain
   $= 0$ 
  using comm-monoid-add-class.sum.neutral by auto
  then have ( $\sum one-chain \in f ' (one-chain1). one-chain-line-integral F$ 
basis one-chain)

```

$-\left(\sum \text{one-chain} \in (f \text{ ' } (\text{one-chain1} - \{p. f p = \{\}\}))\right).$
 $\text{one-chain-line-integral } F \text{ basis one-chain}$
 $= 0$

proof –

have *finte*: *finite* (f ' one-chain1) **using** *finite-chain1* **by** *auto*

show *?thesis*

using *Groups-Big.sum-diff*[*OF finte*, of (f ' (one-chain1 - {p. f p = {}}))

one-chain-line-integral F basis]

0

by *auto*

qed

then show $(\sum \text{one-chain} \in (f \text{ ' } (\text{one-chain1} - \{p. f p = \{\}\})).$
 $\text{one-chain-line-integral } F \text{ basis one-chain}) =$
 $(\sum \text{one-chain} \in (f \text{ ' } (\text{one-chain1})).$
 $\text{one-chain-line-integral } F \text{ basis one-chain})$

by *auto*

qed

have $\bigwedge (k::\text{int}) \gamma. (k, \gamma) \in \text{one-chain1} \implies (f(k, \gamma) = \{\}) \implies f(k, \gamma) \in$
 $\{\text{chain. one-chain-line-integral } F \text{ basis chain} = 0\}$

proof–

fix *k::int*

fix *γ::one-cube*

assume *ass*: $(k, \gamma) \in \text{one-chain1} (f(k, \gamma) = \{\})$

then have *one-chain-line-integral F basis* $(f(k, \gamma)) = 0$

using *one-chain-line-integral-def* **by** *auto*

then show $f(k, \gamma) \in \{p'. (\text{one-chain-line-integral } F \text{ basis } p' = 0)\}$

by *auto*

qed

then have *iii*: $(\sum (k::\text{int}, \gamma) \in \text{one-chain1} - \{p. f p = \{\}\}). k * (\text{line-integral } F$
basis $\gamma))$
 $= (\sum (k::\text{int}, \gamma) \in \text{one-chain1}. k * (\text{line-integral}$
 $F \text{ basis } \gamma))$

proof–

have $\bigwedge k \gamma. (k, \gamma) \in \text{one-chain1} - (\text{one-chain1} - \{p. f p = \{\}\})$
 $\implies k * (\text{line-integral } F \text{ basis } \gamma) = 0$

proof–

fix *k γ*

assume *ass*: $(k, \gamma) \in \text{one-chain1} - (\text{one-chain1} - \{p. f p = \{\}\})$

then have $f(k, \gamma) = \{\}$

by *auto*

then have *one-chain-line-integral F basis* $(f(k, \gamma)) = 0$

by (*auto simp add: one-chain-line-integral-def*)

then have *zero-line-integral: one-chain-line-integral F basis* $(f(k, \gamma)) =$
0

using *one-chain-line-integral-def* **by** *auto*

have *bchain*: *boundary-chain* $(f(k, \gamma))$

using *f-props(1) boundary-chain2 ass*

by (*auto simp add: boundary-chain-def*)

```

have wexist:  $\forall (k, \gamma) \in (f(k, \gamma)). \text{line-integral-exists } F \text{ basis } \gamma$ 
  using f-props(1) line-integral-exists-on-chain2 ass
  by blast
have vpath:  $\forall (k, \gamma) \in (f(k, \gamma)). \text{valid-path } \gamma$ 
  using f-props(1) assms ass
  by blast
have one-chain-line-integral F basis  $(f(k, \gamma)) = k * \text{line-integral } F \text{ basis}$ 
 $\gamma$ 
  proof(cases k = 1)
    assume k-eq-1:  $k = 1$ 
    have one-chain-line-integral F basis  $(f(k, \gamma)) = \text{line-integral } F \text{ basis } \gamma$ 
    using f-props(2) k-eq-1 line-integral-on-path-eq-line-integral-on-equiv-chain
bchain wexist vpath ass finite-basis
    by auto
    then show one-chain-line-integral F basis  $(f(k, \gamma)) = k * \text{line-integral}$ 
F basis  $\gamma$ 
      using k-eq-1
      by auto
    next
      assume  $k \neq 1$ 
      then have k-eq-neg1:  $k = -1$ 
        using ass boundary-chain1
        by (auto simp add: boundary-chain-def)
        have one-chain-line-integral F basis  $(f(k, \gamma)) = \text{line-integral } F \text{ basis}$ 
(reversepath  $\gamma)$ 
          using f-props(2) k-eq-neg1 line-integral-on-path-eq-line-integral-on-equiv-chain
bchain wexist vpath ass finite-basis
          by auto
          then have one-chain-line-integral F basis  $(f(k, \gamma)) = - (\text{line-integral}$ 
F basis  $\gamma)$ 
            using line-integral-on-reverse-path(1) ass line-integral-exists-on-chain1
valid-subdiv-imp-valid-one-chain[OF chain1-eq-chain2 boundary-chain1
boundary-chain2 valid-path]
            by force
            then show one-chain-line-integral F basis  $(f(k, \gamma)) = k * \text{line-integral}$ 
F basis  $\gamma$ 
              using k-eq-neg1
              by auto
          qed
          then show  $k * (\text{line-integral } F \text{ basis } \gamma) = 0$ 
            using zero-line-integral
            by auto
          qed
          then have  $\forall (k::\text{int}, \gamma) \in \text{one-chain1} - \{p. f p = \{\}\}. k * (\text{line-integral } F \text{ basis } \gamma) = 0$  by auto
          then have  $(\sum (k::\text{int}, \gamma) \in \text{one-chain1} - \{p. f p = \{\}\}). k * (\text{line-integral } F \text{ basis } \gamma) = 0$ 
            using Groups-Big.comm-monoid-add-class.sum.neutral
            [of one-chain1 - (one-chain1 - {p. f p = {}}) (\lambda(k::int, \gamma)). k*

```

```

(line-integral F basis  $\gamma$ )]
  by (simp add: split-beta)
  then have  $(\sum_{(k::int,\gamma)\in one-chain1}. k*(line-integral F basis \gamma)) -$ 
 $(\sum_{(k::int,\gamma)\in (one-chain1 - \{p. f p = \{\}\})}. k*(line-integral F$ 
basis  $\gamma)) = 0$ 
    using Groups-Big.sum-diff[OF finite-chain1]
    by (metis (no-types) Diff-subset  $(\sum_{(k,\gamma)\in one-chain1} - (\sum_{(k,\gamma)\in (one-chain1 - \{p. f p = \{\}\})} k * line-integral F basis \gamma) = 0)$   $(\bigwedge B. B \subseteq one-chain1 \implies$ 
 $sum f (one-chain1 - B) = sum f one-chain1 - sum f B)$ )
    then show ?thesis by auto
  qed
  show ?thesis using i ii iii by auto
  qed
  then show ?thesis using one-chain-line-integral-def by auto
  qed
  show ?thesis using 0 1 by auto
  qed
  show one-chain-line-integral F basis one-chain1 = one-chain-line-integral F basis
subdiv using 0 1 by auto
  qed

```

lemma *one-chain-line-integral-eq-line-integral-on-subdivision'*:

```

  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:  $\forall (k, \gamma) \in one-chain1. line-integral-exists F basis$ 
 $\gamma$  and
    valid-path:  $\forall (k, \gamma) \in subdiv. valid-path \gamma$  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
   $\forall (k, \gamma) \in subdiv. line-integral-exists F basis \gamma$ 
  proof -
    obtain f where f-props:
       $((\bigcup (f ` one-chain1)) = subdiv)$ 
       $(\forall (k,\gamma)\in one-chain1. if k = 1 then chain-subdiv-path \gamma (f(k,\gamma)) else chain-subdiv-path$ 
       $(reversepath \gamma) (f(k,\gamma)))$ 
       $(\forall p\in one-chain1. \forall p'\in one-chain1. p \neq p' \implies f p \cap f p' = \{\})$ 
       $(\forall x \in one-chain1. finite (f x))$ 
    using chain1-eq-chain2 chain-subdiv-chain-character by (auto simp add: pairwise-def
chain-subdiv-chain-def)
    have finite-chain2: finite subdiv
      using finite-chain1 f-props(1) f-props(4) by blast
    have  $\bigwedge k \gamma. (k, \gamma) \in subdiv \implies line-integral-exists F basis \gamma$ 
    proof -
      fix k  $\gamma$ 
      assume ass:  $(k, \gamma) \in subdiv$ 
      then obtain  $k' \gamma'$  where kp-gammap:  $(k',\gamma') \in one-chain1 (k,\gamma) \in f(k',\gamma')$ 

```

```

    using f-props(1) by fastforce
  show line-integral-exists F basis  $\gamma$ 
  proof (cases  $k' = 1$ )
    assume k-eq-1:  $k' = 1$ 
    then have i: chain-subdiv-path  $\gamma'$  (f( $k', \gamma'$ ))
      using f-props(2) kp-gammap ass by auto
    have ii: boundary-chain (f( $k', \gamma'$ ))
    using f-props(1) ass assms kp-gammap by (meson UN-I boundary-chain-def)
    have iii: line-integral-exists F basis  $\gamma'$ 
      using assms kp-gammap by blast
    have iv:  $\forall (k, \gamma) \in f(k', \gamma'). \text{ valid-path } \gamma$ 
      using f-props(1) ass assms kp-gammap by blast
    show ?thesis
      using line-integral-on-path-eq-line-integral-on-equiv-chain'(2)[OF i ii iii iv
finite-basis] kp-gammap
      by auto
  next
    assume  $k' \neq 1$ 
    then have k-eq-neg1:  $k' = -1$ 
      using boundary-chain1 kp-gammap
      by (auto simp add: boundary-chain-def)
    then have i: chain-subdiv-path (reversepath  $\gamma'$ ) (f( $k', \gamma'$ ))
      using f-props(2) kp-gammap by auto
    have ii: boundary-chain (f( $k', \gamma'$ ))
      using f-props(1) assms kp-gammap by (meson UN-I boundary-chain-def)
    have iii:  $\forall (k, \gamma) \in f(k', \gamma'). \text{ valid-path } \gamma$ 
      using f-props(1) ass assms kp-gammap by blast
    have iv: valid-path (reversepath  $\gamma'$ )
      using valid-path-equiv-valid-chain-list[OF i ii iii]
      by force
    have line-integral-exists F basis  $\gamma'$ 
      using assms kp-gammap by blast
    then have x: line-integral-exists F basis (reversepath  $\gamma'$ )
      using iv line-integral-on-reverse-path(2) valid-path-reversepath
      by fastforce
    show ?thesis
      using line-integral-on-path-eq-line-integral-on-equiv-chain'(2)[OF i ii x iii
finite-basis] kp-gammap
      by auto
  qed
  qed
  then show  $\forall (k, \gamma) \in \text{subdiv}. \text{ line-integral-exists F basis } \gamma$  by auto
  then show one-chain-line-integral F basis one-chain1 = one-chain-line-integral
F basis subdiv
    using one-chain-line-integral-eq-line-integral-on-sudivision(1) assms
    by auto
  qed

```

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:
 $\text{basis1} \cup \text{basis2} = \text{basis}$ $\text{basis1} \cap \text{basis2} = \{\}$
shows *line-integral* F *basis* $\gamma = (\text{line-integral } F \text{ } \text{basis1} \text{ } \gamma) + (\text{line-integral } F$
basis2 $\gamma)$
line-integral-exists F *basis* γ
apply (*simp add: line-integral-def*)
proof –
have 0 : *integral* $\{0..1\} (\lambda x. (\sum b \in \text{basis1}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b)) +$
 $(\sum b \in \text{basis2}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b))) =$
 $\text{integral } \{0..1\} (\lambda x. \sum b \in \text{basis1}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b)) +$
 $\text{integral } \{0..1\} (\lambda x. \sum b \in \text{basis2}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b))$
using *Henstock-Kurzweil-Integration.integral-add line-integral-exists*
by (*auto simp add: line-integral-exists-def*)
have 1 : *integral* $\{0..1\} (\lambda x. \sum b \in \text{basis}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b)) =$
 $\text{integral } \{0..1\} (\lambda x. (\sum b \in \text{basis1}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b)) +$
 $(\sum b \in \text{basis2}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b)))$
by (*metis (mono-tags, lifting) basis-partition finite-Un finite-basis sum.union-disjoint*)
show *integral* $\{0..1\} (\lambda x. \sum b \in \text{basis}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b)) =$
 $\text{integral } \{0..1\} (\lambda x. \sum b \in \text{basis1}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b)) +$
 $\text{integral } \{0..1\} (\lambda x. \sum b \in \text{basis2}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b))$
using $0\ 1$
by *auto*
have 2 : $((\lambda x. (\sum b \in \text{basis1}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b)) +$
 $(\sum b \in \text{basis2}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b))) \text{ has-integral}$
 $\text{integral } \{0..1\} (\lambda x. \sum b \in \text{basis1}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b)) +$
 $\text{integral } \{0..1\} (\lambda x. \sum b \in \text{basis2}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\}) \cdot b))$
using *Henstock-Kurzweil-Integration.has-integral-add line-integral-exists has-integral-integral*
apply (*auto simp add: line-integral-exists-def*)
by *blast*
have 3 : $(\lambda x. \sum b \in \text{basis}. F (\gamma x) \cdot b * (\text{vector-derivative } \gamma (\text{at } x \text{ within } \{0..1\})$

$\cdot b)) =$
 $(\lambda x. (\sum b \in \text{basis1}. F(\gamma x) \cdot b * (\text{vector-derivative } \gamma \text{ (at } x \text{ within } \{0..1\}) \cdot b)) +$
 $(\sum b \in \text{basis2}. F(\gamma x) \cdot b * (\text{vector-derivative } \gamma$
 $(\text{at } x \text{ within } \{0..1\}) \cdot b)))$
by (*metis (mono-tags, lifting) basis-partition finite-Un finite-basis sum.union-disjoint*)
show *line-integral-exists F basis γ*
apply (*auto simp add: line-integral-exists-def has-integral-integral*)
using 2 3
using *has-integral-integrable-integral* **by** *fastforce*
qed

definition *common-boundary-sudivision-exists* **where**
common-boundary-sudivision-exists one-chain1 one-chain2 \equiv
 $\exists \text{ subdiv. chain-subdiv-chain one-chain1 subdiv} \wedge$
 $\text{chain-subdiv-chain one-chain2 subdiv} \wedge$
 $(\forall (k, \gamma) \in \text{subdiv. valid-path } \gamma) \wedge$
 $\text{boundary-chain subdiv}$

lemma *common-boundary-sudivision-commutative:*
 $(\text{common-boundary-sudivision-exists one-chain1 one-chain2}) = (\text{common-boundary-sudivision-exists one-chain2 one-chain1})$
apply (*simp add: common-boundary-sudivision-exists-def*)
by *blast*

lemma *common-sudivision-imp-eq-line-integral:*
assumes (*common-boundary-sudivision-exists one-chain1 one-chain2*)
boundary-chain one-chain1
boundary-chain one-chain2
 $\forall (k, \gamma) \in \text{one-chain1. line-integral-exists } F \text{ basis } \gamma$
finite one-chain1
finite one-chain2
finite basis
shows $\text{one-chain-line-integral } F \text{ basis one-chain1} = \text{one-chain-line-integral } F$
basis one-chain2
 $\forall (k, \gamma) \in \text{one-chain2. line-integral-exists } F \text{ basis } \gamma$

proof –
obtain *subdiv* **where** *subdiv-props:*
chain-subdiv-chain one-chain1 subdiv
chain-subdiv-chain one-chain2 subdiv
 $(\forall (k, \gamma) \in \text{subdiv. valid-path } \gamma)$
(boundary-chain subdiv)
using *assms*
by (*auto simp add: common-boundary-sudivision-exists-def*)
have *i:* $\forall (k, \gamma) \in \text{subdiv. line-integral-exists } F \text{ basis } \gamma$
using *one-chain-line-integral-eq-line-integral-on-sudivision'(2)[OF subdiv-props(1) assms(2) subdiv-props(4) assms(4) subdiv-props(3) assms(5) assms(7)]*
by *auto*
show $\text{one-chain-line-integral } F \text{ basis one-chain1} = \text{one-chain-line-integral } F \text{ basis one-chain2}$

one-chain2
using *one-chain-line-integral-eq-line-integral-on-sudivision'(1)*[*OF subdiv-props(1)*
assms(2) subdiv-props(4) assms(4) subdiv-props(3) assms(5) assms(7)]
one-chain-line-integral-eq-line-integral-on-sudivision(1)[*OF subdiv-props(2)*
assms(3) subdiv-props(4) i subdiv-props(3) assms(6) assms(7)]
by auto
show $\forall (k, \gamma) \in \text{one-chain2}. \text{line-integral-exists } F \text{ basis } \gamma$
using *one-chain-line-integral-eq-line-integral-on-sudivision(2)*[*OF subdiv-props(2)*
assms(3) subdiv-props(4) i subdiv-props(3) assms(6) assms(7)]
by auto
qed

definition *common-sudiv-exists where*

common-sudiv-exists one-chain1 one-chain2 \equiv
 $\exists \text{subdiv } ps1 \text{ } ps2. \text{chain-subdiv-chain } (\text{one-chain1} - ps1) \text{ subdiv} \wedge$
 $\text{chain-subdiv-chain } (\text{one-chain2} - ps2) \text{ subdiv} \wedge$
 $(\forall (k, \gamma) \in \text{subdiv}. \text{valid-path } \gamma) \wedge$
 $(\text{boundary-chain subdiv}) \wedge$
 $(\forall (k, \gamma) \in ps1. \text{point-path } \gamma) \wedge$
 $(\forall (k, \gamma) \in ps2. \text{point-path } \gamma)$

lemma *common-sudiv-exists-comm:*

shows *common-sudiv-exists C1 C2 = common-sudiv-exists C2 C1*
by (*auto simp add: common-sudiv-exists-def*)

lemma *line-integral-degenerate-chain:*

assumes $(\forall (k, \gamma) \in \text{chain}. \text{point-path } \gamma)$
assumes *finite basis*
shows *one-chain-line-integral F basis chain = 0*
proof (*simp add: one-chain-line-integral-def*)
have $\forall (k, g) \in \text{chain}. \text{line-integral } F \text{ basis } g = 0$
using *assms line-integral-point-path*
by blast
then have $\forall (k, g) \in \text{chain}. \text{real-of-int } k * \text{line-integral } F \text{ basis } g = 0$ **by auto**
then have $\bigwedge p. p \in \text{chain} \implies (\text{case } p \text{ of } (i, f) \Rightarrow \text{real-of-int } i * \text{line-integral } F$
basis f) = 0
by fastforce
then show $(\sum x \in \text{chain}. \text{case } x \text{ of } (k, g) \Rightarrow \text{real-of-int } k * \text{line-integral } F \text{ basis } g) = 0$
by simp
qed

lemma *gen-common-subdiv-imp-common-subdiv:*

shows $(\text{common-sudiv-exists one-chain1 one-chain2}) = (\exists ps1 \text{ } ps2. (\text{common-boundary-sudivision-exists}$
 $(\text{one-chain1} - ps1) (\text{one-chain2} - ps2)) \wedge (\forall (k, \gamma) \in ps1. \text{point-path } \gamma) \wedge (\forall (k,$
 $\gamma) \in ps2. \text{point-path } \gamma))$
by (*auto simp add: common-sudiv-exists-def common-boundary-sudivision-exists-def*)

lemma *common-subdiv-imp-gen-common-subdiv:*

assumes (*common-boundary-sudivision-exists one-chain1 one-chain2*)
shows (*common-sudiv-exists one-chain1 one-chain2*)
using *assms*
apply (*auto simp add: common-sudiv-exists-def common-boundary-sudivision-exists-def*)
by (*metis Diff-empty all-not-in-conv*)

lemma *one-chain-line-integral-point-paths:*

assumes *finite one-chain*
assumes *finite basis*
assumes $(\forall (k, \gamma) \in ps. \text{point-path } \gamma)$
shows *one-chain-line-integral F basis (one-chain - ps) = one-chain-line-integral F basis (one-chain)*
proof –
have $0: (\forall x \in ps. \text{case } x \text{ of } (k, g) \Rightarrow (\text{real-of-int } k * \text{line-integral } F \text{ basis } g) = 0)$
using *line-integral-point-path assms*
by *force*
show *one-chain-line-integral F basis (one-chain - ps) = one-chain-line-integral F basis one-chain*
unfolding *one-chain-line-integral-def* **using** 0 *(finite one-chain)*
by (*force simp add: intro: comm-monoid-add-class.sum.mono-neutral-left*)
qed

lemma *boundary-chain-diff:*

assumes *boundary-chain one-chain*
shows *boundary-chain (one-chain - s)*
using *assms*
by (*auto simp add: boundary-chain-def*)

lemma *gen-common-subdivision-imp-eq-line-integral:*

assumes (*common-sudiv-exists one-chain1 one-chain2*)
boundary-chain one-chain1
boundary-chain one-chain2
 $\forall (k, \gamma) \in \text{one-chain1}. \text{line-integral-exists } F \text{ basis } \gamma$
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*
 $\forall (k, \gamma) \in \text{one-chain2}. \text{line-integral-exists } F \text{ basis } \gamma$
proof –
obtain *ps1 ps2* **where** *gen-subdiv: (common-boundary-sudivision-exists (one-chain1 - ps1) (one-chain2 - ps2))* $(\forall (k, \gamma) \in ps1. \text{point-path } \gamma)$ $(\forall (k, \gamma) \in ps2. \text{point-path } \gamma)$
using *assms(1) gen-common-subdiv-imp-common-subdiv*
by *blast*
show *one-chain-line-integral F basis one-chain1 = one-chain-line-integral F basis one-chain2*
using *one-chain-line-integral-point-paths gen-common-subdiv-imp-common-subdiv assms(2-7) gen-subdiv*

```

    common-subdivision-imp-eq-line-integral(1)[OF gen-subdiv(1) boundary-chain-diff[OF
assms(2)] boundary-chain-diff[OF assms(3)]]
  by auto
  show  $\forall (k, \gamma) \in \text{one-chain2}. \text{line-integral-exists } F \text{ basis } \gamma$ 
  proof -
    obtain subdiv where subdiv-props:
      chain-subdiv-chain (one-chain1-ps1) subdiv
      chain-subdiv-chain (one-chain2-ps2) subdiv
      ( $\forall (k, \gamma) \in \text{subdiv}. \text{valid-path } \gamma$ )
      (boundary-chain subdiv)
    using gen-subdiv(1)
    by (auto simp add: common-boundary-sudivision-exists-def)
    have  $\forall (k, \gamma) \in \text{subdiv}. \text{line-integral-exists } F \text{ basis } \gamma$ 
    using one-chain-line-integral-eq-line-integral-on-sudivision'(2)[OF subdiv-props(1)
boundary-chain-diff[OF assms(2)] subdiv-props(4)] assms(4) subdiv-props(3) assms(5)
assms(7)
      by blast
    then have  $i: \forall (k, \gamma) \in \text{one-chain2-ps2}. \text{line-integral-exists } F \text{ basis } \gamma$ 
    using one-chain-line-integral-eq-line-integral-on-sudivision(2)[OF subdiv-props(2)
boundary-chain-diff[OF assms(3)] subdiv-props(4)] subdiv-props(3) assms(6) assms(7)
      by blast
    then show ?thesis
      using gen-subdiv(3) line-integral-exists-point-path[OF assms(7)]
      by blast
  qed
qed

```

```

lemma common-sudiv-exists-reft:
  assumes common-sudiv-exists C1 C2
  shows common-sudiv-exists C2 C1
  using assms
  apply (simp add: common-sudiv-exists-def)
  by auto

```

```

lemma chain-subdiv-path-singleton:
  shows chain-subdiv-path  $\gamma \{(1, \gamma)\}$ 
  proof -
    have rec-join [(1,  $\gamma$ )] =  $\gamma$ 
      by (simp add: joinpaths-def)
    then have set [(1,  $\gamma$ )] = {(1,  $\gamma$ )} distinct [(1,  $\gamma$ )] rec-join [(1,  $\gamma$ )] =  $\gamma$  valid-chain-list
      [(1,  $\gamma$ )]
      by auto
    then show ?thesis
      by (metis (no-types) chain-subdiv-path.intros)
  qed

```

```

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

```

have *rec-join* $[(-1, \gamma)] = \text{reversepath } \gamma$
by (*simp add: joinpaths-def*)
then have *set* $[(-1, \gamma)] = \{(-1, \gamma)\}$ *distinct* $[(-1, \gamma)]$
rec-join $[(-1, \gamma)] = \text{reversepath } \gamma$ *valid-chain-list* $[(-1, \gamma)]$
by *auto*
then have *chain-subdiv-path* (*reversepath* γ) (*set* $[(-1, \gamma)]$)
using *chain-subdiv-path.intros* **by** *blast*
then show *?thesis*
by *simp*
qed

lemma *chain-subdiv-chain-refl*:
assumes *boundary-chain* C
shows *chain-subdiv-chain* $C C$
using *chain-subdiv-path-singleton chain-subdiv-path-singleton-reverse assms*
unfolding *chain-subdiv-chain-def boundary-chain-def pairwise-def* **using** *case-prodI2*
coeff-cube-to-path.simps
by (*rule-tac* $x = \lambda x. \{x\}$ **in** *exI*) *auto*

definition *reparam-weak* **where**

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

definition *reparam* **where**

reparam $\gamma 1 \gamma 2 \equiv \exists \varphi. (\forall x \in \{0..1\}. \gamma 1 x = (\gamma 2 \circ \varphi) x) \wedge \varphi$ *piecewise-C1-differentiable-on*
 $\{0..1\} \wedge \varphi(0) = 0 \wedge \varphi(1) = 1 \wedge \text{bij-betw } \varphi \{0..1\} \{0..1\} \wedge \varphi^{-1} \{0..1\} \subseteq \{0..1\}$
 $\wedge (\forall x \in \{0..1\}. \text{finite } (\varphi^{-1} \{x\}))$

lemma *reparam-weak-eq-refl*:

shows *reparam-weak* $\gamma 1 \gamma 1$
unfolding *reparam-weak-def*
apply (*rule-tac* $x = \text{id}$ **in** *exI*)
by (*auto simp add: id-def piecewise-C1-differentiable-on-def C1-differentiable-on-def*
continuous-on-id)

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* $F \{b\} \gamma$

proof –

have *gamma2-differentiable*: $(\forall x \in \{0..1\}. \gamma$ *differentiable at* $x)$
using *assms(1)*
by (*auto simp add: valid-path-def C1-differentiable-on-eq*)
then have *gamma2-b-component-differentiable*: $(\forall x \in \{0..1\}. (\lambda x. (\gamma x) \cdot b)$
differentiable at $x)$
by *auto*
then have $(\lambda x. (\gamma x) \cdot b)$ *differentiable-on* $\{0..1\}$

```

    using differentiable-at-withinI
    by (auto simp add: differentiable-on-def)
  then have gamma2-cont-comp: continuous-on {0..1} (λx. (γ x) · b)
    using differentiable-imp-continuous-on
    by auto
  have gamma2-cont:continuous-on {0..1} γ
    using assms(1) C1-differentiable-imp-continuous-on
    by (auto simp add: valid-path-def)
  have iii: continuous-on {0..1} (λx. F (γ x) · b * (vector-derivative γ (at x within
{0..1}) · b))
  proof-
    have 0: continuous-on {0..1} (λx. F (γ x) · b)
      using assms(2) continuous-on-compose[OF gamma2-cont]
      by (auto simp add: path-image-def)
    obtain D where D: (∀ x∈{0..1}. (γ has-vector-derivative D x) (at x)) ∧
continuous-on {0..1} D
      using assms(1)
      by (auto simp add: C1-differentiable-on-def)
    then have *:∀ x∈{0..1}. vector-derivative γ (at x within{0..1}) = D x
      using vector-derivative-at vector-derivative-at-within-ivl
      by fastforce
    then have continuous-on {0..1} (λx. vector-derivative γ (at x within{0..1}))
      using continuous-on-eq D by force
    then have 1: continuous-on {0..1} (λx. (vector-derivative γ (at x within{0..1}))
· b)
      by(auto intro!: continuous-intros)
    show ?thesis
      using continuous-on-mult[OF 0 1] by auto
  qed
  then have (λx. F (γ x) · b * (vector-derivative γ (at x within {0..1}) · b))
integrable-on {0..1}
    using integrable-continuous-real
    by auto
  then show line-integral-exists F {b} γ
    by(auto simp add: line-integral-exists-def)
qed

```

lemma *contour-integral-primitive-lemma:*

fixes $f :: \text{complex} \Rightarrow \text{complex}$ **and** $g :: \text{real} \Rightarrow \text{complex}$

assumes $a \leq b$

and $\bigwedge x. x \in s \implies (f \text{ has-field-derivative } f' x) (at x \text{ within } s)$

and $g \text{ piecewise-differentiable-on } \{a..b\} \bigwedge x. x \in \{a..b\} \implies g x \in s$

shows $((\lambda x. f'(g x) * \text{vector-derivative } g (at x \text{ within } \{a..b\}))$
 $\text{has-integral } (f(g b) - f(g a))) \{a..b\}$

proof –

obtain k **where** k : finite $k \forall x \in \{a..b\} - k. g \text{ differentiable } (at x \text{ within } \{a..b\})$

and cg : continuous-on $\{a..b\}$ g

using $assms$ **by** (auto simp: piecewise-differentiable-on-def)

have cfg : continuous-on $\{a..b\}$ $(\lambda x. f (g x))$

```

apply (rule continuous-on-compose [OF cg, unfolded o-def])
using assms
apply (metis field-differentiable-def field-differentiable-imp-continuous-at continuous-on-eq-continuous-within
continuous-on-subset image-subset-iff)
done
{ fix x::real
  assume a: a < x and b: x < b and xk: x ∉ k
  then have g differentiable at x within {a..b}
    using k by (simp add: differentiable-at-withinI)
  then have (g has-vector-derivative vector-derivative g (at x within {a..b})) (at
x within {a..b})
    by (simp add: vector-derivative-works has-field-derivative-def scaleR-conv-of-real)
  then have gdiff: (g has-derivative (λu. u * vector-derivative g (at x within
{a..b}))) (at x within {a..b})
    by (simp add: has-vector-derivative-def scaleR-conv-of-real)
  have (f has-field-derivative (f' (g x)) (at (g x) within g ' {a..b}))
    using assms by (metis a atLeastAtMost-iff b DERIV-subset image-subset-iff
less-eq-real-def)
  then have fdiff: (f has-derivative (*) (f' (g x)) (at (g x) within g ' {a..b})
    by (simp add: has-field-derivative-def)
  have ((λx. f (g x)) has-vector-derivative f' (g x) * vector-derivative g (at x
within {a..b})) (at x within {a..b})
    using diff-chain-within [OF gdiff fdiff]
    by (simp add: has-vector-derivative-def scaleR-conv-of-real o-def mult-ac)
} note * = this
show ?thesis
apply (rule fundamental-theorem-of-calculus-interior-strong)
using k assms cfg *
apply (auto simp: at-within-Icc-at)
done

```

qed

lemma *line-integral-primitive-lemma:*

fixes *f :: 'a::{euclidean-space,real-normed-field} ⇒ 'a::{euclidean-space,real-normed-field}*
and

g :: real ⇒ 'a

assumes $\bigwedge(a::'a). a \in s \implies (f \text{ has-field-derivative } (f' a)) \text{ (at } a \text{ within } s)$

and *g piecewise-differentiable-on {0..1} ∧ x. x ∈ {0..1} ⇒ g x ∈ s*

and *base-vec ∈ Basis*

shows $((\lambda x. ((f'(g x)) * (\text{vector-derivative } g \text{ (at } x \text{ within } \{0..1\}))) \cdot \text{base-vec})$
 $\text{has-integral } (((f(g 1)) \cdot \text{base-vec} - (f(g 0)) \cdot \text{base-vec})) \{0..1\}$

proof –

obtain *k* **where** *k: finite k ∀ x ∈ {0..1} – k. g differentiable (at x within {0..1})*

and *cg: continuous-on {0..1} g*

using *assms* **by** (auto simp: piecewise-differentiable-on-def)

have *cfg: continuous-on {0..1} (λx. f (g x))*

apply (rule continuous-on-compose [OF cg, unfolded o-def])

using *assms*

apply (metis field-differentiable-def field-differentiable-imp-continuous-at continuous-on-eq-continuous-within)

continuous-on-subset image-subset-iff)

done

{ **fix** $x::\text{real}$

assume $a: 0 < x$ **and** $b: x < 1$ **and** $xk: x \notin k$

then have g *differentiable at x within $\{0..1\}$*

using k **by** (*simp add: differentiable-at-withinI*)

then have (g *has-vector-derivative vector-derivative g (at x within $\{0..1\})$*) (at x within $\{0..1\}$)

by (*simp add: vector-derivative-works has-field-derivative-def scaleR-conv-of-real*)

then have $gdiff: (g$ *has-derivative $(\lambda u. \text{of-real } u * \text{vector-derivative } g \text{ (at } x \text{ within } \{0..1\}))$*) (at x within $\{0..1\}$)

by (*simp add: has-vector-derivative-def scaleR-conv-of-real*)

have (f *has-field-derivative $(f' (g x))$*) (at $(g x)$ within $g^{-1} \{0..1\}$)

using *assms* **by** (*metis a atLeastAtMost-iff b DERIV-subset image-subset-iff less-eq-real-def*)

then have $fdiff: (f$ *has-derivative $(*) (f' (g x))$*) (at $(g x)$ within $g^{-1} \{0..1\}$)

by (*simp add: has-field-derivative-def*)

have ($(\lambda x. f (g x))$ *has-vector-derivative $f' (g x) * \text{vector-derivative } g \text{ (at } x \text{ within } \{0..1\})$*) (at x within $\{0..1\}$)

using *diff-chain-within [OF gdiff fdiff]*

by (*simp add: has-vector-derivative-def scaleR-conv-of-real o-def mult-ac*)

 }

then have $*$: $\bigwedge x. x \in \{0 < .. < 1\} - k \implies ((\lambda x. f (g x))$ *has-vector-derivative $f' (g x) * \text{vector-derivative } g \text{ (at } x \text{ within } \{0..1\})$*) (at x within $\{0..1\}$)

by *auto*

have ($(\lambda x. ((f'(g x)) * ((\text{vector-derivative } g \text{ (at } x \text{ within } \{0..1\}))))$ *has-integral $((f(g 1)) - (f(g 0))) \{0..1\}$*)

using *fundamental-theorem-of-calculus-interior-strong[OF k(1) zero-le-one - cfg]*

using k *assms* cfg **by** (*auto simp: at-within-Icc-at*)

then have ($(\lambda x. (((f'(g x)) * ((\text{vector-derivative } g \text{ (at } x \text{ within } \{0..1\})))) * \text{base-vec})$ *has-integral $((f(g 1)) - (f(g 0))) * \text{base-vec} \{0..1\}$*)

using *has-integral-componentwise-iff assms(4)*

by *fastforce*

then show *?thesis* **using** *inner-mult-left'*

by (*simp add: inner-mult-left' inner-diff-left*)

qed

lemma *reparam-eq-line-integrals:*

assumes *reparam: reparam $\gamma 1 \gamma 2$ and*

pw-smooth: $\gamma 2$ piecewise-C1-differentiable-on $\{0..1\}$ and

cont: continuous-on (path-image $\gamma 2$) $(\lambda x. F x \cdot b)$ and

line-integral-ex: line-integral-exists $F \{b\} \gamma 2$

shows *line-integral $F \{b\} \gamma 1 = \text{line-integral } F \{b\} \gamma 2$*

line-integral-exists $F \{b\} \gamma 1$

proof –

obtain φ **where** *phi: $(\forall x \in \{0..1\}. \gamma 1 x = (\gamma 2 \circ \varphi) x)$ φ piecewise-C1-differentiable-on $\{0..1\}$ $\varphi(0) = 0$ $\varphi(1) = 1$ *bij-betw $\varphi \{0..1\} \{0..1\}$ $\varphi^{-1} \{0..1\} \subseteq \{0..1\} \forall x \in \{0..1\}$.**

```

finite (φ - ' {x})
  using reparam
  by (auto simp add: reparam-def)
obtain s where s: finite s φ C1-differentiable-on {0..1} - s
  using phi
  by(auto simp add: reparam-def piecewise-C1-differentiable-on-def)
let ?s = s ∩ {0..1}
have s-inter: finite ?s φ C1-differentiable-on {0..1} - ?s
  using s
  apply blast
  by (metis Diff-Compl Diff-Diff-Int Diff-eq inf commute s(2))
have cont-phi: continuous-on {0..1} φ
  using phi
  by(auto simp add: reparam-def piecewise-C1-differentiable-on-imp-continuous-on)
obtain s' D where s'-D: finite s' (∀ x ∈ {0 .. 1} - s'. γ2 differentiable at x)
(∀ x ∈ {0..1} - s'. (γ2 has-vector-derivative D x) (at x)) ∧ continuous-on ({0..1}
- s') D
  using pw-smooth
  apply (auto simp add: valid-path-def piecewise-C1-differentiable-on-def C1-differentiable-on-eq)
  by (simp add: vector-derivative-works)
let ?s' = s' ∩ {0..1}
have gamma2-differentiable: finite ?s' (∀ x ∈ {0 .. 1} - ?s'. γ2 differentiable
at x) (∀ x ∈ {0..1} - ?s'. (γ2 has-vector-derivative D x) (at x)) ∧ continuous-on
({0..1} - ?s') D
  using s'-D
  apply blast
  using s'-D(2) apply auto[1]
  by (metis Diff-Int2 inf-top.left-neutral s'-D(3))
then have gamma2-b-component-differentiable: (∀ x ∈ {0 .. 1} - ?s'. (λx. (γ2
x) · b) differentiable at x)
  using differentiable-inner by force
then have (λx. (γ2 x) · b) differentiable-on {0..1} - ?s'
  using differentiable-at-withinI
  by (auto simp add: differentiable-on-def)
then have gama2-cont-comp: continuous-on ({0..1} - ?s') (λx. (γ2 x) · b)
  using differentiable-imp-continuous-on
  by auto

have s-in01: ?s ⊆ {0..1} by auto
have s'-in01: ?s' ⊆ {0..1} by auto
have phi-backimg-s': φ - ' {0..1} ⊆ {0..1} using phi by auto
have inj-on φ {0..1} using phi(5) by (auto simp add: bij-betw-def)
have bij-phi: bij-betw φ {0..1} {0..1} using phi(5) by auto
have finite-bck-img-single: ∀ x ∈ {0..1}. finite (φ - ' {x}) using phi by auto
then have finite-bck-img-single-s': ∀ x ∈ ?s'. finite (φ - ' {x}) by auto
have gamma2-line-integrable: (λx. F (γ2 x) · b * (vector-derivative γ2 (at x
within {0..1}) · b)) integrable-on {0..1}
  using line-integral-ex
  by (simp add: line-integral-exists-def)

```

```

have finite-neg-img: finite ( $\varphi - \cdot ?s'$ )
  using finite-bck-img-single
  by (metis Int-iff finite-Int gamma2-differentiable(1) image-vimage-eq inf-img-fin-dom')
have gamma2-cont:continuous-on ( $\{0..1\} - ?s'$ )  $\gamma 2$ 
  using gamma2-differentiable
  by (meson continuous-at-imp-continuous-on differentiable-imp-continuous-within)
have iii: continuous-on ( $\{0..1\} - ?s'$ ) ( $\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2$ 
(at  $x$  within  $\{0..1\}) \cdot b)$ )
proof-
  have 0: continuous-on ( $\{0..1\} - ?s'$ ) ( $\lambda x. F (\gamma 2 x) \cdot b$ )
    using cont continuous-on-compose[OF gamma2-cont] continuous-on-compose2
gamma2-cont
    unfolding path-image-def by fastforce
  have D: ( $\forall x \in \{0..1\} - ?s'. (\gamma 2 \text{ has-vector-derivative } D x) (at x)) \wedge \text{continuous-on}$ 
( $\{0..1\} - ?s'$ ) D
    using gamma2-differentiable
    by auto
  then have *: $\forall x \in \{0..1\} - ?s'. \text{vector-derivative } \gamma 2 (at x \text{ within } \{0..1\}) = D x$ 
    using vector-derivative-at vector-derivative-at-within-ivl
    by fastforce
  then have continuous-on ( $\{0..1\} - ?s'$ ) ( $\lambda x. \text{vector-derivative } \gamma 2 (at x$ 
within  $\{0..1\})$ )
    using continuous-on-eq D
    by metis
  then have 1: continuous-on ( $\{0..1\} - ?s'$ ) ( $\lambda x. (\text{vector-derivative } \gamma 2 (at x$ 
within  $\{0..1\}) \cdot b)$ )
    by (auto intro!: continuous-intros)
  show ?thesis
    using continuous-on-mult[OF 0 1]
    by auto
qed
have iv:  $\varphi(0) \leq \varphi(1)$ 
  using phi(3) phi(4)
  by (simp add: reparam-def)
have v:  $\varphi' \{0..1\} \subseteq \{0..1\}$ 
  using phi
  by (auto simp add: reparam-def bij-betw-def)
obtain D where D-props: ( $\forall x \in \{0..1\} - ?s. (\varphi \text{ has-vector-derivative } D x) (at$ 
x)
  using s
  by (auto simp add: C1-differentiable-on-def)
then have ( $\bigwedge x. x \in (\{0..1\} - ?s) \implies (\varphi \text{ has-vector-derivative } D x) (at x \text{ within}$ 
 $\{0..1\})$ )
  using has-vector-derivative-at-within
  by blast
then have vi: ( $\bigwedge x. x \in (\{0..1\} - ?s) \implies (\varphi \text{ has-real-derivative } D x) (at x$ 
within  $\{0..1\})$ )
  using has-field-derivative-iff-has-vector-derivative

```

by blast
have $a: ((\lambda x. D x * (F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } (\varphi x) \text{ within } \{0..1\}) \cdot b))) \text{ has-integral } \int_{\{0..1\}} \{\varphi 0.. \varphi 1\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } x \text{ within } \{0..1\}) \cdot b)))$
proof-
have $a: \text{integral } \{\varphi 1.. \varphi 0\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } x \text{ within } \{0..1\}) \cdot b)) = 0$ **using** *integral-singleton integral-empty iv*
by (*simp add: phi(3) phi(4)*)
have $b: ((\lambda x. D x *_R (F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } (\varphi x) \text{ within } \{0..1\}) \cdot b))) \text{ has-integral } \int_{\{0..1\}} \{\varphi 0.. \varphi 1\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } x \text{ within } \{0..1\}) \cdot b)) - \int_{\{0..1\}} \{\varphi 1.. \varphi 0\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } x \text{ within } \{0..1\}) \cdot b)))$
apply(*rule has-integral-substitution-general-[OF s-inter(1) zero-le-one gamma2-differentiable(1) v gamma2-line-integrable iii cont-phi finite-bck-img-single-s]*)
proof-
have *surj-on* $\{0..1\} \varphi$
using *bij-phi*
by (*metis (full-types) bij-betw-def image-subsetI rangeI*)
then show *surj-on* $?s' \varphi$ **using** *bij-phi s'-in01*
by blast
show *inj-on* $\varphi (\{0..1\} \cup (?s \cup \varphi -' ?s'))$
proof-
have $i: \text{inj-on } \varphi \{0..1\}$ **using** *bij-phi*
using *bij-betw-def* **by blast**
have $ii: (\{0..1\} \cup (?s \cup \varphi -' ?s')) = \{0..1\}$ **using** *phi-backimg-s' s-in01 s'-in01*
by blast
show *?thesis* **using** i ii **by auto**
qed
show $\bigwedge x. x \in \{0..1\} - ?s \implies (\varphi \text{ has-real-derivative } D x) (\text{at } x \text{ within } \{0..1\})$
using vi **by blast**
qed
show *?thesis* **using** a b **by auto**
qed
then have $b: \text{integral } \{0..1\} (\lambda x. D x * (F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } (\varphi x) \text{ within } \{0..1\}) \cdot b))) = \int_{\{0..1\}} \{\varphi 0.. \varphi 1\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } x \text{ within } \{0..1\}) \cdot b))$
by auto
have *gamma2-vec-diffable*: $\bigwedge x::\text{real}. x \in \{0..1\} - ((\varphi -' ?s') \cup ?s) \implies ((\gamma 2 \circ \varphi) \text{ has-vector-derivative } \text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x)) (\text{at } x)$
proof-
fix $x::\text{real}$
assume $ass: x \in \{0..1\} - ((\varphi -' ?s') \cup ?s)$

```

have zer-le-x-le-1:  $0 \leq x \wedge x \leq 1$  using ass
  by simp
show  $((\gamma 2 \circ \varphi)$  has-vector-derivative vector-derivative  $(\gamma 2 \circ \varphi)$  (at  $x$ )) (at  $x$ )
proof-
  have **:  $\gamma 2$  differentiable at  $(\varphi x)$ 
    using gamma2-differentiable(2) ass v
    by blast
  have ***:  $\varphi$  differentiable at  $x$ 
    using s ass
    by (auto simp add: C1-differentiable-on-eq)
  then show  $((\gamma 2 \circ \varphi)$  has-vector-derivative vector-derivative  $(\gamma 2 \circ \varphi)$  (at  $x$ ))
    (at  $x$ )
    using differentiable-chain-at[OF *** **]
    by (auto simp add: vector-derivative-works)
  qed
  qed
  then have gamma2-vec-deriv-within:  $\bigwedge x::\text{real}. x \in \{0..1\} \rightarrow ((\varphi - ' ?s') \cup ?s) \Rightarrow$ 
    vector-derivative  $(\gamma 2 \circ \varphi)$  (at  $x$ ) = vector-derivative  $(\gamma 2 \circ \varphi)$  (at  $x$  within
     $\{0..1\}$ )
    using vector-derivative-at-within-ivl[OF gamma2-vec-diffable]
    by auto
  have  $\forall x \in \{0..1\} \rightarrow ((\varphi - ' ?s') \cup ?s). D x * (\text{vector-derivative } \gamma 2 \text{ (at } (\varphi x) \text{ within } \{0..1\}) \cdot b) =$ 
    (vector-derivative  $(\gamma 2 \circ \varphi)$  (at  $x$  within  $\{0..1\}$ )  $\cdot b$ )
  proof
    fix  $x::\text{real}$ 
    assume ass:  $x \in \{0..1\} \rightarrow ((\varphi - ' ?s') \cup ?s)$ 
    then have 0:  $\varphi$  differentiable (at  $x$ )
    using s by (auto simp add: C1-differentiable-on-def differentiable-def has-vector-derivative-def)
    obtain D2 where  $(\varphi$  has-vector-derivative D2) (at  $x$ )
    using D-props ass by blast
    have  $\varphi x \in \{0..1\} \rightarrow ?s'$ 
    using phi(5) ass by (metis Diff-Un Diff-iff Int-iff bij-betw-def image-eqI vimageI)
    then have 1:  $\gamma 2$  differentiable (at  $(\varphi x)$ )
    using gamma2-differentiable
    by auto
    have 3: vector-derivative  $\gamma 2$  (at  $(\varphi x)$ ) = vector-derivative  $\gamma 2$  (at  $(\varphi x)$  within
     $\{0..1\}$ )
    proof-
      have *:  $0 \leq \varphi x \wedge \varphi x \leq 1$  using phi(5) ass
      using  $\langle \varphi x \in \{0..1\} \rightarrow s' \cap \{0..1\} \rangle$  by auto
      then have **:  $(\gamma 2$  has-vector-derivative (vector-derivative  $\gamma 2$  (at  $(\varphi x)$ ))) (at
       $(\varphi x)$ )
      using 1 vector-derivative-works by auto
      show ?thesis
      using * vector-derivative-at-within-ivl[OF **] by auto
    qed
    show  $D x * (\text{vector-derivative } \gamma 2 \text{ (at } (\varphi x) \text{ within } \{0..1\}) \cdot b) = \text{vector-derivative}$ 
     $(\gamma 2 \circ \varphi)$  (at  $x$  within  $\{0..1\}$ )  $\cdot b$ 

```

using *vector-derivative-chain-at*[*OF 0 1*]
apply (*auto simp add: gamma2-vec-deriv-within*[*OF ass, symmetric*] *?**3*[*symmetric*])
using *D-props ass vector-derivative-at*
by *fastforce*
qed
then have $c: \bigwedge x. x \in \{0..1\} - ((\varphi - ' ?s') \cup ?s) \implies D x * (F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } (\varphi x) \text{ within } \{0..1\}) \cdot b)) = F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\}) \cdot b)$
by *auto*
then have $d: \text{integral } (\{0..1\}) (\lambda x. D x * (F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } (\varphi x) \text{ within } \{0..1\}) \cdot b))) = \text{integral } (\{0..1\}) (\lambda x. F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\}) \cdot b))$
proof–
have *negligible* $((\varphi - ' ?s') \cup ?s)$ **using** *finite-neg-img s(1)* **by** *auto*
then show *?thesis*
using *c integral-spike* **by** (*metis(no-types, lifting)*)
qed
have *phi-in-int*: $(\bigwedge x. x \in \{0..1\} \implies \varphi x \in \{0..1\})$ **using** *phi*
using *v* **by** *blast*
then have $e: ((\lambda x. F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\}) \cdot b)) \text{ has-integral } \text{integral } \{\varphi 0.. \varphi 1\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } x \text{ within } \{0..1\}) \cdot b))) \{0..1\}$
proof–
have *negligible* *?s* **using** *s-inter(1)* **by** *auto*
have $0: \text{negligible } ((\varphi - ' ?s') \cup ?s)$ **using** *finite-neg-img s(1)* **by** *auto*
have $c': \forall x \in \{0..1\} - ((\varphi - ' ?s') \cup ?s). D x * (F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } (\varphi x) \text{ within } \{0..1\}) \cdot b)) = F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\}) \cdot b)$
using *c* **by** *blast*
have *has-integral-spike-eq'*: $\bigwedge s t f g y. \text{negligible } s \implies \forall x \in t - s. g x = f x \implies (f \text{ has-integral } y) t = (g \text{ has-integral } y) t$
using *has-integral-spike-eq* **by** *metis*
show *?thesis*
using *a has-integral-spike-eq'*[*OF 0 c'*] **by** *blast*
qed
then have $f: ((\lambda x. F (\gamma 1 x) \cdot b * (\text{vector-derivative } \gamma 1 (\text{at } x \text{ within } \{0..1\}) \cdot b)) \text{ has-integral } \text{integral } \{\varphi 0.. \varphi 1\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } x \text{ within } \{0..1\}) \cdot b))) \{0..1\}$
proof–
assume *ass*: $((\lambda x. F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\}) \cdot b)) \text{ has-integral } \text{integral } \{\varphi 0.. \varphi 1\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } x \text{ within } \{0..1\}) \cdot b))) \{0..1\}$

$\gamma 2$ (at x within $\{0..1\}$) $\cdot b$))
 $\{0..1\}$
have $*$: $\forall x \in \{0..1\} - ((\varphi - ' ?s') \cup ?s) \cup \{0,1\}$. $(\lambda x. F (\gamma 2 (\varphi x)) \cdot b * (vector-derivative (\gamma 2 \circ \varphi) (at x within \{0..1\}) \cdot b)) x = (\lambda x. F (\gamma 1 x) \cdot b * (vector-derivative \gamma 1 (at x within \{0..1\}) \cdot b)) x$
proof–
have $\forall x \in \{0 < .. < 1\} - (\varphi - ' ?s' \cup ?s)$. $(vector-derivative (\gamma 2 \circ \varphi) (at x within \{0..1\}) \cdot b) = (vector-derivative (\gamma 1) (at x within \{0..1\}) \cdot b)$
proof
have i : $open (\{0 < .. < 1\} - ((\varphi - ' ?s') \cup ?s))$ **using** $open-diff s(1)$
 $open-greaterThanLessThan$ $finite-neg-imp$
by $(simp add: open-diff)$
have ii : $\forall x \in \{0 < .. < 1 :: real\} - (\varphi - ' ?s' \cup ?s)$. $(\gamma 2 \circ \varphi) x = \gamma 1 x$ **using** $phi(1)$
by $auto$
fix $x :: real$
assume ass : $x \in \{0 < .. < 1 :: real\} - ((\varphi - ' ?s') \cup ?s)$
then have iii : $(\gamma 2 \circ \varphi$ has-vector-derivative $vector-derivative (\gamma 2 \circ \varphi) (at x within \{0..1\})) (at x)$
by $(metis (no-types) Diff-iff add commute add-strict-mono ass atLeastAtMost-iff gamma2-vec-deriv-within gamma2-vec-diffable greaterThanLessThan-iff less-irrefl not-le)$
then have iv : $(\gamma 1$ has-vector-derivative $vector-derivative (\gamma 2 \circ \varphi) (at x within \{0..1\})) (at x)$
using $has-derivative-transform-within-open i ii$ ass
apply $(auto simp add: has-vector-derivative-def)$
apply $(meson ass has-derivative-transform-within-open i ii)$
apply $(meson ass has-derivative-transform-within-open i ii)$
by $(meson ass has-derivative-transform-within-open i ii)$
have v : $0 \leq x \leq 1$ **using** ass **by** $auto$
have 0 : $vector-derivative \gamma 1 (at x within \{0..1\}) = vector-derivative (\gamma 2 \circ \varphi) (at x within \{0..1\})$
using $vector-derivative-at-within-ivl[OF iv v(1) v(2) zero-less-one]$
by $force$
have 1 : $vector-derivative (\gamma 2 \circ \varphi) (at x within \{0..1\}) = vector-derivative (\gamma 2 \circ \varphi) (at x within \{0..1\})$
using $vector-derivative-at-within-ivl[OF iii v(1) v(2) zero-less-one]$
by $force$
then have $vector-derivative (\gamma 2 \circ \varphi) (at x within \{0..1\}) = vector-derivative \gamma 1 (at x within \{0..1\})$
using $0 1$ **by** $auto$
then show $vector-derivative (\gamma 2 \circ \varphi) (at x within \{0..1\}) \cdot b = vector-derivative \gamma 1 (at x within \{0..1\}) \cdot b$ **by** $auto$
qed
then have i : $\forall x \in \{0..1\} - ((\varphi - ' ?s') \cup ?s) \cup \{0,1\}$. $(vector-derivative (\gamma 2 \circ \varphi) (at x within \{0..1\}) \cdot b) = (vector-derivative (\gamma 1) (at x within \{0..1\}) \cdot b)$

```

    by auto
    have ii:  $\forall x \in \{0..1\} - (((\varphi - ' ?s') \cup ?s) \cup \{0,1\})$ .  $F (\gamma 1 x) \cdot b = F (\gamma 2 (\varphi x)) \cdot b$ 
      using phi(1)
      by auto
    show ?thesis
      using i ii by metis
  qed
  have **: negligible  $((\varphi - ' ?s') \cup ?s \cup \{0, 1\})$  using s(1) finite-neg-img by
  auto
  have has-integral-spike-eq':  $\bigwedge s t g f y$ . negligible  $s \implies \forall x \in t - s$ .  $g x = f x \implies (f \text{ has-integral } y) t = (g \text{ has-integral } y) t$ 
    using has-integral-spike-eq by metis
  show ?thesis
    using has-integral-spike-eq'[OF ** *] ass
    by blast
  qed
  then show line-integral-exists  $F \{b\} \gamma 1$ 
    using phi by(auto simp add: line-integral-exists-def)
  have integral  $(\{0..1\}) (\lambda x. F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\}) \cdot b)) =$ 
    integral  $(\{0..1\}) (\lambda x. F (\gamma 1 x) \cdot b * (\text{vector-derivative } \gamma 1 (\text{at } x \text{ within } \{0..1\}) \cdot b))$ 
    using integral-unique[OF e] integral-unique[OF f]
    by metis
  moreover have integral  $(\{0..1\}) (\lambda x. F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\}) \cdot b)) =$ 
    integral  $(\{0..1\}) (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } x \text{ within } \{0..1\}) \cdot b))$ 
    using b d phi by (auto simp add:)
  ultimately show line-integral  $F \{b\} \gamma 1 = \text{line-integral } F \{b\} \gamma 2$ 
    using phi by(auto simp add: line-integral-def)
  qed

lemma reparam-weak-eq-line-integrals:
  assumes reparam-weak  $\gamma 1 \gamma 2$ 
     $\gamma 2$  C1-differentiable-on  $\{0..1\}$ 
    continuous-on (path-image  $\gamma 2$ )  $(\lambda x. F x \cdot b)$ 
  shows line-integral  $F \{b\} \gamma 1 = \text{line-integral } F \{b\} \gamma 2$ 
    line-integral-exists  $F \{b\} \gamma 1$ 
  proof -
  obtain  $\varphi$  where phi:  $(\forall x \in \{0..1\}. \gamma 1 x = (\gamma 2 \circ \varphi) x)$   $\varphi$  piecewise-C1-differentiable-on  $\{0..1\}$   $\varphi(0) = 0$   $\varphi(1) = 1$   $\varphi - ' \{0..1\} = \{0..1\}$ 
    using assms(1)
    by (auto simp add: reparam-weak-def)
  obtain  $s$  where s: finite  $s$   $\varphi$  C1-differentiable-on  $\{0..1\} - s$ 
    using phi
    by(auto simp add: reparam-weak-def piecewise-C1-differentiable-on-def)

```



```

have cont-phi: continuous-on {0..1}  $\varphi$ 
  using phi
by (auto simp add: reparam-weak-def piecewise-C1-differentiable-on-imp-continuous-on)
have gamma2-differentiable:  $(\forall x \in \{0 .. 1\}. \gamma 2 \text{ differentiable at } x)$ 
  using assms(2)
  by (auto simp add: valid-path-def C1-differentiable-on-eq)
then have gamma2-b-component-differentiable:  $(\forall x \in \{0 .. 1\}. (\lambda x. (\gamma 2 x) \cdot b)$ 
differentiable at x)
  by auto
then have  $(\lambda x. (\gamma 2 x) \cdot b)$  differentiable-on {0..1}
  using differentiable-at-withinI
  by (auto simp add: differentiable-on-def)
then have gamma2-cont-comp: continuous-on {0..1}  $(\lambda x. (\gamma 2 x) \cdot b)$ 
  using differentiable-imp-continuous-on
  by auto
have gamma2-cont: continuous-on {0..1}  $\gamma 2$ 
  using assms(2) C1-differentiable-imp-continuous-on
  by (auto simp add: valid-path-def)
have iii: continuous-on {0..1}  $(\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 \text{ (at } x$ 
within } \{0..1\}) \cdot b))
proof-
  have 0: continuous-on {0..1}  $(\lambda x. F (\gamma 2 x) \cdot b)$ 
    using assms(3) continuous-on-compose[OF gamma2-cont]
    by (auto simp add: path-image-def)
  obtain D where D:  $(\forall x \in \{0..1\}. (\gamma 2 \text{ has-vector-derivative } D x) \text{ (at } x)) \wedge$ 
continuous-on {0..1} D
    using assms(2) by (auto simp add: C1-differentiable-on-def)
  then have  $*:\forall x \in \{0..1\}. \text{vector-derivative } \gamma 2 \text{ (at } x \text{ within } \{0..1\}) = D x$ 
    using vector-derivative-at vector-derivative-at-within-ivl
    by fastforce
  then have continuous-on {0..1}  $(\lambda x. \text{vector-derivative } \gamma 2 \text{ (at } x \text{ within } \{0..1\}))$ 
    using continuous-on-eq D by force
  then have 1: continuous-on {0..1}  $(\lambda x. (\text{vector-derivative } \gamma 2 \text{ (at } x \text{ within } \{0..1\}))$ 
· b)
    by (auto intro!: continuous-intros)
  show ?thesis
    using continuous-on-mult[OF 0 1] by auto
qed
have iv:  $\varphi(0) \leq \varphi(1)$ 
  using phi(3) phi(4) by (simp add: reparam-weak-def)
have v:  $\varphi \cdot \{0..1\} \subseteq \{0..1\}$ 
  using phi(5) by (simp add: reparam-weak-def)
obtain D where D-props:  $(\forall x \in \{0..1\} - s. (\varphi \text{ has-vector-derivative } D x) \text{ (at$ 
x))
  using s
  by (auto simp add: C1-differentiable-on-def)
then have  $(\bigwedge x. x \in (\{0..1\} - s) \implies (\varphi \text{ has-vector-derivative } D x) \text{ (at } x \text{ within}$ 
{0..1}))
  using has-vector-derivative-at-within by blast

```

then have vi : $(\bigwedge x. x \in \{0..1\} - s) \implies (\varphi \text{ has-real-derivative } D x) \text{ (at } x \text{ within } \{0..1\})$
using *has-field-derivative-iff-has-vector-derivative*
by *blast*
have a : $((\lambda x. D x * (F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } \gamma 2 \text{ (at } (\varphi x) \text{ within } \{0..1\}) \cdot b))) \text{ has-integral } \int_{\{0..1\}} \{\varphi \ 0.. \varphi \ 1\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 \text{ (at } x \text{ within } \{0..1\}) \cdot b)))$
using *has-integral-substitution-strong*[*OF s(1) zero-le-one iv v iii cont-phi vi*]
by *simp*
then have b : $\text{integral } \{0..1\} (\lambda x. D x * (F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } \gamma 2 \text{ (at } (\varphi x) \text{ within } \{0..1\}) \cdot b))) = \int_{\{0..1\}} \{\varphi \ 0.. \varphi \ 1\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 \text{ (at } x \text{ within } \{0..1\}) \cdot b))$
by *auto*
have $\text{gamma2-vec-diffable}$: $\bigwedge x::\text{real}. x \in \{0..1\} - s \implies ((\gamma 2 \circ \varphi) \text{ has-vector-derivative } \text{vector-derivative } (\gamma 2 \circ \varphi) \text{ (at } x)) \text{ (at } x)$
proof–
fix $x::\text{real}$
assume ass : $x \in \{0..1\} - s$
have zer-le-x-le-1 : $0 \leq x \wedge x \leq 1$ **using** ass **by** *auto*
show $((\gamma 2 \circ \varphi) \text{ has-vector-derivative } \text{vector-derivative } (\gamma 2 \circ \varphi) \text{ (at } x)) \text{ (at } x)$
proof–
have $**$: $\gamma 2$ *differentiable at* (φx)
using *phi gamma2-differentiable*
by *(auto simp add: zer-le-x-le-1)*
have $***$: φ *differentiable at* x
using $s \text{ ass}$
by *(auto simp add: C1-differentiable-on-eq)*
then show $((\gamma 2 \circ \varphi) \text{ has-vector-derivative } \text{vector-derivative } (\gamma 2 \circ \varphi) \text{ (at } x)) \text{ (at } x)$
using *differentiable-chain-at*[*OF *** ***]
by *(auto simp add: vector-derivative-works)*
qed
qed
then have $\text{gamma2-vec-deriv-within}$: $\bigwedge x::\text{real}. x \in \{0..1\} - s \implies \text{vector-derivative } (\gamma 2 \circ \varphi) \text{ (at } x) = \text{vector-derivative } (\gamma 2 \circ \varphi) \text{ (at } x \text{ within } \{0..1\})$
using *vector-derivative-at-within-ivl*[*OF gamma2-vec-diffable*]
by *auto*
have $\forall x \in \{0..1\} - s. D x * (\text{vector-derivative } \gamma 2 \text{ (at } (\varphi x) \text{ within } \{0..1\}) \cdot b) = (\text{vector-derivative } (\gamma 2 \circ \varphi) \text{ (at } x \text{ within } \{0..1\}) \cdot b)$
proof
fix $x::\text{real}$
assume ass : $x \in \{0..1\} - s$
then have 0 : φ *differentiable at* x
using s
by *(auto simp add: C1-differentiable-on-def differentiable-def has-vector-derivative-def)*
obtain $D2$ **where** $(\varphi \text{ has-vector-derivative } D2) \text{ (at } x)$

```

    using D-props ass
  by blast
  have  $\varphi x \in \{0..1\}$ 
  using phi(5) ass
  by (auto simp add: reparam-weak-def)
  then have 1:  $\gamma 2$  differentiable (at ( $\varphi x$ ))
  using gamma2-differentiable
  by auto
  have 3: vector-derivative  $\gamma 2$  (at ( $\varphi x$ )) = vector-derivative  $\gamma 2$  (at ( $\varphi x$ ) within
 $\{0..1\}$ )
  proof-
    have *:  $0 \leq \varphi x \wedge \varphi x \leq 1$  using phi(5) ass by auto
    then have **: ( $\gamma 2$  has-vector-derivative (vector-derivative  $\gamma 2$  (at ( $\varphi x$ )))) (at
( $\varphi x$ ))
      using 1 vector-derivative-works
      by auto
    show ?thesis
      using * vector-derivative-at-within-ivl[OF **]
      by auto
  qed
  show  $D x * (\text{vector-derivative } \gamma 2 \text{ (at } (\varphi x) \text{ within } \{0..1\}) \cdot b) = \text{vector-derivative}$ 
( $\gamma 2 \circ \varphi$ ) (at  $x$  within  $\{0..1\}) \cdot b$ 
    using vector-derivative-chain-at[OF 0 1]
  apply (auto simp add: gamma2-vec-deriv-within[OF ass, symmetric] 3[symmetric])
  using D-props ass vector-derivative-at
  by fastforce
  qed
  then have c:  $\bigwedge x. x \in (\{0..1\} - s) \implies D x * (F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative}$ 
 $\gamma 2$  (at ( $\varphi x$ ) within  $\{0..1\}) \cdot b)) =$ 

$$F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) \text{ (at } x$$

within  $\{0..1\}) \cdot b)$ 
    by auto
  then have d:  $\text{integral } (\{0..1\}) (\lambda x. D x * (F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative}$ 
 $\gamma 2$  (at ( $\varphi x$ ) within  $\{0..1\}) \cdot b))) =$ 

$$\text{integral } (\{0..1\}) (\lambda x. F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) \text{ (at } x$$

within  $\{0..1\}) \cdot b))$ 
  proof-
    have negligible s using s(1) by auto
    then show ?thesis
      using c integral-spike by (metis(no-types, lifting))
  qed
  have phi-in-int:  $(\bigwedge x. x \in \{0..1\} \implies \varphi x \in \{0..1\})$  using phi
  by (auto simp add:)
  then have e:  $((\lambda x. F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) \text{ (at } x$ 
within  $\{0..1\}) \cdot b))$  has-integral

$$\text{integral } \{\varphi 0.. \varphi 1\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative}$$

 $\gamma 2 \text{ (at } x \text{ within } \{0..1\}) \cdot b)) \{0..1\}$ 
  proof-
    have 0: negligible s using s(1) by auto

```

have $c': \forall x \in \{0..1\} - s. D x * (F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } (\varphi x) \text{ within } \{0..1\}) \cdot b)) =$
 $F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\}) \cdot b)$
using c **by** *auto*
have $\text{has-integral-spike-eq}': \bigwedge s t f g y. \text{negligible } s \implies$
 $\forall x \in t - s. g x = f x \implies (f \text{ has-integral } y) t$
 $= (g \text{ has-integral } y) t$
using $\text{has-integral-spike-eq}$ **by** *metis*
show *?thesis*
using $a \text{ has-integral-spike-eq}' [OF 0 c']$ **by** *blast*
qed
then have $f: ((\lambda x. F (\gamma 1 x) \cdot b * (\text{vector-derivative } \gamma 1 (\text{at } x \text{ within } \{0..1\}) \cdot b)) \text{ has-integral}$
 $\int_{\{0..1\}} \{\varphi 0.. \varphi 1\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } x \text{ within } \{0..1\}) \cdot b)))$
proof-
assume $\text{ass}: ((\lambda x. F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\}) \cdot b)) \text{ has-integral}$
 $\int_{\{0..1\}} \{\varphi 0.. \varphi 1\} (\lambda x. F (\gamma 2 x) \cdot b * (\text{vector-derivative } \gamma 2 (\text{at } x \text{ within } \{0..1\}) \cdot b)))$
have $*: \forall x \in \{0..1\} - (s \cup \{0,1\}). (\lambda x. F (\gamma 2 (\varphi x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\}) \cdot b)) x =$
 $(\lambda x. F (\gamma 1 x) \cdot b * (\text{vector-derivative } \gamma 1 (\text{at } x \text{ within } \{0..1\}) \cdot b)) x$
proof-
have $\forall x \in \{0 < .. < 1\} - s. (\text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\}) \cdot b) = (\text{vector-derivative } \gamma 1 (\text{at } x \text{ within } \{0..1\}) \cdot b)$
proof
have $i: \text{open } (\{0 < .. < 1\} - s)$ **using** *open-diff s open-greaterThanLessThan*
by *blast*
have $ii: \forall x \in \{0 < .. < 1 :: \text{real}\} - s. (\gamma 2 \circ \varphi) x = \gamma 1 x$ **using** *phi(1)*
by *auto*
fix $x :: \text{real}$
assume $\text{ass}: x \in \{0 < .. < 1 :: \text{real}\} - s$
then have $iii: (\gamma 2 \circ \varphi \text{ has-vector-derivative } \text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\})) (\text{at } x)$
using *has-vector-derivative-at-within gamma2-vec-deriv-within gamma2-vec-diffable*
by *auto*

then have $iv: (\gamma 1 \text{ has-vector-derivative } \text{vector-derivative } (\gamma 2 \circ \varphi) (\text{at } x \text{ within } \{0..1\})) (\text{at } x)$
using *has-derivative-transform-within-open i ii ass*
apply (*auto simp add: has-vector-derivative-def*)
by *force*
have $v: 0 \leq x \leq 1$ **using** ass **by** *auto*
have $0: \text{vector-derivative } \gamma 1 (\text{at } x \text{ within } \{0..1\}) = \text{vector-derivative } (\gamma 2 \circ \varphi)$

φ) (at x within $\{0..1\}$)
using *vector-derivative-at-within-ivl*[*OF iv v(1) v(2) zero-less-one*]
by force
have 1 : *vector-derivative* $(\gamma 2 \circ \varphi)$ (at x within $\{0..1\}$) = *vector-derivative*
 $(\gamma 2 \circ \varphi)$ (at x within $\{0..1\}$)
using *vector-derivative-at-within-ivl*[*OF iii v(1) v(2) zero-less-one*]
by force
then have *vector-derivative* $(\gamma 2 \circ \varphi)$ (at x within $\{0..1\}$) = *vector-derivative*
 $\gamma 1$ (at x within $\{0..1\}$)
using $0\ 1$ **by auto**
then show *vector-derivative* $(\gamma 2 \circ \varphi)$ (at x within $\{0..1\}$) $\cdot b$ = *vector-derivative*
 $\gamma 1$ (at x within $\{0..1\}$) $\cdot b$ **by auto**
qed
then have i : $\forall x \in \{0..1\} - (s \cup \{0,1\})$. (*vector-derivative* $(\gamma 2 \circ \varphi)$ (at x
within $\{0..1\}$) $\cdot b$) = (*vector-derivative* $\gamma 1$) (at x within $\{0..1\}$) $\cdot b$)
by auto
have ii : $\forall x \in \{0..1\} - (s \cup \{0,1\})$. $F (\gamma 1\ x) \cdot b = F (\gamma 2 (\varphi\ x)) \cdot b$
using *phi(1)* **by auto**
show *?thesis*
using $i\ ii$ **by auto**
qed
have $**$: *negligible* $(s \cup \{0,1\})$ **using** $s(1)$ **by auto**
have *has-integral-spike-eq'*: $\bigwedge s\ t\ g\ f\ y$. *negligible* $s \implies$
 $\forall x \in t - s$. $g\ x = f\ x \implies (f\ \text{has-integral}\ y)\ t = (g$
has-integral $y)\ t$
using *has-integral-spike-eq* **by metis**
show *?thesis*
using *has-integral-spike-eq'*[*OF ** **] *ass*
by blast
qed
then show *line-integral-exists* $F\ \{b\}\ \gamma 1$
using *phi* **by** (*auto simp add: line-integral-exists-def*)
have *integral* $(\{0..1\}) (\lambda x. F (\gamma 2 (\varphi\ x)) \cdot b * (\text{vector-derivative } (\gamma 2 \circ \varphi)$ (at x
within $\{0..1\}$) $\cdot b)) =$
integral $(\{0..1\}) (\lambda x. F (\gamma 1\ x) \cdot b * (\text{vector-derivative } \gamma 1$ (at x
within $\{0..1\}$) $\cdot b))$
using *integral-unique*[*OF e*] *integral-unique*[*OF f*]
by metis
moreover have *integral* $(\{0..1\}) (\lambda x. F (\gamma 2 (\varphi\ x)) \cdot b * (\text{vector-derivative } (\gamma 2$
 $\circ \varphi)$ (at x within $\{0..1\}$) $\cdot b)) =$
integral $(\{0..1\}) (\lambda x. F (\gamma 2\ x) \cdot b * (\text{vector-derivative } \gamma 2$ (at
 x within $\{0..1\}$) $\cdot b))$
using $b\ d\ phi$ **by** (*auto simp add:*)
ultimately show *line-integral* $F\ \{b\}\ \gamma 1 = \text{line-integral } F\ \{b\}\ \gamma 2$
using *phi* **by** (*auto simp add: line-integral-def*)
qed

lemma *line-integral-sum-basis*:

assumes *finite* (*basis::('a::euclidean-space) set*) $\forall b \in \text{basis}$. *line-integral-exists* F

```

{b}  $\gamma$ 
  shows  $\text{line-integral } F \text{ basis } \gamma = (\sum_{b \in \text{basis}} \text{line-integral } F \{b\} \gamma)$ 
     $\text{line-integral-exists } F \text{ basis } \gamma$ 
  using assms
proof(induction basis)
  show  $\text{line-integral } F \{ \} \gamma = (\sum_{b \in \{ \}} \text{line-integral } F \{b\} \gamma)$ 
    by(auto simp add: line-integral-def)
  show  $\forall b \in \{ \}. \text{line-integral-exists } F \{b\} \gamma \implies \text{line-integral-exists } F \{ \} \gamma$ 
    by(simp add: line-integral-exists-def integrable-0)
next
  fix basis::('a::euclidean-space) set
  fix x::'a::euclidean-space
  fix  $\gamma$ 
  assume ind-hyp:  $(\forall b \in \text{basis}. \text{line-integral-exists } F \{b\} \gamma \implies \text{line-integral-exists } F \text{ basis } \gamma)$ 
     $(\forall b \in \text{basis}. \text{line-integral-exists } F \{b\} \gamma \implies \text{line-integral } F \text{ basis } \gamma = (\sum_{b \in \text{basis}} \text{line-integral } F \{b\} \gamma))$ 
  assume step: finite basis
     $x \notin \text{basis}$ 
     $\forall b \in \text{insert } x \text{ basis}. \text{line-integral-exists } F \{b\} \gamma$ 
  then have 0:  $\text{line-integral-exists } F \{x\} \gamma$  by auto
  have 1:  $\text{line-integral-exists } F \text{ basis } \gamma$ 
    using ind-hyp step by auto
  then show  $\text{line-integral-exists } F (\text{insert } x \text{ basis}) \gamma$ 
    using step(1) step(2) line-integral-sum-gen(2)[OF - 0 1] by simp
  have 3: finite (insert x basis) using step(1) by auto
  have  $\text{line-integral } F \text{ basis } \gamma = (\sum_{b \in \text{basis}} \text{line-integral } F \{b\} \gamma)$ 
    using ind-hyp step by auto
  then show  $\text{line-integral } F (\text{insert } x \text{ basis}) \gamma = (\sum_{b \in \text{insert } x \text{ basis}} \text{line-integral } F \{b\} \gamma)$ 
    using step(1) step(2) line-integral-sum-gen(1)[OF 3 0 1]
    by force
qed

```

lemma *reparam-weak-eq-line-integrals-basis*:

```

  assumes reparam-weak  $\gamma1 \ \gamma2$ 
     $\gamma2 \text{ } C1\text{-differentiable-on } \{0..1\}$ 
     $\forall b \in \text{basis}. \text{continuous-on } (\text{path-image } \gamma2) (\lambda x. F x \cdot b)$ 
    finite basis
  shows  $\text{line-integral } F \text{ basis } \gamma1 = \text{line-integral } F \text{ basis } \gamma2$ 
     $\text{line-integral-exists } F \text{ basis } \gamma1$ 
proof-
  show  $\text{line-integral-exists } F \text{ basis } \gamma1$ 
    using reparam-weak-eq-line-integrals(2)[OF assms(1) assms(2)] assms(3-4)
     $\text{line-integral-sum-basis(2)[OF assms(4)]}$ 
    by(simp add: subset-iff)
  show  $\text{line-integral } F \text{ basis } \gamma1 = \text{line-integral } F \text{ basis } \gamma2$ 
    using reparam-weak-eq-line-integrals[OF assms(1) assms(2)] assms(3-4) line-integral-sum-basis(1)[OF assms(4)]

```

$\text{line-integral-exists-smooth-one-base}[OF \text{ assms}(2)]$
 $\text{by}(\text{simp add: subset-iff})$
qed

lemma *reparam-eq-line-integrals-basis:*

assumes *reparam* $\gamma 1 \ \gamma 2$
 $\gamma 2$ *piecewise-C1-differentiable-on* $\{0..1\}$
 $\forall b \in \text{basis. continuous-on}(\text{path-image } \gamma 2) (\lambda x. F x \cdot b)$
finite basis
 $\forall b \in \text{basis. line-integral-exists } F \ \{b\} \ \gamma 2$
shows *line-integral* $F \ \text{basis} \ \gamma 1 = \text{line-integral } F \ \text{basis} \ \gamma 2$
line-integral-exists $F \ \text{basis} \ \gamma 1$

proof –

show *line-integral-exists* $F \ \text{basis} \ \gamma 1$
using *reparam-eq-line-integrals*(2)[*OF assms*(1) *assms*(2)] *assms*(3–5) *line-integral-sum-basis*(2)[*OF assms*(4)]
 $\text{by}(\text{simp add: subset-iff})$
show *line-integral* $F \ \text{basis} \ \gamma 1 = \text{line-integral } F \ \text{basis} \ \gamma 2$
using *reparam-eq-line-integrals*[*OF assms*(1) *assms*(2)] *assms*(3–5) *line-integral-sum-basis*(1)[*OF assms*(4)]
 $\text{by}(\text{simp add: subset-iff})$
qed

lemma *line-integral-exists-smooth:*

assumes γ *C1-differentiable-on* $\{0..1\}$
 $\forall (b::'a::\text{euclidean-space}) \in \text{basis. continuous-on}(\text{path-image } \gamma) (\lambda x. F x \cdot b)$
finite basis
shows *line-integral-exists* $F \ \text{basis} \ \gamma$
using *reparam-weak-eq-line-integrals-basis*(2)[*OF reparam-weak-eq-refl*[**where** $? \gamma 1.0 = \gamma$]] *assms*
by *fastforce*

lemma *smooth-path-imp-reverse:*

assumes g *C1-differentiable-on* $\{0..1\}$
shows (*reversepath* g) *C1-differentiable-on* $\{0..1\}$
using *assms continuous-on-const*
apply (*auto simp: reversepath-def*)
apply (*rule C1-differentiable-compose* [of $\lambda x::\text{real. } 1-x - g$, *unfolded o-def*])
apply (*auto simp: C1-differentiable-on-eq*)
apply (*simp add: finite-vimageI inj-on-def*)
done

lemma *piecewise-smooth-path-imp-reverse:*

assumes g *piecewise-C1-differentiable-on* $\{0..1\}$
shows (*reversepath* g) *piecewise-C1-differentiable-on* $\{0..1\}$
using *assms valid-path-reversepath*
using *valid-path-def* **by** *blast*

definition *chain-reparam-weak-chain* **where**

chain-reparam-weak-chain one-chain1 one-chain2 \equiv
 $\exists f. \text{bij } f \wedge f \text{ 'one-chain1 = one-chain2} \wedge (\forall (k,\gamma) \in \text{one-chain1. if } k = \text{fst}$
 $(f(k,\gamma)) \text{ then reparam-weak } \gamma (\text{snd } (f(k,\gamma))) \text{ else reparam-weak } \gamma (\text{reversepath } (\text{snd}$
 $(f(k,\gamma))))$

lemma *chain-reparam-weak-chain-line-integral*:

assumes *chain-reparam-weak-chain one-chain1 one-chain2*

$\forall (k2,\gamma2) \in \text{one-chain2. } \gamma2 \text{ C1-differentiable-on } \{0..1\}$

$\forall (k2,\gamma2) \in \text{one-chain2.} \forall b \in \text{basis. continuous-on } (\text{path-image } \gamma2) (\lambda x. F x \cdot b)$
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

$\forall (k, \gamma) \in \text{one-chain1. line-integral-exists } F \text{ basis } \gamma$

proof –

obtain *f where f: bij f*

$(\forall (k,\gamma) \in \text{one-chain1. if } k = \text{fst } (f(k,\gamma)) \text{ then reparam-weak } \gamma (\text{snd } (f(k,\gamma)))$
else reparam-weak } \gamma (\text{reversepath } (\text{snd } (f(k,\gamma))))

f 'one-chain1 = one-chain2

using *assms(1)*

by (*auto simp add: chain-reparam-weak-chain-def*)

have $0: \forall x \in \text{one-chain1. (case } x \text{ of } (k, \gamma) \Rightarrow (\text{real-of-int } k * \text{line-integral } F \text{ basis}$
 $\gamma) = (\text{case } f \text{ x of } (k, \gamma) \Rightarrow \text{real-of-int } k * \text{line-integral } F \text{ basis } \gamma) \wedge$
 $\text{line-integral-exists } F \text{ basis } \gamma)$

proof

{fix *k1* $\gamma1$

assume *ass1: (k1, $\gamma1$) \in one-chain1*

have *real-of-int k1 * line-integral F basis $\gamma1 = (\text{case } (f (k1,\gamma1)) \text{ of } (k2, \gamma2)$*
 $\Rightarrow \text{real-of-int } k2 * \text{line-integral } F \text{ basis } \gamma2) \wedge$
 $\text{line-integral-exists } F \text{ basis } \gamma1$

proof(*cases*)

assume *ass2: k1 = 1*

let *?k2 = fst (f (k1, $\gamma1$))*

let *? $\gamma2 = \text{snd } (f (k1, \gamma1))$*

have *real-of-int k1 * line-integral F basis $\gamma1 = \text{real-of-int } ?k2 * \text{line-integral}$*
 $F \text{ basis } ?\gamma2 \wedge$

$\text{line-integral-exists } F \text{ basis } \gamma1$

proof(*cases*)

assume *ass3: ?k2 = 1*

then have $0: \text{reparam-weak } \gamma1 ?\gamma2$

using *ass1 ass2 f(2)*

by *auto*

have $1: ?\gamma2 \text{ C1-differentiable-on } \{0..1\}$

using *f(3) assms(2) ass1*

by *force*

have $2: \forall b \in \text{basis. continuous-on } (\text{path-image } ?\gamma2) (\lambda x. F x \cdot b)$

using *f(3) assms(3) ass1*

by *force*


```

show real-of-int  $k1$  * line-integral  $F$  basis  $\gamma1$  = real-of-int  $?k2$  * line-integral
 $F$  basis  $?\gamma2$   $\wedge$ 
    line-integral-exists  $F$  basis  $\gamma1$ 
    using assms reparam-weak-eq-line-integrals-basis[ $OF$  0 1 2 assms(4)]
    ass2 ass3
    by auto
next
assume  $?k2 \neq 1$ 
then have ass3:  $?k2 = -1$ 
    using bound2 ass1 f(3) unfolding boundary-chain-def by force
then have 0: reparam-weak  $\gamma1$  (reversepath  $?\gamma2$ )
    using ass1 ass2 f(2)
    by auto
have 1: (reversepath  $?\gamma2$ ) C1-differentiable-on {0..1}
    using f(3) assms(2) ass1 smooth-path-imp-reverse
    by force
have 2:  $\forall b \in \text{basis}. \text{continuous-on } (\text{path-image } (\text{reversepath } ?\gamma2)) (\lambda x. F$ 
 $x \cdot b)$ 
    using f(3) assms(3) ass1 path-image-reversepath
    by force
have 3: line-integral  $F$  basis  $?\gamma2$  = - line-integral  $F$  basis (reversepath
 $?\gamma2$ )
proof-
    have i:valid-path (reversepath  $?\gamma2$ )
    using 1 C1-differentiable-imp-piecewise
    by (auto simp add: valid-path-def)
    show ?thesis
    using line-integral-on-reverse-path(1)[ $OF$  i line-integral-exists-smooth[ $OF$ 
1 2 ]] assms
    by auto
qed
show real-of-int  $k1$  * line-integral  $F$  basis  $\gamma1$  = real-of-int  $?k2$  * line-integral
 $F$  basis  $?\gamma2$   $\wedge$ 
    line-integral-exists  $F$  basis  $\gamma1$ 
    using assms reparam-weak-eq-line-integrals-basis[ $OF$  0 1 2 assms(4)]
    ass2 ass3 3
    by auto
qed
then show real-of-int  $k1$  * line-integral  $F$  basis  $\gamma1$  = (case  $f$  ( $k1$ ,  $\gamma1$ ) of
( $k2$ ,  $\gamma2$ )  $\Rightarrow$  real-of-int  $k2$  * line-integral  $F$  basis  $\gamma2$ )  $\wedge$ 
    line-integral-exists  $F$  basis  $\gamma1$ 
    by (simp add: case-prod-beta')
next
assume  $k1 \neq 1$ 
then have ass2:  $k1 = -1$ 
    using bound1 ass1 f(3) unfolding boundary-chain-def by force
let  $?k2 = \text{fst } (f$  ( $k1$ ,  $\gamma1$ ))
let  $?\gamma2 = \text{snd } (f$  ( $k1$ ,  $\gamma1$ ))
have real-of-int  $k1$  * line-integral  $F$  basis  $\gamma1$  = real-of-int  $?k2$  * line-integral

```

```

F basis ? $\gamma$ 2  $\wedge$ 
      line-integral-exists F basis  $\gamma$ 1
  proof(cases)
    assume ass3: ? $k$ 2 = 1
    then have 0: reparam-weak  $\gamma$ 1 (reversepath ? $\gamma$ 2)
      using ass1 ass2 f(2)
      by auto
    have 1: (reversepath ? $\gamma$ 2) C1-differentiable-on {0..1}
      using f(3) assms(2) ass1 smooth-path-imp-reverse
      by force
    have 2:  $\forall b \in \text{basis}. \text{continuous-on } (\text{path-image } (\text{reversepath } ?\gamma 2)) (\lambda x. F$ 
x  $\cdot b)$ 
      using f(3) assms(3) ass1 path-image-reversepath
      by force
    have 3: line-integral F basis ? $\gamma$ 2 = - line-integral F basis (reversepath
? $\gamma$ 2)
  proof-
    have i: valid-path (reversepath ? $\gamma$ 2)
      using 1 C1-differentiable-imp-piecewise
      by (auto simp add: valid-path-def)
    show ?thesis
      using line-integral-on-reverse-path(1)[OF i line-integral-exists-smooth[OF
1 2 assms(4)]]
      by auto
    qed
    show real-of-int k1 * line-integral F basis  $\gamma$ 1 = real-of-int ? $k$ 2 * line-integral
F basis ? $\gamma$ 2  $\wedge$ 
      line-integral-exists F basis  $\gamma$ 1
      using assms reparam-weak-eq-line-integrals-basis[OF 0 1 2 assms(4)]
      ass2 ass3 3
      by auto
  next
    assume ? $k$ 2  $\neq$  1
    then have ass3: ? $k$ 2 = -1
      using bound2 ass1 f(3) unfolding boundary-chain-def by force
    then have 0: reparam-weak  $\gamma$ 1 ? $\gamma$ 2
      using ass1 ass2 f(2) by auto
    have 1: ? $\gamma$ 2 C1-differentiable-on {0..1}
      using f(3) assms(2) ass1 by force
    have 2:  $\forall b \in \text{basis}. \text{continuous-on } (\text{path-image } ?\gamma 2) (\lambda x. F$ 
x  $\cdot b)$ 
      using f(3) assms(3) ass1 by force
    show real-of-int k1 * line-integral F basis  $\gamma$ 1 = real-of-int ? $k$ 2 * line-integral
F basis ? $\gamma$ 2  $\wedge$ 
      line-integral-exists F basis  $\gamma$ 1
      using assms reparam-weak-eq-line-integrals-basis[OF 0 1 2 assms(4)]
      ass2 ass3
      by auto
    qed
    then show real-of-int k1 * line-integral F basis  $\gamma$ 1 = (case f (k1,  $\gamma$ 1) of

```

```

(k2, γ2) ⇒ real-of-int k2 * line-integral F basis γ2) ∧
      line-integral-exists F basis γ1
  by (simp add: case-prod-beta')
qed
}
then show ∧x. x ∈ one-chain1 ⇒
      (case x of (k, γ) ⇒ (real-of-int k * line-integral F basis
γ) = (case f x of (k, γ) ⇒ real-of-int k * line-integral F basis γ) ∧
      line-integral-exists F basis γ)

  by (auto simp add: case-prod-beta')
qed
show one-chain-line-integral F basis one-chain1 = one-chain-line-integral F basis
one-chain2
  using 0 by (simp add: one-chain-line-integral-def sum-bij[OF f(1) - f(3)]
split-beta)
  show ∀(k, γ) ∈ one-chain1. line-integral-exists F basis γ
  using 0 by blast
qed

```

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,γ2) ∈ one-chain2. γ2 piecewise-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
and line: ∀(k2,γ2) ∈ one-chain2. (∀ b ∈ basis. line-integral-exists F {b} γ2)
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 -
obtain f where f: bij f
  (∀(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))))))
  f ' one-chain1 = one-chain2
  using assms(1)
  by (auto simp add: chain-reparam-chain-def)
have integ-exist-b: ∀(k1,γ1) ∈ one-chain1. ∀ b ∈ basis. line-integral-exists F {b}

```

```

(snd (f (k1, γ1)))
  using line f by fastforce
  have valid-cubes: ∀ (k1,γ1)∈one-chain1. valid-path (snd (f (k1, γ1)))
  using assms(2) f(3) valid-path-def by fastforce
  have integ-rev-exist-b: ∀ (k1,γ1)∈one-chain1. ∀ b∈basis. line-integral-exists F {b}
(reversepath (snd (f (k1, γ1))))
  using line-integral-on-reverse-path(2) integ-exist-b valid-cubes
  by blast
  have 0: ∀ x∈one-chain1. (case x of (k, γ) ⇒ (real-of-int k * line-integral F basis
γ) = (case f x of (k, γ) ⇒ real-of-int k * line-integral F basis γ) ∧
line-integral-exists F basis γ)

proof
  {fix k1 γ1
  assume ass1: (k1,γ1) ∈one-chain1
  have real-of-int k1 * line-integral F basis γ1 = (case (f (k1,γ1)) of (k2, γ2)
⇒ real-of-int k2 * line-integral F basis γ2) ∧
line-integral-exists F basis γ1
  proof(cases)
    assume ass2: k1 = 1
    let ?k2 = fst (f (k1, γ1))
    let ?γ2 = snd (f (k1, γ1))
    have real-of-int k1 * line-integral F basis γ1 = real-of-int ?k2 * line-integral
F basis ?γ2 ∧
line-integral-exists F basis γ1
  proof(cases)
    assume ass3: ?k2 = 1
    then have 0: reparam γ1 ?γ2
      using ass1 ass2 f(2)
      by auto
    have 1: ?γ2 piecewise-C1-differentiable-on {0..1}
      using f(3) assms(2) ass1
      by force
    have 2: ∀ b∈basis. continuous-on (path-image ?γ2) (λx. F x · b)
      using f(3) assms(3) ass1
      by force
    show real-of-int k1 * line-integral F basis γ1 = real-of-int ?k2 * line-integral
F basis ?γ2 ∧
line-integral-exists F basis γ1
  using assms reparam-eq-line-integrals-basis[OF 0 1 2 assms(4)] integ-exist-b
ass1 ass2 ass3
  by auto
  next
  assume ?k2 ≠ 1
  then have ass3: ?k2 = -1
    using bound2 ass1 f(3) unfolding boundary-chain-def by force
  then have 0: reparam γ1 (reversepath ?γ2)
    using ass1 ass2 f(2)
    by auto
  have 1: (reversepath ?γ2) piecewise-C1-differentiable-on {0..1}

```

```

    using f(3) assms(2) ass1 piecewise-smooth-path-imp-reverse
    by force
    have 2:  $\forall b \in \text{basis}. \text{continuous-on } (\text{path-image } (\text{reversepath } ?\gamma 2)) (\lambda x. F$ 
x · b)
    using f(3) assms(3) ass1 path-image-reversepath
    by force
    have 3:  $\text{line-integral } F \text{ basis } ?\gamma 2 = - \text{line-integral } F \text{ basis } (\text{reversepath}$ 
? $\gamma 2)$ 
    proof-
    have i:  $\text{valid-path } (\text{reversepath } ?\gamma 2)$ 
    using 1 C1-differentiable-imp-piecewise
    by (auto simp add: valid-path-def)
    have ii:  $\text{line-integral-exists } F \text{ basis } (\text{snd } (f (k1, \gamma 1)))$ 
    using assms(4) line-integral-sum-basis(2) integ-exist-b ass1
    by fastforce
    show ?thesis
    using i ii line-integral-on-reverse-path(1) valid-path-reversepath by
blast
    qed
    show  $\text{real-of-int } k1 * \text{line-integral } F \text{ basis } \gamma 1 = \text{real-of-int } ?k2 * \text{line-integral}$ 
 $F \text{ basis } ?\gamma 2 \wedge$ 
 $\text{line-integral-exists } F \text{ basis } \gamma 1$ 
    using assms reparam-eq-line-integrals-basis[OF 0 1 2 assms(4)] integ-rev-exist-b
    ass1 ass2 ass3 3
    by auto
    qed
    then show  $\text{real-of-int } k1 * \text{line-integral } F \text{ basis } \gamma 1 = (\text{case } f (k1, \gamma 1) \text{ of}$ 
( $k2, \gamma 2$ )  $\Rightarrow \text{real-of-int } k2 * \text{line-integral } F \text{ basis } \gamma 2) \wedge$ 
 $\text{line-integral-exists } F \text{ basis } \gamma 1$ 
    by (simp add: case-prod-beta^)
    next
    assume  $k1 \neq 1$ 
    then have  $\text{ass2}: k1 = -1$ 
    using bound1 ass1 f(3) unfolding boundary-chain-def by force
    let  $?k2 = \text{fst } (f (k1, \gamma 1))$ 
    let  $?\gamma 2 = \text{snd } (f (k1, \gamma 1))$ 
    have  $\text{real-of-int } k1 * \text{line-integral } F \text{ basis } \gamma 1 = \text{real-of-int } ?k2 * \text{line-integral}$ 
 $F \text{ basis } ?\gamma 2 \wedge$ 
 $\text{line-integral-exists } F \text{ basis } \gamma 1$ 
    proof(cases)
    assume  $\text{ass3}: ?k2 = 1$ 
    then have 0:  $\text{reparam } \gamma 1 (\text{reversepath } ?\gamma 2)$ 
    using ass1 ass2 f(2)
    by auto
    have 1:  $(\text{reversepath } ?\gamma 2) \text{ piecewise-C1-differentiable-on } \{0..1\}$ 
    using f(3) assms(2) ass1 piecewise-smooth-path-imp-reverse
    by force
    have 2:  $\forall b \in \text{basis}. \text{continuous-on } (\text{path-image } (\text{reversepath } ?\gamma 2)) (\lambda x. F$ 
x · b)

```

```

    using f(3) assms(3) ass1 path-image-reversepath
    by force
    have 3: line-integral F basis ?γ2 = - line-integral F basis (reversepath
?γ2)
  proof -
    have i: valid-path (reversepath ?γ2)
      using 1 C1-differentiable-imp-piecewise
      by (auto simp add: valid-path-def)
    show ?thesis
      using line-integral-on-reverse-path(1)[OF i] integ-rev-exist-b
      using ass1 assms(4) line-integral-sum-basis(2) by fastforce
    qed
    show real-of-int k1 * line-integral F basis γ1 = real-of-int ?k2 * line-integral
F basis ?γ2 ∧
      line-integral-exists F basis γ1
      using assms reparam-eq-line-integrals-basis[OF 0 1 2 assms(4)]
      ass2 ass3 3
      using ass1 integ-rev-exist-b by auto
  next
    assume ?k2 ≠ 1
    then have ass3: ?k2 = -1
      using bound2 ass1 f(3) unfolding boundary-chain-def by force
    then have 0: reparam γ1 ?γ2
      using ass1 ass2 f(2) by auto
    have 1: ?γ2 piecewise-C1-differentiable-on {0..1}
      using f(3) assms(2) ass1
      by force
    have 2: ∀ b ∈ basis. continuous-on (path-image ?γ2) (λx. F x · b)
      using f(3) assms(3) ass1
      by force
    show real-of-int k1 * line-integral F basis γ1 = real-of-int ?k2 * line-integral
F basis ?γ2 ∧
      line-integral-exists F basis γ1
      using assms reparam-eq-line-integrals-basis[OF 0 1 2 assms(4)]
      ass2 ass3
      using ass1 integ-exist-b by auto
    qed
    then show real-of-int k1 * line-integral F basis γ1 = (case f (k1, γ1) of
(k2, γ2) ⇒ real-of-int k2 * line-integral F basis γ2) ∧
      line-integral-exists F basis γ1
      by (simp add: case-prod-beta')
    qed
  }
  then show ∧x. x ∈ one-chain1 ⇒
    (case x of (k, γ) ⇒ (real-of-int k * line-integral F basis
γ) = (case f x of (k, γ) ⇒ real-of-int k * line-integral F basis γ) ∧
      line-integral-exists F basis γ)
    by (auto simp add: case-prod-beta')
  qed

```

```

show one-chain-line-integral F basis one-chain1 = one-chain-line-integral F basis
one-chain2
  using 0 by (simp add: one-chain-line-integral-def sum-bij[OF f(1) - f(3)])
prod.case-eq-if)
  show  $\forall (k, \gamma) \in \text{one-chain1}. \text{line-integral-exists } F \text{ basis } \gamma$ 
  using 0 by blast
qed

```

```

lemma path-image-rec-join:
  fixes  $\gamma :: \text{real} \Rightarrow (\text{real} \times \text{real})$ 
  fixes  $k :: \text{int}$ 
  fixes  $l$ 
  shows  $\bigwedge k \gamma. (k, \gamma) \in \text{set } l \implies \text{valid-chain-list } l \implies \text{path-image } \gamma \subseteq \text{path-image}$ 
(rec-join l)
proof(induction l)
  case Nil
  then show ?case by auto
next
  case ass: (Cons a l)
  obtain  $k' \gamma'$  where  $a = (k', \gamma')$  by force
  have  $\text{path-image } \gamma \subseteq \text{path-image } (\text{rec-join } ((k', \gamma') \# l))$ 
  proof(cases)
    assume  $l = []$ 
    then show  $\text{path-image } \gamma \subseteq \text{path-image } (\text{rec-join } ((k', \gamma') \# l))$ 
    using ass(2-3) a
    by(auto simp add:)
  next
    assume  $l \neq []$ 
    then obtain  $b \ l'$  where  $b \# l = l = b \# l'$ 
    by (meson rec-join.elims)
    obtain  $k'' \gamma''$  where  $b = (k'', \gamma'')$  by force
    show ?thesis
    using ass path-image-reversepath b-l path-image-join
    by(fastforce simp add: a)
  qed
  then show ?case
  using  $a$  by auto
qed

```

```

lemma path-image-rec-join-2:
  fixes  $l$ 
  shows  $l \neq [] \implies \text{valid-chain-list } l \implies \text{path-image } (\text{rec-join } l) \subseteq (\bigcup (k, \gamma) \in \text{set}$ 
l. path-image } \gamma)
proof(induction l)
  case Nil
  then show ?case by auto
next
  case ass: (Cons a l)
  obtain  $k' \gamma'$  where  $a = (k', \gamma')$  by force

```

```

have path-image (rec-join (a # l))  $\subseteq$  ( $\bigcup (k, y) \in \text{set } (a \# l)$ ). path-image y)
proof(cases)
  assume l=[]
  then show path-image (rec-join (a # l))  $\subseteq$  ( $\bigcup (k, y) \in \text{set } (a \# l)$ ). path-image
y)
  using step a by(auto simp add:)
next
  assume l $\neq$ []
  then obtain b l' where b-l: l = b # l'
  by (meson rec-join.elims)
  obtain k''  $\gamma''$  where b: b = (k'', $\gamma''$ ) by force
  show ?thesis
  using ass
    path-image-reversepath b-l path-image-join
  apply(auto simp add: a)
  apply blast
  by fastforce
qed
then show ?case
  using a by auto
qed

```

```

lemma continuous-on-closed-UN:
  assumes finite S
  shows (( $\bigwedge s. s \in S \implies \text{closed } s$ )  $\implies$  ( $\bigwedge s. s \in S \implies \text{continuous-on } s f$ )  $\implies$ 
continuous-on ( $\bigcup S$ ) f)
  using assms
proof(induction S)
  case empty
  then show ?case by auto
next
  case (insert x F)
  then show ?case using continuous-on-closed-Un closed-Union
  by (simp add: closed-Union continuous-on-closed-Un)
qed

```

```

lemma chain-reparam-weak-path-line-integral:
  assumes path-eq-chain: chain-reparam-weak-path  $\gamma$  one-chain and
  boundary-chain: boundary-chain one-chain and
  line-integral-exists:  $\forall b \in \text{basis}. \forall (k::\text{int}, \gamma) \in \text{one-chain}. \text{line-integral-exists } F$ 
{b}  $\gamma$  and
  valid-path:  $\forall (k::\text{int}, \gamma) \in \text{one-chain}. \text{valid-path } \gamma$  and
  finite-basis: finite basis and
  cont:  $\forall b \in \text{basis}. \forall (k, \gamma_2) \in \text{one-chain}. \text{continuous-on } (\text{path-image } \gamma_2) (\lambda x. F x$ 
. b) and
  finite-one-chain: finite one-chain
shows line-integral F basis  $\gamma = \text{one-chain-line-integral } F$  basis one-chain
line-integral-exists F basis  $\gamma$ 

```



```

proof –
  obtain  $l$  where  $l$ -props: set  $l = \text{one-chain distinct } l \text{ reparam } \gamma \text{ (rec-join } l)$ 
  valid-chain-list  $l \neq []$ 
  using chain-reparam-weak-path-def assms
  by auto
  have  $bchain$ - $l$ : boundary-chain (set  $l$ )
  using  $l$ -props boundary-chain
  by (simp add: boundary-chain-def)
  have cont-forall:  $\forall b \in \text{basis}. \text{continuous-on } (\bigcup (k, \gamma) \in \text{one-chain}. \text{path-image } \gamma)$ 
  ( $\lambda x. F x \cdot b$ )
  proof
    fix  $b$ 
    assume ass:  $b \in \text{basis}$ 
    have continuous-on ( $\bigcup ((\text{path-image} \circ \text{snd}) \text{ ‘one-chain}))$  ( $\lambda x. F x \cdot b$ )
      apply(rule continuous-on-closed-UN[where  $?S = (\text{path-image} \circ \text{snd}) \text{ ‘one-chain}$  ]])
    proof
      show finite one-chain using finite-one-chain by auto
      show  $\bigwedge s. s \in (\text{path-image} \circ \text{snd}) \text{ ‘one-chain} \implies \text{closed } s$ 
        using closed-valid-path-image[OF] valid-path
        by fastforce
      show  $\bigwedge s. s \in (\text{path-image} \circ \text{snd}) \text{ ‘one-chain} \implies \text{continuous-on } s$  ( $\lambda x. F x \cdot b$ )
        using cont ass by force
    qed
  then show continuous-on ( $\bigcup (k, \gamma) \in \text{one-chain}. \text{path-image } \gamma)$  ( $\lambda x. F x \cdot b$ )
    by (auto simp add: Union-eq case-prod-beta)
  qed
show line-integral-exists  $F$  basis  $\gamma$ 
proof (rule reparam-eq-line-integrals-basis[OF l-props(3) - - finite-basis])
  let  $? \gamma 2.0 = \text{rec-join } l$ 
  show  $? \gamma 2.0$  piecewise-C1-differentiable-on  $\{0..1\}$ 
    apply(simp only: valid-path-def[symmetric])
    apply(rule joined-is-valid)
    using assms l-props by auto
  show  $\forall b \in \text{basis}. \text{continuous-on } (\text{path-image } (\text{rec-join } l))$  ( $\lambda x. F x \cdot b$ ) using
path-image-rec-join-2[OF l-props(5) l-props(4)] l-props(1)
    using cont-forall continuous-on-subset by blast
  show  $\forall b \in \text{basis}. \text{line-integral-exists } F \{b\}$  (rec-join  $l$ )
  proof
    fix  $b$ 
    assume ass:  $b \in \text{basis}$ 
    show line-integral-exists  $F \{b\}$  (rec-join  $l$ )
    proof (rule line-integral-exists-on-rec-join)
      show boundary-chain (set  $l$ )
        using  $l$ -props boundary-chain by auto
      show valid-chain-list  $l$  using  $l$ -props by auto
      show  $\bigwedge k \gamma. (k, \gamma) \in \text{set } l \implies \text{valid-path } \gamma$  using  $l$ -props assms by auto
      show  $\forall (k, \gamma) \in \text{set } l. \text{line-integral-exists } F \{b\} \gamma$  using  $l$ -props line-integral-exists

```

```

ass by blast
  qed
  qed
  qed
  show line-integral F basis  $\gamma$  = one-chain-line-integral F basis one-chain
  proof-
    have i: line-integral F basis (rec-join l) = one-chain-line-integral F basis
one-chain
    proof (rule one-chain-line-integral-rec-join)
      show set l = one-chain distinct l valid-chain-list l using l-props by auto
      show boundary-chain one-chain using boundary-chain by auto
      show  $\forall (k, \gamma) \in \text{one-chain}. \text{line-integral-exists } F \text{ basis } \gamma$ 
        using line-integral-sum-basis(2)[OF finite-basis] line-integral-exists by blast
      show  $\forall (k, \gamma) \in \text{one-chain}. \text{valid-path } \gamma$  using valid-path by auto
      show finite basis using finite-basis by auto
    qed
    have ii: line-integral F basis  $\gamma$  = line-integral F basis (rec-join l)
    proof (rule reparam-eq-line-integrals-basis[OF l-props(3) - - finite-basis])
      let ? $\gamma$ 2.0=rec-join l
      show ? $\gamma$ 2.0 piecewise-C1-differentiable-on {0..1}
        apply (simp only: valid-path-def[symmetric])
        apply (rule joined-is-valid)
        using assms l-props by auto
      show  $\forall b \in \text{basis}. \text{continuous-on (path-image (rec-join l)) } (\lambda x. F x \cdot b)$  using
path-image-rec-join-2[OF l-props(5) l-props(4)] l-props(1)
        using cont-forall continuous-on-subset by blast
      show  $\forall b \in \text{basis}. \text{line-integral-exists } F \{b\} (\text{rec-join } l)$ 
    proof
      fix b
      assume ass: b ∈ basis
      show line-integral-exists F {b} (rec-join l)
      proof (rule line-integral-exists-on-rec-join)
        show boundary-chain (set l)
          using l-props boundary-chain by auto
        show valid-chain-list l using l-props by auto
        show  $\bigwedge k \gamma. (k, \gamma) \in \text{set } l \implies \text{valid-path } \gamma$  using l-props assms by auto
        show  $\forall (k, \gamma) \in \text{set } l. \text{line-integral-exists } F \{b\} \gamma$  using l-props line-integral-exists
      qed
    qed
  ass by blast
  qed
  qed
  qed
  show line-integral F basis  $\gamma$  = one-chain-line-integral F basis one-chain using
i ii by auto
  qed
  qed

```

definition *chain-reparam-chain'* where
chain-reparam-chain' one-chain1 subdiv
 $\equiv \exists f. ((\bigcup (f \text{ ' one-chain1})) = \text{subdiv}) \wedge$

$(\forall \text{cube} \in \text{one-chain1}. \text{chain-reparam-weak-path } (\text{rec-join } [\text{cube}]) (f \text{ cube})) \wedge$
 $(\forall p \in \text{one-chain1}. \forall p' \in \text{one-chain1}. p \neq p' \longrightarrow f p \cap f p' = \{\}) \wedge$
 $(\forall x \in \text{one-chain1}. \text{finite } (f x))$

lemma *chain-reparam-chain'-imp-finite-subdiv*:

assumes *finite one-chain1*
chain-reparam-chain' one-chain1 subdiv
shows *finite subdiv*
using *assms*
by (*auto simp add: chain-reparam-chain'-def*)

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: $\forall b \in \text{basis}. \forall (k :: \text{int}, \gamma) \in \text{subdiv}. \text{line-integral-exists}$
 $F \{b\} \gamma$ **and**
valid-path: $\forall (k, \gamma) \in \text{subdiv}. \text{valid-path } \gamma$ and
valid-path-2: $\forall (k, \gamma) \in \text{one-chain1}. \text{valid-path } \gamma$ and
finite-chain1: finite one-chain1 and
finite-basis: finite basis and
cont-field: $\forall b \in \text{basis}. \forall (k, \gamma2) \in \text{subdiv}. \text{continuous-on } (\text{path-image } \gamma2) (\lambda x. F$
 $x \cdot b)$
shows *one-chain-line-integral F basis one-chain1 = one-chain-line-integral F*
basis subdiv
 $\forall (k, \gamma) \in \text{one-chain1}. \text{line-integral-exists } F \text{ basis } \gamma$

proof –

obtain *f* **where** *f*-props:

$((\bigcup (f \text{ 'one-chain1})) = \text{subdiv})$
 $(\forall \text{cube} \in \text{one-chain1}. \text{chain-reparam-weak-path } (\text{rec-join } [\text{cube}]) (f \text{ cube}))$
 $(\forall p \in \text{one-chain1}. \forall p' \in \text{one-chain1}. p \neq p' \longrightarrow f p \cap f p' = \{\})$
 $(\forall x \in \text{one-chain1}. \text{finite } (f x))$
using *chain1-eq-chain2*
by (*auto simp add: chain-reparam-chain'-def*)

then have *0: one-chain-line-integral F basis subdiv = one-chain-line-integral F*
basis $(\bigcup (f \text{ 'one-chain1}))$

by *auto*

{fix *k* γ

assume *ass: $(k, \gamma) \in \text{one-chain1}$*

have *line-integral-exists F basis $\gamma \wedge$*

*one-chain-line-integral F basis $(f (k, \gamma)) = k * \text{line-integral F basis } \gamma$*

proof (*cases k = 1*)

assume *k-eq-1: k = 1*

then have *i: chain-reparam-weak-path $\gamma (f(k, \gamma))$*

using *f-props(2) ass by auto*

have *ii: boundary-chain $(f(k, \gamma))$*

using *f-props(1) ass assms unfolding boundary-chain-def*

by *blast*

```

have iii:  $\forall b \in \text{basis}. \forall (k, \gamma) \in f (k, \gamma). \text{line-integral-exists } F \{b\} \gamma$ 
  using f-props(1) ass assms
  by blast
have iv:  $\forall (k, \gamma) \in f (k, \gamma). \text{valid-path } \gamma$ 
  using f-props(1) ass assms
  by blast
have v:  $\forall b \in \text{basis}. \forall (k, \gamma^2) \in f (k, \gamma). \text{continuous-on } (\text{path-image } \gamma^2) (\lambda x. F$ 
x  $\cdot b)$ 
  using f-props(1) ass assms
  by blast
have line-integral-exists F basis  $\gamma \wedge \text{one-chain-line-integral } F \text{ basis } (f (k, \gamma))$ 
= line-integral F basis  $\gamma$ 
  using chain-reparam-weak-path-line-integral[OF i ii iii iv finite-basis v] ass
f-props(4)
  by (auto)
then show line-integral-exists F basis  $\gamma \wedge \text{one-chain-line-integral } F \text{ basis } (f$ 
(k,  $\gamma)) = k * \text{line-integral } F \text{ basis } \gamma$ 
  using k-eq-1 by auto
next
assume k  $\neq 1$ 
then have k-eq-neg1: k = -1
  using ass boundary-chain1
  by (auto simp add: boundary-chain-def)
then have i: chain-reparam-weak-path (reversepath  $\gamma$ ) (f(k, $\gamma$ ))
  using f-props(2) ass by auto
have ii: boundary-chain (f(k, $\gamma$ ))
  using f-props(1) ass assms unfolding boundary-chain-def
  by blast
have iii:  $\forall b \in \text{basis}. \forall (k, \gamma) \in f (k, \gamma). \text{line-integral-exists } F \{b\} \gamma$ 
  using f-props(1) ass assms
  by blast
have iv:  $\forall (k, \gamma) \in f (k, \gamma). \text{valid-path } \gamma$ 
  using f-props(1) ass assms by blast
have v:  $\forall b \in \text{basis}. \forall (k, \gamma^2) \in f (k, \gamma). \text{continuous-on } (\text{path-image } \gamma^2) (\lambda x. F$ 
x  $\cdot b)$ 
  using f-props(1) ass assms by blast
have x: line-integral-exists F basis (reversepath  $\gamma$ )  $\wedge \text{one-chain-line-integral } F$ 
basis (f (k,  $\gamma$ )) = line-integral F basis (reversepath  $\gamma$ )
  using chain-reparam-weak-path-line-integral[OF i ii iii iv finite-basis v] ass
f-props(4)
  by auto
have valid-path (reversepath  $\gamma$ )
  using f-props(1) ass assms by auto
then have line-integral-exists F basis  $\gamma \wedge \text{one-chain-line-integral } F \text{ basis } (f$ 
(k,  $\gamma)) = - (\text{line-integral } F \text{ basis } \gamma)$ 
  using line-integral-on-reverse-path reversepath-reversepath x ass
  by metis
then show line-integral-exists F basis  $\gamma \wedge \text{one-chain-line-integral } F \text{ basis } (f$ 
(k:int,  $\gamma)) = k * \text{line-integral } F \text{ basis } \gamma$ 

```

```

    using k-eq-neg1 by auto
  qed}
  note cube-line-integ = this
  have finite-chain2: finite subdiv
    using finite-chain1 f-props(1) f-props(4) by auto
  show line-integral-exists-on-chain1:  $\forall (k, \gamma) \in \text{one-chain1}. \text{line-integral-exists } F$ 
  basis  $\gamma$ 
    using cube-line-integ by auto
  have 1:  $\text{one-chain-line-integral } F \text{ basis } (\bigcup (f \text{ ' one-chain1})) = \text{one-chain-line-integral}$ 
   $F \text{ basis one-chain1}$ 
  proof -
    have 0:  $\text{one-chain-line-integral } F \text{ basis } (\bigcup (f \text{ ' one-chain1})) =$ 
       $(\sum \text{one-chain} \in (f \text{ ' one-chain1}). \text{one-chain-line-integral } F$ 
  basis one-chain)
    proof -
      have finite:  $\forall \text{chain} \in (f \text{ ' one-chain1}). \text{finite chain}$ 
      using f-props(1) finite-chain2
      by (meson Sup-upper finite-subset)
      have disj:  $\forall A \in f \text{ ' one-chain1}. \forall B \in f \text{ ' one-chain1}. A \neq B \longrightarrow A \cap B = \{\}$ 
      apply (simp add: image-def)
      using f-props(3)
      by metis
      show  $\text{one-chain-line-integral } F \text{ basis } (\bigcup (f \text{ ' one-chain1})) =$ 
       $(\sum \text{one-chain} \in (f \text{ ' one-chain1}). \text{one-chain-line-integral}$ 
   $F \text{ basis one-chain})$ 
      using Groups-Big.comm-monoid-add-class.sum.Union-disjoint[OF finite
  disj]
      one-chain-line-integral-def
      by auto
    qed
  have 1:  $(\sum \text{one-chain} \in (f \text{ ' one-chain1}). \text{one-chain-line-integral } F \text{ basis one-chain})$ 
  =
       $\text{one-chain-line-integral } F \text{ basis one-chain1}$ 
  proof -
    have  $(\sum \text{one-chain} \in (f \text{ ' one-chain1}). \text{one-chain-line-integral } F \text{ basis one-chain})$ 
  =
       $(\sum (k, \gamma) \in \text{one-chain1}. k * (\text{line-integral } F \text{ basis } \gamma))$ 
  proof -
    have  $i: (\sum \text{one-chain} \in (f \text{ ' } (\text{one-chain1} - \{p. f p = \{\}\})). \text{one-chain-line-integral}$ 
   $F \text{ basis one-chain}) =$ 
       $(\sum (k, \gamma) \in \text{one-chain1} - \{p. f p = \{\}\}. k * (\text{line-integral}$ 
   $F \text{ basis } \gamma))$ 
  proof -
    have  $\text{inj-on } f \text{ } (\text{one-chain1} - \{p. f p = \{\}\})$ 
    unfolding inj-on-def
    using f-props(3) by force
    then have 0:  $(\sum \text{one-chain} \in (f \text{ ' } (\text{one-chain1} - \{p. f p = \{\}\})).$ 
   $\text{one-chain-line-integral } F \text{ basis one-chain})$ 
      =  $(\sum (k, \gamma) \in (\text{one-chain1} - \{p. f p$ 

```

```

= {})). one-chain-line-integral F basis (f (k, γ))
  using Groups-Big.comm-monoid-add-class.sum.reindex
  by auto
  have  $\bigwedge k \gamma. (k, \gamma) \in (\text{one-chain1} - \{p. f p = \{\}\}) \implies$ 
    one-chain-line-integral F basis (f(k, γ))
= k* (line-integral F basis γ)
  using cube-line-integ by auto
  then have  $(\sum (k, \gamma) \in (\text{one-chain1} - \{p. f p = \{\}\})).$  one-chain-line-integral
F basis (f (k, γ))
    =  $(\sum (k, \gamma) \in (\text{one-chain1} - \{p. f p = \{\}\})).$  k* (line-integral F
basis γ)
  by (auto intro!: Finite-Cartesian-Product.sum-cong-aux)
  then show  $(\sum \text{one-chain} \in (f' (\text{one-chain1} - \{p. f p = \{\}\})).$ 
one-chain-line-integral F basis one-chain) =
     $(\sum (k, \gamma) \in (\text{one-chain1} - \{p. f p =$ 
 $\{\}\}).$  k* (line-integral F basis γ))
  using 0 by auto
  qed
  have  $\bigwedge (k::int) \gamma. (k, \gamma) \in \text{one-chain1} \implies (f (k, \gamma) = \{\}) \implies (k, \gamma) \in$ 
 $\{(k', \gamma'). k' * (\text{line-integral F basis } \gamma') = 0\}$ 
  proof-
  fix k::int
  fix γ::real⇒real*real
  assume ass:(k, γ) ∈ one-chain1
  (f (k, γ) = {})
  then have zero-line-integral:one-chain-line-integral F basis (f (k, γ)) = 0
  using one-chain-line-integral-def
  by auto
  show  $(k, \gamma) \in \{(k'::int, \gamma'). k' * \text{line-integral F basis } \gamma' = 0\}$ 
  using zero-line-integral cube-line-integ ass
  by force
  qed
  then have ii:(∑ one-chain ∈ (f' (one-chain1 - {p. f p = {}})). one-chain-line-integral
F basis one-chain) =
     $(\sum \text{one-chain} \in (f' (\text{one-chain1})).$ 
one-chain-line-integral F basis one-chain)
  proof -
  have  $\bigwedge \text{one-chain}. \text{one-chain} \in (f' (\text{one-chain1})) - (f' (\text{one-chain1} -$ 
 $\{p. f p = \{\}\})) \implies$ 
    one-chain-line-integral F basis
one-chain = 0
  proof -
  fix one-chain
  assume one-chain ∈ (f' (one-chain1)) - (f' (one-chain1 - {p. f p =
 $\{\}\}))$ 
  then have one-chain = {}
  by auto
  then show one-chain-line-integral F basis one-chain = 0
  by (auto simp add: one-chain-line-integral-def)

```

qed
then have $0:(\sum \text{one-chain} \in f'(\text{one-chain1}) - (f'(\text{one-chain1} - \{p.f p = \{\}\}))). \text{one-chain-line-integral } F \text{ basis one-chain}$
 $= 0$
using *Groups-Big.comm-monoid-add-class.sum.neutral*
by auto
then have $(\sum \text{one-chain} \in f'(\text{one-chain1}). \text{one-chain-line-integral } F$
basis one-chain)
 $- (\sum \text{one-chain} \in (f'(\text{one-chain1} -$
 $\{p.f p = \{\}\})). \text{one-chain-line-integral } F \text{ basis one-chain})$
 $= 0$
proof –
have *finte: finite (f' one-chain1)* **using** *finite-chain1* **by auto**
show *?thesis*
using *Groups-Big.sum-diff[OF finte, of (f' (one-chain1 - {p.f p =*
 $\{\}\}))$
 $\text{one-chain-line-integral } F \text{ basis}]$
 0
by auto
qed
then show $(\sum \text{one-chain} \in (f'(\text{one-chain1} - \{p.f p = \{\}\}))).$
 $\text{one-chain-line-integral } F \text{ basis one-chain} =$
 $(\sum \text{one-chain} \in (f'(\text{one-chain1})).$
 $\text{one-chain-line-integral } F \text{ basis one-chain})$
by auto
qed
have $\bigwedge (k::\text{int}) \gamma. (k, \gamma) \in \text{one-chain1} \implies (f(k, \gamma) = \{\}) \implies f(k, \gamma) \in$
 $\{\text{chain. one-chain-line-integral } F \text{ basis chain} = 0\}$
proof–
fix $k::\text{int}$
fix $\gamma::\text{real} \implies \text{real} * \text{real}$
assume $\text{ass}:(k, \gamma) \in \text{one-chain1} \ (f(k, \gamma) = \{\})$
then have $\text{one-chain-line-integral } F \text{ basis } (f(k, \gamma)) = 0$
using *one-chain-line-integral-def*
by auto
then show $f(k, \gamma) \in \{p'. (\text{one-chain-line-integral } F \text{ basis } p' = 0)\}$
by auto
qed
then have $\text{iii}:(\sum (k::\text{int}, \gamma) \in \text{one-chain1} - \{p.f p = \{\}\}). k * (\text{line-integral } F$
 $\text{basis } \gamma))$
 $= (\sum (k::\text{int}, \gamma) \in \text{one-chain1}. k * (\text{line-integral}$
 $F \text{ basis } \gamma))$
proof–
have $\bigwedge k \gamma. (k, \gamma) \in \text{one-chain1} - (\text{one-chain1} - \{p.f p = \{\}\})$
 $\implies k * (\text{line-integral } F \text{ basis } \gamma) = 0$
proof–
fix $k \gamma$
assume $\text{ass}:(k, \gamma) \in \text{one-chain1} - (\text{one-chain1} - \{p.f p = \{\}\})$
then have $f(k, \gamma) = \{\}$

by auto
 then have one-chain-line-integral F basis $(f(k, \gamma)) = 0$
 by (auto simp add: one-chain-line-integral-def)
 then have zero-line-integral:one-chain-line-integral F basis $(f(k, \gamma)) = 0$
 using one-chain-line-integral-def
 by auto
 then show $k * (\text{line-integral F basis } \gamma) = 0$
 using zero-line-integral cube-line-integ ass
 by force
 qed
 then have $\forall (k::\text{int}, \gamma) \in \text{one-chain1} - (\text{one-chain1} - \{p. f p = \{\}\})$.
 $k * (\text{line-integral F basis } \gamma) = 0$ by auto
 then have $(\sum (k::\text{int}, \gamma) \in \text{one-chain1} - (\text{one-chain1} - \{p. f p = \{\}\}))$.
 $k * (\text{line-integral F basis } \gamma) = 0$
 using Groups-Big.comm-monoid-add-class.sum.neutral
 [of one-chain1 - (one-chain1 - {p. f p = {}})] $(\lambda(k::\text{int}, \gamma). k * (\text{line-integral F basis } \gamma))$
 by (simp add: split-beta)
 then have $(\sum (k::\text{int}, \gamma) \in \text{one-chain1}. k * (\text{line-integral F basis } \gamma) - (\sum (k::\text{int}, \gamma) \in (\text{one-chain1} - \{p. f p = \{\}\}). k * (\text{line-integral F basis } \gamma)) = 0$
 using Groups-Big.sum-diff[OF finite-chain1]
 by (metis (no-types) Diff-subset $(\sum (k, \gamma) \in \text{one-chain1} - (\text{one-chain1} - \{p. f p = \{\}\}). k * \text{line-integral F basis } \gamma) = 0$) $(\bigwedge f B. B \subseteq \text{one-chain1} \implies \text{sum } f (\text{one-chain1} - B) = \text{sum } f \text{ one-chain1} - \text{sum } f B)$)
 then show ?thesis by auto
 qed
 show ?thesis using i ii iii by auto
 qed
 then show ?thesis using one-chain-line-integral-def by auto
 qed
 show ?thesis using 0 1 by auto
 qed
 show one-chain-line-integral F basis one-chain1 = one-chain-line-integral F basis subdiv using 0 1 by auto
 qed

lemma chain-reparam-chain'-line-integral-smooth-cubes:

assumes chain-reparam-chain' one-chain1 one-chain2
 $\forall (k2, \gamma2) \in \text{one-chain2}. \gamma2 \text{ C1-differentiable-on } \{0..1\}$
 $\forall b \in \text{basis}. \forall (k2, \gamma2) \in \text{one-chain2}. \text{continuous-on } (\text{path-image } \gamma2) (\lambda x. F x \cdot b)$
 finite basis
 finite one-chain1
 boundary-chain one-chain1
 boundary-chain one-chain2
 $\forall (k, \gamma) \in \text{one-chain1}. \text{valid-path } \gamma$
 shows one-chain-line-integral F basis one-chain1 = one-chain-line-integral F basis one-chain2

$\forall (k, \gamma) \in \text{one-chain1}. \text{line-integral-exists } F \text{ basis } \gamma$
proof –
 {**fix** b
 assume $b \in \text{basis}$
 fix $k \ \gamma$
 assume $(k, \gamma) \in \text{one-chain2}$
 have $\text{line-integral-exists } F \ \{b\} \ \gamma$
 apply (*rule line-integral-exists-smooth*)
 using $\langle (k, \gamma) \in \text{one-chain2} \rangle \text{ assms}(2)$ **apply** *blast*
 using *assms*
 using $\langle (k, \gamma) \in \text{one-chain2} \rangle \langle b \in \text{basis} \rangle$ **apply** *blast*
 using $\langle b \in \text{basis} \rangle$ **by** *blast*}
then have $a: \forall b \in \text{basis}. \forall (k, \gamma) \in \text{one-chain2}. \text{line-integral-exists } F \ \{b\} \ \gamma$
 by *auto*
have $b: \forall (k2, \gamma2) \in \text{one-chain2}. \text{valid-path } \gamma2$
 using *assms(2)*
 by (*simp add: C1-differentiable-imp-piecewise case-prod-beta valid-path-def*)

show $\text{one-chain-line-integral } F \text{ basis } \text{one-chain1} = \text{one-chain-line-integral } F \text{ basis } \text{one-chain2}$
 by (*rule chain-reparam-chain'-line-integral[OF assms(1) assms(6) assms(7) a b assms(8) assms(5) assms(4) assms(3)]*)
 show $\forall (k, \gamma) \in \text{one-chain1}. \text{line-integral-exists } F \text{ basis } \gamma$
 by (*rule chain-reparam-chain'-line-integral[OF assms(1) assms(6) assms(7) a b assms(8) assms(5) assms(4) assms(3)]*)
qed

lemma *chain-subdiv-path-pathimg-subset*:
 assumes *chain-subdiv-path* γ *subdiv*
 shows $\forall (k', \gamma') \in \text{subdiv}. (\text{path-image } \gamma') \subseteq \text{path-image } \gamma$
 using *assms chain-subdiv-path.simps path-image-rec-join*
 by (*auto simp add: chain-subdiv-path.simps*)

lemma *reparam-path-image*:
 assumes *reparam* $\gamma1 \ \gamma2$
 shows $\text{path-image } \gamma1 = \text{path-image } \gamma2$
 using *assms*
 apply (*auto simp add: reparam-def path-image-def image-def bij-betw-def*)
 apply (*force dest!: equalityD2*)
 done

lemma *chain-reparam-weak-path-pathimg-subset*:
 assumes *chain-reparam-weak-path* γ *subdiv*
 shows $\forall (k', \gamma') \in \text{subdiv}. (\text{path-image } \gamma') \subseteq \text{path-image } \gamma$
 using *assms*
 apply (*auto simp add: chain-reparam-weak-path-def*)
 using *path-image-rec-join reparam-path-image* **by** *blast*

lemma *chain-subdiv-chain-pathimg-subset'*:

assumes *chain-subdiv-chain one-chain subdiv*
assumes $(k, \gamma) \in \text{subdiv}$
shows $\exists k' \gamma'. (k', \gamma') \in \text{one-chain} \wedge \text{path-image } \gamma \subseteq \text{path-image } \gamma'$
using *assms unfolding chain-subdiv-chain-def pairwise-def*
apply *auto*
by (*metis chain-subdiv-path.cases coeff-cube-to-path.simps path-image-rec-join path-image-reversepath*)

lemma *chain-subdiv-chain-pathimg-subset*:
assumes *chain-subdiv-chain one-chain subdiv*
shows $\bigcup (\text{path-image } \cdot \{ \gamma. \exists k. (k, \gamma) \in \text{subdiv} \}) \subseteq \bigcup (\text{path-image } \cdot \{ \gamma. \exists k. (k, \gamma) \in \text{one-chain} \})$
using *assms unfolding chain-subdiv-chain-def pairwise-def*
apply *auto*
by (*metis UN-iff assms chain-subdiv-chain-pathimg-subset' subsetCE case-prodI2*)

lemma *chain-reparam-chain'-pathimg-subset'*:
assumes *chain-reparam-chain' one-chain subdiv*
assumes $(k, \gamma) \in \text{subdiv}$
shows $\exists k' \gamma'. (k', \gamma') \in \text{one-chain} \wedge \text{path-image } \gamma \subseteq \text{path-image } \gamma'$
using *assms chain-reparam-weak-path-pathimg-subset*
apply (*auto simp add: chain-reparam-chain'-def set-eq-iff*)
using *path-image-reversepath case-prodE case-prodD old.prod.exhaust*
apply (*simp add: list.distinct(1) list.inject rec-join.elims*)
by (*smt case-prodD coeff-cube-to-path.simps rec-join.simps(2) reversepath-simps(2) surj-pair*)

definition *common-reparam-exists:: (int × (real ⇒ real × real)) set ⇒ (int × (real ⇒ real × real)) set ⇒ bool* **where**
common-reparam-exists one-chain1 one-chain2 \equiv
 $(\exists \text{subdiv } ps1 \text{ } ps2.$
 $\text{chain-reparam-chain}' (\text{one-chain1} - ps1) \text{ subdiv} \wedge$
 $\text{chain-reparam-chain}' (\text{one-chain2} - ps2) \text{ subdiv} \wedge$
 $(\forall (k, \gamma) \in \text{subdiv}. \gamma \text{ C1-differentiable-on } \{0..1\}) \wedge$
 $\text{boundary-chain subdiv} \wedge$
 $(\forall (k, \gamma) \in ps1. \text{point-path } \gamma) \wedge$
 $(\forall (k, \gamma) \in ps2. \text{point-path } \gamma))$

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)
 $\forall (k2, \gamma2) \in \text{one-chain2}. \forall b \in \text{basis}. \text{continuous-on } (\text{path-image } \gamma2) (\lambda x. F x \cdot$
b)
(common-reparam-exists one-chain1 one-chain2)
 $(\forall (k, \gamma) \in \text{one-chain1}. \text{valid-path } \gamma)$

$(\forall (k, \gamma) \in \text{one-chain2}. \text{valid-path } \gamma)$
shows $\text{one-chain-line-integral } F \text{ basis one-chain1} = \text{one-chain-line-integral } F$
 basis one-chain2
 $\forall (k, \gamma) \in \text{one-chain1}. \text{line-integral-exists } F \text{ basis } \gamma$
proof –
obtain subdiv ps1 ps2 **where** props :
 $\text{chain-reparam-chain}' (\text{one-chain1} - \text{ps1}) \text{ subdiv}$
 $\text{chain-reparam-chain}' (\text{one-chain2} - \text{ps2}) \text{ subdiv}$
 $(\forall (k, \gamma) \in \text{subdiv}. \gamma \text{ C1-differentiable-on } \{0..1\})$
 $\text{boundary-chain subdiv}$
 $(\forall (k, \gamma) \in \text{ps1}. \text{point-path } \gamma)$
 $(\forall (k, \gamma) \in \text{ps2}. \text{point-path } \gamma)$
using assms
by $(\text{auto simp add: common-reparam-exists-def})$
have $\text{subdiv-valid: } (\forall (k, \gamma) \in \text{subdiv}. \text{valid-path } \gamma)$
apply $(\text{simp add: valid-path-def})$
using $\text{props}(3)$
using $\text{C1-differentiable-imp-piecewise}$ **by** blast
have $\text{onechain-boundary1: boundary-chain } (\text{one-chain1} - \text{ps1})$ **using** $\text{assms}(4)$
by $(\text{auto simp add: boundary-chain-def})$
have $\text{onechain-boundary2: boundary-chain } (\text{one-chain2} - \text{ps1})$ **using** $\text{assms}(5)$
by $(\text{auto simp add: boundary-chain-def})$
{fix $k2 \gamma2 b$
assume $\text{ass: } (k2, \gamma2) \in \text{subdiv } b \in \text{basis}$
have $\bigwedge k \gamma. (k, \gamma) \in \text{subdiv} \implies \exists k' \gamma'. (k', \gamma') \in \text{one-chain2} \wedge \text{path-image } \gamma$
 $\subseteq \text{path-image } \gamma'$
by $(\text{meson chain-reparam-chain}'\text{-pathimg-subset}' \text{ props Diff-subset subsetCE})$
then have $\text{continuous-on } (\text{path-image } \gamma2) (\lambda x. F x \cdot b)$
using $\text{assms}(6) \text{ continuous-on-subset}[\text{where } ?f = (\lambda x. F x \cdot b)] \text{ ass}$
apply $(\text{auto simp add: subset-iff})$
by $(\text{metis (mono-tags, lifting) case-prodD})$
then have $\text{cont-field: } \forall b \in \text{basis}. \forall (k2, \gamma2) \in \text{subdiv}. \text{continuous-on } (\text{path-image } \gamma2)$
 $(\lambda x. F x \cdot b)$
by auto
have $\text{one-chain1-ps-valid: } (\forall (k, \gamma) \in \text{one-chain1} - \text{ps1}. \text{valid-path } \gamma)$ **using**
 assms **by** auto
have $\text{one-chain2-ps-valid: } (\forall (k, \gamma) \in \text{one-chain2} - \text{ps1}. \text{valid-path } \gamma)$ **using**
 assms **by** auto
have $0: \text{one-chain-line-integral } F \text{ basis } (\text{one-chain1} - \text{ps1}) = \text{one-chain-line-integral } F$
 basis subdiv
apply $(\text{rule chain-reparam-chain}'\text{-line-integral-smooth-cubes}[\text{OF props}(1) \text{ props}(3)$
 $\text{cont-field finite-basis}])$
using props assms
apply blast
using props assms
using $\text{onechain-boundary1}$ **apply** blast
using props assms
apply blast
using $\text{one-chain1-ps-valid}$ **by** blast

```

have 1:  $\forall (k, \gamma) \in (\text{one-chain1} - \text{ps1}). \text{line-integral-exists } F \text{ basis } \gamma$ 
apply (rule chain-reparam-chain'-line-integral-smooth-cubes[OF props(1) props(3)]
cont-field finite-basis)
using props assms
apply blast
using props assms
using onechain-boundary1 apply blast
using props assms
apply blast
using one-chain1-ps-valid by blast
have 2:  $\text{one-chain-line-integral } F \text{ basis } (\text{one-chain2} - \text{ps2}) = \text{one-chain-line-integral}$ 
F basis subdiv
apply (rule chain-reparam-chain'-line-integral-smooth-cubes[OF props(2) props(3)]
cont-field finite-basis)
using props assms
apply blast
apply (simp add: assms(5) boundary-chain-diff)
apply (simp add: props(4))
by (simp add: assms(9))
have 3:  $\forall (k, \gamma) \in (\text{one-chain2} - \text{ps2}). \text{line-integral-exists } F \text{ basis } \gamma$ 
apply (rule chain-reparam-chain'-line-integral-smooth-cubes[OF props(2) props(3)]
cont-field finite-basis)
using props assms
apply blast
apply (simp add: assms(5) boundary-chain-diff)
apply (simp add: props(4))
by (simp add: assms(9))
show line-int-ex-chain1:  $\forall (k, \gamma) \in \text{one-chain1}. \text{line-integral-exists } F \text{ basis } \gamma$ 
using 0
using 1 finite-basis line-integral-exists-point-path props(5) by fastforce
then show one-chain-line-integral F basis one-chain1 = one-chain-line-integral
F basis one-chain2
using 0 1 2 3
using assms(2-3) finite-basis one-chain-line-integral-point-paths props(5) props(6)
by auto
qed

```

```

definition subcube ::  $\text{real} \Rightarrow \text{real} \Rightarrow (\text{int} \times (\text{real} \Rightarrow \text{real} \times \text{real})) \Rightarrow (\text{int} \times (\text{real} \Rightarrow \text{real} \times \text{real}))$  where
  subcube a b cube = (fst cube, subpath a b (snd cube))

```

```

lemma subcube-valid-path:
  assumes valid-path (snd cube)  $a \in \{0..1\}$   $b \in \{0..1\}$ 
  shows valid-path (snd (subcube a b cube))
  using valid-path-subpath[OF assms] by (auto simp add: subcube-def)

```

```

end
theory Green
  imports Paths Derivs Integrals General-Utils

```

begin

lemma *frontier-Un-subset-Un-frontier*:

frontier ($s \cup t$) \subseteq (*frontier* s) \cup (*frontier* t)

by (*simp add: frontier-def Un-Diff*) (*auto simp add: closure-def interior-def islimpt-def*)

definition *has-partial-derivative*:: ($'a::\text{euclidean-space}$) \Rightarrow ($'b::\text{euclidean-space}$) \Rightarrow $'a \Rightarrow ('a \Rightarrow 'b) \Rightarrow ('a) \Rightarrow \text{bool}$ **where**

has-partial-derivative F *base-vec* F' a
 $\equiv ((\lambda x::'a::\text{euclidean-space}. F((a - ((a \cdot \text{base-vec}) *_R \text{base-vec})) + (x \cdot \text{base-vec}) *_R \text{base-vec}))$
has-derivative F') (*at* a)

definition *has-partial-vector-derivative*:: ($'a::\text{euclidean-space}$) \Rightarrow ($'b::\text{euclidean-space}$) \Rightarrow $'a \Rightarrow ('b) \Rightarrow ('a) \Rightarrow \text{bool}$ **where**

has-partial-vector-derivative F *base-vec* F' a
 $\equiv ((\lambda x. F((a - ((a \cdot \text{base-vec}) *_R \text{base-vec})) + x *_R \text{base-vec}))$
has-vector-derivative F') (*at* ($a \cdot \text{base-vec}$))

definition *partially-vector-differentiable* **where**

partially-vector-differentiable F *base-vec* $p \equiv (\exists F'. \text{has-partial-vector-derivative } F \text{ base-vec } F' p)$

definition *partial-vector-derivative*:: ($'a::\text{euclidean-space}$) \Rightarrow ($'b::\text{euclidean-space}$) \Rightarrow $'a \Rightarrow 'a \Rightarrow 'b$ **where**

partial-vector-derivative F *base-vec* a
 $\equiv (\text{vector-derivative } (\lambda x. F((a - ((a \cdot \text{base-vec}) *_R \text{base-vec})) + x *_R \text{base-vec}))$
base-vec) (*at* ($a \cdot \text{base-vec}$)))

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 –

obtain F' **where** F' -*prop*: $((\lambda x. F((a - ((a \cdot \text{base-vec}) *_R \text{base-vec})) + x *_R \text{base-vec}))$

has-vector-derivative F') (*at* ($a \cdot \text{base-vec}$))

using *assms partially-vector-differentiable-def has-partial-vector-derivative-def*

by *blast*

show *?thesis*

using *Derivative.differentiableI-vector[OF F'-prop]*

by (*simp add: vector-derivative-works partial-vector-derivative-def[symmetric]*
has-partial-vector-derivative-def[symmetric])

qed

lemma *fundamental-theorem-of-calculus-partial-vector*:

fixes a $b::\text{real}$ **and**

$F::('a::\text{euclidean-space} \Rightarrow 'b::\text{euclidean-space})$ **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: $i \in \text{Basis } j \in \text{Basis}$ and
no-i-component: $c \cdot i = 0$ and
has-partial-deriv: $\forall p \in D. \text{has-partial-vector-derivative } (\lambda x. (F x) \cdot j) i (F-j-i$
p) p and
domain-subset-of-D: $\{x *_R i + c \mid x. a \leq x \wedge x \leq b\} \subseteq D$
shows $((\lambda x. F-j-i(x *_R i + c)) \text{has-integral}$
 $F(b *_R i + c) \cdot j - F(a *_R i + c) \cdot j) (\text{cbox } a b)$
proof –
let *?domain* = $\{v. \exists x. a \leq x \wedge x \leq b \wedge v = x *_R i + c\}$
have $\forall x \in ?\text{domain}. \text{has-partial-vector-derivative } (\lambda x. (F x) \cdot j) i (F-j-i x) x$
using *has-partial-deriv domain-subset-of-D*
by *auto*
then have $\forall x \in (\text{cbox } a b). ((\lambda x. F(x *_R i + c) \cdot j) \text{has-vector-derivative } (F-j-i(x$
 $x *_R i + c))) (\text{at } x)$
proof(*clarsimp simp add: has-partial-vector-derivative-def*)
fix *x*::real
assume *range-of-x*: $a \leq x \wedge x \leq b$
assume *ass2*: $\forall x. (\exists z \geq a. z \leq b \wedge x = z *_R i + c) \longrightarrow$
 $((\lambda z. F(x - (x \cdot i) *_R i + z *_R i) \cdot j) \text{has-vector-derivative } F-j-i$
 $x) (\text{at } (x \cdot i)))$
have $((\lambda z. F((x *_R i + c) - ((x *_R i + c) \cdot i) *_R i + z *_R i) \cdot j)$
 $\text{has-vector-derivative } F-j-i(x *_R i + c)) (\text{at } ((x *_R i + c) \cdot i))$
using *range-of-x ass2 by auto*
then have *0*: $((\lambda x. F(c + x *_R i) \cdot j) \text{has-vector-derivative } F-j-i(x *_R i +$
 $c)) (\text{at } x)$
by (*simp add: assms(2) inner-left-distrib no-i-component*)
have *1*: $(\lambda x. F(x *_R i + c) \cdot j) = (\lambda x. F(c + x *_R i) \cdot j)$
by (*simp add: add commute*)
then show $((\lambda x. F(x *_R i + c) \cdot j) \text{has-vector-derivative } F-j-i(x *_R i + c))$
 $(\text{at } x)$
using *0 and 1 by auto*
qed
then have $\forall x \in (\text{cbox } a b). ((\lambda x. F(x *_R i + c) \cdot j) \text{has-vector-derivative } (F-j-i(x$
 $x *_R i + c))) (\text{at-within } x (\text{cbox } a b))$
using *has-vector-derivative-at-within*
by *blast*
then show $((\lambda x. F-j-i(x *_R i + c)) \text{has-integral}$
 $F(b *_R i + c) \cdot j - F(a *_R i + c) \cdot j) (\text{cbox } a b)$
using *fundamental-theorem-of-calculus[of a b (λx::real. F(x *_R i + c) · j)*
 $(\lambda x::\text{real}. F-j-i(x *_R i + c))]$ and
a-leq-b
by *auto*
qed

lemma *fundamental-theorem-of-calculus-partial-vector-gen*:

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

proof –

let $?domain = \{v. \exists x. k1 \leq x \wedge x \leq k2 \wedge v = x *_R i + c\}$
have $\forall x \in ?domain. has\text{-}partial\text{-}vector\text{-}derivative\ F\ i\ (F-i\ x)\ x$
using $has\text{-}partial\text{-}deriv\ domain\text{-}subset\text{-}of\text{-}D$
by $auto$
then have $\forall x \in (cbox\ k1\ k2). ((\lambda x. F(x *_R i + c))\ has\text{-}vector\text{-}derivative\ (F-i(x *_R i + c)))\ (at\ x)$
proof $(clarsimp\ simp\ add: has\text{-}partial\text{-}vector\text{-}derivative\text{-}def)$
fix $x::real$
assume $range\text{-}of\text{-}x: k1 \leq x \leq k2$
assume $ass2: \forall x. (\exists z \geq k1. z \leq k2 \wedge x = z *_R i + c) \longrightarrow$
 $((\lambda z. F(x - (x \cdot i) *_R i + z *_R i))\ has\text{-}vector\text{-}derivative\ F-i\ x)$
 $(at\ (x \cdot i))$
have $((\lambda z. F((x *_R i + c) - ((x *_R i + c) \cdot i) *_R i + z *_R i))\ has\text{-}vector\text{-}derivative$
 $F-i\ (x *_R i + c))\ (at\ ((x *_R i + c) \cdot i))$
using $range\text{-}of\text{-}x\ ass2$ **by** $auto$
then have $0: ((\lambda x. F(c + x *_R i))\ has\text{-}vector\text{-}derivative\ F-i\ (x *_R i + c))$
 $(at\ x)$
by $(simp\ add: inner\text{-}commute\ inner\text{-}right\text{-}distrib\ no\text{-}i\text{-component}\ unit\text{-}len)$
have $1: (\lambda x. F(x *_R i + c)) = (\lambda x. F(c + x *_R i))$
by $(simp\ add: add.\text{commute})$
then show $((\lambda x. F(x *_R i + c))\ has\text{-}vector\text{-}derivative\ F-i\ (x *_R i + c))\ (at\ x)$
using 0 **and** 1 **by** $auto$
qed
then have $\forall x \in (cbox\ k1\ k2). ((\lambda x. F(x *_R i + c))\ has\text{-}vector\text{-}derivative\ (F-i(x *_R i + c)))\ (at\ \text{within}\ x\ (cbox\ k1\ k2))$
using $has\text{-}vector\text{-}derivative\text{-}at\ \text{within}$
by $blast$
then show $(\lambda x. F-i(x *_R i + c))\ has\text{-}integral$
 $F(k2 *_R i + c) - F(k1 *_R i + c)$ $(cbox\ k1\ k2)$
using $fundamental\text{-}theorem\text{-}of\text{-}calculus[of\ k1\ k2\ (\lambda x::real. F(x *_R i + c))]$ **and**
 $(\lambda x::real. F-i(x *_R i + c))$ **and**
 $a\text{-leq-}b$
by $auto$
qed

lemma $add\text{-}scale\text{-}img:$

assumes $a < b$ **shows** $(\lambda x::real. a + (b - a) * x) ' \{0 .. 1\} = \{a .. b\}$

```

using assms
apply (auto simp: algebra-simps affine-ineq image-iff)
using less-eq-real-def apply force
apply (rule-tac x=(x-a)/(b-a) in bezI)
apply (auto simp: field-simps)
done

```

```

lemma add-scale-imp':
  assumes  $a \leq b$ 
  shows  $(\lambda x::\text{real}. a + (b - a) * x) \cdot \{0 .. 1\} = \{a .. b\}$ 
proof (cases a = b)
  case True
  then show ?thesis by force
next
  case False
  then show ?thesis
    using add-scale-imp assms by auto
qed

```

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

```

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

```

```

lemma analytically-valid-imp-part-deriv-integrable-on:
  assumes analytically-valid  $(s::(\text{real}*\text{real}) \text{ set})$   $(f::(\text{real}*\text{real}) \Rightarrow \text{real}) \ i$ 
  shows  $(\text{partial-vector-derivative } f \ i)$  integrable-on s
proof —
  have integrable lborel  $(\lambda p. \text{partial-vector-derivative } f \ i \ p * \text{indicator } s \ p)$ 
  using assms
  by (simp add: analytically-valid-def indic-ident)
  then have integrable lborel  $(\lambda p::(\text{real}*\text{real}). \text{if } p \in s \text{ then } \text{partial-vector-derivative}$ 
 $f \ i \ p \text{ else } 0)$ 
  using indic-ident[of partial-vector-derivative f i]
  by (simp add: indic-ident)
  then have  $(\lambda x. \text{if } x \in s \text{ then } \text{partial-vector-derivative } f \ i \ x \text{ else } 0)$  integrable-on
UNIV
  using Equivalence-Lebesgue-Henstock-Integration.integrable-on-lborel
  by auto

```


then show (*partial-vector-derivative f i*) *integrable-on s*
using *integrable-restrict-UNIV*
by auto
qed

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

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 $\exists a b g1 g2. a < b \wedge (\forall x \in \{a..b\}. g2\ x \leq g1\ x) \wedge$
 $cubeImage\ twoC = \{(y,x). x \in \{a..b\} \wedge y \in \{g2\ x..g1\ x\}\}$
 $\wedge twoC = (\lambda(y, x). ((1 - y) * g2 ((1 - x) * a + x * b) + y * g1$
 $((1 - x) * a + x * b), (1 - x) * a + x * b))$
 $\wedge g1\ piecewise-C1-differentiable-on\ \{a..b\} \wedge g2\ piecewise-C1-differentiable-on$
 $\{a..b\}$

using *assms*

proof (*simp add: typeII-twoCube-def cubeImage-def image-def*)

assume $\exists a b. a < b \wedge (\exists g1 g2. (\forall x \in \{a..b\}. g2\ x \leq g1\ x) \wedge$
 $twoC = (\lambda(y, x). ((1 - y) * g2 ((1 - x) * a + x * b) + y$
 $* g1 ((1 - x) * a + x * b), (1 - x) * a + x * b)) \wedge$
 $g1\ piecewise-C1-differentiable-on\ \{a..b\} \wedge g2\ piecewise-C1-differentiable-on$
 $\{a..b\})$

then obtain *a b g1 g2* **where**

twoCisTypeII: $a < b$
 $(\forall x \in \{a..b\}. g2\ x \leq g1\ x)$
 $twoC = (\lambda(y, x). ((1 - y) * g2 ((1 - x) * a + x * b) + y * g1 ((1 - x) * a$
 $+ x * b), (1 - x) * a + x * b))$
 $g1\ piecewise-C1-differentiable-on\ \{a..b\}$
 $g2\ piecewise-C1-differentiable-on\ \{a..b\}$

by auto

have *ab1*: $a - x * a + x * b \leq b$ **if** *a1*: $0 \leq x \leq 1$ **for** *x*

using that apply (*simp add: algebra-simps*)

by (*metis affine-ineq less-eq-real-def mult.commute twoCisTypeII(1)*)

have *ex*: $\exists z \in Green.unit-cube.$

$(d, c) =$

```

      (case z of
      (y, x) =>
      (g2 (a - x * a + x * b) - y * g2 (a - x * a + x * b) + y * g1
(a - x * a + x * b),
      a - x * a + x * b))
    if c-bounds: a ≤ c c ≤ b and d-bounds: g2 c ≤ d d ≤ g1 c for c d
  proof -
    have b-minus-a-nzero: b - a ≠ 0 using twoCisTypeII(1) by auto
    have x-witness: ∃ k1. c = (1 - k1)*a + k1 * b ∧ 0 ≤ k1 ∧ k1 ≤ 1
    apply (rule-tac x=(c - a)/(b - a) in exI)
    using c-bounds ⟨a < b⟩ apply (simp add: divide-simps algebra-simps)
    using sum-sqs-eq by blast
    have y-witness: ∃ k2. d = (1 - k2)*(g2 c) + k2 * (g1 c) ∧ 0 ≤ k2 ∧ k2 ≤ 1
  proof (cases g1 c - g2 c = 0)
    case True
      with d-bounds show ?thesis by (fastforce simp add: algebra-simps)
    next
      case False
        let ?k2 = (d - g2 c)/(g1 c - g2 c)
        have k2-in-bounds: 0 ≤ ?k2 ∧ ?k2 ≤ 1
          using twoCisTypeII(2) c-bounds d-bounds False by simp
        have d = (1 - ?k2) * g2 c + ?k2 * g1 c
          using False sum-sqs-eq by (fastforce simp add: divide-simps algebra-simps)
        with k2-in-bounds show ?thesis
          by fastforce
      qed
    show ∃ x∈unit-cube. (d, c) = (case x of (y, x) => (g2 (a - x * a + x * b) -
y * g2 (a - x * a + x * b) + y * g1 (a - x * a + x * b), a - x * a + x * b))
      using x-witness y-witness by (force simp add: left-diff-distrib)
    qed
  have {y. ∃ x∈unit-cube. y = twoC x} = {(y, x). a ≤ x ∧ x ≤ b ∧ g2 x ≤ y ∧
y ≤ g1 x}
    apply (auto simp add: twoCisTypeII ab1 left-diff-distrib ex)
    using order.order-iff-strict twoCisTypeII(1) apply fastforce
    apply (smt affine-ineq atLeastAtMost-iff mult-left-mono twoCisTypeII)+
    done
  then show ∃ a b. a < b ∧ (∃ g1 g2. (∀ x∈{a..b}. g2 x ≤ g1 x) ∧
      {y. ∃ x∈unit-cube. y = twoC x} = {(y, x). a ≤ x ∧ x ≤ b
∧ g2 x ≤ y ∧ y ≤ g1 x}) ∧
      twoC = (λ(y, x). ((1 - y) * g2 ((1 - x) * a + x * b) + y
* g1 ((1 - x) * a + x * b), (1 - x) * a + x * b)) ∧
      g1 piecewise-C1-differentiable-on {a..b} ∧ g2 piecewise-C1-differentiable-on
{a..b})
    using twoCisTypeII by blast
  qed

```

definition *horizontal-boundary* :: *two-cube* ⇒ *one-chain* **where**
horizontal-boundary twoC ≡ {(1, (λx. twoC(x,0))), (-1, (λx. twoC(x,1)))}

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

definition *boundary* :: *two-cube* \Rightarrow *one-chain* **where**
boundary twoC $\equiv \text{horizontal-boundary twoC} \cup \text{vertical-boundary twoC}$

definition *valid-two-cube* **where**
valid-two-cube twoC $\equiv \text{card}(\text{boundary twoC}) = 4$

definition *two-chain-integral*:: *two-chain* $\Rightarrow ((\text{real} * \text{real}) \Rightarrow (\text{real})) \Rightarrow \text{real}$ **where**
two-chain-integral twoChain F $\equiv \sum C \in \text{twoChain}. (\text{integral}(\text{cubeImage } C) F)$

definition *valid-two-chain* **where**
valid-two-chain twoChain $\equiv (\forall \text{twoCube} \in \text{twoChain}. \text{valid-two-cube twoCube})$
 $\wedge \text{pairwise}(\lambda c1 c2. ((\text{boundary } c1) \cap (\text{boundary } c2)) = \{\}) \text{twoChain} \wedge \text{inj-on cubeImage twoChain}$

definition *two-chain-boundary*:: *two-chain* \Rightarrow *one-chain* **where**
two-chain-boundary twoChain $\equiv \bigcup(\text{boundary } ` \text{twoChain})$

definition *gen-division* **where**
gen-division s S $\equiv (\text{finite } S \wedge (\bigcup S = s) \wedge \text{pairwise}(\lambda X Y. \text{negligible}(X \cap Y)) S)$

definition *two-chain-horizontal-boundary*:: *two-chain* \Rightarrow *one-chain* **where**
two-chain-horizontal-boundary twoChain $\equiv \bigcup(\text{horizontal-boundary } ` \text{twoChain})$

definition *two-chain-vertical-boundary*:: *two-chain* \Rightarrow *one-chain* **where**
two-chain-vertical-boundary twoChain $\equiv \bigcup(\text{vertical-boundary } ` \text{twoChain})$

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

lemma *sum-zero-set*:
assumes $\forall x \in s. f x = 0$ *finite s finite t*
shows $\text{sum } f (s \cup t) = \text{sum } f t$
using *assms*
by (*simp add: IntE sum.union-inter-neutral sup-commute*)

abbreviation *valid-typeII-division s twoChain* $\equiv ((\forall \text{twoCube} \in \text{twoChain}. \text{typeII-twoCube } \text{twoCube}) \wedge$

$(\text{gen-division } s (\text{cubeImage } \text{twoChain})) \wedge$
 $(\text{valid-two-chain } \text{twoChain}))$

lemma *two-chain-vertical-boundary-is-boundary-chain*:

shows *boundary-chain* (*two-chain-vertical-boundary twoChain*)

by (*simp add: boundary-chain-def two-chain-vertical-boundary-def vertical-boundary-def*)

lemma *two-chain-horizontal-boundary-is-boundary-chain*:

shows *boundary-chain* (*two-chain-horizontal-boundary twoChain*)

by (*simp add: boundary-chain-def two-chain-horizontal-boundary-def horizontal-boundary-def*)

definition *typeI-twoCube* :: *two-cube* \Rightarrow *bool* **where**

typeI-twoCube (*twoC*::*two-cube*)

$\equiv \exists a b g1 g2. a < b \wedge (\forall x \in \{a..b\}. g2 x \leq g1 x) \wedge$

$\text{twoC} = (\lambda(x,y). ((1-x)*a + x*b,$

$(1 - y) * (g2 ((1-x)*a + x*b)) + y * (g1$

$((1-x)*a + x*b)))) \wedge$

g1 *piecewise-C1-differentiable-on* $\{a..b\} \wedge$

g2 *piecewise-C1-differentiable-on* $\{a..b\}$

lemma *typeI-twoCubeImg*:

assumes *typeI-twoCube twoC*

shows $\exists a b g1 g2. a < b \wedge (\forall x \in \{a .. b\}. g2 x \leq g1 x) \wedge$

$\text{cubeImage } \text{twoC} = \{(x,y). x \in \{a..b\} \wedge y \in \{g2 x .. g1 x\}\} \wedge$

$\text{twoC} = (\lambda(x, y). ((1 - x) * a + x * b, (1 - y) * g2 ((1 - x)$

$* a + x * b) + y * g1 ((1 - x) * a + x * b))) \wedge$

g1 *piecewise-C1-differentiable-on* $\{a .. b\} \wedge$ *g2* *piecewise-C1-differentiable-on*

$\{a .. b\}$

proof –

have $\exists a b. a < b \wedge$

$(\exists g1 g2. (\forall x \in \{a..b\}. g2 x \leq g1 x) \wedge$

$\text{twoC} = (\lambda(x, y). ((1 - x) * a + x * b, (1 - y) * g2 ((1 - x) *$

$a + x * b) + y * g1 ((1 - x) * a + x * b))) \wedge$

g1 *piecewise-C1-differentiable-on* $\{a .. b\} \wedge$ *g2* *piecewise-C1-differentiable-on*

$\{a .. b\}$)

using *assms* **by** (*simp add: typeI-twoCube-def*)

then obtain *a b g1 g2* **where**

twoCisTypeI: $a < b$

$(\forall x \in \{a..b\}. g2 x \leq g1 x)$

$\text{twoC} = (\lambda(x, y). ((1 - x) * a + x * b, (1 - y) * g2 ((1 - x) * a + x *$

$b) + y * g1 ((1 - x) * a + x * b)))$

g1 *piecewise-C1-differentiable-on* $\{a .. b\}$

g2 *piecewise-C1-differentiable-on* $\{a .. b\}$

by *auto*

have *ex*: $\exists z \in \text{Green.unit-cube}.$

$(c, d) =$

(case z of

```

      (x, y) ⇒
      (a - x * a + x * b,
      g2 (a - x * a + x * b) - y * g2 (a - x * a + x * b) + y * g1 (a
- x * a + x * b)))
    if c-bounds: a ≤ c c ≤ b and d-bounds: g2 c ≤ d d ≤ g1 c for c d
  proof -
    have x-witness: ∃ k1. c = (1 - k1)*a + k1 * b ∧ 0 ≤ k1 ∧ k1 ≤ 1
  proof -
    let ?k1 = (c - a)/(b - a)
    have k1-in-bounds: 0 ≤ (c - a)/(b - a) ∧ (c - a)/(b - a) ≤ 1
      using twoCisTypeI(1) c-bounds by simp
    have c = (1 - ?k1)*a + ?k1 * b
      using twoCisTypeI(1) sum-sqs-eq
      by (auto simp add: divide-simps algebra-simps)
    then show ?thesis
      using twoCisTypeI k1-in-bounds by fastforce
  qed
  have y-witness: ∃ k2. d = (1 - k2)*(g2 c) + k2 * (g1 c) ∧ 0 ≤ k2 ∧ k2 ≤ 1
  proof (cases g1 c - g2 c = 0)
    case True
      with d-bounds show ?thesis
        by force
    next
      case False
        let ?k2 = (d - g2 c)/(g1 c - g2 c)
        have k2-in-bounds: 0 ≤ ?k2 ∧ ?k2 ≤ 1 using twoCisTypeI(2) c-bounds
d-bounds False by simp
        have d = (1 - ?k2) * g2 c + ?k2 * g1 c
          using False apply (simp add: divide-simps algebra-simps)
          using sum-sqs-eq by fastforce
        then show ?thesis using k2-in-bounds by fastforce
      qed
    show ∃ x ∈ unit-cube. (c, d) =
      (case x of (x, y) ⇒ (a - x * a + x * b, g2 (a - x * a + x * b) - y *
g2 (a - x * a + x * b) + y * g1 (a - x * a + x * b)))
      using x-witness y-witness by (force simp add: left-diff-distrib)
    qed
    have {y. ∃ x ∈ unit-cube. y = twoC x} = {(x, y). a ≤ x ∧ x ≤ b ∧ g2 x ≤ y ∧
y ≤ g1 x}
      apply (auto simp add: twoCisTypeI left-diff-distrib ex)
      using less-eq-real-def twoCisTypeI(1) apply auto[1]
      apply (smt affine-ineq twoCisTypeI)
      apply (smt affine-ineq atLeastAtMost-iff mult-left-mono twoCisTypeI)+
    done
  then show ?thesis
    unfolding cubeImage-def image-def using twoCisTypeI by auto
  qed

```

lemma typeI-cube-explicit-spec:

assumes *typeI-twoCube twoC*
shows $\exists a b g1 g2. a < b \wedge (\forall x \in \{a .. b\}. g2\ x \leq g1\ x) \wedge$
 $cubeImage\ twoC = \{(x,y). x \in \{a..b\} \wedge y \in \{g2\ x .. g1\ x\}\}$
 $\wedge twoC = (\lambda(x, y). ((1 - x) * a + x * b, (1 - y) * g2\ ((1 -$
 $x) * a + x * b) + y * g1\ ((1 - x) * a + x * b)))$
 $\wedge g1\ piecewise-C1-differentiable-on\ \{a .. b\} \wedge g2\ piecewise-C1-differentiable-on$
 $\{a .. b\}$
 $\wedge (\lambda x. twoC(x, 0)) = (\lambda x. (a + (b - a) * x, g2\ (a + (b - a) *$
 $x)))$
 $\wedge (\lambda y. twoC(1, y)) = (\lambda x. (b, g2\ b + x *R\ (g1\ b - g2\ b)))$
 $\wedge (\lambda x. twoC(x, 1)) = (\lambda x. (a + (b - a) * x, g1\ (a + (b - a) *$
 $x)))$
 $\wedge (\lambda y. twoC(0, y)) = (\lambda x. (a, g2\ a + x *R\ (g1\ a - g2\ a)))$

proof –

let *?bottom-edge* = $(\lambda x. twoC(x, 0))$
let *?right-edge* = $(\lambda y. twoC(1, y))$
let *?top-edge* = $(\lambda x. twoC(x, 1))$
let *?left-edge* = $(\lambda y. twoC(0, y))$
obtain *a b g1 g2* **where**
twoCisTypeI: $a < b$
 $(\forall x \in cbox\ a\ b. g2\ x \leq g1\ x)$
 $cubeImage\ twoC = \{(x,y). x \in cbox\ a\ b \wedge y \in cbox\ (g2\ x)\ (g1\ x)\}$
 $twoC = (\lambda(x, y). ((1 - x) * a + x * b, (1 - y) * g2\ ((1 - x) * a + x * b)$
 $+ y * g1\ ((1 - x) * a + x * b)))$
 $g1\ piecewise-C1-differentiable-on\ \{a .. b\}$
 $g2\ piecewise-C1-differentiable-on\ \{a .. b\}$
using *assms and typeI-twoCubeImg[oftwoC]* **by** *auto*
have *bottom-edge-explicit*: $?bottom-edge = (\lambda x. (a + (b - a) * x, g2\ (a + (b -$
 $a) * x)))$
by (*simp add: twoCisTypeI(4) algebra-simps*)
have *right-edge-explicit*: $?right-edge = (\lambda x. (b, g2\ b + x *R\ (g1\ b - g2\ b)))$
by (*simp add: twoCisTypeI(4) algebra-simps*)
have *top-edge-explicit*: $?top-edge = (\lambda x. (a + (b - a) * x, g1\ (a + (b - a) *$
 $x)))$
by (*simp add: twoCisTypeI(4) algebra-simps*)
have *left-edge-explicit*: $?left-edge = (\lambda x. (a, g2\ a + x *R\ (g1\ a - g2\ a)))$
by (*simp add: twoCisTypeI(4) algebra-simps*)
show *?thesis*
using *bottom-edge-explicit right-edge-explicit top-edge-explicit left-edge-explicit*
twoCisTypeI
by *auto*
qed

lemma *typeI-twoCube-smooth-edges*:

assumes *typeI-twoCube twoC*

$(k, \gamma) \in boundary\ twoC$

shows $\gamma\ piecewise-C1-differentiable-on\ \{0..1\}$

proof –

let *?bottom-edge* = $(\lambda x. twoC(x, 0))$

```

let ?right-edge = ( $\lambda y. \text{twoC}(1, y)$ )
let ?top-edge = ( $\lambda x. \text{twoC}(x, 1)$ )
let ?left-edge = ( $\lambda y. \text{twoC}(0, y)$ )
obtain a b g1 g2 where
  twoCisTypeI:  $a < b$ 
  ( $\forall x \in \text{cbox } a \ b. g2 \ x \leq g1 \ x$ )
  cubeImage twoC =  $\{(x,y). x \in \text{cbox } a \ b \wedge y \in \text{cbox } (g2 \ x) \ (g1 \ x)\}$ 
  twoC = ( $\lambda(x, y). ((1 - x) * a + x * b, (1 - y) * g2 \ ((1 - x) * a + x * b)$ 
+  $y * g1 \ ((1 - x) * a + x * b))$ )
  g1 piecewise-C1-differentiable-on  $\{a .. b\}$ 
  g2 piecewise-C1-differentiable-on  $\{a .. b\}$ 
  ( $\lambda x. \text{twoC}(x, 0) = (\lambda x. (a + (b - a) * x, g2 \ (a + (b - a) * x))$ )
  ( $\lambda y. \text{twoC}(1, y) = (\lambda x. (b, g2 \ b + x *_{R} (g1 \ b - g2 \ b))$ )
  ( $\lambda x. \text{twoC}(x, 1) = (\lambda x. (a + (b - a) * x, g1 \ (a + (b - a) * x))$ )
  ( $\lambda y. \text{twoC}(0, y) = (\lambda x. (a, g2 \ a + x *_{R} (g1 \ a - g2 \ a))$ )
  using assms and typeI-cube-explicit-spec[oftwoC]
  by auto
have bottom-edge-smooth: ( $\lambda x. \text{twoC}(x, 0)$ ) piecewise-C1-differentiable-on  $\{0..1\}$ 
proof -
  have  $\forall x. (\lambda x. (a + (b - a) * x))^{-1} \{x\} = \{(x - a)/(b - a)\}$ 
  using twoCisTypeI(1)
  by(auto simp add: Set.vimage-def)
  then have finite-vimg:  $\bigwedge x. \text{finite}(\{0..1\} \cap (\lambda x. (a + (b - a) * x))^{-1} \{x\})$ 
by auto
  have scale-shif-smth: ( $\lambda x. (a + (b - a) * x)$ ) C1-differentiable-on  $\{0..1\}$  using
scale-shift-smooth by auto
  then have scale-shif-pw-smth: ( $\lambda x. (a + (b - a) * x)$ ) piecewise-C1-differentiable-on
 $\{0..1\}$  using C1-differentiable-imp-piecewise by blast
  have g2-smooth: g2 piecewise-C1-differentiable-on ( $\lambda x. a + (b - a) * x$ ) ‘
 $\{0..1\}$  using add-scale-img[OF twoCisTypeI(1)] twoCisTypeI(6) by auto
  have ( $\lambda x. g2 \ (a + (b - a) * x)$ ) piecewise-C1-differentiable-on  $\{0..1\}$ 
  using Cauchy-Integral-Theorem.piecewise-C1-differentiable-compose[OF scale-shif-pw-smth
g2-smooth finite-vimg]
  by (auto simp add: o-def)
  then have ( $\lambda x::\text{real}. (a + (b - a) * x, g2 \ (a + (b - a) * x))$ ) piecewise-C1-differentiable-on
 $\{0..1\}$ 
  using all-components-smooth-one-pw-smooth-is-pw-smooth[where  $f = (\lambda x::\text{real}.$ 
 $(a + (b - a) * x, g2 \ (a + (b - a) * x))$ )]
  apply (simp only: real-pair-basis)
  by fastforce
  then show ?thesis using twoCisTypeI(7) by auto
qed
have top-edge-smooth: ?top-edge piecewise-C1-differentiable-on  $\{0..1\}$ 
proof -
  have  $\forall x. (\lambda x. (a + (b - a) * x))^{-1} \{x\} = \{(x - a)/(b - a)\}$ 
  using twoCisTypeI(1)
  by(auto simp add: Set.vimage-def)
  then have finite-vimg:  $\bigwedge x. \text{finite}(\{0..1\} \cap (\lambda x. (a + (b - a) * x))^{-1} \{x\})$ 
by auto

```

```

have scale-shif-smth: ( $\lambda x. (a + (b - a) * x)$ ) C1-differentiable-on {0..1} using
scale-shift-smooth by auto
then have scale-shif-pw-smth: ( $\lambda x. (a + (b - a) * x)$ ) piecewise-C1-differentiable-on
{0..1} using C1-differentiable-imp-piecewise by blast
have g1-smooth: g1 piecewise-C1-differentiable-on ( $\lambda x. a + (b - a) * x$ ) ‘
{0..1} using add-scale-img[OF twoCisTypeI(1)] twoCisTypeI(5) by auto
have ( $\lambda x. g1 (a + (b - a) * x)$ ) piecewise-C1-differentiable-on {0..1}
using Cauchy-Integral-Theorem.piecewise-C1-differentiable-compose[OF scale-shif-pw-smth
g1-smooth finite-vimg]
by (auto simp add: o-def)
then have ( $\lambda x. (a + (b - a) * x, g1 (a + (b - a) * x))$ ) piecewise-C1-differentiable-on
{0..1}
using all-components-smooth-one-pw-smooth-is-pw-smooth[where  $f = (\lambda x. (a + (b - a) * x, g1 (a + (b - a) * x))$ )]
apply (simp only: real-pair-basis)
by fastforce
then show ?thesis using twoCisTypeI(9) by auto
qed
have right-edge-smooth: ?right-edge piecewise-C1-differentiable-on {0..1}
proof –
have ( $\lambda x. (g2 b + x *_R (g1 b - g2 b))$ ) C1-differentiable-on {0..1}
using scale-shift-smooth C1-differentiable-imp-piecewise by auto
then have ( $\lambda x. (g2 b + x *_R (g1 b - g2 b))$ ) piecewise-C1-differentiable-on
{0..1}
using C1-differentiable-imp-piecewise by fastforce
then have ( $\lambda x. (b, g2 b + x *_R (g1 b - g2 b))$ ) piecewise-C1-differentiable-on
{0..1}
using pair-prod-smooth-pw-smooth by auto
then show ?thesis
using twoCisTypeI(8) by auto
qed
have left-edge-smooth: ?left-edge piecewise-C1-differentiable-on {0..1}
proof –
have ( $\lambda x. (g2 a + x *_R (g1 a - g2 a))$ ) C1-differentiable-on {0..1}
using scale-shift-smooth by auto
then have ( $\lambda x. (g2 a + x *_R (g1 a - g2 a))$ ) piecewise-C1-differentiable-on
{0..1}
using C1-differentiable-imp-piecewise by fastforce
then have ( $\lambda x. (a, g2 a + x *_R (g1 a - g2 a))$ ) piecewise-C1-differentiable-on
{0..1}
using pair-prod-smooth-pw-smooth by auto
then show ?thesis
using twoCisTypeI(10) by auto
qed
have  $\gamma = ?bottom-edge \vee \gamma = ?right-edge \vee \gamma = ?top-edge \vee \gamma = ?left-edge$ 
using assms by (auto simp add: boundary-def horizontal-boundary-def vertical-boundary-def)
then show ?thesis
using left-edge-smooth right-edge-smooth top-edge-smooth bottom-edge-smooth
by auto

```


qed

lemma *two-chain-integral-eq-integral-divisible*:

assumes *f-integrable*: $\forall \text{twoCube} \in \text{twoChain}. F \text{ integrable-on } \text{cubeImage } \text{twoCube}$

and

gen-division: *gen-division* s (*cubeImage* ‘ *twoChain*) **and**

valid-two-chain: *valid-two-chain* *twoChain*

shows $\text{integral } s F = \text{two-chain-integral } \text{twoChain } F$

proof –

show $\text{integral } s F = \text{two-chain-integral } \text{twoChain } F$

proof (*simp add*: *two-chain-integral-def*)

have *partial-deriv-integrable*:

$\forall \text{twoCube} \in \text{twoChain}. ((F) \text{ has-integral } (\text{integral } (\text{cubeImage } \text{twoCube}) (F)))$
(*cubeImage* *twoCube*)

using *f-integrable* **by** *auto*

then have *partial-deriv-integrable*:

$\bigwedge \text{twoCubeImg}. \text{twoCubeImg} \in \text{cubeImage } ‘ \text{twoChain} \implies (F \text{ has-integral } (\text{integral } (\text{twoCubeImg}) F))$ (*twoCubeImg*)

using *Henstock-Kurzweil-Integration.integrable-neg* **by** *force*

have *finite-images*: *finite* (*cubeImage* ‘ *twoChain*)

using *gen-division gen-division-def* **by** *auto*

have *negligible-images*: *pairwise* ($\lambda S S'. \text{negligible } (S \cap S')$) (*cubeImage* ‘ *twoChain*)

using *gen-division* **by** (*auto simp add*: *gen-division-def pairwise-def*)

have *inj*: *inj-on* *cubeImage* *twoChain*

using *valid-two-chain* **by** (*simp add*: *inj-on-def valid-two-chain-def*)

have $\text{integral } s F = (\sum \text{twoCubeImg} \in \text{cubeImage } ‘ \text{twoChain}. \text{integral } \text{twoCubeImg } F)$

using *has-integral-Union*[*OF finite-images partial-deriv-integrable negligible-images*]
gen-division

by (*auto simp add*: *gen-division-def*)

also have $\dots = (\sum C \in \text{twoChain}. \text{integral } (\text{cubeImage } C) F)$

using *sum.reindex inj* **by** *auto*

finally show $\text{integral } s F = (\sum C \in \text{twoChain}. \text{integral } (\text{cubeImage } C) F)$.

qed

qed

definition *only-vertical-division* **where**

only-vertical-division one-chain two-chain \equiv

$\exists \mathcal{V} \mathcal{H}. \text{finite } \mathcal{H} \wedge \text{finite } \mathcal{V} \wedge$

$(\forall (k, \gamma) \in \mathcal{V}.)$

$(\exists (k', \gamma') \in \text{two-chain-vertical-boundary } \text{two-chain}.$

$(\exists a \in \{0..1\}. \exists b \in \{0..1\}. a \leq b \wedge \text{subpath } a b \gamma' = \gamma))) \wedge$

$(\text{common-sudiv-exists } (\text{two-chain-horizontal-boundary } \text{two-chain}) \mathcal{H}$

$\vee \text{common-reparam-exists } \mathcal{H} (\text{two-chain-horizontal-boundary } \text{two-chain}))$

\wedge

$\text{boundary-chain } \mathcal{H} \wedge \text{one-chain} = \mathcal{V} \cup \mathcal{H} \wedge$

$(\forall (k, \gamma) \in \mathcal{H}. \text{valid-path } \gamma)$

abbreviation *valid-typeI-division s twoChain*
 $\equiv (\forall \text{twoCube} \in \text{twoChain}. \text{typeI-twoCube } \text{twoCube}) \wedge$
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, γ) \in boundary twoC*

shows *continuous-on (γ ' {0 .. 1}) F*

proof –

obtain *a b g1 g2 where*

twoCisTypeI: a < b

($\forall x \in \text{cbox } a \ b. g2 \ x \leq g1 \ x$)

cubeImage twoC = {(x,y). x \in cbox a b \wedge y \in cbox (g2 x) (g1 x)}

*twoC = ($\lambda(x, y). ((1 - x) * a + x * b, (1 - y) * g2 ((1 - x) * a + x * b)$*
 $+ y * g1 ((1 - x) * a + x * b)))$

g1 piecewise-C1-differentiable-on {a .. b}

g2 piecewise-C1-differentiable-on {a .. b}

*($\lambda x. \text{twoC}(x, 0) = (\lambda x. (a + (b - a) * x, g2 (a + (b - a) * x))$)*

*($\lambda y. \text{twoC}(1, y) = (\lambda x. (b, g2 b + x *_R (g1 b - g2 b))$)*

*($\lambda x. \text{twoC}(x, 1) = (\lambda x. (a + (b - a) * x, g1 (a + (b - a) * x))$)*

*($\lambda y. \text{twoC}(0, y) = (\lambda x. (a, g2 a + x *_R (g1 a - g2 a))$)*

using *typeI-twoC and typeI-cube-explicit-spec[oftwoC]*

by *auto*

let *?bottom-edge = ($\lambda x. \text{twoC}(x, 0)$)*

let *?right-edge = ($\lambda y. \text{twoC}(1, y)$)*

let *?top-edge = ($\lambda x. \text{twoC}(x, 1)$)*

let *?left-edge = ($\lambda y. \text{twoC}(0, y)$)*

let *?Dg1 = {p. $\exists x. x \in \text{cbox } a \ b \wedge p = (x, g1(x))$ }*

have *line-is-pair-img: ?Dg1 = ($\lambda x. (x, g1(x))$) ' (cbox a b)*

using *image-def by auto*

have *field-cont-on-top-edge-image: continuous-on ?Dg1 F*

by (*rule continuous-on-subset [OF field-cont]*) (*auto simp: twoCisTypeI(2)*
twoCisTypeI(3))

have *top-edge-is-compos-of-scal-and-g1:*

*($\lambda x. \text{twoC}(x, 1) = (\lambda x. (x, g1(x))) \circ (\lambda x. a + (b - a) * x)$)*

using *twoCisTypeI by auto*

have *Dg1-is-bot-edge-pathimg: path-image ($\lambda x. \text{twoC}(x, 1)$) = ?Dg1*

using *line-is-pair-img and top-edge-is-compos-of-scal-and-g1 image-comp path-image-def*
add-scale-img and twoCisTypeI(1)

by (*metis (no-types, lifting) cbox-interval*)

then have *cont-on-top: continuous-on (path-image ?top-edge) F*

using *field-cont-on-top-edge-image by auto*

let *?Dg2 = {p. $\exists x. x \in \text{cbox } a \ b \wedge p = (x, g2(x))$ }*

have *line-is-pair-img: ?Dg2 = ($\lambda x. (x, g2(x))$) ' (cbox a b)*

using *image-def by auto*

have *field-cont-on-bot-edge-image: continuous-on ?Dg2 F*

apply (*rule continuous-on-subset [OF field-cont]*)

```

    using twoCisTypeI(2) twoCisTypeI(3) by auto
  have bot-edge-is-compos-of-scal-and-g2:  $(\lambda x. \text{twoC}(x, 0)) = (\lambda x. (x, g2(x))) \circ$ 
 $(\lambda x. a + (b - a) * x)$ 
    using twoCisTypeI by auto
  have Dg2-is-bot-edge-pathimg:
    path-image  $(\lambda x. \text{twoC}(x, 0)) = ?Dg2$ 
  using line-is-pair-img and bot-edge-is-compos-of-scal-and-g2 image-comp path-image-def
  add-scale-img and twoCisTypeI(1)
    by (metis (no-types, lifting) cbox-interval)
  then have cont-on-bot: continuous-on (path-image ?bottom-edge) F
    using field-cont-on-bot-edge-image by auto
  let ?D-left-edge =  $\{p. \exists y. y \in \text{cbox } (g2 \ a) \ (g1 \ a) \wedge p = (a, y)\}$ 
  have field-cont-on-left-edge-image: continuous-on ?D-left-edge F
    apply (rule continuous-on-subset [OF field-cont])
    using twoCisTypeI(1) twoCisTypeI(3) by auto
  have  $g2 \ a \leq g1 \ a$  using twoCisTypeI(1) twoCisTypeI(2) by auto
  then have  $(\lambda x. g2 \ a + (g1 \ a - g2 \ a) * x) \text{ ' } \{(0::\text{real})..1\} = \{g2 \ a .. g1 \ a\}$ 
    using add-scale-img'[of g2 a g1 a] by blast
  then have left-eq: ?D-left-edge = ?left-edge '  $\{0..1\}$ 
    unfolding twoCisTypeI(10)
  by (auto simp add: subset-iff image-def set-eq-iff Semiring-Normalization.comm-semiring-1-class.semiring-no)
  then have cont-on-left: continuous-on (path-image ?left-edge) F
    using field-cont-on-left-edge-image path-image-def
    by (metis left-eq field-cont-on-left-edge-image path-image-def)
  let ?D-right-edge =  $\{p. \exists y. y \in \text{cbox } (g2 \ b) \ (g1 \ b) \wedge p = (b, y)\}$ 
  have field-cont-on-right-edge-image: continuous-on ?D-right-edge F
    apply (rule continuous-on-subset [OF field-cont])
    using twoCisTypeI(1) twoCisTypeI(3) by auto
  have  $g2 \ b \leq g1 \ b$  using twoCisTypeI(1) twoCisTypeI(2) by auto
  then have  $(\lambda x. g2 \ b + (g1 \ b - g2 \ b) * x) \text{ ' } \{(0::\text{real})..1\} = \{g2 \ b .. g1 \ b\}$ 
    using add-scale-img'[of g2 b g1 b] by blast
  then have right-eq: ?D-right-edge = ?right-edge '  $\{0..1\}$ 
    unfolding twoCisTypeI(8)
  by (auto simp add: subset-iff image-def set-eq-iff Semiring-Normalization.comm-semiring-1-class.semiring-no)
  then have cont-on-right:
    continuous-on (path-image ?right-edge) F
    using field-cont-on-right-edge-image path-image-def
    by (metis right-eq field-cont-on-right-edge-image path-image-def)
  have all-edge-cases:
     $(\gamma = ?bottom-edge \vee \gamma = ?right-edge \vee \gamma = ?top-edge \vee \gamma = ?left-edge)$ 
  using assms by (auto simp add: boundary-def horizontal-boundary-def vertical-boundary-def)
  show ?thesis
    apply (simp add: path-image-def[symmetric])
    using cont-on-top cont-on-bot cont-on-right cont-on-left all-edge-cases
    by blast
qed

```

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 –

let $?bottom-edge = (\lambda x. twoC(0, x))$
let $?right-edge = (\lambda y. twoC(y, 1))$
let $?top-edge = (\lambda x. twoC(1, x))$
let $?left-edge = (\lambda y. twoC(y, 0))$
obtain $a b g1 g2$ **where**
 $twoCisTypeII: a < b$
 $(\forall x \in cbox\ a\ b. g2\ x \leq g1\ x)$
 $cubeImage\ twoC = \{(y, x). x \in cbox\ a\ b \wedge y \in cbox\ (g2\ x)\ (g1\ x)\}$
 $twoC = (\lambda(y, x). ((1 - y) * g2 ((1 - x) * a + x * b) + y * g1 ((1 - x) * a$
 $+ x * b), (1 - x) * a + x * b))$
 $g1\ piecewise-C1-differentiable-on\ \{a .. b\}$
 $g2\ piecewise-C1-differentiable-on\ \{a .. b\}$
using $assms$ **and** $typeII-twoCubeImg[oftwoC]$ **by** $auto$
have $bottom-edge-explicit: ?bottom-edge = (\lambda x. (g2 (a + (b - a) * x), a + (b$
 $- a) * x))$
by $(simp\ add: twoCisTypeII(4)\ algebra-simps)$
have $right-edge-explicit: ?right-edge = (\lambda x. (g2 b + x *R (g1 b - g2 b), b))$
by $(simp\ add: twoCisTypeII(4)\ algebra-simps)$
have $top-edge-explicit: ?top-edge = (\lambda x. (g1 (a + (b - a) * x), a + (b - a) *$
 $x))$
by $(simp\ add: twoCisTypeII(4)\ algebra-simps)$
have $left-edge-explicit: ?left-edge = (\lambda x. (g2 a + x *R (g1 a - g2 a), a))$
by $(simp\ add: twoCisTypeII(4)\ algebra-simps)$
show $?thesis$
using $bottom-edge-explicit\ right-edge-explicit\ top-edge-explicit\ left-edge-explicit$
 $twoCisTypeII$
by $auto$
qed

lemma $typeII-twoCube-smooth-edges:$

assumes $typeII-twoCube\ twoC\ (k, \gamma) \in boundary\ twoC$

shows $\gamma\ piecewise-C1-differentiable-on\ \{0..1\}$

proof –

let $?bottom-edge = (\lambda x. twoC(0, x))$

let $?right-edge = (\lambda y. twoC(y, 1))$

let $?top-edge = (\lambda x. twoC(1, x))$

let $?left-edge = (\lambda y. twoC(y, 0))$

obtain $a b g1 g2$ **where**

```

twoCisTypeII: a < b
(∀ x ∈ cbox a b. g2 x ≤ g1 x)
cubeImage twoC = {(y, x). x ∈ cbox a b ∧ y ∈ cbox (g2 x) (g1 x)}
twoC = (λ(y, x). ((1 - y) * g2 ((1 - x) * a + x * b) + y * g1 ((1 - x) * a
+ x * b), (1 - x) * a + x * b))
g1 piecewise-C1-differentiable-on {a .. b}
g2 piecewise-C1-differentiable-on {a .. b}
(λx. twoC(0, x)) = (λx. (g2 (a + (b - a) * x), a + (b - a) * x))
(λy. twoC(y, 1)) = (λx. (g2 b + x *R (g1 b - g2 b), b))
(λx. twoC(1, x)) = (λx. (g1 (a + (b - a) * x), a + (b - a) * x))
(λy. twoC(y, 0)) = (λx. (g2 a + x *R (g1 a - g2 a), a))
using assms and typeII-cube-explicit-spec[oftwoC]
by auto
have bottom-edge-smooth: ?bottom-edge piecewise-C1-differentiable-on {0..1}
proof -
  have ∀ x. (λx. (a + (b - a) * x)) -' {x} = {(x - a)/(b - a)}
    using twoCisTypeII(1) by auto
  then have finite-vimg: ∧x. finite({0..1} ∩ (λx. (a + (b - a) * x)) -' {x})
by auto
  have scale-shif-pw-smth: (λx. (a + (b - a) * x)) piecewise-C1-differentiable-on
{0..1}
    using scale-shift-smooth C1-differentiable-imp-piecewise by blast
    have g2-smooth: g2 piecewise-C1-differentiable-on (λx. a + (b - a) * x) -'
{0..1} using add-scale-img[OF twoCisTypeII(1)] twoCisTypeII(6) by auto
    have (λx. g2 (a + (b - a) * x)) piecewise-C1-differentiable-on {0..1}
    using Cauchy-Integral-Theorem.piecewise-C1-differentiable-compose[OF scale-shif-pw-smth
g2-smooth finite-vimg]
    by (auto simp add: o-def)
    then have (λx::real. (g2 (a + (b - a) * x), a + (b - a) * x)) piecewise-C1-differentiable-on
{0..1}
    using all-components-smooth-one-pw-smooth-is-pw-smooth[where f = (λx::real.
(g2 (a + (b - a) * x), a + (b - a) * x))]
    by (fastforce simp add: real-pair-basis)
    then show ?thesis using twoCisTypeII(7) by auto
qed
have top-edge-smooth: ?top-edge piecewise-C1-differentiable-on {0..1}
proof -
  have ∀ x. (λx. (a + (b - a) * x)) -' {x} = {(x - a)/(b - a)}
    using twoCisTypeII(1) by auto
  then have finite-vimg: ∧x. finite({0..1} ∩ (λx. (a + (b - a) * x)) -' {x})
by auto
  have scale-shif-pw-smth: (λx. (a + (b - a) * x)) piecewise-C1-differentiable-on
{0..1}
    using scale-shift-smooth C1-differentiable-imp-piecewise by blast
    have g1-smooth: g1 piecewise-C1-differentiable-on (λx. a + (b - a) * x) -'
{0..1} using add-scale-img[OF twoCisTypeII(1)] twoCisTypeII(5) by auto
    have (λx. g1 (a + (b - a) * x)) piecewise-C1-differentiable-on {0..1}
    using piecewise-C1-differentiable-compose[OF scale-shif-pw-smth g1-smooth
finite-vimg]

```

```

    by (auto simp add: o-def)
  then have (λx::real. (g1 (a + (b - a) * x), a + (b - a) * x)) piecewise-C1-differentiable-on
{0..1}
    using all-components-smooth-one-pw-smooth-is-pw-smooth[where f = (λx::real.
(g1 (a + (b - a) * x), a + (b - a) * x))]
    by (fastforce simp add: real-pair-basis)
  then show ?thesis using twoCisTypeII(9) by auto
qed
have right-edge-smooth: ?right-edge piecewise-C1-differentiable-on {0..1}
proof -
  have (λx. (g2 b + x *R (g1 b - g2 b))) piecewise-C1-differentiable-on {0..1}
    by (simp add: C1-differentiable-imp-piecewise)
  then have (λx. (g2 b + x *R (g1 b - g2 b), b)) piecewise-C1-differentiable-on
{0..1}
    using all-components-smooth-one-pw-smooth-is-pw-smooth[of (1,0) (λx. (g2
b + x *R (g1 b - g2 b), b))]
    by (auto simp add: real-pair-basis)
  then show ?thesis
    using twoCisTypeII(8) by auto
qed
have left-edge-smooth: ?left-edge piecewise-C1-differentiable-on {0..1}
proof -
  have 0: (λx. (g2 a + x *R (g1 a - g2 a))) C1-differentiable-on {0..1}
    using C1-differentiable-imp-piecewise by fastforce
  have (λx. (g2 a + x *R (g1 a - g2 a), a)) piecewise-C1-differentiable-on {0..1}
    using C1-differentiable-imp-piecewise[OF C1-differentiable-on-pair[OF 0 C1-differentiable-on-const[of
a {0..1}]]]
    by force
  then show ?thesis
    using twoCisTypeII(10) by auto
qed
have γ = ?bottom-edge ∨ γ = ?right-edge ∨ γ = ?top-edge ∨ γ = ?left-edge
  using assms by (auto simp add: boundary-def horizontal-boundary-def vertical-boundary-def)
then show ?thesis
  using left-edge-smooth right-edge-smooth top-edge-smooth bottom-edge-smooth
by auto
qed

```

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,γ) ∈ boundary twoC
shows continuous-on (γ ‘ {0 .. 1}) F
proof -
  obtain a b g1 g2 where
    twoCisTypeII: a < b

```

```

(∀ x ∈ cbox a b. g2 x ≤ g1 x)
cubeImage twoC = {(y, x). x ∈ cbox a b ∧ y ∈ cbox (g2 x) (g1 x)}
twoC = (λ(y, x). ((1 - y) * g2 ((1 - x) * a + x * b) + y * g1 ((1 - x) * a
+ x * b), (1 - x) * a + x * b))
g1 piecewise-C1-differentiable-on {a .. b}
g2 piecewise-C1-differentiable-on {a .. b}
(λx. twoC(0, x)) = (λx. (g2 (a + (b - a) * x), a + (b - a) * x))
(λy. twoC(y, 1)) = (λx. (g2 b + x *R (g1 b - g2 b), b))
(λx. twoC(1, x)) = (λx. (g1 (a + (b - a) * x), a + (b - a) * x))
(λy. twoC(y, 0)) = (λx. (g2 a + x *R (g1 a - g2 a), a))
using typeII-twoC and typeII-cube-explicit-spec[oftwoC]
by auto
let ?bottom-edge = (λx. twoC(0, x))
let ?right-edge = (λy. twoC(y, 1))
let ?top-edge = (λx. twoC(1, x))
let ?left-edge = (λy. twoC(y, 0))
let ?Dg1 = {p. ∃ x. x ∈ cbox a b ∧ p = (g1(x), x)}
have line-is-pair-img: ?Dg1 = (λx. (g1(x), x)) ‘ (cbox a b)
using image-def by auto
have field-cont-on-top-edge-image: continuous-on ?Dg1 F
by (rule continuous-on-subset [OF field-cont]) (auto simp: twoCisTypeII(2)
twoCisTypeII(3))
have top-edge-is-compos-of-scal-and-g1:
(λx. twoC(1, x)) = (λx. (g1(x), x)) ∘ (λx. a + (b - a) * x)
using twoCisTypeII by auto
have Dg1-is-bot-edge-pathimg:
path-image (λx. twoC(1, x)) = ?Dg1
using line-is-pair-img and top-edge-is-compos-of-scal-and-g1 image-comp path-image-def
add-scale-img and twoCisTypeII(1)
by (metis (no-types, lifting) cbox-interval)
then have cont-on-top: continuous-on (path-image ?top-edge) F
using field-cont-on-top-edge-image by auto
let ?Dg2 = {p. ∃ x. x ∈ cbox a b ∧ p = (g2(x), x)}
have line-is-pair-img: ?Dg2 = (λx. (g2(x), x)) ‘ (cbox a b)
using image-def by auto
have field-cont-on-bot-edge-image: continuous-on ?Dg2 F
by (rule continuous-on-subset [OF field-cont]) (auto simp add: twoCisTypeII(2)
twoCisTypeII(3))
have bot-edge-is-compos-of-scal-and-g2:
(λx. twoC(0, x)) = (λx. (g2(x), x)) ∘ (λx. a + (b - a) * x)
using twoCisTypeII by auto
have Dg2-is-bot-edge-pathimg: path-image (λx. twoC(0, x)) = ?Dg2
unfolding path-image-def
using line-is-pair-img and bot-edge-is-compos-of-scal-and-g2 image-comp add-scale-img
[OF ⟨a < b⟩]
by (metis (no-types, lifting) box-real(2))
then have cont-on-bot: continuous-on (path-image ?bottom-edge) F
using field-cont-on-bot-edge-image
by auto

```

```

let ?D-left-edge = {p.  $\exists y. y \in \text{cbox } (g2 \ a) \ (g1 \ a) \wedge p = (y, \ a)$ }
have field-cont-on-left-edge-image: continuous-on ?D-left-edge F
  apply (rule continuous-on-subset [OF field-cont])
  using twoCisTypeII(1) twoCisTypeII(3) by auto
have g2 a ≤ g1 a using twoCisTypeII(1) twoCisTypeII(2) by auto
then have (λx. g2 a + x * (g1 a - g2 a)) ‘ {0::real}..1 } = {g2 a .. g1 a}
  using add-scale-imag'[of g2 a g1 a] by (auto simp add: ac-simps)
with ⟨g2 a ≤ g1 a⟩ have left-eq: ?D-left-edge = ?left-edge ‘ {0..1}
  by (simp only: twoCisTypeII(10)) auto
then have cont-on-left: continuous-on (path-image ?left-edge) F
  using field-cont-on-left-edge-image path-image-def
  by (metis left-eq field-cont-on-left-edge-image path-image-def)
let ?D-right-edge = {p.  $\exists y. y \in \text{cbox } (g2 \ b) \ (g1 \ b) \wedge p = (y, \ b)$ }
have field-cont-on-right-edge-image: continuous-on ?D-right-edge F
  apply (rule continuous-on-subset [OF field-cont])
  using twoCisTypeII(1) twoCisTypeII(3) by auto
have g2 b ≤ g1 b using twoCisTypeII(1) twoCisTypeII(2) by auto
then have (λx. g2 b + x * (g1 b - g2 b)) ‘ {0::real}..1 } = {g2 b .. g1 b}
  using add-scale-imag'[of g2 b g1 b] by (auto simp add: ac-simps)
with ⟨g2 b ≤ g1 b⟩ have right-eq: ?D-right-edge = ?right-edge ‘ {0..1}
  by (simp only: twoCisTypeII(8)) auto
then have cont-on-right:
  continuous-on (path-image ?right-edge) F
  using field-cont-on-right-edge-image path-image-def
  by (metis right-eq field-cont-on-right-edge-image path-image-def)
have all-edge-cases:
  (γ = ?bottom-edge ∨ γ = ?right-edge ∨ γ = ?top-edge ∨ γ = ?left-edge)
  using assms unfolding boundary-def horizontal-boundary-def vertical-boundary-def
by blast
show ?thesis
  apply (simp add: path-image-def[symmetric])
  using cont-on-top cont-on-bot cont-on-right cont-on-left all-edge-cases
  by blast
qed

```

lemma *two-cube-boundary-is-boundary*: boundary-chain (boundary C)
 by (auto simp add: boundary-chain-def boundary-def horizontal-boundary-def vertical-boundary-def)

lemma *common-boundary-subdiv-exists-refl*:
 assumes $\forall (k, \gamma) \in \text{boundary } \text{two}C. \text{valid-path } \gamma$
 shows common-boundary-sudivision-exists (boundary twoC) (boundary twoC)
 using assms chain-subdiv-chain-refl common-boundary-sudivision-exists-def two-cube-boundary-is-boundary
 by blast

lemma *common-boundary-subdiv-exists-refl'*:
 assumes $\forall (k, \gamma) \in C. \text{valid-path } \gamma$
 boundary-chain (C::(int × (real ⇒ real × real))) set)
 shows common-boundary-sudivision-exists (C) (C)

using *assms chain-subdiv-chain-refl common-boundary-sudivision-exists-def* **by**
blast

lemma *gen-common-boundary-subdiv-exists-refl-twochain-boundary*:

assumes $\forall (k,\gamma) \in C. \text{valid-path } \gamma$
 $\text{boundary-chain } (C::(\text{int} \times (\text{real} \Rightarrow \text{real} \times \text{real})) \text{ set})$
shows *common-sudiv-exists* $(C) (C)$

using *assms chain-subdiv-chain-refl common-boundary-sudivision-exists-def common-subdiv-imp-gen-common*
by *blast*

lemma *two-chain-boundary-is-boundary-chain*:

shows *boundary-chain* $(\text{two-chain-boundary } \text{twoChain})$

by $(\text{simp add: boundary-chain-def two-chain-boundary-def boundary-def horizontal-boundary-def vertical-boundary-def})$

lemma *typeI-edges-are-valid-paths*:

assumes *typeI-twoCube* $\text{twoC } (k,\gamma) \in \text{boundary } \text{twoC}$
shows *valid-path* γ

using *typeI-twoCube-smooth-edges[OF assms]* *C1-differentiable-imp-piecewise*
by $(\text{auto simp: valid-path-def})$

lemma *typeII-edges-are-valid-paths*:

assumes *typeII-twoCube* $\text{twoC } (k,\gamma) \in \text{boundary } \text{twoC}$
shows *valid-path* γ

using *typeII-twoCube-smooth-edges[OF assms]* *C1-differentiable-imp-piecewise*
by $(\text{auto simp: valid-path-def})$

lemma *finite-two-chain-vertical-boundary*:

assumes *finite two-chain*
shows *finite* $(\text{two-chain-vertical-boundary } \text{two-chain})$

using *assms* **by** $(\text{simp add: two-chain-vertical-boundary-def vertical-boundary-def})$

lemma *finite-two-chain-horizontal-boundary*:

assumes *finite two-chain*
shows *finite* $(\text{two-chain-horizontal-boundary } \text{two-chain})$

using *assms* **by** $(\text{simp add: two-chain-horizontal-boundary-def horizontal-boundary-def})$

locale *R2* =

fixes $i j$

assumes *i-is-x-axis*: $i = (1::\text{real}, 0::\text{real})$ **and**
j-is-y-axis: $j = (0::\text{real}, 1::\text{real})$

begin

lemma *analytically-valid-y*:

assumes *analytically-valid* $s F i$

shows $(\lambda x. \text{integral UNIV } (\lambda y. (\text{partial-vector-derivative } F i) (y, x) * (\text{indicator } s (y, x)))) \in \text{borel-measurable lborel}$

proof –

have $\{(1::\text{real}, 0::\text{real}), (0, 1)\} - \{(1, 0)\} = \{(0::\text{real}, 1::\text{real})\}$

by force
with *assms* **show** *?thesis*
using *assms* **by**(*simp add: real-pair-basis analytically-valid-def i-is-x-axis j-is-y-axis*)
qed

lemma *analytically-valid-x*:
assumes *analytically-valid s F j*
shows $(\lambda x. \text{integral UNIV } (\lambda y. ((\text{partial-vector-derivative } F j) (x, y)) * (\text{indicator } s (x, y)))) \in \text{borel-measurable lborel}$
proof –
have $\{(1::\text{real}, 0::\text{real}), (0, 1)\} - \{(0, 1)\} = \{(1::\text{real}, 0::\text{real})\}$
by force
with *assms* **show** *?thesis*
by(*simp add: real-pair-basis analytically-valid-def i-is-x-axis j-is-y-axis*)
qed

lemma *Greens-thm-type-I*:
fixes $F:: ((\text{real} * \text{real}) \Rightarrow (\text{real} * \text{real}))$ **and**
 $\text{gamma1 gamma2 gamma3 gamma4} :: (\text{real} \Rightarrow (\text{real} * \text{real}))$ **and**
 $a:: \text{real}$ **and** $b:: \text{real}$ **and**
 $g1:: (\text{real} \Rightarrow \text{real})$ **and** $g2:: (\text{real} \Rightarrow \text{real})$
assumes *Dy-def: Dy-pair = $\{x::\text{real}, y\} . x \in \text{cbox } a b \wedge y \in \text{cbox } (g2 x) (g1 x)$* **and**
 $\text{gamma1-def: gamma1} = (\lambda x. (a + (b - a) * x, g2(a + (b - a) * x)))$ **and**
 $\text{gamma1-smooth: gamma1}$ *piecewise-C1-differentiable-on $\{0..1\}$* **and**
 $\text{gamma2-def: gamma2} = (\lambda x. (b, g2(b) + x *_{\mathbb{R}} (g1(b) - g2(b))))$ **and**
 $\text{gamma3-def: gamma3} = (\lambda x. (a + (b - a) * x, g1(a + (b - a) * x)))$ **and**
 $\text{gamma3-smooth: gamma3}$ *piecewise-C1-differentiable-on $\{0..1\}$* **and**
 $\text{gamma4-def: gamma4} = (\lambda x. (a, g2(a) + x *_{\mathbb{R}} (g1(a) - g2(a))))$ **and**
F-i-analytically-valid: analytically-valid Dy-pair $(\lambda p. F(p) \cdot i)$ j **and**
 $g2\text{-leq-g1: } \forall x \in \text{cbox } a b. (g2 x) \leq (g1 x)$ **and**
 $a\text{-lt-}b: a < b$
shows $(\text{line-integral } F \{i\} \text{ gamma1}) +$
 $(\text{line-integral } F \{i\} \text{ gamma2}) -$
 $(\text{line-integral } F \{i\} \text{ gamma3}) -$
 $(\text{line-integral } F \{i\} \text{ gamma4})$
 $= (\text{integral Dy-pair } (\lambda a. - (\text{partial-vector-derivative } (\lambda p. F(p) \cdot i) j$
 $a)))$
 $\text{line-integral-exists } F \{i\} \text{ gamma4}$
 $\text{line-integral-exists } F \{i\} \text{ gamma3}$
 $\text{line-integral-exists } F \{i\} \text{ gamma2}$
 $\text{line-integral-exists } F \{i\} \text{ gamma1}$
proof –
let $?F\text{-}b' = \text{partial-vector-derivative } (\lambda a. (F a) \cdot i)$ j
have *F-first-is-continuous: continuous-on Dy-pair $(\lambda a. F(a) \cdot i)$*
using *F-i-analytically-valid*
by (*auto simp add: analytically-valid-def*)
let $?f = (\lambda x. \text{if } x \in (\text{Dy-pair}) \text{ then } (\text{partial-vector-derivative } (\lambda a. (F a) \cdot i) j$

x else 0)
have *f-lesbegue-integrable*: integrable lborel ?f
using *F-i-analytically-valid*
by (auto simp add: analytically-valid-def indic-ident)
have *partially-vec-diff*: $\forall a \in Dy\text{-pair}. \text{partially-vector-differentiable } (\lambda a. (F a) \cdot$
 $i) j a$
using *F-i-analytically-valid*
by (auto simp add: analytically-valid-def indicator-def)
have *x-axis-integral-measurable*: $(\lambda x. \text{integral UNIV } (\lambda y. ?f(x, y))) \in \text{borel-measurable}$
lborel
proof –
have $(\lambda p. (?F\text{-}b' p) * \text{indicator } (Dy\text{-pair}) p) = (\lambda x. \text{if } x \in (Dy\text{-pair}) \text{ then}$
 $(?F\text{-}b') x \text{ else } 0)$
using *indic-ident*[of ?F-b'] **by** auto
then have $\bigwedge x y. ?F\text{-}b'(x, y) * \text{indicator } (Dy\text{-pair}) (x, y) = (\lambda x. \text{if } x \in (Dy\text{-pair})$
 $\text{then } (?F\text{-}b') x \text{ else } 0) (x, y)$
by auto
then show *?thesis*
using *analytically-valid-x*[OF *F-i-analytically-valid*]
by (auto simp add: indicator-def)
qed
have *F-partially-differentiable*: $\forall a \in Dy\text{-pair}. \text{has-partial-vector-derivative } (\lambda x. (F$
 $x) \cdot i) j (?F\text{-}b' a) a$
using *partial-vector-derivative-works-partially-vec-diff*
by fastforce
have *g1-g2-continuous*: continuous-on (cbox a b) g1
continuous-on (cbox a b) g2
proof –
have *shift-scale-cont*: continuous-on {a..b} $(\lambda x. (x - a) * (1 / (b - a)))$
by (intro continuous-intros)
have *shift-scale-inv*: $(\lambda x. a + (b - a) * x) \circ (\lambda x. (x - a) * (1 / (b - a))) = id$
using *a-lt-b* **by** (auto simp add: o-def)
have *img-shift-scale*: $(\lambda x. (x - a) * (1 / (b - a))) \{a..b\} = \{0..1\}$
using *a-lt-b* **apply** (auto simp: divide-simps image-iff)
apply (rule-tac $x = x * (b - a) + a$ **in** *bexI*)
using *le-diff-eq* **by** fastforce+
have *gamma1-y-component*: $(\lambda x. g2(a + (b - a) * x)) = g2 \circ (\lambda x. (a + (b -$
 $a) * x))$
by auto
have *continuous-on {0..1}* $(\lambda x. g2(a + (b - a) * x))$
using *continuous-on-inner*[OF *piecewise-C1-differentiable-on-imp-continuous-on*[OF
gamma1-smooth], of $(\lambda x. j)$, OF *continuous-on-const*]
by (simp add: *gamma1-def j-is-y-axis*)
then have *continuous-on {a..b}* $((\lambda x. g2(a + (b - a) * x)) \circ (\lambda x. (x -$
 $a) * (1 / (b - a))))$
using *img-shift-scale continuous-on-compose shift-scale-cont*
by force
then have *continuous-on {a..b}* $(g2 \circ (\lambda x. (a + (b - a) * x)) \circ (\lambda x. (x -$
 $a) * (1 / (b - a))))$

```

    using gamma1-y-component by auto
  then show continuous-on (cbox a b) g2
    using a-lt-b by (simp add: shift-scale-inv)
  have gamma3-y-component:  $(\lambda x. g1(a + (b - a) * x)) = g1 \circ (\lambda x. (a + (b - a) * x))$ 
    by auto
  have continuous-on {0..1}  $(\lambda x. g1(a + (b - a) * x))$ 
    using continuous-on-inner[OF piecewise-C1-differentiable-on-imp-continuous-on[OF gamma3-smooth], of  $(\lambda x. j)$ , OF continuous-on-const]
    by (simp add: gamma3-def j-is-y-axis)
  then have continuous-on {a..b}  $((\lambda x. g1(a + (b - a) * x)) \circ (\lambda x. (x - a) * (1/(b-a))))$ 
    using img-shift-scale continuous-on-compose shift-scale-cont
    by force
  then have continuous-on {a..b}  $(g1 \circ (\lambda x. (a + (b - a) * x)) \circ (\lambda x. (x - a) * (1/(b-a))))$ 
    using gamma3-y-component by auto
  then show continuous-on (cbox a b) g1
    using a-lt-b by (simp add: shift-scale-inv)
qed
have g2-scale-j-contin: continuous-on (cbox a b)  $(\lambda x. (0, g2 x))$ 
  by (intro continuous-intros g1-g2-continuous)
let ?Dg2 =  $\{p. \exists x. x \in \text{cbox } a \ b \wedge p = (x, g2(x))\}$ 
have line-is-pair-img:  $?Dg2 = (\lambda x. (x, g2(x))) \text{ ' } (cbox a \ b)$ 
  using image-def by auto
have g2-path-continuous: continuous-on (cbox a b)  $(\lambda x. (x, g2(x)))$ 
  by (intro continuous-intros g1-g2-continuous)
have field-cont-on-gamma1-image: continuous-on ?Dg2  $(\lambda a. F(a) \cdot i)$ 
  apply (rule continuous-on-subset [OF F-first-is-continuous])
  by (auto simp add: Dy-def g2-leq-g1)
have gamma1-is-compos-of-scal-and-g2:
   $gamma1 = (\lambda x. (x, g2(x))) \circ (\lambda x. a + (b - a) * x)$ 
  using gamma1-def by auto
have add-scale-img:
 $(\lambda x. a + (b - a) * x) \text{ ' } \{0 .. 1\} = \{a .. b\}$  using add-scale-img and a-lt-b by
auto
then have Dg2-is-gamma1-pathimg: path-image gamma1 = ?Dg2
  by (metis (no-types, lifting) box-real(2) gamma1-is-compos-of-scal-and-g2 image-comp
line-is-pair-img path-image-def)
have Base-vecs:  $i \in \text{Basis } j \in \text{Basis } i \neq j$ 
  using real-pair-basis and i-is-x-axis and j-is-y-axis by auto
have gamma1-as-euclid-space-fun:  $gamma1 = (\lambda x. (a + (b - a) * x) *_R i + (0, g2(a + (b - a) * x)))$ 
  using i-is-x-axis gamma1-def by auto
have 0: line-integral F {i} gamma1 = integral (cbox a b)  $(\lambda x. F(x, g2(x)) \cdot i)$ 
  line-integral-exists F {i} gamma1
  using line-integral-on-pair-path-strong [OF norm-Basis[OF Base-vecs(1)] -
gamma1-as-euclid-space-fun, of F]
  gamma1-def gamma1-smooth g2-scale-j-contin a-lt-b add-scale-img

```

Dg2-is-gamma1-pathing and field-cont-on-gamma1-image
by (*auto simp: pathstart-def pathfinish-def i-is-x-axis*)
then show (*line-integral-exists F {i} gamma1*) **by metis**
have *gamma2-x-const*: $\forall x. \text{gamma2 } x \cdot i = b$
by (*simp add: i-is-x-axis gamma2-def*)
have 1: (*line-integral F {i} gamma2*) = 0 (*line-integral-exists F {i} gamma2*)
using *line-integral-on-pair-straight-path[OF gamma2-x-const] straight-path-differentiable-x gamma2-def*
by (*auto simp add: mult.commute*)
then show (*line-integral-exists F {i} gamma2*) **by metis**
have *continuous-on (cbox a b) (λx. F(x, g2(x)) · i)*
using *line-is-pair-img and g2-path-continuous and field-cont-on-gamma1-image Topological-Spaces.continuous-on-compose i-is-x-axis j-is-y-axis*
by auto
then have 6: ($\lambda x. F(x, g2(x)) \cdot i$) *integrable-on (cbox a b)*
using *integrable-continuous [of a b (λx. F(x, g2(x)) · i)] by auto*
have *g1-scale-j-contin: continuous-on (cbox a b) (λx. (0, g1 x))*
by (*intro continuous-intros g1-g2-continuous*)
let ?Dg1 = {*p. ∃ x. x ∈ cbox a b ∧ p = (x, g1(x))*}
have *line-is-pair-img: ?Dg1 = (λx. (x, g1(x))) ' (cbox a b)*
using *image-def by auto*
have *g1-path-continuous: continuous-on (cbox a b) (λx. (x, g1(x)))*
by (*intro continuous-intros g1-g2-continuous*)
have *field-cont-on-gamma3-image: continuous-on ?Dg1 (λa. F(a) · i)*
apply (*rule continuous-on-subset [OF F-first-is-continuous]*)
by (*auto simp add: Dy-def g2-leq-g1*)
have *gamma3-is-compos-of-scal-and-g1:*
 $\text{gamma3} = (\lambda x. (x, g1(x))) \circ (\lambda x. a + (b - a) * x)$
using *gamma3-def by auto*
then have *Dg1-is-gamma3-pathimg: path-image gamma3 = ?Dg1*
by (*metis (no-types, lifting) box-real(2) image-comp line-is-pair-img local.add-scale-img path-image-def*)
have *Base-vecs: i ∈ Basis j ∈ Basis i ≠ j*
using *real-pair-basis and i-is-x-axis and j-is-y-axis by auto*
have *gamma3-as-euclid-space-fun: gamma3 = (λx. (a + (b - a) * x) *_R i + (0, g1 (a + (b - a) * x)))*
using *i-is-x-axis gamma3-def by auto*
have 2: *line-integral F {i} gamma3 = integral (cbox a b) (λx. F(x, g1(x)) · i)*
line-integral-exists F {i} gamma3
using *line-integral-on-pair-path-strong [OF norm-Basis[OF Base-vecs(1)] - gamma3-as-euclid-space-fun, of F]*
gamma3-def and gamma3-smooth and g1-scale-j-contin and a-lt-b and add-scale-img
Dg1-is-gamma3-pathimg and field-cont-on-gamma3-image
by (*auto simp: pathstart-def pathfinish-def i-is-x-axis*)
then show (*line-integral-exists F {i} gamma3*) **by metis**
have *gamma4-x-const*: $\forall x. \text{gamma4 } x \cdot i = a$
using *gamma4-def*
by (*auto simp add: real-inner-class.inner-add-left inner-not-same-Basis i-is-x-axis*)

```

have 3: (line-integral F {i} gamma4) = 0 (line-integral-exists F {i} gamma4)
using line-integral-on-pair-straight-path[OF gamma4-x-const] straight-path-differentiable-x
gamma4-def
by (auto simp add: mult.commute)
then show (line-integral-exists F {i} gamma4)
by metis
have continuous-on (cbox a b) (λx. F(x, g1(x)) · i)
using line-is-pair-img and g1-path-continuous and field-cont-on-gamma3-image
continuous-on-compose i-is-x-axis j-is-y-axis
by auto
then have 7: (λx. F(x, g1(x)) · i) integrable-on (cbox a b)
using integrable-continuous [of a b (λx. F(x, g1(x)) · i)]
by auto
have partial-deriv-one-d-integrable:
((λy. ?F-b'(xc, y)) has-integral
F(xc, g1(xc)) · i - F(xc, g2(xc)) · i) (cbox (g2 xc) (g1 xc))
if xc ∈ cbox a b for xc
proof -
have {(xc', y). y ∈ cbox (g2 xc) (g1 xc) ∧ xc' = xc} ⊆ Dy-pair
using that by (auto simp add: Dy-def)
then show ((λy. ?F-b'(xc, y)) has-integral F(xc, g1 xc) · i - F(xc, g2 xc) ·
i) (cbox (g2 xc) (g1 xc))
using that and Base-vecs and F-partially-differentiable
and Dy-def [symmetric] and g2-leq-g1
and fundamental-theorem-of-calculus-partial-vector
[of g2 xc g1 xc j i xc *R i Dy-pair F ?F-b']
by (auto simp add: Groups.ab-semigroup-add-class.add.commute i-is-x-axis
j-is-y-axis)
qed
have partial-deriv-integrable: (?F-b') integrable-on Dy-pair
by (simp add: F-i-analytically-valid analytically-valid-imp-part-deriv-integrable-on)
have 4: integral Dy-pair ?F-b'
= integral (cbox a b) (λx. integral (cbox (g2 x) (g1 x)) (λy. ?F-b'(x, y)))
proof -
have x-axis-gauge-integrable:
∧x. (λy. ?f(x,y)) integrable-on UNIV
proof -
fix x::real
have ∀x. x ∉ cbox a b → (λy. ?f(x, y)) = (λy. 0)
by (auto simp add: Dy-def)
then have f-integrable-x-not-in-range:
∀x. x ∉ cbox a b → (λy. ?f(x, y)) integrable-on UNIV
by (simp add: integrable-0)
let ?F-b'-oneD = (λx. (λy. if y ∈ (cbox (g2 x) (g1 x)) then ?F-b'(x,y) else
0))
have f-value-x-in-range: ∀x ∈ cbox a b. ?F-b'-oneD x = (λy. ?f(x,y))
by (auto simp add: Dy-def)
have ∀x ∈ cbox a b. ?F-b'-oneD x integrable-on UNIV
using has-integral-integrable integrable-restrict-UNIV partial-deriv-one-d-integrable

```

by *blast*
then have *f-integrable-x-in-range*:
 $\forall x. x \in \text{cbox } a \ b \longrightarrow (\lambda y. ?f(x, y)) \text{ integrable-on UNIV}$
using *f-value-x-in-range by auto*
show $(\lambda y. ?f(x, y)) \text{ integrable-on UNIV}$
using *f-integrable-x-not-in-range and f-integrable-x-in-range by auto*
qed
have *arg*: $(\lambda a. \text{if } a \in \text{Dy-pair} \text{ then } \text{partial-vector-derivative } (\lambda a. F \ a \cdot i) \ j \ a \text{ else } 0) =$
 $(\lambda x. \text{if } x \in \text{Dy-pair} \text{ then } \text{if } x \in \text{Dy-pair} \text{ then } \text{partial-vector-derivative } (\lambda a. F \ a \cdot i) \ j \ x \text{ else } 0 \text{ else } 0)$
by *auto*
have *arg2*: $\text{Dy-pair} = \{(x, y). (\forall i \in \text{Basis}. a \cdot i \leq x \cdot i \wedge x \cdot i \leq b \cdot i) \wedge (\forall i \in \text{Basis}. g2 \ x \cdot i \leq y \cdot i \wedge y \cdot i \leq g1 \ x \cdot i)\}$
using *Dy-def by auto*
have *arg3*: $\bigwedge x. x \in \text{Dy-pair} \implies (\lambda x. \text{if } x \in \text{Dy-pair} \text{ then } \text{partial-vector-derivative } (\lambda a. F \ a \cdot i) \ j \ x \text{ else } 0) \ x$
 $= (\lambda x. \text{partial-vector-derivative } (\lambda a. F \ a \cdot i) \ j \ x) \ x$
by *auto*
have *arg4*: $\bigwedge x. x \in (\text{cbox } a \ b) \implies$
 $(\lambda x. \text{integral } (\text{cbox } (g2 \ x) \ (g1 \ x)) (\lambda y. \text{if } (x, y) \in \text{Dy-pair} \text{ then } \text{partial-vector-derivative } (\lambda a. F \ a \cdot i) \ j \ (x, y) \text{ else } 0)) \ x =$
 $(\lambda x. \text{integral } (\text{cbox } (g2 \ x) \ (g1 \ x)) (\lambda y. \text{partial-vector-derivative } (\lambda a. F \ a \cdot i) \ j \ (x, y))) \ x$
apply (*simp add: Dy-def*)
by (*smt Henstock-Kurzweil-Integration.integral-cong atLeastAtMost-iff*)
show *?thesis*
using *gauge-integral-Fubini-curve-bounded-region-x*
 $[OF \ \text{f-lesbeque-integrable } x\text{-axis-gauge-integrable } x\text{-axis-integral-measurable}$
 $\text{arg } \text{arg2}]$
 $\text{Henstock-Kurzweil-Integration.integral-cong}[OF \ \text{arg3}, \text{ of } \text{Dy-pair } (\lambda x. x)]$
 $\text{Henstock-Kurzweil-Integration.integral-cong}[OF \ \text{arg4}, \text{ of } \text{cbox } a \ b \ (\lambda x. x)]$
by *auto*
qed
have *5*: $(\text{integral } \text{Dy-pair } (\lambda a. (?F\text{-}b' \ a)))$
 $= \text{integral } (\text{cbox } a \ b) (\lambda x. F(x, g1(x)) \cdot i - F(x, g2(x)) \cdot i)$
using *4 Henstock-Kurzweil-Integration.integral-cong partial-deriv-one-d-integrable*
integrable-def
by (*smt integral-unique*)
show $(\text{line-integral } F \ \{i\} \ \text{gamma1}) + (\text{line-integral } F \ \{i\} \ \text{gamma2}) -$
 $(\text{line-integral } F \ \{i\} \ \text{gamma3}) - (\text{line-integral } F \ \{i\} \ \text{gamma4})$
 $= (\text{integral } \text{Dy-pair } (\lambda a. - (?F\text{-}b' \ a)))$
using *0(1) 1(1) 2(1) 3(1) 5 6 7*
by (*simp add: Henstock-Kurzweil-Integration.integral-diff*)
qed

theorem *Greens-thm-type-II*:
fixes $F :: (\text{real} * \text{real}) \Rightarrow (\text{real} * \text{real})$ **and**
 $\text{gamma4 } \text{gamma3 } \text{gamma2 } \text{gamma1} :: (\text{real} \Rightarrow (\text{real} * \text{real}))$ **and**

$a:: \text{real}$ and $b:: \text{real}$ and
 $g1:: (\text{real} \Rightarrow \text{real})$ and $g2:: (\text{real} \Rightarrow \text{real})$
assumes $Dx\text{-def}: Dx\text{-pair} = \{(x::\text{real}, y) . y \in \text{cbox } a \ b \wedge x \in \text{cbox } (g2 \ y) \ (g1 \ y)\}$ and
 $gamma4\text{-def}: gamma4 = (\lambda x. (g2(a + (b - a) * x), a + (b - a) * x))$ and
 $gamma4\text{-smooth}: gamma4 \text{ piecewise-}C1\text{-differentiable-on } \{0..1\}$ and
 $gamma3\text{-def}: gamma3 = (\lambda x. (g2(b) + x *_R (g1(b) - g2(b)), b))$ and
 $gamma2\text{-def}: gamma2 = (\lambda x. (g1(a + (b - a) * x), a + (b - a) * x))$ and
 $gamma2\text{-smooth}: gamma2 \text{ piecewise-}C1\text{-differentiable-on } \{0..1\}$ and
 $gamma1\text{-def}: gamma1 = (\lambda x. (g2(a) + x *_R (g1(a) - g2(a)), a))$ and
 $F\text{-}j\text{-analytically-valid}: \text{analytically-valid } Dx\text{-pair } (\lambda p. F(p) \cdot j) \ i$ and
 $g2\text{-leq-}g1: \forall x \in \text{cbox } a \ b. (g2 \ x) \leq (g1 \ x)$ and
 $a\text{-lt-}b: a < b$
shows $-(\text{line-integral } F \ \{j\} \ gamma4) -$
 $(\text{line-integral } F \ \{j\} \ gamma3) +$
 $(\text{line-integral } F \ \{j\} \ gamma2) +$
 $(\text{line-integral } F \ \{j\} \ gamma1)$
 $= (\text{integral } Dx\text{-pair } (\lambda a. (\text{partial-vector-derivative } (\lambda a. (F \ a) \cdot j) \ i$
 $a)))$
 $\text{line-integral-exists } F \ \{j\} \ gamma4$
 $\text{line-integral-exists } F \ \{j\} \ gamma3$
 $\text{line-integral-exists } F \ \{j\} \ gamma2$
 $\text{line-integral-exists } F \ \{j\} \ gamma1$
proof –
let $?F\text{-}a' = \text{partial-vector-derivative } (\lambda a. (F \ a) \cdot j) \ i$
have $F\text{-first-is-continuous}: \text{continuous-on } Dx\text{-pair } (\lambda a. F(a) \cdot j)$
using $F\text{-}j\text{-analytically-valid}$
by $(\text{auto simp add: analytically-valid-def})$
let $?f = (\lambda x. \text{if } x \in (Dx\text{-pair}) \text{ then } (\text{partial-vector-derivative } (\lambda a. (F \ a) \cdot j) \ i)$
 $x \text{ else } 0)$
have $f\text{-lesbegue-integrable}: \text{integrable l borel } ?f$
using $F\text{-}j\text{-analytically-valid}$
by $(\text{auto simp add: analytically-valid-def indic-ident})$
have $\text{partially-vec-diff}: \forall a \in Dx\text{-pair}. \text{partially-vector-differentiable } (\lambda a. (F \ a) \cdot j) \ i \ a$
using $F\text{-}j\text{-analytically-valid}$
by $(\text{auto simp add: analytically-valid-def indicator-def})$
have $\bigwedge x \ y. ?F\text{-}a' \ (x, y) * \text{indicator } (Dx\text{-pair}) \ (x, y) = (\lambda x. \text{if } x \in (Dx\text{-pair}) \text{ then } ?F\text{-}a' \ x \text{ else } 0) \ (x, y)$
using $\text{indic-ident}[of \ ?F\text{-}a']$ **by** auto
then have $y\text{-axis-integral-measurable}: (\lambda x. \text{integral UNIV } (\lambda y. ?f(y, x))) \in$
 $\text{borel-measurable l borel}$
using $\text{analytically-valid-y}[OF \ F\text{-}j\text{-analytically-valid}]$
by $(\text{auto simp add: indicator-def})$
have $F\text{-partially-differentiable}: \forall a \in Dx\text{-pair}. \text{has-partial-vector-derivative } (\lambda x. (F \ x) \cdot j) \ i \ (?F\text{-}a' \ a) \ a$
using $\text{partial-vector-derivative-works partially-vec-diff}$ **by** fastforce
have $g1\text{-}g2\text{-continuous}: \text{continuous-on } (\text{cbox } a \ b) \ g1 \ \text{continuous-on } (\text{cbox } a \ b) \ g2$
proof –


```

have shift-scale-cont: continuous-on {a..b} ( $\lambda x. (x - a) * (1 / (b - a))$ )
  by (intro continuous-intros)
have shift-scale-inv: ( $\lambda x. a + (b - a) * x$ )  $\circ$  ( $\lambda x. (x - a) * (1 / (b - a))$ ) = id
  using a-lt-b by (auto simp add: o-def)
have img-shift-scale:
  ( $\lambda x. (x - a) * (1 / (b - a))$ ) ‘{a..b} = {0..1}
  apply (auto simp: divide-simps image-iff)
  apply (rule-tac x=x * (b - a) + a in bezI)
  using a-lt-b by (auto simp: algebra-simps mult-le-cancel-left affine-ineq)
have continuous-on {0..1} ( $\lambda x. g2(a + (b - a) * x)$ )
  using continuous-on-inner[OF piecewise-C1-differentiable-on-imp-continuous-on [OF
gamma4-smooth], of ( $\lambda x. i$ ), OF continuous-on-const]
  by (simp add: gamma4-def i-is-x-axis)
  then have continuous-on {a..b} (( $\lambda x. g2(a + (b - a) * x)$ )  $\circ$  ( $\lambda x. (x -$ 
a) * (1 / (b - a))))
  using img-shift-scale continuous-on-compose shift-scale-cont by force
  then show continuous-on (cbox a b) g2
  using a-lt-b by (simp add: shift-scale-inv)
  have continuous-on {0..1} ( $\lambda x. g1(a + (b - a) * x)$ )
  using continuous-on-inner[OF piecewise-C1-differentiable-on-imp-continuous-on [OF
gamma2-smooth], of ( $\lambda x. i$ ), OF continuous-on-const]
  by (simp add: gamma2-def i-is-x-axis)
  then have continuous-on {a..b} (( $\lambda x. g1(a + (b - a) * x)$ )  $\circ$  ( $\lambda x. (x -$ 
a) * (1 / (b - a))))
  using img-shift-scale continuous-on-compose shift-scale-cont by force
  then show continuous-on (cbox a b) g1
  using a-lt-b by (simp add: shift-scale-inv)
qed
have g2-scale-i-contin: continuous-on (cbox a b) ( $\lambda x. (g2 x, 0)$ )
  by (intro continuous-intros g1-g2-continuous)
let ?Dg2 = {p.  $\exists x. x \in \text{cbox } a \ b \wedge p = (g2(x), x)$ }
have line-is-pair-img: ?Dg2 = ( $\lambda x. (g2(x), x)$ ) ‘(cbox a b)
  using image-def by auto
have g2-path-continuous: continuous-on (cbox a b) ( $\lambda x. (g2(x), x)$ )
  by (intro continuous-intros g1-g2-continuous)
have field-cont-on-gamma4-image: continuous-on ?Dg2 ( $\lambda a. F(a) \cdot j$ )
  by (rule continuous-on-subset [OF F-first-is-continuous]) (auto simp: Dx-def
g2-leq-g1)
have gamma4-is-compos-of-scal-and-g2: gamma4 = ( $\lambda x. (g2(x), x)$ )  $\circ$  ( $\lambda x. a +$ 
(b - a) * x)
  using gamma4-def by auto
have add-scale-img:
  ( $\lambda x. a + (b - a) * x$ ) ‘{0 .. 1} = {a .. b} using add-scale-img and a-lt-b by
auto
then have Dg2-is-gamma4-pathimg: path-image gamma4 = ?Dg2
  using line-is-pair-img and gamma4-is-compos-of-scal-and-g2 image-comp path-image-def
  by (metis (no-types, lifting) cbox-interval)
have Base-vecs: i  $\in$  Basis j  $\in$  Basis i  $\neq$  j
  using real-pair-basis and i-is-x-axis and j-is-y-axis by auto

```

```

have gamma4-as-euclid-space-fun: gamma4 = (λx. (a + (b - a) * x) *R j +
(g2 (a + (b - a) * x), 0))
  using j-is-y-axis gamma4-def
  by auto
have 0: (line-integral F {j} gamma4) = integral (cbox a b) (λx. F(g2(x), x) ·
j)
  line-integral-exists F {j} gamma4
  using line-integral-on-pair-path-strong [OF norm-Basis[OF Base-vecs(2)] -
gamma4-as-euclid-space-fun]
  gamma4-def gamma4-smooth g2-scale-i-contin a-lt-b add-scale-img
  Dg2-is-gamma4-pathimg and field-cont-on-gamma4-image
  by (auto simp: pathstart-def pathfinish-def j-is-y-axis)
then show line-integral-exists F {j} gamma4 by metis
have gamma3-y-const: ∀ x. gamma3 x · j = b
  by (simp add: gamma3-def j-is-y-axis)
have 1: (line-integral F {j} gamma3) = 0 (line-integral-exists F {j} gamma3)
  using line-integral-on-pair-straight-path[OF gamma3-y-const] straight-path-differentiable-y
gamma3-def
  by (auto simp add: mult.commute)
then show line-integral-exists F {j} gamma3 by auto
have continuous-on (cbox a b) (λx. F(g2(x), x) · j)
  by (smt Collect-mono-iff continuous-on-compose2 continuous-on-eq field-cont-on-gamma4-image
g2-path-continuous line-is-pair-img)
then have 6: (λx. F(g2(x), x) · j) integrable-on (cbox a b)
  using integrable-continuous by blast
have g1-scale-i-contin: continuous-on (cbox a b) (λx. (g1 x, 0))
  by (intro continuous-intros g1-g2-continuous)
let ?Dg1 = {p. ∃ x. x ∈ cbox a b ∧ p = (g1(x), x)}
have line-is-pair-img: ?Dg1 = (λx. (g1(x), x) ) ‘ (cbox a b)
  using image-def by auto
have g1-path-continuous: continuous-on (cbox a b) (λx. (g1(x), x))
  by (intro continuous-intros g1-g2-continuous)
have field-cont-on-gamma2-image: continuous-on ?Dg1 (λa. F(a) · j)
  by (rule continuous-on-subset [OF F-first-is-continuous]) (auto simp: Dx-def
g2-leq-g1)
have gamma2 = (λx. (g1(x), x)) ◦ (λx. a + (b - a) * x)
  using gamma2-def by auto
then have Dg1-is-gamma2-pathimg: path-image gamma2 = ?Dg1
  using line-is-pair-img image-comp path-image-def add-scale-img
  by (metis (no-types, lifting) cbox-interval)
have Base-vecs: i ∈ Basis j ∈ Basis i ≠ j
  using real-pair-basis and i-is-x-axis and j-is-y-axis by auto
have gamma2-as-euclid-space-fun: gamma2 = (λx. (a + (b - a) * x) *R j +
(g1 (a + (b - a) * x), 0))
  using j-is-y-axis gamma2-def by auto
have 2: (line-integral F {j} gamma2) = integral (cbox a b) (λx. F(g1(x), x) ·
j)
  (line-integral-exists F {j} gamma2)
  using line-integral-on-pair-path-strong [OF norm-Basis[OF Base-vecs(2)] -

```

```

gamma2-as-euclid-space-fun]
  gamma2-def and gamma2-smooth and g1-scale-i-contin and a-lt-b and
add-scale-img
  Dg1-is-gamma2-pathing and field-cont-on-gamma2-image
  by (auto simp: pathstart-def pathfinish-def j-is-y-axis)
  then show line-integral-exists F {j} gamma2 by metis
  have gamma1-y-const:  $\forall x. \text{gamma1 } x \cdot j = a$ 
  using gamma1-def
  by (auto simp add: real-inner-class.inner-add-left
    euclidean-space-class.inner-not-same-Basis j-is-y-axis)
  have 3: (line-integral F {j} gamma1) = 0 (line-integral-exists F {j} gamma1)
  using line-integral-on-pair-straight-path[OF gamma1-y-const] straight-path-differentiable-y
gamma1-def
  by (auto simp add: mult.commute)
  then show line-integral-exists F {j} gamma1 by auto
  have continuous-on (cbox a b) ( $\lambda x. F(g1(x), x) \cdot j$ )
  by (smt Collect-mono-iff continuous-on-compose2 continuous-on-eq field-cont-on-gamma2-image
g1-path-continuous line-is-pair-img)
  then have 7: ( $\lambda x. F(g1(x), x) \cdot j$ ) integrable-on (cbox a b)
  using integrable-continuous [of a b ( $\lambda x. F(g1(x), x) \cdot j$ )]
  by auto
  have partial-deriv-one-d-integrable:
    (( $\lambda y. ?F\text{-}a'(y, xc)$ ) has-integral  $F(g1(xc), xc) \cdot j - F(g2(xc), xc) \cdot j$ ) (cbox
(g2 xc) (g1 xc))
  if  $xc \in \text{cbox } a \text{ } b$  for  $xc::\text{real}$ 
  proof -
  have  $\{(y, xc'). y \in \text{cbox } (g2 \text{ } xc) (g1 \text{ } xc) \wedge xc' = xc\} \subseteq \text{Dx-pair}$ 
  using that by (auto simp add: Dx-def)
  then show ?thesis
  using that and Base-vecs and F-partially-differentiable
    and Dx-def [symmetric] and g2-leq-g1
    and fundamental-theorem-of-calculus-partial-vector
    [of g2 xc g1 xc i j xc *R j Dx-pair F ?F-a']
  by (auto simp add: Groups.ab-semigroup-add-class.add.commute i-is-x-axis
j-is-y-axis)
  qed
  have ?f integrable-on UNIV
  by (simp add: f-lesbegue-integrable integrable-on-lborel)
  then have partial-deriv-integrable: ?F-a' integrable-on Dx-pair
  using integrable-restrict-UNIV by auto
  have 4: integral Dx-pair ?F-a' = integral (cbox a b) ( $\lambda x. \text{integral } (\text{cbox } (g2 \text{ } x)
(g1 \text{ } x)) (\lambda y. ?F\text{-}a'(y, x))$ )
  proof -
  have y-axis-gauge-integrable: ( $\lambda y. ?f(y, x)$ ) integrable-on UNIV for x
  proof -
  let ?F-a'-oneD = ( $\lambda x. (\lambda y. \text{if } y \in (\text{cbox } (g2 \text{ } x) (g1 \text{ } x)) \text{ then } ?F\text{-}a'(y, x) \text{ else }
0)$ )
  have  $\forall x. x \notin \text{cbox } a \text{ } b \longrightarrow (\lambda y. ?f(y, x)) = (\lambda y. 0)$ 
  by (auto simp add: Dx-def)

```

then have *f-integrable-x-not-in-range*:
 $\forall x. x \notin \text{cbox } a \ b \longrightarrow (\lambda y. ?f \ (y, x)) \text{ integrable-on UNIV}$
by (*simp add: integrable-0*)
have $\forall x \in \text{cbox } a \ b. ?F\text{-}a'\text{-oneD } x = (\lambda y. ?f(y, x))$
using *g2-leq-g1* **by** (*auto simp add: Dx-def*)
moreover have $\forall x \in \text{cbox } a \ b. ?F\text{-}a'\text{-oneD } x \text{ integrable-on UNIV}$
using *has-integral-integrable integrable-restrict-UNIV partial-deriv-one-d-integrable*
by *blast*
ultimately have $\forall x. x \in \text{cbox } a \ b \longrightarrow (\lambda y. ?f \ (y, x)) \text{ integrable-on UNIV}$
by *auto*
then show $(\lambda y. ?f \ (y, x)) \text{ integrable-on UNIV}$
using *f-integrable-x-not-in-range* **by** *auto*
qed
have *arg*: $(\lambda a. \text{if } a \in \text{Dx-pair} \text{ then } \text{partial-vector-derivative } (\lambda a. F \ a \cdot j) \ i \ a \text{ else } 0) =$
 $(\lambda x. \text{if } x \in \text{Dx-pair} \text{ then } \text{if } x \in \text{Dx-pair} \text{ then } \text{partial-vector-derivative}$
 $(\lambda a. F \ a \cdot j) \ i \ x \text{ else } 0 \text{ else } 0)$
by *auto*
have *arg2*: $\text{Dx-pair} = \{(y, x). (\forall i \in \text{Basis}. a \cdot i \leq x \cdot i \wedge x \cdot i \leq b \cdot i) \wedge$
 $(\forall i \in \text{Basis}. g2 \ x \cdot i \leq y \cdot i \wedge y \cdot i \leq g1 \ x \cdot i)\}$
using *Dx-def* **by** *auto*
have *arg3*: $\bigwedge x. x \in \text{Dx-pair} \implies (\lambda x. \text{if } x \in \text{Dx-pair} \text{ then } \text{partial-vector-derivative}$
 $(\lambda a. F \ a \cdot j) \ i \ x \text{ else } 0) \ x$
 $= (\lambda x. \text{partial-vector-derivative } (\lambda a. F \ a \cdot j) \ i \ x) \ x$
by *auto*
have *arg4*: $\bigwedge x. x \in (\text{cbox } a \ b) \implies$
 $(\lambda x. \text{integral } (\text{cbox } (g2 \ x) \ (g1 \ x)) \ (\lambda y. \text{if } (y, x) \in \text{Dx-pair} \text{ then}$
 $\text{partial-vector-derivative } (\lambda a. F \ a \cdot j) \ i \ (y, x) \text{ else } 0)) \ x =$
 $(\lambda x. \text{integral } (\text{cbox } (g2 \ x) \ (g1 \ x)) \ (\lambda y. \text{partial-vector-derivative}$
 $(\lambda a. F \ a \cdot j) \ i \ (y, x))) \ x$
apply (*clarsimp simp: Dx-def*)
by (*smt Henstock-Kurzweil-Integration.integral-cong atLeastAtMost-iff*)
show *?thesis*
using *gauge-integral-Fubini-curve-bounded-region-y*
 $[OF \ \text{f-lesbegue-integrable } y\text{-axis-gauge-integrable } y\text{-axis-integral-measurable}$
 $\text{arg } \text{arg2}]$
 $\text{Henstock-Kurzweil-Integration.integral-cong}[OF \ \text{arg3}, \text{ of } \text{Dx-pair } (\lambda x. x)]$
 $\text{Henstock-Kurzweil-Integration.integral-cong}[OF \ \text{arg4}, \text{ of } \text{cbox } a \ b \ (\lambda x. x)]$
by *auto*
qed
have $((\text{integral } \text{Dx-pair } (\lambda a. (?F\text{-}a' \ a)))$
 $= \text{integral } (\text{cbox } a \ b) \ (\lambda x. F(g1(x), x) \cdot j - F(g2(x), x) \cdot j))$
using *4 Henstock-Kurzweil-Integration.integral-cong partial-deriv-one-d-integrable*
integrable-def
by (*smt integral-unique*)
then have *integral Dx-pair* $(\lambda a. - (?F\text{-}a' \ a))$
 $= - \text{integral } (\text{cbox } a \ b) \ (\lambda x. F(g1(x), x) \cdot j - F(g2(x), x) \cdot j)$
using *partial-deriv-integrable* **and** *integral-neg* **by** *auto*
then have *5*: *integral Dx-pair* $(\lambda a. - (?F\text{-}a' \ a))$

```

      = integral (cbox a b) (λx. ( F(g2(x), x) · j - F(g1(x), x) · j))
using 6 7
    and integral-neg [of - (λx. F(g1 x, x) · j - F(g2 x, x) · j)] by auto
have (line-integral F {j} gamma4) + (line-integral F {j} gamma3) -
      (line-integral F {j} gamma2) - (line-integral F {j} gamma1)
      = (integral Dx-pair (λa. -(?F-a' a)))
using 0 1 2 3 5 6 7
      Henstock-Kurzweil-Integration.integral-diff[of (λx. F(g2(x), x) · j)
      cbox a b (λx. F(g1(x), x) · j)] by (auto)
then show -(line-integral F {j} gamma4) -
      (line-integral F {j} gamma3) +
      (line-integral F {j} gamma2) +
      (line-integral F {j} gamma1)
      = (integral Dx-pair (λa. (?F-a' a)))
    by (simp add: ⟨integral Dx-pair (λa. - ?F-a' a) = - integral (cbox a b) (λx.
      F(g1 x, x) · j - F(g2 x, x) · j)⟩ ⟨integral Dx-pair ?F-a' = integral (cbox a b) (λx.
      F(g1 x, x) · j - F(g2 x, x) · j)⟩)
qed

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) (λx. (F x) · j) i
begin

lemma GreenThm-typeII-twoCube:
  shows integral (cubeImage twoC) (λa. partial-vector-derivative (λx. (F x) · j) i
  a) = one-chain-line-integral F {j} (boundary twoC)
    ∀(k,γ) ∈ boundary twoC. line-integral-exists F {j} γ
proof -
  let ?bottom-edge = (λx. twoC(x, 0))
  let ?right-edge = (λy. twoC(1, y))
  let ?top-edge = (λx. twoC(x, 1))
  let ?left-edge = (λy. twoC(0, y))
  have line-integral-around-boundary:
    one-chain-line-integral F {j} (boundary twoC) =
      line-integral F {j} ?bottom-edge + line-integral F {j} ?right-edge
      - line-integral F {j} ?top-edge - line-integral F {j} ?left-edge
proof (simp add: one-chain-line-integral-def horizontal-boundary-def vertical-boundary-def
  boundary-def)
  have finite1: finite {(- 1::int, λy. twoC (0, y)), (1, λy. twoC (1, y)), (- 1,
  λx. twoC (x, 1))} by auto
  then have sum-step1: (∑ (k, g) ∈ {(- 1::int, λy. twoC (0, y)), (1, λy. twoC
  (1, y)), (1, λx. twoC (x, 0)), (- 1, λx. twoC (x, 1))}. k * line-integral F {j} g)
  =

```

$line_integral\ F\ \{j\}\ (\lambda x. twoC\ (x, 0)) + (\sum (k, g) \in \{(-1::int), \lambda y. twoC\ (0, y), (1, \lambda y. twoC\ (1, y)), (-1, \lambda x. twoC\ (x, 1))\}. k * line_integral\ F\ \{j\}\ g)$
using *sum.insert-remove* [*OF finite1*]
using *valid-two-cube*
apply (*simp only: one-chain-line-integral-def horizontal-boundary-def vertical-boundary-def boundary-def valid-two-cube-def*)
by (*auto simp: card-insert-if split: if-split-asm*)
have *three-distinct-edges: card* $\{(-1::int, \lambda y. twoC\ (0, y)), (1, \lambda y. twoC\ (1, y)), (-1, \lambda x. twoC\ (x, 1))\} = 3$
using *valid-two-cube*
apply (*simp add: one-chain-line-integral-def horizontal-boundary-def vertical-boundary-def boundary-def valid-two-cube-def*)
by (*auto simp: card-insert-if split: if-split-asm*)
have *finite2: finite* $\{(-1::int, \lambda y. twoC\ (0, y)), (1, \lambda y. twoC\ (1, y))\}$ **by** *auto*
then have *sum-step2: $(\sum (k, g) \in \{(-1::int), \lambda y. twoC\ (0, y), (1, \lambda y. twoC\ (1, y)), (-1, \lambda x. twoC\ (x, 1))\}. k * line_integral\ F\ \{j\}\ g) = (- line_integral\ F\ \{j\}\ (\lambda x. twoC\ (x, 1))) + (\sum (k, g) \in \{(-1::int), \lambda y. twoC\ (0, y), (1, \lambda y. twoC\ (1, y))\}. k * line_integral\ F\ \{j\}\ g)$*
using *sum.insert-remove* [*OF finite2*] *three-distinct-edges*
by (*auto simp: card-insert-if split: if-split-asm*)
have *two-distinct-edges: card* $\{(-1::int, \lambda y. twoC\ (0, y)), (1, \lambda y. twoC\ (1, y))\} = 2$
using *three-distinct-edges*
by (*simp add: one-chain-line-integral-def horizontal-boundary-def vertical-boundary-def boundary-def*)
have *finite3: finite* $\{(-1::int, \lambda y. twoC\ (0, y))\}$ **by** *auto*
then have *sum-step3: $(\sum (k, g) \in \{(-1::int), \lambda y. twoC\ (0, y), (1, \lambda y. twoC\ (1, y))\}. k * line_integral\ F\ \{j\}\ g) = (line_integral\ F\ \{j\}\ (\lambda y. twoC\ (1, y))) + (\sum (k, g) \in \{(-1::real), \lambda y. twoC\ (0, y)\}. k * line_integral\ F\ \{j\}\ g)$*
using *sum.insert-remove* [*OF finite2*] *three-distinct-edges*
by (*auto simp: card-insert-if split: if-split-asm*)
show $(\sum x \in \{(-1::int, \lambda y. twoC\ (0, y)), (1, \lambda y. twoC\ (1, y)), (1, \lambda x. twoC\ (x, 0)), (-1, \lambda x. twoC\ (x, 1))\}. case\ x\ of\ (k, g) \Rightarrow k * line_integral\ F\ \{j\}\ g) = line_integral\ F\ \{j\}\ (\lambda x. twoC\ (x, 0)) + line_integral\ F\ \{j\}\ (\lambda y. twoC\ (1, y)) - line_integral\ F\ \{j\}\ (\lambda x. twoC\ (x, 1)) - line_integral\ F\ \{j\}\ (\lambda y. twoC\ (0, y))$
using *sum-step1 sum-step2 sum-step3* **by** *auto*
qed
obtain *a b g1 g2* **where**
 $twoCisTypeII: a < b$
 $(\forall x \in cbox\ a\ b. g2\ x \leq g1\ x)$
 $cubeImage\ twoC = \{(y, x). x \in cbox\ a\ b \wedge y \in cbox\ (g2\ x)\ (g1\ x)\}$
 $twoC = (\lambda(y, x). ((1 - y) * g2\ ((1 - x) * a + x * b) + y * g1\ ((1 - x) * a + x * b), (1 - x) * a + x * b))$
 $g1\ piecewise-C1-differentiable-on\ \{a .. b\}$
 $g2\ piecewise-C1-differentiable-on\ \{a .. b\}$

```

    using two-cube and typeII-twoCubeImg[oftwoC]
    by auto
    have left-edge-explicit: ?left-edge = ( $\lambda x. (g2 (a + (b - a) * x), a + (b - a) * x)$ )
    by (simp add: twoCisTypeII(4) algebra-simps)
    have left-edge-smooth: ?left-edge piecewise-C1-differentiable-on {0..1}
    proof -
      have  $\forall x. (\lambda x. (a + (b - a) * x))^{-1} \{x\} = \{(x - a)/(b - a)\}$ 
      using twoCisTypeII(1) by(auto simp add: Set.vimage-def)
      then have finite-vimg:  $\bigwedge x. \text{finite}(\{0..1\} \cap (\lambda x. (a + (b - a) * x))^{-1} \{x\})$ 
    by auto
      have scale-shif-smth: ( $\lambda x. (a + (b - a) * x)$ ) C1-differentiable-on {0..1} using
    scale-shift-smooth by auto
      then have scale-shif-pw-smth: ( $\lambda x. (a + (b - a) * x)$ ) piecewise-C1-differentiable-on
    {0..1} using C1-differentiable-imp-piecewise by blast
      have g2-smooth: g2 piecewise-C1-differentiable-on ( $\lambda x. a + (b - a) * x$ ) ‘
    {0..1} using add-scale-img[OF twoCisTypeII(1)] twoCisTypeII(6) by auto
      have ( $\lambda x. g2 (a + (b - a) * x)$ ) piecewise-C1-differentiable-on {0..1}
      using Cauchy-Integral-Theorem.piecewise-C1-differentiable-compose[OF scale-shif-pw-smth
    g2-smooth finite-vimg]
      by (auto simp add: o-def)
      then have ( $\lambda x::\text{real}. (g2 (a + (b - a) * x), a + (b - a) * x)$ ) piecewise-C1-differentiable-on
    {0..1}
      using all-components-smooth-one-pw-smooth-is-pw-smooth[where f = ( $\lambda x::\text{real}.$ 
    ( $g2 (a + (b - a) * x), a + (b - a) * x$ ))]
      by (fastforce simp add: real-pair-basis)
      then show ?thesis using left-edge-explicit by auto
    qed
    have top-edge-explicit: ?top-edge = ( $\lambda x. (g2 b + x *_R (g1 b - g2 b), b)$ )
    and right-edge-explicit: ?right-edge = ( $\lambda x. (g1 (a + (b - a) * x), a + (b - a) * x)$ )
    by (simp-all add: twoCisTypeII(4) algebra-simps)
    have right-edge-smooth: ?right-edge piecewise-C1-differentiable-on {0..1}
    proof -
      have  $\forall x. (\lambda x. (a + (b - a) * x))^{-1} \{x\} = \{(x - a)/(b - a)\}$ 
      using twoCisTypeII(1) by(auto simp add: Set.vimage-def)
      then have finite-vimg:  $\bigwedge x. \text{finite}(\{0..1\} \cap (\lambda x. (a + (b - a) * x))^{-1} \{x\})$ 
    by auto
      then have scale-shif-pw-smth: ( $\lambda x. (a + (b - a) * x)$ ) piecewise-C1-differentiable-on
    {0..1}
      using C1-differentiable-imp-piecewise [OF scale-shift-smooth] by auto
      have g1-smooth: g1 piecewise-C1-differentiable-on ( $\lambda x. a + (b - a) * x$ ) ‘
    {0..1} using add-scale-img[OF twoCisTypeII(1)] twoCisTypeII(5) by auto
      have ( $\lambda x. g1 (a + (b - a) * x)$ ) piecewise-C1-differentiable-on {0..1}
      using piecewise-C1-differentiable-compose[OF scale-shif-pw-smth g1-smooth
    finite-vimg]
      by (auto simp add: o-def)
      then have ( $\lambda x::\text{real}. (g1 (a + (b - a) * x), a + (b - a) * x)$ ) piecewise-C1-differentiable-on
    {0..1}

```

```

using all-components-smooth-one-pw-smooth-is-pw-smooth[where  $f = (\lambda x::real.$ 
( $g1 (a + (b - a) * x), a + (b - a) * x$ ))]
by (fastforce simp add: real-pair-basis)
then show ?thesis using right-edge-explicit by auto
qed
have bottom-edge-explicit: ?bottom-edge = ( $\lambda x. (g2 a + x *R (g1 a - g2 a), a)$ )
by (simp add: twoCisTypeII(4) algebra-simps)
show integral (cubeImage twoC) ( $\lambda a. partial-vector-derivative (\lambda x. (F x) \cdot j)$  i
a) = one-chain-line-integral F {j} (boundary twoC)
using Greens-thm-type-II[OF twoCisTypeII(3) left-edge-explicit left-edge-smooth
top-edge-explicit right-edge-explicit right-edge-smooth
bottom-edge-explicit f-analytically-valid
twoCisTypeII(2) twoCisTypeII(1)]
line-integral-around-boundary
by auto
have line-integral-exists F {j}  $\gamma$  if  $(k, \gamma) \in boundary twoC$  for  $k \ \gamma$ 
proof -
have line-integral-exists F {j} ( $\lambda y. twoC (0, y)$ )
line-integral-exists F {j} ( $\lambda x. twoC (x, 1)$ )
line-integral-exists F {j} ( $\lambda y. twoC (1, y)$ )
line-integral-exists F {j} ( $\lambda x. twoC (x, 0)$ )
using Greens-thm-type-II[OF twoCisTypeII(3) left-edge-explicit left-edge-smooth
top-edge-explicit right-edge-explicit right-edge-smooth
bottom-edge-explicit f-analytically-valid
twoCisTypeII(2) twoCisTypeII(1)] line-integral-around-boundary
by auto
then show line-integral-exists F {j}  $\gamma$ 
using that by (auto simp add: boundary-def horizontal-boundary-def vertical-boundary-def)
qed
then show  $\forall (k, \gamma) \in boundary twoC. line-integral-exists F \{j\} \ \gamma$  by auto
qed

```

```

lemma line-integral-exists-on-typeII-Cube-boundaries':
assumes  $(k, \gamma) \in boundary twoC$ 
shows line-integral-exists F {j}  $\gamma$ 
using assms GreenThm-typeII-twoCube(2) by blast

```

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:  $\forall two-cube \in two-chain. valid-two-cube two-cube$ 
using valid-typeII-div
by (auto simp add: valid-two-chain-def)

```


lemma *typeII-chain-line-integral-exists-boundary'*:
shows $\forall (k,\gamma) \in \text{two-chain-vertical-boundary two-chain. line-integral-exists } F \{j\}$
 γ
proof –
have *integ-exis*: $\forall (k,\gamma) \in \text{two-chain-boundary two-chain. line-integral-exists } F$
 $\{j\} \gamma$
using *green-typeII-cube.line-integral-exists-on-typeII-Cube-boundaries'*[of *i j*]
F-anal-valid valid-typeII-div
apply (*auto simp add: two-chain-boundary-def*)
using *R2-axioms green-typeII-cube-axioms-def green-typeII-cube-def two-chain-valid-valid-cubes*
by *blast*
then show *integ-exis-vert*:
 $\forall (k,\gamma) \in \text{two-chain-vertical-boundary two-chain. line-integral-exists } F \{j\} \gamma$
by (*simp add: two-chain-boundary-def two-chain-vertical-boundary-def boundary-def*)
qed

lemma *typeII-chain-line-integral-exists-boundary''*:
 $\forall (k,\gamma) \in \text{two-chain-horizontal-boundary two-chain. line-integral-exists } F \{j\} \gamma$
proof –
have *integ-exis*: $\forall (k,\gamma) \in \text{two-chain-boundary two-chain. line-integral-exists } F$
 $\{j\} \gamma$
using *green-typeII-cube.line-integral-exists-on-typeII-Cube-boundaries'*[of *i j*]
valid-typeII-div
apply (*simp add: two-chain-boundary-def boundary-def*)
using *F-anal-valid R2-axioms green-typeII-cube-axioms-def green-typeII-cube-def*
two-chain-valid-valid-cubes **by** *fastforce*
then show *integ-exis-vert*:
 $\forall (k,\gamma) \in \text{two-chain-horizontal-boundary two-chain. line-integral-exists } F \{j\} \gamma$
by (*simp add: two-chain-boundary-def two-chain-horizontal-boundary-def boundary-def*)
qed

lemma *typeII-cube-line-integral-exists-boundary*:
 $\forall (k,\gamma) \in \text{two-chain-boundary two-chain. line-integral-exists } F \{j\} \gamma$
using *valid-typeII-div typeII-chain-line-integral-exists-boundary' typeII-chain-line-integral-exists-boundary''*
apply (*auto simp add: two-chain-boundary-def two-chain-horizontal-boundary-def*
two-chain-vertical-boundary-def)
using *boundary-def* **by** *auto*

lemma *type-II-chain-horiz-bound-valid*:
 $\forall (k,\gamma) \in \text{two-chain-horizontal-boundary two-chain. valid-path } \gamma$
using *valid-typeII-div typeII-edges-are-valid-paths*
by (*force simp add: two-chain-boundary-def two-chain-horizontal-boundary-def*
boundary-def)

lemma *type-II-chain-vert-bound-valid*:
 $\forall (k,\gamma) \in \text{two-chain-vertical-boundary two-chain. valid-path } \gamma$
using *typeII-edges-are-valid-paths valid-typeII-div*
by (*force simp add: two-chain-boundary-def two-chain-vertical-boundary-def boundary-def*)

lemma *members-of-only-horiz-div-line-integrable'*:
assumes *only-horizontal-division one-chain two-chain*
 $(k::int, \gamma) \in \text{one-chain}$
 $(k::int, \gamma) \in \text{one-chain}$
finite two-chain
 $\forall \text{two-cube} \in \text{two-chain. valid-two-cube two-cube}$
shows *line-integral-exists F {j} γ*
proof –
have *integ-exis: $\forall (k, \gamma) \in \text{two-chain-boundary two-chain. line-integral-exists F {j} $\gamma$$*
using *typeII-cube-line-integral-exists-boundary* **by** *blast*
have *integ-exis-vert:*
 $\forall (k, \gamma) \in \text{two-chain-vertical-boundary two-chain. line-integral-exists F {j} $\gamma$$
using *typeII-chain-line-integral-exists-boundary'* **assms** **by** *auto*
have *integ-exis-horiz:*
 $(\bigwedge k \gamma. (\exists (k', \gamma') \in \text{two-chain-horizontal-boundary two-chain. } \exists a \in \{0..1\}. \exists b \in \{0..1\}. a \leq b \wedge \text{subpath } a \text{ } b \text{ } \gamma' = \gamma) \implies$
 $\text{line-integral-exists } F \text{ } \{j\} \text{ } \gamma)$
using *integ-exis type-II-chain-horiz-bound-valid*
using *line-integral-exists-subpath*[of *F {j}*]
by (*fastforce simp add: two-chain-boundary-def two-chain-horizontal-boundary-def two-chain-vertical-boundary-def boundary-def*)
obtain $\mathcal{H} \mathcal{V}$ **where** *hv-props: finite \mathcal{H}*
 $(\forall (k, \gamma) \in \mathcal{H}. (\exists (k', \gamma') \in \text{two-chain-horizontal-boundary two-chain. } (\exists a \in \{0 .. 1\}. \exists b \in \{0..1\}. a \leq b \wedge \text{subpath } a \text{ } b \text{ } \gamma' = \gamma)))$
 $\text{one-chain} = \mathcal{H} \cup \mathcal{V}$
 $((\text{common-sudiv-exists } (\text{two-chain-vertical-boundary two-chain}) \mathcal{V})$
 $\vee \text{common-reparam-exists } \mathcal{V} \text{ } (\text{two-chain-vertical-boundary two-chain}))$
finite \mathcal{V}
boundary-chain \mathcal{V}
 $\forall (k, \gamma) \in \mathcal{V}. \text{valid-path } \gamma$
using *assms(1) unfolding only-horizontal-division-def* **by** *blast*
have *finite-j: finite {j} by auto*
show *line-integral-exists F {j} γ*
proof(*cases common-sudiv-exists (two-chain-vertical-boundary two-chain) \mathcal{V}*)
case *True*
show *?thesis*
using *gen-common-subdivision-imp-eq-line-integral(2)[OF True two-chain-vertical-boundary-is-boundary-cl*
hv-props(6) integ-exis-vert finite-two-chain-vertical-boundary[OF assms(4)] hv-props(5)
finite-j]
 integ-exis-horiz [of γ] *assms(3) case-prod-conv hv-props(2) hv-props(3)*
by *fastforce*
next
case *False*
have *i: {j} \subseteq Basis using j-is-y-axis real-pair-basis by auto*
have *ii: $\forall (k2, \gamma2) \in \text{two-chain-vertical-boundary two-chain. } \forall b \in \{j\}. \text{continuous-on}$*
(path-image $\gamma2$) ($\lambda x. F x \cdot b$)
using *F-anal-valid field-cont-on-typeII-region-cont-on-edges valid-typeII-div*

by (*fastforce simp add: analytically-valid-def two-chain-vertical-boundary-def boundary-def path-image-def*)
show *line-integral-exists* $F \{j\} \gamma$
using *common-reparam-exists-imp-eq-line-integral(2)[OF finite-j hv-props(5)]*

finite-two-chain-vertical-boundary[OF assms(4)] hv-props(6) two-chain-vertical-boundary-is-boundary-ch
ii - hv-props(7) type-II-chain-vert-bound-valid
integ-axis-horiz[of γ] assms(3) hv-props False
by *fastforce*

qed
qed

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 (*simp add: two-chain-boundary-def one-chain-line-integral-def two-chain-integral-def*)

let $?F\text{-}a' =$ *partial-vector-derivative ($\lambda a. (F a) \cdot j$) i*
have $(\sum_{(k,g) \in \text{boundary twoCube}} k * \text{line-integral } F \{j\} g) = \text{integral (cubeImage twoCube) } (\lambda a. ?F\text{-}a' a)$

if *twoCube* \in *two-chain* **for** *twoCube*
using *green-typeII-cube.GreenThm-typeII-twoCube(1) valid-typeII-div F-anal-valid one-chain-line-integral-def valid-two-chain-def*

by (*simp add: R2-axioms green-typeII-cube-axioms-def green-typeII-cube-def that*)

then have *double-sum-eq-sum:*
 $(\sum \text{twoCube} \in (\text{two-chain}). (\sum_{(k,g) \in \text{boundary twoCube}} k * \text{line-integral } F \{j\} g))$
 $= (\sum \text{twoCube} \in (\text{two-chain}). \text{integral (cubeImage twoCube) } (\lambda a. ?F\text{-}a' a))$

using *Finite-Cartesian-Product.sum-cong-aux* **by** *auto*
have *pairwise-disjoint-boundaries:* $\forall x \in (\text{boundary 'two-chain}). (\forall y \in (\text{boundary 'two-chain}). (x \neq y \longrightarrow (x \cap y = \{\})))$
using *valid-typeII-div* **by** (*fastforce simp add: image-def valid-two-chain-def pairwise-def*)

have *finite-boundaries:* $\forall B \in (\text{boundary 'two-chain}). \text{finite } B$
using *valid-typeII-div image-iff* **by** (*fastforce simp add: valid-two-cube-def valid-two-chain-def*)

have *boundary-inj: inj-on boundary two-chain*
using *valid-typeII-div* **by** (*force simp add: valid-two-cube-def valid-two-chain-def pairwise-def inj-on-def*)

have $(\sum x \in (\bigcup x \in \text{two-chain}. \text{boundary } x). \text{case } x \text{ of } (k, g) \Rightarrow k * \text{line-integral } F \{j\} g) =$
 $(\sum \text{twoCube} \in (\text{two-chain}). (\sum_{(k,g) \in \text{boundary twoCube}} k * \text{line-integral } F \{j\} g))$

using *sum.reindex[OF boundary-inj, of $\lambda x. (\sum_{(k,g) \in x} k * \text{line-integral } F \{j\} g)$]*

using *sum.Union-disjoint[OF finite-boundaries pairwise-disjoint-boundaries, of $\lambda x. \text{case } x \text{ of } (k, g) \Rightarrow (k::\text{int}) * \text{line-integral } F \{j\} g$]*

by auto
then show $(\sum C \in \text{two-chain. integral (cubeImage } C) (\lambda a. ?F\text{-}a' a)) = (\sum x \in (\bigcup x \in \text{two-chain. boundary } x). \text{ case } x \text{ of } (k, g) \Rightarrow k * \text{line-integral } F \{j\} g)$
using double-sum-eq-sum by auto
qed

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 –
let $?F\text{-}a' = (\text{partial-vector-derivative } (\lambda x. (F x) \cdot j) i)$
have *integral s ($\lambda x. ?F\text{-}a' x$) = two-chain-integral two-chain ($\lambda a. ?F\text{-}a' a$)*
proof (*simp add: two-chain-integral-def*)
have *partial-deriv-integrable:*
($?F\text{-}a'$ has-integral (integral (cubeImage twoCube) $?F\text{-}a'$)) (cubeImage twoCube)
if *twoCube \in two-chain for twoCube*
by (*simp add: analytically-valid-imp-part-deriv-integrable-on F-anal-valid integrable-integral that*)
then have *partial-deriv-integrable:*
 $\bigwedge \text{twoCubeImg. twoCubeImg} \in \text{cubeImage ' two-chain} \implies ((\lambda x. ?F\text{-}a' x)$
has-integral (integral (twoCubeImg) ($\lambda x. ?F\text{-}a' x$))) (twoCubeImg)
using *integrable-neg by force*
have *finite-images: finite (cubeImage ' two-chain)*
using *gen-division gen-division-def by auto*
have *negligible-images: pairwise ($\lambda S S'. \text{negligible } (S \cap S')$) (cubeImage ' two-chain)*
using *gen-division by (auto simp add: gen-division-def pairwise-def)*
have *inj-on cubeImage two-chain using valid-typeII-div valid-two-chain-def by auto*
then have $(\sum \text{twoCubeImg} \in \text{cubeImage ' two-chain. integral twoCubeImg } (\lambda x. ?F\text{-}a' x))$
 $= (\sum C \in \text{two-chain. integral (cubeImage } C) (\lambda a. ?F\text{-}a' a))$
using *sum.reindex by auto*
then show *integral s ($\lambda x. ?F\text{-}a' x$) = ($\sum C \in \text{two-chain. integral (cubeImage } C) (\lambda a. ?F\text{-}a' a)$)*
using *has-integral-Union[OF finite-images partial-deriv-integrable negligible-images]*
gen-division
by (*auto simp add: gen-division-def*)
qed
then show *?thesis*
using *GreenThm-typeII-twoChain F-anal-valid*
by auto
qed

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*

$F \{j\} \gamma$
proof –
let $?F\text{-}a' = (\text{partial-vector-derivative } (\lambda x. (F x) \cdot j) i)$

have *horiz-line-integral-zero*:
 $\text{one-chain-line-integral } F \{j\} (\text{two-chain-horizontal-boundary two-chain}) = 0$
proof (*simp add: one-chain-line-integral-def*)
have $\text{line-integral } F \{j\} (\text{snd oneCube}) = 0$
if $\text{oneCube} \in \text{two-chain-horizontal-boundary}(\text{two-chain})$ **for** oneCube
proof –
from *that* **obtain** $x \ y1 \ y2 \ k$
where $\text{horiz-edge-def: oneCube} = (k, (\lambda t::\text{real}. ((1 - t) * (y2) + t * y1, x::\text{real})))$
using *valid-typeII-div*
by (*auto simp add: typeII-twoCube-def two-chain-horizontal-boundary-def horizontal-boundary-def*)
let $?horiz\text{-edge} = (\text{snd oneCube})$
have $\text{horiz-edge-y-const: } \forall t. (?horiz\text{-edge } t) \cdot j = x$
by (*auto simp add: horiz-edge-def real-inner-class.inner-add-left euclidean-space-class.inner-not-same-Basis j-is-y-axis*)
have $\text{horiz-edge-is-straight-path: } ?horiz\text{-edge} = (\lambda t. (y2 + t * (y1 - y2), x))$
by (*auto simp: horiz-edge-def algebra-simps*)
have $\forall x. ?horiz\text{-edge}$ *differentiable at* x
using *horiz-edge-is-straight-path straight-path-differentiable-y*
by (*metis mult-commute-abs*)
then show $\text{line-integral } F \{j\} (\text{snd oneCube}) = 0$
using *line-integral-on-pair-straight-path(1) j-is-y-axis real-pair-basis horiz-edge-y-const*
by *blast*
qed
then show $(\sum x \in \text{two-chain-horizontal-boundary two-chain. case } x \text{ of } (k, g) \Rightarrow k * \text{line-integral } F \{j\} g) = 0$
by (*force intro: sum.neutral*)
qed

have *boundary-is-finite: finite (two-chain-boundary two-chain)*
unfolding *two-chain-boundary-def*
proof (*rule finite-UN-I*)
show *finite two-chain*
using *valid-typeII-div finite-image-iff gen-division-def valid-two-chain-def* **by** *auto*
show $\bigwedge a. a \in \text{two-chain} \implies \text{finite (boundary } a)$
by (*simp add: boundary-def horizontal-boundary-def vertical-boundary-def*)
qed
have *boundary-is-vert-hor*:
 $\text{two-chain-boundary two-chain} =$
 $(\text{two-chain-vertical-boundary two-chain}) \cup$
 $(\text{two-chain-horizontal-boundary two-chain})$
by (*auto simp add: two-chain-boundary-def two-chain-vertical-boundary-def two-chain-horizontal-boundary-def boundary-def*)

then have *hor-vert-finite*:
finite (*two-chain-vertical-boundary two-chain*)
finite (*two-chain-horizontal-boundary two-chain*)
using *boundary-is-finite* **by** *auto*
have *horiz-verti-disjoint*:
(*two-chain-vertical-boundary two-chain*) \cap
(*two-chain-horizontal-boundary two-chain*) = {}
proof (*simp add: two-chain-vertical-boundary-def two-chain-horizontal-boundary-def*
horizontal-boundary-def
vertical-boundary-def)
show ($\bigcup x \in \text{two-chain. } \{(-1, \lambda y. x(0, y)), (1::\text{int}, \lambda y. x(1::\text{real}, y))\} \cap$
 $(\bigcup x \in \text{two-chain. } \{(1, \lambda z. x(z, 0)), (-1, \lambda z. x(z, 1))\}) = \{\}$)
proof –
have $\{(-1, \lambda y. \text{twoCube}(0, y)), (1::\text{int}, \lambda y. \text{twoCube}(1, y))\} \cap$
 $\{(1, \lambda z. \text{twoCube2}(z, 0)), (-1, \lambda z. \text{twoCube2}(z, 1))\} = \{\}$
if *twoCube* \in *two-chain* *twoCube2* \in *two-chain* **for** *twoCube twoCube2*
proof (*cases twoCube = twoCube2*)
case *True*
have *card-4*: *card* $\{(-1, \lambda y. \text{twoCube2}(0::\text{real}, y)), (1::\text{int}, \lambda y. \text{twoCube2}$
 $(1, y)), (1, \lambda x. \text{twoCube2}(x, 0)), (-1, \lambda x. \text{twoCube2}(x, 1))\} = 4$
using *valid-typeII-div valid-two-chain-def that(2)*
by (*auto simp add: boundary-def vertical-boundary-def horizontal-boundary-def*
valid-two-cube-def)
show *?thesis*
using *card-4* **by** (*auto simp: True card-insert-if split: if-split-asm*)
next
case *False*
show *?thesis*
using *valid-typeII-div valid-two-chain-def*
by (*simp add: boundary-def vertical-boundary-def horizontal-boundary-def*
pairwise-def <twoCube \neq twoCube2> that)
qed
then have $\bigcup ((\lambda \text{twoCube. } \{(-1, \lambda y. \text{twoCube}(0::\text{real}, y)), (1::\text{real}, \lambda y.$
 $\text{twoCube}(1::\text{real}, y))\}) \text{ ‘two-chain})$
 $\cap \bigcup ((\lambda \text{twoCube. } \{(1::\text{int}, \lambda z. \text{twoCube}(z,$
 $0::\text{real}), (-1, \lambda z. \text{twoCube}(z, 1::\text{real}))\}) \text{ ‘two-chain})$
 $= \{\}$
by (*fastforce simp add: Union-disjoint*)
then show *?thesis* **by** *force*
qed
qed
have *one-chain-line-integral F {j}* (*two-chain-boundary two-chain*)
 $= \text{one-chain-line-integral } F \{j\} (\text{two-chain-vertical-boundary two-chain}) +$
 $\text{one-chain-line-integral } F \{j\} (\text{two-chain-horizontal-boundary two-chain})$
using *boundary-is-vert-hor horiz-verti-disjoint*
by (*simp add: hor-vert-finite sum.union-disjoint one-chain-line-integral-def*)
then have *y-axis-line-integral-is-only-vertical*:
one-chain-line-integral F {j} (*two-chain-boundary two-chain*)
 $= \text{one-chain-line-integral } F \{j\} (\text{two-chain-vertical-boundary$

```

two-chain)
  using horiz-line-integral-zero
  by auto

obtain  $\mathcal{H} \mathcal{V}$  where hv-props: finite  $\mathcal{H}$ 
  ( $\forall (k, \gamma) \in \mathcal{H}. (\exists (k', \gamma') \in \text{two-chain-horizontal-boundary two-chain.}$ 
    ( $\exists a \in \{0 .. 1\}.$ 
      ( $\exists b \in \{0..1\}.$ 
        ( $a \leq b \wedge \text{subpath } a \ b \ \gamma' = \gamma)))$ 
     $\gamma = \mathcal{H} \cup \mathcal{V}$ 
    (common-sudiv-exists (two-chain-vertical-boundary two-chain)  $\mathcal{V}$ 
       $\vee$  common-reparam-exists  $\mathcal{V}$  (two-chain-vertical-boundary two-chain))
    finite  $\mathcal{V}$ 
    boundary-chain  $\mathcal{V}$ 
     $\forall (k, \gamma) \in \mathcal{V}. \text{valid-path } \gamma$ 
    using only-horizontal-division
    by (fastforce simp add: only-horizontal-division-def)
  have finite {j} by auto
  then have eq-integrals: one-chain-line-integral  $F \{j\} \mathcal{V} = \text{one-chain-line-integral}$ 
 $F \{j\}$  (two-chain-vertical-boundary two-chain)
  proof (cases common-sudiv-exists (two-chain-vertical-boundary two-chain)  $\mathcal{V}$ )
    case True then show ?thesis
      using gen-common-subdivision-imp-eq-line-integral(1)[OF True two-chain-vertical-boundary-is-boundary-of- $\mathcal{V}$ ]
hv-props(6) - hor-vert-finite(1) hv-props(5)
      typeII-chain-line-integral-exists-boundary'
      by force
    next
    case False
      have integ-exis-vert:
         $\forall (k, \gamma) \in \text{two-chain-vertical-boundary two-chain. line-integral-exists } F \{j\} \ \gamma$ 
        using typeII-chain-line-integral-exists-boundary' assms
        by (fastforce simp add: valid-two-chain-def)
      have integ-exis:  $\forall (k, \gamma) \in \text{two-chain-boundary two-chain. line-integral-exists } F$ 
 $\{j\} \ \gamma$ 
        using typeII-cube-line-integral-exists-boundary by blast
      have valid-paths:  $\forall (k, \gamma) \in \text{two-chain-horizontal-boundary two-chain. valid-path}$ 
 $\gamma$ 
        using typeII-edges-are-valid-paths valid-typeII-div
        by (fastforce simp add: two-chain-boundary-def two-chain-horizontal-boundary-def
boundary-def)
      have integ-exis-horiz:
        ( $\bigwedge k \ \gamma. (\exists (k', \gamma') \in \text{two-chain-horizontal-boundary two-chain. } \exists a \in \{0..1\}.$ 
 $\exists b \in \{0..1\}. a \leq b \wedge \text{subpath } a \ b \ \gamma' = \gamma) \implies$ 
          line-integral-exists } F {j} \ \gamma)
        using integ-exis valid-paths line-integral-exists-subpath[of F {j}]
        by (fastforce simp add: two-chain-boundary-def two-chain-horizontal-boundary-def
two-chain-vertical-boundary-def boundary-def)
      have finite-j: finite {j} by auto
      have i: {j}  $\subseteq$  Basis using j-is-y-axis real-pair-basis by auto

```

have *ii*: $\forall (k2, \gamma2) \in \text{two-chain-vertical-boundary two-chain}. \forall b \in \{j\}. \text{continuous-on}$
 (path-image $\gamma2$) $(\lambda x. F x \cdot b)$
using *valid-typeII-div field-cont-on-typeII-region-cont-on-edges F-anal-valid*
by (fastforce simp add: *analytically-valid-def two-chain-vertical-boundary-def*
boundary-def path-image-def)
show *one-chain-line-integral F {j} $\mathcal{V} = \text{one-chain-line-integral F {j}}$ (two-chain-vertical-boundary*
two-chain)
using *hv-props(4) False common-reparam-exists-imp-eq-line-integral(1)[OF*
finite-j hv-props(5) hor-vert-finite(1) hv-props(6) two-chain-vertical-boundary-is-boundary-chain
ii
 - *hv-props(7) type-II-chain-vert-bound-valid]*
by fastforce
qed

have *line-integral-on-path*:
one-chain-line-integral F {j} $\gamma =$
one-chain-line-integral F {j} (two-chain-vertical-boundary two-chain)
proof (simp only: *one-chain-line-integral-def*)
have *line-integral F {j} (snd oneCube) = 0* **if** *oneCube: oneCube $\in \mathcal{H}$ for*
oneCube
proof -
obtain *k γ where k-gamma: (k, γ) = oneCube*
using *prod.collapse by blast*
then obtain *k' γ' a b where kp-gammap:*
(k'::int, γ') \in two-chain-horizontal-boundary two-chain
a \in {0 .. 1}
b \in {0..1}
subpath a b $\gamma' = \gamma$
using *hv-props oneCube*
by (smt case-prodE split-conv)
obtain *x y1 y2 where horiz-edge-def: (k', γ') = (k', ($\lambda t::real. ((1 - t) * (y2)$*
*+ t * y1, x::real)))*
using *valid-typeII-div kp-gammap*
by (auto simp add: *typeII-twoCube-def two-chain-horizontal-boundary-def*
horizontal-boundary-def)
have *horiz-edge-y-const: $\forall t. \gamma (t) \cdot j = x$*
using *horiz-edge-def kp-gammap(4)*
by (auto simp add: *real-inner-class.inner-add-left*
euclidean-space-class.inner-not-same-Basis j-is-y-axis subpath-def)
have *horiz-edge-is-straight-path:*
 *$\gamma = (\lambda t::real. ((1*y2 - a*y2) + a*y1 + ((b-a)*y1 - (b - a)*y2)*t,$*
x::real))
proof -
fix *t::real*
have *(1 - (b - a)*t + a) * (y2) + ((b-a)*t + a) * y1 = (1 - (b - a)*t*
*+ a) * (y2) + ((b-a)*t + a) * y1*
by auto
then have *$\gamma = (\lambda t::real. ((1 - (b - a)*t - a) * (y2) + ((b-a)*t + a) *$*
y1, x::real))


```

using horiz-edge-def Product-Type.snd-conv Product-Type.fst-conv kp-gammap(4)
by (simp add: subpath-def diff-diff-eq[symmetric])
also have ... = ( $\lambda t::real. ((1*y2 - (b - a)*y2*t - a*y2) + ((b-a)*y1*t$ 
+  $a*y1)$ ,  $x::real$ )
by(auto simp add: ring-class.ring-distrib(2) Groups.diff-diff-eq left-diff-distrib)
also have ... = ( $\lambda t::real. ((1*y2 - a*y2) + a*y1 + ((b-a)*y1 - (b -$ 
 $a)*y2)*t$ ,  $x::real$ )
by (force simp add: left-diff-distrib)
finally show  $\gamma = (\lambda t::real. ((1*y2 - a*y2) + a*y1 + ((b-a)*y1 - (b -$ 
 $a)*y2)*t$ ,  $x::real$ ) .
qed
show line-integral F {j} (snd oneCube) = 0
proof -
have  $\forall x. \gamma$  differentiable at x
by (simp add: horiz-edge-is-straight-path straight-path-differentiable-y)
then have line-integral F {j}  $\gamma = 0$ 
by (simp add: horiz-edge-y-const line-integral-on-pair-straight-path(1))
then show ?thesis
using Product-Type.snd-conv k-gamma by auto
qed
qed
then have  $\forall x \in \mathcal{H}. (case\ x\ of\ (k, g) \Rightarrow (k::int) * line-integral\ F\ \{j\}\ g) = 0$ 
by auto
then show ( $\sum x \in \gamma. case\ x\ of\ (k, g) \Rightarrow real-of-int\ k * line-integral\ F\ \{j\}\ g$ ) =
( $\sum x \in two-chain-vertical-boundary\ two-chain. case\ x\ of\ (k, g) \Rightarrow$ 
 $real-of-int\ k * line-integral\ F\ \{j\}\ g$ )
using hv-props(1) hv-props(3) hv-props(5) sum-zero-set hor-vert-finite(1)
eq-integrals
by (clarsimp simp add: one-chain-line-integral-def sum-zero-set)
qed
then have one-chain-line-integral F {j}  $\gamma =$ 
one-chain-line-integral F {j} (two-chain-vertical-boundary
two-chain)
using line-integral-on-path by auto
then have one-chain-line-integral F {j}  $\gamma =$ 
one-chain-line-integral F {j} (two-chain-boundary two-chain)
using y-axis-line-integral-is-only-vertical by auto
then show ?thesis
using valid-typeII-div GreenThm-typeII-divisible by auto
qed

```

lemma *GreenThm-typeII-divisible-region-boundary:*

assumes

two-cubes-trace-vertical-boundaries:

two-chain-vertical-boundary two-chain $\subseteq \gamma$ and

boundary-of-region-is-subset-of-partition-boundary:

$\gamma \subseteq two-chain-boundary\ two-chain$

shows *integral s (partial-vector-derivative ($\lambda x. (F\ x) \cdot j$) i) = one-chain-line-integral*
F {j} γ

```

proof –
  let ?F-a' = (partial-vector-derivative (λx. (F x) · j) i)

  have horiz-line-integral-zero:
    one-chain-line-integral F {j} (two-chain-horizontal-boundary two-chain) = 0
  proof (simp add: one-chain-line-integral-def)
    have line-integral F {j} (snd oneCube) = 0
    if one: oneCube ∈ two-chain-horizontal-boundary(two-chain) for oneCube
    proof –
      obtain x y1 y2 k where horiz-edge-def: oneCube = (k, (λt::real. ((1 - t) *
        (y2) + t * y1, x::real)))
      using valid-typeII-div one
      by (auto simp add: typeII-twoCube-def two-chain-horizontal-boundary-def
        horizontal-boundary-def)
      let ?horiz-edge = (snd oneCube)
      have horiz-edge-y-const: ∀ t. (?horiz-edge t) · j = x
      using horiz-edge-def
      by (auto simp add: real-inner-class.inner-add-left
        euclidean-space-class.inner-not-same-Basis j-is-y-axis)
      have horiz-edge-is-straight-path:
        ?horiz-edge = (λt. (y2 + t * (y1 - y2), x))
      by (simp add: add-diff-eq diff-add-eq mult.commute right-diff-distrib horiz-edge-def)
      show line-integral F {j} (snd oneCube) = 0
      by (metis horiz-edge-is-straight-path horiz-edge-y-const line-integral-on-pair-straight-path(1)
        mult.commute straight-path-differentiable-y)
    qed
    then show (∑ x∈two-chain-horizontal-boundary two-chain. case x of (k, g) ⇒
      k * line-integral F {j} g) = 0
    by (simp add: prod.case-eq-if)
    qed

  have boundary-is-finite: finite (two-chain-boundary two-chain)
  unfolding two-chain-boundary-def
  proof (rule finite-UN-I)
    show finite two-chain
    using valid-typeII-div finite-image-iff gen-division-def valid-two-chain-def by
    auto
    show ∧a. a ∈ two-chain ⇒ finite (boundary a)
    by (simp add: boundary-def horizontal-boundary-def vertical-boundary-def)
  qed
  have boundary-is-vert-hor:
    two-chain-boundary two-chain = (two-chain-vertical-boundary two-chain) ∪
    (two-chain-horizontal-boundary two-chain)
  by(auto simp add: two-chain-boundary-def two-chain-vertical-boundary-def two-chain-horizontal-boundary-def
    boundary-def)
  then have hor-vert-finite:
    finite (two-chain-vertical-boundary two-chain)
    finite (two-chain-horizontal-boundary two-chain)
    using boundary-is-finite by auto

```

have *horiz-verti-disjoint*:
 $(\text{two-chain-vertical-boundary two-chain}) \cap (\text{two-chain-horizontal-boundary two-chain})$
 $= \{\}$
proof (*simp add: two-chain-vertical-boundary-def two-chain-horizontal-boundary-def*
horizontal-boundary-def
vertical-boundary-def)
show $(\bigcup_{x \in \text{two-chain}} \{(-1, \lambda y. x(0, y)), (1::\text{int}, \lambda y. x(1::\text{real}, y))\}) \cap$
 $(\bigcup_{x \in \text{two-chain}} \{(1, \lambda z. x(z, 0)), (-1, \lambda z. x(z, 1))\}) = \{\}$
proof –
have $\{(-1, \lambda y. \text{twoCube}(0, y)), (1::\text{int}, \lambda y. \text{twoCube}(1, y))\} \cap$
 $\{(1, \lambda y. \text{twoCube2}(y, 0)), (-1, \lambda y. \text{twoCube2}(y, 1))\} = \{\}$
if *twoCube* \in *two-chain* *twoCube2* \in *two-chain* **for** *twoCube* *twoCube2*
proof (*cases twoCube = twoCube2*)
case *True*
have $\text{card} \{(-1, \lambda y. \text{twoCube2}(0::\text{real}, y)), (1::\text{int}, \lambda y. \text{twoCube2}(1, y)),$
 $(1, \lambda x. \text{twoCube2}(x, 0)), (-1, \lambda x. \text{twoCube2}(x, 1))\} = 4$
using *valid-typeII-div valid-two-chain-def that(2)*
by (*auto simp add: boundary-def vertical-boundary-def horizontal-boundary-def*
valid-two-cube-def)
then show *?thesis*
by (*auto simp: True card-insert-if split: if-split-asm*)
next
case *False* **show** *?thesis*
using *valid-typeII-div valid-two-chain-def*
by (*simp add: boundary-def vertical-boundary-def horizontal-boundary-def*
pairwise-def <twoCube \neq twoCube2> that(1) that(2))
qed
then have $\bigcup ((\lambda \text{twoCube}. \{(-1, \lambda y. \text{twoCube}(0::\text{real}, y)), (1::\text{real}, \lambda y.$
 $\text{twoCube}(1::\text{real}, y))\}) \text{ 'two-chain})$
 $\cap \bigcup ((\lambda \text{twoCube}. \{(1::\text{int}, \lambda z. \text{twoCube}(z,$
 $0::\text{real}), (-1, \lambda z. \text{twoCube}(z, 1::\text{real}))\}) \text{ 'two-chain})$
 $= \{\}$
by (*force simp add: Complete-Lattices.Union-disjoint*)
then show *?thesis* **by** *force*
qed
qed
have *one-chain-line-integral F {j} (two-chain-boundary two-chain)*
 $= \text{one-chain-line-integral } F \{j\} (\text{two-chain-vertical-boundary two-chain}) +$
 $\text{one-chain-line-integral } F \{j\} (\text{two-chain-horizontal-boundary two-chain})$
using *boundary-is-vert-hor horiz-verti-disjoint*
by (*auto simp add: one-chain-line-integral-def hor-vert-finite sum.union-disjoint*)
then have *y-axis-line-integral-is-only-vertical*:
 $\text{one-chain-line-integral } F \{j\} (\text{two-chain-boundary two-chain})$
 $= \text{one-chain-line-integral } F \{j\} (\text{two-chain-vertical-boundary two-chain})$
using *horiz-line-integral-zero* **by** *auto*

have $\exists \mathcal{H}. \mathcal{H} \subseteq (\text{two-chain-horizontal-boundary two-chain}) \wedge$
 $\gamma = \mathcal{H} \cup (\text{two-chain-vertical-boundary two-chain})$
proof

```

let ? $\mathcal{H} = \gamma - (\text{two-chain-vertical-boundary two-chain})$ 
show ? $\mathcal{H} \subseteq \text{two-chain-horizontal-boundary two-chain} \wedge \gamma = ?\mathcal{H} \cup \text{two-chain-vertical-boundary}$ 
two-chain
  using two-cubes-trace-vertical-boundaries
  boundary-of-region-is-subset-of-partition-boundary boundary-is-vert-hor
  by blast
qed
then obtain  $\mathcal{H}$  where
  h-props:  $\mathcal{H} \subseteq (\text{two-chain-horizontal-boundary two-chain})$ 
   $\gamma = \mathcal{H} \cup (\text{two-chain-vertical-boundary two-chain})$ 
  by auto
have h-vert-disj:  $\mathcal{H} \cap (\text{two-chain-vertical-boundary two-chain}) = \{\}$ 
  using horiz-vert-disjoint h-props(1) by auto
have h-finite: finite  $\mathcal{H}$ 
  using hor-vert-finite h-props(1) Finite-Set.rev-finite-subset by force
have line-integral-on-path:
  one-chain-line-integral F {j}  $\gamma =$ 
  one-chain-line-integral F {j}  $\mathcal{H} + \text{one-chain-line-integral F \{j\} (two-chain-vertical-boundary}$ 
two-chain)
  by (auto simp add: one-chain-line-integral-def h-props sum.union-disjoint[OF
h-finite hor-vert-finite(1) h-vert-disj])

have one-chain-line-integral F {j}  $\mathcal{H} = 0$ 
proof (simp add: one-chain-line-integral-def)
  have line-integral F {j} (snd oneCube) = 0
  if oneCube: oneCube  $\in \text{two-chain-horizontal-boundary(two-chain)}$  for oneCube
  proof -
    obtain x y1 y2 k where horiz-edge-def: oneCube = (k, ( $\lambda t::\text{real}.$  ((1 - t) *
(y2) + t * y1, x::real)))
    using valid-typeII-div oneCube
    by (auto simp add: typeII-twoCube-def two-chain-horizontal-boundary-def
horizontal-boundary-def)
    let ?horiz-edge = (snd oneCube)
    have horiz-edge-y-const:  $\forall t. (?horiz-edge t) \cdot j = x$ 
    by (simp add: j-is-y-axis horiz-edge-def)
    have horiz-edge-is-straight-path:
      ?horiz-edge = ( $\lambda t. (y2 + t * (y1 - y2), x)$ )
    by (simp add: add-diff-eq diff-add-eq mult.commute right-diff-distrib horiz-edge-def)
    show line-integral F {j} (snd oneCube) = 0
    by (simp add: horiz-edge-is-straight-path j-is-y-axis line-integral-on-pair-straight-path(1)
mult.commute straight-path-differentiable-y)
  qed
then have  $\forall \text{oneCube} \in \mathcal{H}. \text{line-integral F \{j\} (snd oneCube) = 0}$ 
  using h-props by auto
then have  $\forall x \in \mathcal{H}. (\text{case } x \text{ of } (k, g) \Rightarrow (k::\text{int}) * \text{line-integral F \{j\} } g) = 0$ 
  by auto
then show ( $\sum x \in \mathcal{H}. \text{case } x \text{ of } (k, g) \Rightarrow k * \text{line-integral F \{j\} } g) = 0$ 
  using sum-not-0 by blast
qed

```

then have *one-chain-line-integral* $F \{j\} \gamma =$
one-chain-line-integral $F \{j\}$ (*two-chain-vertical-boundary*
two-chain)
using *line-integral-on-path* **by** *auto*
then have *one-chain-line-integral* $F \{j\} \gamma =$
one-chain-line-integral $F \{j\}$ (*two-chain-boundary* *two-chain*)
using *y-axis-line-integral-is-only-vertical* **by** *auto*
then show *?thesis*
using *valid-typeII-div* *GreenThm-typeII-divisible* **by** *auto*
qed

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*) $(\lambda a. - \text{partial-vector-derivative } (\lambda p. F p \cdot i)$
 $j a) = \text{one-chain-line-integral } F \{i\}$ (*boundary* *twoC*)
 $\forall (k, \gamma) \in \text{boundary } \text{twoC}. \text{line-integral-exists } F \{i\} \gamma$

proof –

let *?bottom-edge* = $(\lambda x. \text{twoC}(x, 0))$

let *?right-edge* = $(\lambda y. \text{twoC}(1, y))$

let *?top-edge* = $(\lambda x. \text{twoC}(x, 1))$

let *?left-edge* = $(\lambda y. \text{twoC}(0, y))$

have *line-integral-around-boundary*:

one-chain-line-integral $F \{i\}$ (*boundary* *twoC*) =

line-integral $F \{i\}$ *?bottom-edge* + *line-integral* $F \{i\}$ *?right-edge*

– *line-integral* $F \{i\}$ *?top-edge* – *line-integral* $F \{i\}$ *?left-edge*

proof (*simp* *add*: *one-chain-line-integral-def* *horizontal-boundary-def* *vertical-boundary-def* *boundary-def*)

have *finite1*: *finite* $\{(- 1::\text{int}, \lambda y. \text{twoC}(0, y)), (1, \lambda y. \text{twoC}(1, y)), (- 1,$
 $\lambda x. \text{twoC}(x, 1))\}$ **by** *auto*

have *sum-step1*: $(\sum (k, g) \in \{(- 1::\text{int}, \lambda y. \text{twoC}(0, y)), (1, \lambda y. \text{twoC}(1,$
 $y)), (1, \lambda x. \text{twoC}(x, 0)), (- 1, \lambda x. \text{twoC}(x, 1))\}. k * \text{line-integral } F \{i\} g) =$
 $\text{line-integral } F \{i\} (\lambda x. \text{twoC}(x, 0)) + (\sum (k, g) \in \{(- 1::\text{int},$
 $\lambda y. \text{twoC}(0, y)), (1, \lambda y. \text{twoC}(1, y)), (- 1, \lambda x. \text{twoC}(x, 1))\}. k * \text{line-integral } F \{i\} g)$

using *sum.insert-remove* [*OF* *finite1*] *valid-two-cube*

by (*auto* *simp*: *horizontal-boundary-def* *vertical-boundary-def* *boundary-def* *valid-two-cube-def* *card-insert-if-split*: *if-split-asm*)

have *three-distinct-edges*: *card* $\{(- 1::\text{int}, \lambda y. \text{twoC}(0, y)), (1, \lambda y. \text{twoC}(1,$
 $y)), (- 1, \lambda x. \text{twoC}(x, 1))\} = 3$

using *valid-two-cube*

apply (*simp add: one-chain-line-integral-def horizontal-boundary-def vertical-boundary-def boundary-def valid-two-cube-def*)
by (*auto simp: card-insert-if split: if-split-asm*)
have *finite2: finite* $\{(- 1::int, \lambda y. twoC (0, y)), (1, \lambda y. twoC (1, y))\}$ **by** *auto*
have *sum-step2: $(\sum (k, g) \in \{(- 1::int, \lambda y. twoC (0, y)), (1, \lambda y. twoC (1, y)), (-1, \lambda x. twoC (x, 1))\}. k * line-integral F \{i\} g =$*
 $(- line-integral F \{i\} (\lambda x. twoC (x, 1))) + (\sum (k, g) \in \{(- 1::int, \lambda y. twoC (0, y)), (1, \lambda y. twoC (1, y))\}. k * line-integral F \{i\} g)$
using *sum.insert-remove [OF finite2] three-distinct-edges*
by (*auto simp: card-insert-if split: if-split-asm*)
have *two-distinct-edges: card* $\{(- 1::int, \lambda y. twoC (0, y)), (1, \lambda y. twoC (1, y))\} = 2$
using *three-distinct-edges*
by (*simp add: one-chain-line-integral-def horizontal-boundary-def vertical-boundary-def boundary-def*)
have *finite3: finite* $\{(- 1::int, \lambda y. twoC (0, y))\}$ **by** *auto*
have *sum-step3: $(\sum (k, g) \in \{(- 1::int, \lambda y. twoC (0, y)), (1, \lambda y. twoC (1, y))\}. k * line-integral F \{i\} g =$*
 $(line-integral F \{i\} (\lambda y. twoC (1, y))) + (\sum (k, g) \in \{(- 1::real, \lambda y. twoC (0, y))\}. k * line-integral F \{i\} g)$
using *sum.insert-remove [OF finite2] three-distinct-edges*
by (*auto simp: card-insert-if split: if-split-asm*)
show $(\sum x \in \{(- 1::int, \lambda y. twoC (0, y)), (1, \lambda y. twoC (1, y)), (1, \lambda x. twoC (x, 0)), (- 1, \lambda x. twoC (x, 1))\}. case x of (k, g) \Rightarrow k * line-integral F \{i\} g =$
 $line-integral F \{i\} (\lambda x. twoC (x, 0)) + line-integral F \{i\} (\lambda y. twoC (1, y)) - line-integral F \{i\} (\lambda x. twoC (x, 1)) - line-integral F \{i\} (\lambda y. twoC (0, y))$
using *sum-step1 sum-step2 sum-step3 by auto*
qed
obtain *a b g1 g2 where*
twoCisTypeI: a < b
 $(\forall x \in cbox a b. g2 x \leq g1 x)$
 $cubeImage twoC = \{(x, y). x \in cbox a b \wedge y \in cbox (g2 x) (g1 x)\}$
 $twoC = (\lambda(x, y). ((1 - x) * a + x * b, (1 - y) * g2 ((1 - x) * a + x * b) + y * g1 ((1 - x) * a + x * b)))$
g1 piecewise-C1-differentiable-on $\{a .. b\}$
g2 piecewise-C1-differentiable-on $\{a .. b\}$
 $(\lambda x. twoC(x, 0)) = (\lambda x. (a + (b - a) * x, g2 (a + (b - a) * x)))$
 $(\lambda y. twoC(1, y)) = (\lambda x. (b, g2 b + x *_R (g1 b - g2 b)))$
 $(\lambda x. twoC(x, 1)) = (\lambda x. (a + (b - a) * x, g1 (a + (b - a) * x)))$
 $(\lambda y. twoC(0, y)) = (\lambda x. (a, g2 a + x *_R (g1 a - g2 a)))$
using *two-cube and typeI-cube-explicit-spec[oftwoC]* **by** *auto*
have *bottom-edge-smooth: $(\lambda x. twoC (x, 0))$ piecewise-C1-differentiable-on* $\{0..1\}$
using *typeI-twoCube-smooth-edges two-cube boundary-def vertical-boundary-def horizontal-boundary-def*
by *auto*
have *top-edge-smooth: ?top-edge piecewise-C1-differentiable-on* $\{0..1\}$
using *typeI-twoCube-smooth-edges two-cube boundary-def vertical-boundary-def*

horizontal-boundary-def
by auto
show *integral (cubeImage twoC) (λa. - partial-vector-derivative (λp. F p · i) j a) = one-chain-line-integral F {i} (boundary twoC)*
using *Greens-thm-type-I[OF twoCisTypeI(3) twoCisTypeI(7) bottom-edge-smooth twoCisTypeI(8) twoCisTypeI(9) top-edge-smooth twoCisTypeI(10) f-analytically-valid twoCisTypeI(2) twoCisTypeI(1)] line-integral-around-boundary*
by auto
have *line-integral-exists F {i} (λy. twoC (0, y))*
line-integral-exists F {i} (λx. twoC (x, 1))
line-integral-exists F {i} (λy. twoC (1, y))
line-integral-exists F {i} (λx. twoC (x, 0))
using *Greens-thm-type-I[OF twoCisTypeI(3) twoCisTypeI(7) bottom-edge-smooth twoCisTypeI(8) twoCisTypeI(9) top-edge-smooth twoCisTypeI(10) f-analytically-valid twoCisTypeI(2) twoCisTypeI(1)] line-integral-around-boundary*
by auto
then have *line-integral-exists F {i} γ if (k,γ) ∈ boundary twoC for k γ*
using that by *(auto simp add: boundary-def horizontal-boundary-def vertical-boundary-def)*
then show *∀(k,γ) ∈ boundary twoC. line-integral-exists F {i} γ by auto*
qed

lemma *line-integral-exists-on-typeI-Cube-boundaries'*:
assumes *(k,γ) ∈ boundary twoC*
shows *line-integral-exists F {i} γ*
using *assms two-cube valid-two-cube f-analytically-valid GreenThm-typeI-twoCube(2)*
by blast

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)
(λx. (F x) · i) j
begin

lemma *two-chain-valid-valid-cubes: ∀ two-cube ∈ two-chain. valid-two-cube two-cube*
using *valid-typeI-div*
by *(auto simp add: valid-two-chain-def)*

lemma *typeI-cube-line-integral-exists-boundary'*:
assumes *∀ two-cube ∈ two-chain. typeI-twoCube two-cube*
assumes *∀ twoC ∈ two-chain. analytically-valid (cubeImage twoC) (λx. (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*
γ

proof –
have *integ-exis*: $\forall (k, \gamma) \in \text{two-chain-boundary two-chain. line-integral-exists } F$
 $\{i\} \gamma$
using *green-typeI-cube.line-integral-exists-on-typeI-Cube-boundaries'* [of *i j*] *assms*
using *R2-axioms green-typeI-cube-axioms-def green-typeI-cube-def two-chain-boundary-def*
by *fastforce*
then show *integ-exis-vert*:
 $\forall (k, \gamma) \in \text{two-chain-vertical-boundary two-chain. line-integral-exists } F \{i\} \gamma$
by (*simp add: two-chain-boundary-def two-chain-vertical-boundary-def boundary-def*)
qed

lemma *typeI-cube-line-integral-exists-boundary''*:

$\forall (k, \gamma) \in \text{two-chain-horizontal-boundary two-chain. line-integral-exists } F \{i\} \gamma$
proof –
have *integ-exis*: $\forall (k, \gamma) \in \text{two-chain-boundary two-chain. line-integral-exists } F$
 $\{i\} \gamma$
using *green-typeI-cube.line-integral-exists-on-typeI-Cube-boundaries'* [of *i j*] *valid-typeI-div*
apply (*simp add: two-chain-boundary-def boundary-def*)
using *F-anal-valid R2-axioms green-typeI-cube-axioms-def green-typeI-cube-def*
two-chain-valid-valid-cubes **by** *fastforce*
then show *integ-exis-horiz*:
 $\forall (k, \gamma) \in \text{two-chain-horizontal-boundary two-chain. line-integral-exists } F \{i\} \gamma$
by (*simp add: two-chain-boundary-def two-chain-horizontal-boundary-def boundary-def*)
qed

lemma *typeI-cube-line-integral-exists-boundary*:

$\forall (k, \gamma) \in \text{two-chain-boundary two-chain. line-integral-exists } F \{i\} \gamma$
using *typeI-cube-line-integral-exists-boundary' typeI-cube-line-integral-exists-boundary''*
apply (*auto simp add: two-chain-boundary-def two-chain-horizontal-boundary-def*
two-chain-vertical-boundary-def)
by (*meson R2-axioms green-typeI-chain.F-anal-valid green-typeI-chain-axioms*
green-typeI-cube.line-integral-exists-on-typeI-Cube-boundaries' green-typeI-cube-axioms-def
green-typeI-cube-def two-chain-valid-valid-cubes valid-typeI-div)

lemma *type-I-chain-horiz-bound-valid*:

$\forall (k, \gamma) \in \text{two-chain-horizontal-boundary two-chain. valid-path } \gamma$
using *typeI-edges-are-valid-paths valid-typeI-div*
by (*force simp add: two-chain-boundary-def two-chain-horizontal-boundary-def*
boundary-def)

lemma *type-I-chain-vert-bound-valid*:

assumes $\forall \text{two-cube} \in \text{two-chain. typeI-twoCube two-cube}$
shows $\forall (k, \gamma) \in \text{two-chain-vertical-boundary two-chain. valid-path } \gamma$
using *typeI-edges-are-valid-paths assms(1)*
by (*force simp add: two-chain-boundary-def two-chain-vertical-boundary-def boundary-def*)

lemma *members-of-only-vertical-div-line-integrable'*:

assumes *only-vertical-division one-chain two-chain*
 $(k::\text{int}, \gamma) \in \text{one-chain}$


```

    (k::int, γ)∈one-chain
    finite two-chain
  shows line-integral-exists F {i} γ
proof –
  have integ-exis: ∀(k,γ) ∈ two-chain-boundary two-chain. line-integral-exists F
  {i} γ
  using typeI-cube-line-integral-exists-boundary by blast
  have integ-exis-horiz:
    ∀(k,γ) ∈ two-chain-horizontal-boundary two-chain. line-integral-exists F {i} γ
  using typeI-cube-line-integral-exists-boundary'' assms by auto
  have valid-paths: ∀(k,γ) ∈ two-chain-vertical-boundary two-chain. valid-path γ
  using type-I-chain-vert-bound-valid valid-typeI-div by linarith
  have integ-exis-vert:
    (∧k γ. (∃(k', γ')∈two-chain-vertical-boundary two-chain. ∃ a∈{0..1}. ∃ b∈{0..1}.
    a ≤ b ∧ subpath a b γ' = γ) ⇒
      line-integral-exists F {i} γ)
  using integ-exis valid-paths line-integral-exists-subpath[of F {i}]
  by (fastforce simp add: two-chain-boundary-def two-chain-horizontal-boundary-def
    two-chain-vertical-boundary-def boundary-def)
obtain  $\mathcal{H}$   $\mathcal{V}$  where hv-props: finite  $\mathcal{V}$ 
    (∀(k,γ) ∈  $\mathcal{V}$ . (∃(k', γ') ∈ two-chain-vertical-boundary two-chain.
      (∃ a ∈ {0 .. 1}. ∃ b ∈ {0..1}. a ≤ b ∧ subpath a b γ' = γ)))
    one-chain =  $\mathcal{H} \cup \mathcal{V}$ 
    (common-sudiv-exists (two-chain-horizontal-boundary two-chain)  $\mathcal{H}$ )
    ∨ common-reparam-exists  $\mathcal{H}$  (two-chain-horizontal-boundary two-chain)
    finite  $\mathcal{H}$ 
    boundary-chain  $\mathcal{H}$ 
    ∀(k,γ)∈ $\mathcal{H}$ . valid-path γ
  using assms(1) unfolding only-vertical-division-def by blast
  have finite-i: finite {i} by auto
  show line-integral-exists F {i} γ
  proof(cases common-sudiv-exists (two-chain-horizontal-boundary two-chain)  $\mathcal{H}$ )
    case True
      show ?thesis
      using gen-common-subdivision-imp-eq-line-integral(2)[OF True two-chain-horizontal-boundary-is-boundary
        hv-props(6) integ-exis-horiz finite-two-chain-horizontal-boundary[OF assms(4)] hv-props(5)
        finite-i]
        integ-exis-vert[of γ] assms(3) case-prod-conv hv-props(2) hv-props(3)
      by fastforce
    next
      case False
      have i: {i} ⊆ Basis using i-is-x-axis real-pair-basis by auto
      have ii: ∀(k2, γ2)∈two-chain-horizontal-boundary two-chain. ∀ b∈{i}. continuous-on
      (path-image γ2) (λx. F x · b)
      using assms field-cont-on-typeI-region-cont-on-edges F-anal-valid valid-typeI-div
      by (fastforce simp add: analytically-valid-def two-chain-horizontal-boundary-def
        boundary-def path-image-def)
      show line-integral-exists F {i} γ
      using common-reparam-exists-imp-eq-line-integral(2)[OF finite-i hv-props(5)]

```

finite-two-chain-horizontal-boundary[*OF assms*(4)] *hv-props*(6) *two-chain-horizontal-boundary-is-boundary-ch*
ii

- *hv-props*(7) *type-I-chain-horiz-bound-valid*
integ-exis-vert[of γ] *False*
assms(3) *hv-props*(2-4) **by** *fastforce*

qed
qed

lemma *GreenThm-typeI-two-chain*:

two-chain-integral two-chain ($\lambda a. - \text{partial-vector-derivative } (\lambda x. (F x) \cdot i) j a$)
= *one-chain-line-integral* $F \{i\}$ (*two-chain-boundary two-chain*)

proof (*simp add: two-chain-boundary-def one-chain-line-integral-def two-chain-integral-def*)

let $?F\text{-}b' = \text{partial-vector-derivative } (\lambda x. (F x) \cdot i) j$
have *all-two-cubes-have-four-distict-edges*: $\forall \text{twoCube} \in \text{two-chain}. \text{card } (\text{boundary twoCube}) = 4$

using *valid-typeI-div valid-two-chain-def valid-two-cube-def* **by** *auto*

have *no-shared-edges-have-similar-orientations*:

$\forall \text{twoCube1} \in \text{two-chain}. \forall \text{twoCube2} \in \text{two-chain}. \text{twoCube1} \neq \text{twoCube2} \longrightarrow$
 $\text{boundary twoCube1} \cap \text{boundary twoCube2} = \{\}$

using *valid-typeI-div valid-two-chain-def*

by (*auto simp add: pairwise-def*)

have $(\sum (k,g) \in \text{boundary twoCube}. k * \text{line-integral } F \{i\} g) = \text{integral } (\text{cubeImage twoCube}) (\lambda a. - ?F\text{-}b' a)$

if $\text{twoCube} \in \text{two-chain}$ **for** twoCube

proof -

have *analytically-val*: *analytically-valid* (*cubeImage twoCube*) ($\lambda x. F x \cdot i$) j

using *that F-anal-valid* **by** *auto*

show $(\sum (k, g) \in \text{boundary twoCube}. k * \text{line-integral } F \{i\} g) = \text{integral } (\text{cubeImage twoCube}) (\lambda a. - ?F\text{-}b' a)$

using *green-typeI-cube.GreenThm-typeI-twoCube*

apply (*simp add: one-chain-line-integral-def*)

by (*simp add: R2-axioms analytically-val green-typeI-cube-axioms-def green-typeI-cube-def that two-chain-valid-valid-cubes valid-typeI-div*)

qed

then have *double-sum-eq-sum*:

$(\sum \text{twoCube} \in (\text{two-chain}). (\sum (k,g) \in \text{boundary twoCube}. k * \text{line-integral } F \{i\} g))$

= $(\sum \text{twoCube} \in (\text{two-chain}). \text{integral } (\text{cubeImage twoCube}) (\lambda a. - ?F\text{-}b' a))$

using *Finite-Cartesian-Product.sum-cong-aux* **by** *auto*

have *pairwise-disjoint-boundaries*: $\forall x \in (\text{boundary 'two-chain}). (\forall y \in (\text{boundary 'two-chain}). (x \neq y \longrightarrow (x \cap y = \{\})))$

using *no-shared-edges-have-similar-orientations*

by (*force simp add: image-def disjoint-iff-not-equal*)

have *finite-boundaries*: $\forall B \in (\text{boundary 'two-chain}). \text{finite } B$

using *all-two-cubes-have-four-distict-edges*

using *image-iff* **by** *fastforce*

have *boundary-inj*: *inj-on boundary two-chain*

using *all-two-cubes-have-four-distict-edges* **and** *no-shared-edges-have-similar-orientations*

by (*force simp add: inj-on-def*)
have $(\sum x \in (\bigcup (\text{boundary}' \text{two-chain})). \text{case } x \text{ of } (k, g) \Rightarrow k * \text{line-integral } F \{i\} g)$
 $= (\text{sum} \circ \text{sum}) (\lambda x. \text{case } x \text{ of } (k, g) \Rightarrow (k::\text{int}) * \text{line-integral } F \{i\} g)$
(*boundary' two-chain*)
using *sum.Union-disjoint[OF finite-boundaries pairwise-disjoint-boundaries]*
by *simp*
also have $\dots = (\sum \text{twoCube} \in (\text{two-chain}). (\sum (k,g) \in \text{boundary twoCube}. k * \text{line-integral } F \{i\} g))$
using *sum.reindex[OF boundary-inj, of $\lambda x. (\sum (k,g) \in x. k * \text{line-integral } F \{i\} g)$]*
by *auto*
finally show $(\sum C \in \text{two-chain}. - \text{integral } (\text{cubeImage } C) (\text{partial-vector-derivative } (\lambda x. F x \cdot i) j)) = (\sum x \in (\bigcup x \in \text{two-chain}. \text{boundary } x). \text{case } x \text{ of } (k, g) \Rightarrow \text{real-of-int } k * \text{line-integral } F \{i\} g)$
using *double-sum-eq-sum* **by** *auto*
qed

lemma *GreenThm-typeI-divisible:*

assumes *gen-division: gen-division s (cubeImage' two-chain)*
shows *integral s ($\lambda x. - \text{partial-vector-derivative } (\lambda a. F(a) \cdot i) j x$) = one-chain-line-integral $F \{i\}$ (two-chain-boundary two-chain)*
proof –
let $?F\text{-}b' = \text{partial-vector-derivative } (\lambda a. F(a) \cdot i) j$
have *integral s ($\lambda x. - ?F\text{-}b' x$) = two-chain-integral two-chain ($\lambda a. - ?F\text{-}b' a$)*
proof (*simp add: two-chain-integral-def*)
have $(\sum f \in \text{two-chain}. \text{integral } (\text{cubeImage } f) (\text{partial-vector-derivative } (\lambda p. F p \cdot i) j)) = \text{integral } s (\text{partial-vector-derivative } (\lambda p. F p \cdot i) j)$
by (*metis analytically-valid-imp-part-deriv-integrable-on F-anal-valid gen-division two-chain-integral-def two-chain-integral-eq-integral-divisible valid-typeI-div*)
then show – *integral s (partial-vector-derivative ($\lambda a. F a \cdot i$) j) = ($\sum C \in \text{two-chain}. - \text{integral } (\text{cubeImage } C) (\text{partial-vector-derivative } (\lambda a. F a \cdot i) j)$)*
by (*simp add: sum-negf*)
qed
then show *?thesis*
using *GreenThm-typeI-two-chain assms* **by** *auto*
qed

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 $\subseteq \gamma$ and
boundary-of-region-is-subset-of-partition-boundary:
 $\gamma \subseteq \text{two-chain-boundary two-chain}$
shows *integral s ($\lambda x. - \text{partial-vector-derivative } (\lambda a. F(a) \cdot i) j x$) = one-chain-line-integral $F \{i\} \gamma$*
proof –
let $?F\text{-}b' = \text{partial-vector-derivative } (\lambda a. F(a) \cdot i)$

have *all-two-cubes-have-four-distict-edges*: $\forall \text{twoCube} \in \text{two-chain}. \text{card} (\text{boundary twoCube}) = 4$
using *valid-typeI-div valid-two-chain-def valid-two-cube-def* **by** *auto*
have *no-shared-edges-have-similar-orientations*:
 $\forall \text{twoCube1} \in \text{two-chain}. \forall \text{twoCube2} \in \text{two-chain}.$
 $\text{twoCube1} \neq \text{twoCube2} \longrightarrow \text{boundary twoCube1} \cap \text{boundary twoCube2} = \{\}$
using *valid-typeI-div valid-two-chain-def* **by** (*auto simp add: pairwise-def*)

have *vert-line-integral-zero*:
 $\text{one-chain-line-integral } F \{i\} (\text{two-chain-vertical-boundary two-chain}) = 0$
proof (*simp add: one-chain-line-integral-def*)
have $\text{line-integral } F \{i\} (\text{snd oneCube}) = 0$
if $\text{oneCube} \in \text{two-chain-vertical-boundary}(\text{two-chain})$ **for** oneCube
proof –
obtain $x \ y1 \ y2 \ k$ **where** *vert-edge-def*: $\text{oneCube} = (k, (\lambda t::\text{real}. (x::\text{real}, (1 - t) * (y2) + t * y1)))$
using *valid-typeI-div oneCube*
by (*auto simp add: typeI-twoCube-def two-chain-vertical-boundary-def vertical-boundary-def*)
let $?vert\text{-edge} = (\text{snd oneCube})$
have *vert-edge-x-const*: $\forall t. (?vert\text{-edge } t) \cdot i = x$
by (*simp add: i-is-x-axis vert-edge-def*)
have *vert-edge-is-straight-path*: $?vert\text{-edge} = (\lambda t. (x, y2 + t * (y1 - y2)))$
using *vert-edge-def Product-Type.snd-conv*
by (*auto simp add: mult.commute right-diff-distrib'*)
show *?thesis*
by (*simp add: i-is-x-axis line-integral-on-pair-straight-path(1) mult.commute straight-path-differentiable-x vert-edge-is-straight-path*)
qed
then show $(\sum x \in \text{two-chain-vertical-boundary two-chain}. \text{case } x \text{ of } (k, g) \Rightarrow k * \text{line-integral } F \{i\} g) = 0$
using *comm-monoid-add-class.sum.neutral* **by** (*simp add: prod.case-eq-if*)
qed

have *boundary-is-finite*: $\text{finite} (\text{two-chain-boundary two-chain})$
unfolding *two-chain-boundary-def*
by (*metis all-two-cubes-have-four-distict-edges card.infinite finite-UN-I finite-imageD*)

$\text{gen-division gen-division-def zero-neq-numeral valid-typeI-div valid-two-chain-def}$
have *boundary-is-vert-hor*: $(\text{two-chain-boundary two-chain}) =$
 $(\text{two-chain-vertical-boundary two-chain}) \cup$
 $(\text{two-chain-horizontal-boundary two-chain})$
by (*auto simp add: two-chain-boundary-def two-chain-vertical-boundary-def two-chain-horizontal-boundary-def boundary-def*)
then have *hor-vert-finite*:
 $\text{finite} (\text{two-chain-vertical-boundary two-chain})$
 $\text{finite} (\text{two-chain-horizontal-boundary two-chain})$
using *boundary-is-finite* **by** *auto*
have *horiz-verti-disjoint*:

$(two-chain-vertical-boundary\ two-chain) \cap (two-chain-horizontal-boundary\ two-chain)$
 $= \{\}$
proof (*simp add: two-chain-vertical-boundary-def two-chain-horizontal-boundary-def*
horizontal-boundary-def
vertical-boundary-def)
show $(\bigcup_{x \in two-chain} \{(-1, \lambda y. x(0, y)), (1::int, \lambda y. x(1::real, y))\}) \cap$
 $(\bigcup_{x \in two-chain} \{(1, \lambda z. x(z, 0)), (-1, \lambda z. x(z, 1))\}) = \{\}$
proof –
have $\{(-1, \lambda y. twoCube(0, y)), (1::int, \lambda y. twoCube(1, y))\} \cap$
 $\{(1, \lambda z. twoCube2(z, 0)), (-1, \lambda z. twoCube2(z, 1))\} = \{\}$
if $twoCube \in two-chain\ twoCube2 \in two-chain$ **for** $twoCube\ twoCube2$
proof (*cases twoCube = twoCube2*)
case *True*
have $card\ \{(-1::int, \lambda y. twoCube2(0::real, y)), (1::int, \lambda y. twoCube2(1,$
 $y)), (1, \lambda x. twoCube2(x, 0)), (-1, \lambda x. twoCube2(x, 1))\} = 4$
using *all-two-cubes-have-four-distict-edges that(2)*
by (*auto simp add: boundary-def vertical-boundary-def horizontal-boundary-def*)
then show *?thesis*
by (*auto simp: True card-insert-if split: if-split-asm*)
next
case *False*
then show *?thesis*
using *no-shared-edges-have-similar-orientations*
by (*simp add: that boundary-def vertical-boundary-def horizontal-boundary-def*)
qed
then have $\bigcup ((\lambda twoCube. \{(-1::int, \lambda y. twoCube(0, y)), (1, \lambda y. twoCube$
 $(1, y))\}) \text{ ‘ } two-chain)$
 $\cap \bigcup ((\lambda twoCube. \{(1, \lambda y. twoCube(y, 0)), (-1, \lambda z. twoCube(z,$
 $1))\}) \text{ ‘ } two-chain) = \{\}$
using *Complete-Lattices.Union-disjoint by force*
then show *?thesis by force*
qed
qed
have $one-chain-line-integral\ F\ \{i\}\ (two-chain-boundary\ two-chain)$
 $= one-chain-line-integral\ F\ \{i\}\ (two-chain-vertical-boundary\ two-chain) +$
 $one-chain-line-integral\ F\ \{i\}\ (two-chain-horizontal-boundary\ two-chain)$
using *boundary-is-vert-hor horiz-verti-disjoint*
by (*auto simp add: one-chain-line-integral-def hor-vert-finite sum.union-disjoint*)
then have *x-axis-line-integral-is-only-horizontal:*
 $one-chain-line-integral\ F\ \{i\}\ (two-chain-boundary\ two-chain)$
 $= one-chain-line-integral\ F\ \{i\}\ (two-chain-horizontal-boundary\ two-chain)$
using *vert-line-integral-zero by auto*

have $\exists \mathcal{V}. \mathcal{V} \subseteq (two-chain-vertical-boundary\ two-chain) \wedge \gamma = \mathcal{V} \cup (two-chain-horizontal-boundary$
 $two-chain)$
proof
let $? \mathcal{V} = \gamma - (two-chain-horizontal-boundary\ two-chain)$
show $? \mathcal{V} \subseteq two-chain-vertical-boundary\ two-chain \wedge \gamma = ? \mathcal{V} \cup two-chain-horizontal-boundary$
 $two-chain$

```

    using two-cubes-trace-horizontal-boundaries
      boundary-of-region-is-subset-of-partition-boundary boundary-is-vert-hor
    by blast
  qed
  then obtain  $\mathcal{V}$  where
    v-props:  $\mathcal{V} \subseteq (\text{two-chain-vertical-boundary two-chain}) \gamma = \mathcal{V} \cup (\text{two-chain-horizontal-boundary two-chain})$ 
  by auto
  have v-horiz-disj:  $\mathcal{V} \cap (\text{two-chain-horizontal-boundary two-chain}) = \{\}$ 
  using horiz-verti-disjoint v-props(1) by auto
  have v-finite: finite  $\mathcal{V}$ 
  using hor-vert-finite v-props(1) Finite-Set.rev-finite-subset by force
  have line-integral-on-path:  $\text{one-chain-line-integral } F \{i\} \gamma = \text{one-chain-line-integral } F \{i\} \mathcal{V} + \text{one-chain-line-integral } F \{i\} (\text{two-chain-horizontal-boundary two-chain})$ 
  by (auto simp add: one-chain-line-integral-def v-props sum.union-disjoint[OF v-finite hor-vert-finite(2) v-horiz-disj])

  have one-chain-line-integral  $F \{i\} \mathcal{V} = 0$ 
  proof (simp add: one-chain-line-integral-def)
    have line-integral  $F \{i\} (\text{snd oneCube}) = 0$ 
    if oneCube:  $\text{oneCube} \in \text{two-chain-vertical-boundary}(\text{two-chain})$  for oneCube
    proof -
      obtain  $x \ y1 \ y2 \ k$  where vert-edge-def:  $\text{oneCube} = (k, (\lambda t::\text{real}. (x::\text{real}, (1 - t) * (y2) + t * y1)))$ 
      using valid-typeI-div oneCube
      by (auto simp add: typeI-twoCube-def two-chain-vertical-boundary-def vertical-boundary-def)
      let ?vert-edge = (snd oneCube)
      have vert-edge-x-const:  $\forall t. (?vert-edge \ t) \cdot i = x$ 
      by (simp add: i-is-x-axis vert-edge-def)
      have vert-edge-is-straight-path:
        ?vert-edge =  $(\lambda t. (x, y2 + t * (y1 - y2)))$ 
      by (auto simp: vert-edge-def algebra-simps)
      have  $\forall x. ?vert-edge$  differentiable at  $x$ 
      by (metis mult.commute vert-edge-is-straight-path straight-path-differentiable-x)
      then show line-integral  $F \{i\} (\text{snd oneCube}) = 0$ 
      using line-integral-on-pair-straight-path(1) vert-edge-x-const by blast
    qed
  qed
  then have  $\forall \text{oneCube} \in \mathcal{V}. \text{line-integral } F \{i\} (\text{snd oneCube}) = 0$ 
  using v-props by auto
  then show  $(\sum x \in \mathcal{V}. \text{case } x \text{ of } (k, g) \Rightarrow k * \text{line-integral } F \{i\} g) = 0$ 
  using comm-monoid-add-class.sum.neutral by (simp add: prod.case-eq-if)
  qed
  then have one-chain-line-integral  $F \{i\} \gamma = \text{one-chain-line-integral } F \{i\} (\text{two-chain-boundary two-chain})$ 
  using x-axis-line-integral-is-only-horizontal by (simp add: line-integral-on-path)
  then show ?thesis
  using assms and GreenThm-typeI-divisible by auto

```

qed

lemma *GreenThm-typeI-divisible-region-boundary-gen:*

assumes *valid-typeI-div: valid-typeI-division s two-chain and*

f-analytically-valid: \forall twoC \in 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 –

let *?F-b' = partial-vector-derivative ($\lambda a. F(a) \cdot i$)*

have *all-two-cubes-have-four-distict-edges: \forall twoCube \in two-chain. card (boundary twoCube) = 4*

using *valid-typeI-div valid-two-chain-def valid-two-cube-def*

by *auto*

have *no-shared-edges-have-similar-orientations:*

\forall twoCube1 \in two-chain. \forall twoCube2 \in two-chain.

twoCube1 \neq twoCube2 \longrightarrow boundary twoCube1 \cap boundary twoCube2 = {}

using *valid-typeI-div valid-two-chain-def by (auto simp add: pairwise-def)*

have *vert-line-integral-zero:*

one-chain-line-integral F {i} (two-chain-vertical-boundary two-chain) = 0

proof *(simp add: one-chain-line-integral-def)*

have *line-integral F {i} (snd oneCube) = 0*

if *oneCube: oneCube \in two-chain-vertical-boundary(two-chain) for oneCube*

proof –

obtain *x y1 y2 k where vert-edge-def: oneCube = (k, ($\lambda t::real. (x::real, (1 - t) * (y2) + t * y1)$))*

using *valid-typeI-div oneCube*

by *(auto simp add: typeI-twoCube-def two-chain-vertical-boundary-def vertical-boundary-def)*

let *?vert-edge = (snd oneCube)*

have *vert-edge-x-const: $\forall t. (?vert-edge t) \cdot i = x$*

by *(simp add: i-is-x-axis vert-edge-def)*

have *vert-edge-is-straight-path: ?vert-edge = ($\lambda t. (x, y2 + t * (y1 - y2)$))*

by *(auto simp: vert-edge-def algebra-simps)*

show *?thesis*

by *(simp add: i-is-x-axis line-integral-on-pair-straight-path(1) mult.commute straight-path-differentiable-x vert-edge-is-straight-path)*

qed

then show *($\sum x \in$ two-chain-vertical-boundary two-chain. case x of (k, g) \Rightarrow k * line-integral F {i} g) = 0*

using *comm-monoid-add-class.sum.neutral by (simp add: prod.case-eq-if)*

qed

have *boundary-is-finite: finite (two-chain-boundary two-chain)*

unfolding *two-chain-boundary-def*

proof *(rule finite-UN-I)*

```

show finite two-chain
  using assms(1) finite-imageD gen-division-def valid-two-chain-def by auto
show  $\bigwedge a. a \in \text{two-chain} \implies \text{finite } (\text{boundary } a)$ 
  by (simp add: boundary-def horizontal-boundary-def vertical-boundary-def)
qed
have boundary-is-vert-hor: two-chain-boundary two-chain =
  (two-chain-vertical-boundary two-chain)  $\cup$  (two-chain-horizontal-boundary
two-chain)
  by (auto simp add: two-chain-boundary-def two-chain-vertical-boundary-def
two-chain-horizontal-boundary-def boundary-def)
  then have hor-vert-finite:
    finite (two-chain-vertical-boundary two-chain)
    finite (two-chain-horizontal-boundary two-chain)
  using boundary-is-finite by auto
  have horiz-verti-disjoint:
    (two-chain-vertical-boundary two-chain)  $\cap$  (two-chain-horizontal-boundary two-chain)
= {}
  proof (simp add: two-chain-vertical-boundary-def two-chain-horizontal-boundary-def
horizontal-boundary-def
vertical-boundary-def)
  show  $(\bigcup x \in \text{two-chain}. \{(-1, \lambda y. x(0, y)), (1::\text{int}, \lambda y. x(1::\text{real}, y))\})$ 
 $\cap$   $(\bigcup x \in \text{two-chain}. \{(1, \lambda y. x(y, 0)), (-1, \lambda y. x(y, 1))\}) = \{\}$ 
  proof -
  have  $\{(-1, \lambda y. \text{twoCube}(0, y)), (1::\text{int}, \lambda y. \text{twoCube}(1, y))\} \cap$ 
 $\{(1, \lambda y. \text{twoCube2}(y, 0)), (-1, \lambda y. \text{twoCube2}(y, 1))\} = \{\}$ 
  if twoCube  $\in$  two-chain twoCube2  $\in$  two-chain for twoCube twoCube2
  proof (cases twoCube = twoCube2)
  case True
  have card  $\{(-1::\text{int}, \lambda y. \text{twoCube2}(0, y)), (1::\text{int}, \lambda y. \text{twoCube2}(1, y)),$ 
 $(1, \lambda x. \text{twoCube2}(x, 0)), (-1, \lambda x. \text{twoCube2}(x, 1))\} = 4$ 
  using all-two-cubes-have-four-distict-edges that(2)
  by (auto simp add: boundary-def vertical-boundary-def horizontal-boundary-def)
  then show ?thesis
  by (auto simp: card-insert-if True split: if-split-asm)
  next
  case False
  then show ?thesis
  using no-shared-edges-have-similar-orientations
  by (simp add: that boundary-def vertical-boundary-def horizontal-boundary-def)
qed
  then have  $\bigcup ((\lambda \text{twoCube}. \{(-1, \lambda y. \text{twoCube}(0, y)), (1, \lambda y. \text{twoCube}(1,$ 
 $y))\}) \text{ 'two-chain})$ 
 $\cap \bigcup ((\lambda \text{twoCube}. \{(1::\text{int}, \lambda y. \text{twoCube}(y, 0)), (-1, \lambda y. \text{twoCube}$ 
 $(y, 1))\}) \text{ 'two-chain})$ 
= {}
  using Complete-Lattices.Union-disjoint by force
  then show ?thesis by force
qed
qed

```



```

have one-chain-line-integral  $F \{i\}$  (two-chain-boundary two-chain)
  = one-chain-line-integral  $F \{i\}$  (two-chain-vertical-boundary two-chain) +
    one-chain-line-integral  $F \{i\}$  (two-chain-horizontal-boundary two-chain)
using boundary-is-vert-hor horiz-verti-disjoint
by (auto simp add: one-chain-line-integral-def hor-vert-finite sum.union-disjoint)
then have x-axis-line-integral-is-only-horizontal:
  one-chain-line-integral  $F \{i\}$  (two-chain-boundary two-chain)
  = one-chain-line-integral  $F \{i\}$  (two-chain-horizontal-boundary two-chain)
using vert-line-integral-zero by auto

obtain  $\mathcal{H} \mathcal{V}$  where hv-props: finite  $\mathcal{V}$ 
  ( $\forall (k, \gamma) \in \mathcal{V}. (\exists (k', \gamma') \in \text{two-chain-vertical-boundary two-chain}.$ 
    ( $\exists a \in \{0 .. 1\}, \exists b \in \{0..1\}. a \leq b \wedge \text{subpath } a \ b \ \gamma' = \gamma))$ )  $\gamma = \mathcal{H} \cup \mathcal{V}$ 
  (common-sudiv-exists (two-chain-horizontal-boundary two-chain)  $\mathcal{H}$ )
   $\vee$  common-reparam-exists  $\mathcal{H}$  (two-chain-horizontal-boundary two-chain)
  finite  $\mathcal{H}$ 
  boundary-chain  $\mathcal{H}$ 
   $\forall (k, \gamma) \in \mathcal{H}. \text{valid-path } \gamma$ 
using only-vertical-division by (auto simp add: only-vertical-division-def)
have finite  $\{i\}$  by auto
then have eq-integrals: one-chain-line-integral  $F \{i\}$   $\mathcal{H} =$  one-chain-line-integral
 $F \{i\}$  (two-chain-horizontal-boundary two-chain)
proof(cases common-sudiv-exists (two-chain-horizontal-boundary two-chain)  $\mathcal{H}$ )
  case True
    then show ?thesis
      using gen-common-subdivision-imp-eq-line-integral(1)[OF True two-chain-horizontal-boundary-is-boundary
hv-props(6) - hor-vert-finite(2) hv-props(5)]
        typeI-cube-line-integral-exists-boundary''
      by force
    next
      case False
        have integ-exis-horiz:
           $\forall (k, \gamma) \in \text{two-chain-horizontal-boundary two-chain}. \text{line-integral-exists } F \{i\}$ 
 $\gamma$ 
          using typeI-cube-line-integral-exists-boundary'' assms
          by (fastforce simp add: valid-two-chain-def)
        have integ-exis:  $\forall (k, \gamma) \in \text{two-chain-boundary two-chain}. \text{line-integral-exists } F$ 
 $\{i\} \ \gamma$ 
          using typeI-cube-line-integral-exists-boundary by blast
        have valid-paths:  $\forall (k, \gamma) \in \text{two-chain-vertical-boundary two-chain}. \text{valid-path } \gamma$ 
          using typeI-edges-are-valid-paths assms
          by (fastforce simp add: two-chain-boundary-def two-chain-vertical-boundary-def
boundary-def)
        have integ-exis-vert:
          ( $\bigwedge k \ \gamma. (\exists (k', \gamma') \in \text{two-chain-vertical-boundary two-chain}. \exists a \in \{0..1\}.$ 
 $\exists b \in \{0..1\}. a \leq b \wedge \text{subpath } a \ b \ \gamma' = \gamma) \implies$ 
            line-integral-exists  $F \{i\} \ \gamma$ )
          using integ-exis valid-paths line-integral-exists-subpath[of  $F \{i\}$ ]
          by (fastforce simp add: two-chain-boundary-def two-chain-vertical-boundary-def

```

```

    two-chain-horizontal-boundary-def boundary-def)
  have finite-i: finite {i} by auto
  have i: {i} ⊆ Basis using i-is-x-axis real-pair-basis by auto
  have ii: ∀ (k2, γ2) ∈ two-chain-horizontal-boundary two-chain. ∀ b ∈ {i}. continuous-on
(path-image γ2) (λx. F x · b)
    using assms(1) field-cont-on-typeI-region-cont-on-edges assms(2)
  by (fastforce simp add: analytically-valid-def two-chain-horizontal-boundary-def
boundary-def path-image-def)
  have *: common-reparam-exists ℋ (two-chain-horizontal-boundary two-chain)
    using hv-props(4) False by auto
  show one-chain-line-integral F {i} ℋ = one-chain-line-integral F {i} (two-chain-horizontal-boundary
two-chain)
    using common-reparam-exists-imp-eq-line-integral(1)[OF finite-i hv-props(5)
hor-vert-finite(2) hv-props(6) two-chain-horizontal-boundary-is-boundary-chain ii
* hv-props(7) type-I-chain-horiz-bound-valid]
    by fastforce
  qed

have line-integral-on-path:
  one-chain-line-integral F {i} γ =
  one-chain-line-integral F {i} (two-chain-horizontal-boundary two-chain)
proof (auto simp add: one-chain-line-integral-def)
  have line-integral F {i} (snd oneCube) = 0 if oneCube: oneCube ∈ V for
oneCube
  proof -
    obtain k γ where k-gamma: (k,γ) = oneCube
      by (metis coeff-cube-to-path.cases)
    then obtain k' γ' a b where kp-gammap:
      (k'::int, γ') ∈ two-chain-vertical-boundary two-chain
      a ∈ {0 .. 1}
      b ∈ {0..1}
      subpath a b γ' = γ
      using hv-props oneCube
      by (smt case-prodE split-conv)
    obtain x y1 y2 where vert-edge-def: (k',γ') = (k', (λt::real. (x::real, (1 -
t) * (y2) + t * y1)))
      using valid-typeI-div kp-gammap
      by (auto simp add: typeI-twoCube-def two-chain-vertical-boundary-def
vertical-boundary-def)
    have vert-edge-x-const: ∀ t. γ (t) · i = x
      by (metis (no-types, lifting) Pair-inject fstI i-is-x-axis inner-Pair-0(2)
kp-gammap(4) real-inner-1-right subpath-def vert-edge-def)
    have γ = (λt::real. (x::real, (1 - (b - a)*t - a) * (y2) + ((b-a)*t + a) *
y1))
      using vert-edge-def Product-Type.snd-conv Product-Type.fst-conv kp-gammap(4)
      by (simp add: subpath-def diff-diff-eq[symmetric])
    also have ... = (λt::real. (x::real, (1*y2 - a*y2) + a*y1 + ((b-a)*y1 -
(b - a)*y2)*t))
      by (simp add: algebra-simps)
  end
end

```

finally have *vert-edge-is-straight-path*:
 $\gamma = (\lambda t::\text{real}. (x::\text{real}, (1*y2 - a*y2) + a*y1 + ((b-a)*y1 - (b - a)*y2)*t))$.
show *line-integral F {i} (snd oneCube) = 0*
proof –
have $\forall x. \gamma$ *differentiable at x*
by (*simp add: straight-path-differentiable-x vert-edge-is-straight-path*)
then have *line-integral F {i} $\gamma = 0$*
using *line-integral-on-pair-straight-path(1) vert-edge-x-const* **by** *blast*
then show *?thesis*
using *Product-Type.snd-conv k-gamma* **by** *auto*
qed
qed
then have $\forall x \in \mathcal{V}. (\text{case } x \text{ of } (k, g) \Rightarrow (k::\text{int}) * \text{line-integral } F \{i\} g) = 0$
by *auto*
then show $(\sum x \in \gamma. \text{case } x \text{ of } (k, g) \Rightarrow \text{real-of-int } k * \text{line-integral } F \{i\} g) =$
 $(\sum x \in \text{two-chain-horizontal-boundary two-chain. case } x \text{ of } (k, g) \Rightarrow$
 $\text{of-int } k * \text{line-integral } F \{i\} g)$
using *hv-props(1) hv-props(3) hv-props(5) sum-zero-set hor-vert-finite(2)*
eq-integrals
apply (*auto simp add: one-chain-line-integral-def*)
by (*smt Un-commute sum-zero-set*)
qed
then have *one-chain-line-integral F {i} $\gamma =$*
one-chain-line-integral F {i} (two-chain-boundary two-chain)
using *x-axis-line-integral-is-only-horizontal line-integral-on-path* **by** *auto*
then show *?thesis*
using *assms GreenThm-typeI-divisible* **by** *auto*
qed
end

locale *green-typeI-typeII-chain* = *R2: R2 i j + T1: green-typeI-chain i j F two-chain-typeI*
 $+ T2: \text{green-typeII-chain } i j F \text{ two-chain-typeII}$ **for** $i j F \text{ two-chain-typeI two-chain-typeII}$
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 $\subseteq \gamma$ and

typeII-two-cubes-trace-vertical-boundaries:

two-chain-vertical-boundary two-chain-typeII $\subseteq \gamma$ and

boundary-of-region-is-subset-of-partition-boundaries:

$\gamma \subseteq \text{two-chain-boundary two-chain-typeI}$

$\gamma \subseteq \text{two-chain-boundary two-chain-typeII}$

shows *integral s ($\lambda x. \text{partial-vector-derivative } (\lambda a. F a \cdot j) i x - \text{partial-vector-derivative}$*
 $(\lambda a. F a \cdot i) j x)$

$= \text{one-chain-line-integral } F \{i, j\} \gamma$
proof –
let $?F\text{-}b' = \text{partial-vector-derivative } (\lambda a. F a \cdot i) j$
let $?F\text{-}a' = \text{partial-vector-derivative } (\lambda a. F a \cdot j) i$
have $\text{typeI-regions-integral: integral } s (\lambda x. - \text{partial-vector-derivative } (\lambda a. F a \cdot i) j x) = \text{one-chain-line-integral } F \{i\} \gamma$
using $T1.\text{GreenThm-typeI-divisible-region-boundary}$
 $\text{gen-divisions}(1) \text{ typeI-two-cubes-trace-horizontal-boundaries}$
 $\text{boundary-of-region-is-subset-of-partition-boundaries}(1)$
by blast
have $\text{typeII-regions-integral: integral } s (\text{partial-vector-derivative } (\lambda x. F x \cdot j) i) = \text{one-chain-line-integral } F \{j\} \gamma$
 $= \text{one-chain-line-integral } F \{j\} \gamma$
using $T2.\text{GreenThm-typeII-divisible-region-boundary}$ $\text{gen-divisions}(2)$
 $\text{typeII-two-cubes-trace-vertical-boundaries}$
 $\text{boundary-of-region-is-subset-of-partition-boundaries}(2)$
by auto
have $\text{integral-dis: integral } s (\lambda x. ?F\text{-}a' x - ?F\text{-}b' x) = \text{integral } s (\lambda x. ?F\text{-}a' x) + \text{integral } s (\lambda x. - ?F\text{-}b' x)$
proof –
have $\forall \text{twoCube} \in \text{two-chain-typeII}. (?F\text{-}a' \text{ has-integral integral } (\text{cubeImage } \text{twoCube}) ?F\text{-}a') (\text{cubeImage } \text{twoCube})$
by $(\text{simp add: analytically-valid-imp-part-deriv-integrable-on } T2.F\text{-anal-valid has-integral-iff})$
then have $\bigwedge u. u \in (\text{cubeImage } \text{'two-chain-typeII}) \implies (?F\text{-}a' \text{ has-integral integral } u ?F\text{-}a') u$
by auto
then have $(?F\text{-}a' \text{ has-integral } (\sum \text{img} \in \text{cubeImage } \text{'two-chain-typeII}. \text{integral } \text{img } ?F\text{-}a')) s$
using $\text{gen-divisions}(2) \text{ unfolding } \text{gen-division-def}$
by $(\text{metis has-integral-Union})$
then have $F\text{-}a'\text{-integrable: } (?F\text{-}a' \text{ integrable-on } s) \text{ by } \text{auto}$
have $\forall \text{twoCube} \in \text{two-chain-typeI}. (?F\text{-}b' \text{ has-integral integral } (\text{cubeImage } \text{twoCube}) ?F\text{-}b') (\text{cubeImage } \text{twoCube})$
using $\text{analytically-valid-imp-part-deriv-integrable-on } T1.F\text{-anal-valid}$ **by** blast
then have $\bigwedge u. u \in (\text{cubeImage } \text{'two-chain-typeI}) \implies (?F\text{-}b' \text{ has-integral integral } u ?F\text{-}b') u$
by auto
then have $(?F\text{-}b' \text{ has-integral } (\sum \text{img} \in \text{cubeImage } \text{'two-chain-typeI}. \text{integral } \text{img } ?F\text{-}b')) s$
using $\text{gen-divisions}(1) \text{ unfolding } \text{gen-division-def}$
by $(\text{metis has-integral-Union})$
then show $?thesis$
by $(\text{simp add: } F\text{-}a'\text{-integrable } \text{Henstock-Kurzweil-Integration.integral-diff has-integral-iff})$
qed
have $\text{line-integral-dist: one-chain-line-integral } F \{i, j\} \gamma = \text{one-chain-line-integral } F \{i\} \gamma + \text{one-chain-line-integral } F \{j\} \gamma$
proof $(\text{simp add: one-chain-line-integral-def})$
have $k * \text{line-integral } F \{i, j\} g = k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g$

if $kg: (k,g) \in \gamma$ **for** $k g$
proof –
obtain *twoCube-typeI* **where** *twoCube-typeI-props*:
twoCube-typeI \in *two-chain-typeI*
 $(k, g) \in$ *boundary twoCube-typeI*
typeI-twoCube twoCube-typeI
continuous-on (cubeImage twoCube-typeI) ($\lambda x. F(x) \cdot i$)
using *boundary-of-region-is-subset-of-partition-boundaries(1) two-chain-boundary-def*
T1.valid-typeI-div
T1.F-anal-valid kg
by (*auto simp add: analytically-valid-def*)
obtain *twoCube-typeII* **where** *twoCube-typeII-props*:
twoCube-typeII \in *two-chain-typeII*
 $(k, g) \in$ *boundary twoCube-typeII*
typeII-twoCube twoCube-typeII
continuous-on (cubeImage twoCube-typeII) ($\lambda x. F(x) \cdot j$)
using *boundary-of-region-is-subset-of-partition-boundaries(2) two-chain-boundary-def*
T2.valid-typeII-div
kg T2.F-anal-valid
by (*auto simp add: analytically-valid-def*)
have *line-integral F {i, j} g = line-integral F {i} g + line-integral F {j} g*
proof –
have *int-exists-i: line-integral-exists F {i} g*
using *T1.typeI-cube-line-integral-exists-boundary assms kg*
by (*auto simp add: valid-two-chain-def*)
have *int-exists-j: line-integral-exists F {j} g*
using *T2.typeII-cube-line-integral-exists-boundary assms kg*
by (*auto simp add: valid-two-chain-def*)
have *finite: finite {i, j} by auto*
show *?thesis*
using *line-integral-sum-gen[OF finite int-exists-i int-exists-j] R2.i-is-x-axis*
R2.j-is-y-axis
by *auto*
qed
then show $k * \text{line-integral } F \{i, j\} g = k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g$
line-integral F {j} g
by (*simp add: distrib-left*)
qed
then have *line-integral-distrib*:
 $(\sum (k,g) \in \gamma. k * \text{line-integral } F \{i, j\} g) =$
 $(\sum (k,g) \in \gamma. k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g)$
by (*force intro: sum.cong split-cong*)
have $(\lambda x. (\text{case } x \text{ of } (k, g) \Rightarrow (k::\text{int}) * \text{line-integral } F \{i\} g) + (\text{case } x \text{ of } (k, g) \Rightarrow (k::\text{int}) * \text{line-integral } F \{j\} g)) =$
 $(\lambda x. (\text{case } x \text{ of } (k, g) \Rightarrow (k * \text{line-integral } F \{i\} g)$
 $+ (k::\text{int}) * \text{line-integral } F \{j\} g))$
using *comm-monoid-add-class.sum.distrib by auto*
then show $(\sum (k, g) \in \gamma. k * \text{line-integral } F \{i, j\} g) =$
 $(\sum (k, g) \in \gamma. (k::\text{int}) * \text{line-integral } F \{i\} g) + (\sum (k, g) \in \gamma. (k::\text{int}) *$

line-integral $F \{j\} g$
using *comm-monoid-add-class.sum.distrib*[of $(\lambda(k, g). k * \text{line-integral } F \{i\} g) (\lambda(k, g). k * \text{line-integral } F \{j\} g) \gamma$]
line-integral-distrib
by *presburger*
qed
show *?thesis*
using *integral-dis line-integral-dist typeI-regions-integral typeII-regions-integral*
by *auto*
qed

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-typeII-one-chains-have-gen-common-subdiv:
common-sudiv-exists one-chain-typeI one-chain-typeII
shows *integral s* $(\lambda x. \text{partial-vector-derivative } (\lambda x. (F x) \cdot j) i x - \text{partial-vector-derivative } (\lambda x. (F x) \cdot i) j x) = \text{one-chain-line-integral } F \{i, j\} \text{ one-chain-typeI}$
integral s $(\lambda x. \text{partial-vector-derivative } (\lambda x. (F x) \cdot j) i x - \text{partial-vector-derivative } (\lambda x. (F x) \cdot i) j x) = \text{one-chain-line-integral } F \{i, j\} \text{ one-chain-typeII}$
proof –
let *?F-b'* = *partial-vector-derivative* $(\lambda x. (F x) \cdot i) j$
let *?F-a'* = *partial-vector-derivative* $(\lambda x. (F x) \cdot j) i$
have *one-chain-i-integrals:*
one-chain-line-integral $F \{i\} \text{ one-chain-typeI} = \text{one-chain-line-integral } F \{i\} \text{ one-chain-typeII} \wedge$
 $(\forall (k, \gamma) \in \text{one-chain-typeI}. \text{line-integral-exists } F \{i\} \gamma) \wedge$
 $(\forall (k, \gamma) \in \text{one-chain-typeII}. \text{line-integral-exists } F \{i\} \gamma)$
proof (*intro conjI*)
have *finite two-chain-typeI*
using *T1.valid-typeI-div finite-image-iff*
by (*auto simp add: gen-division-def valid-two-chain-def*)
then show *ii:* $\forall (k, \gamma) \in \text{one-chain-typeI}. \text{line-integral-exists } F \{i\} \gamma$
using *T1.members-of-only-vertical-div-line-integrable' assms*
by *fastforce*
have *finite (two-chain-horizontal-boundary two-chain-typeI)*
by (*meson T1.valid-typeI-div finite-imageD finite-two-chain-horizontal-boundary gen-division-def valid-two-chain-def*)
then have *finite one-chain-typeI*
using *only-vertical-division(1) only-vertical-division-def by auto*
moreover have *finite one-chain-typeII*
using *only-horizontal-division(1) only-horizontal-division-def by auto*
ultimately show *one-chain-line-integral* $F \{i\} \text{ one-chain-typeI} = \text{one-chain-line-integral } F \{i\} \text{ one-chain-typeII}$

and $\forall (k, \gamma) \in \text{one-chain-typeII}. \text{line-integral-exists } F \{i\} \gamma$
using *gen-common-subdivision-imp-eq-line-integral*[*OF typeI-and-typII-one-chains-have-gen-common-subdiv*
only-vertical-division(2) only-horizontal-division(2)] *ii*
by auto
qed
have *one-chain-j-integrals*:
one-chain-line-integral $F \{j\}$ *one-chain-typeII* = *one-chain-line-integral* $F \{j\}$
one-chain-typeI \wedge
 $(\forall (k, \gamma) \in \text{one-chain-typeII}. \text{line-integral-exists } F \{j\} \gamma) \wedge$
 $(\forall (k, \gamma) \in \text{one-chain-typeI}. \text{line-integral-exists } F \{j\} \gamma)$
proof (*intro conjI*)
have *finite two-chain-typeII*
using *T2.valid-typeII-div finite-image-iff*
by (*auto simp add: gen-division-def valid-two-chain-def*)
then show *ii*: $\forall (k, \gamma) \in \text{one-chain-typeII}. \text{line-integral-exists } F \{j\} \gamma$
using *T2.members-of-only-horiz-div-line-integrable' assms T2.two-chain-valid-valid-cubes*
by blast
have *typeII-and-typI-one-chains-have-common-subdiv*: *common-sudiv-exists one-chain-typeII*
one-chain-typeI
by (*simp add: common-sudiv-exists-comm typeI-and-typII-one-chains-have-gen-common-subdiv*)
have *iv*: *finite one-chain-typeI*
using *only-vertical-division(1) only-vertical-division-def* **by auto**
moreover have *iv'*: *finite one-chain-typeII*
using *only-horizontal-division(1) only-horizontal-division-def* **by auto**
ultimately show *one-chain-line-integral* $F \{j\}$ *one-chain-typeII* =
one-chain-line-integral $F \{j\}$ *one-chain-typeI*
 $\forall (k, \gamma) \in \text{one-chain-typeI}. \text{line-integral-exists } F \{j\} \gamma$
using *gen-common-subdivision-imp-eq-line-integral*[*OF typeII-and-typI-one-chains-have-common-subdiv*
only-horizontal-division(2) only-vertical-division(2)] *ii* **ii**
by auto
qed
have *typeI-regions-integral*:
integral $s (\lambda x. - ?F-b' x) = \text{one-chain-line-integral } F \{i\}$ *one-chain-typeI*
using *T1.GreenThm-typeI-divisible-region-boundary-gen T1.valid-typeI-div*
T1.F-anal-valid only-vertical-division(1)
by auto
have *typeII-regions-integral*:
integral $s ?F-a' = \text{one-chain-line-integral } F \{j\}$ *one-chain-typeII*
using *T2.GreenThm-typeII-divisible-region-boundary-gen T2.valid-typeII-div*
T2.F-anal-valid only-horizontal-division(1)
by auto
have *line-integral-dist*:
one-chain-line-integral $F \{i, j\}$ *one-chain-typeI* = *one-chain-line-integral* $F \{i\}$
one-chain-typeI + *one-chain-line-integral* $F \{j\}$ *one-chain-typeI* \wedge
one-chain-line-integral $F \{i, j\}$ *one-chain-typeII* = *one-chain-line-integral* F
 $\{i\}$ *one-chain-typeII* + *one-chain-line-integral* $F \{j\}$ *one-chain-typeII*
proof (*simp add: one-chain-line-integral-def*)
have *line-integral-distrib*:
 $(\sum (k, g) \in \text{one-chain-typeI}. k * \text{line-integral } F \{i, j\} g) =$

$(\sum (k,g) \in \text{one-chain-typeI}. k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g) \wedge$
 $(\sum (k,g) \in \text{one-chain-typeII}. k * \text{line-integral } F \{i, j\} g) =$
 $(\sum (k,g) \in \text{one-chain-typeII}. k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g)$

proof –

have $0: k * \text{line-integral } F \{i, j\} g = k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g$

if $(k,g) \in \text{one-chain-typeII}$ **for** $k g$

proof –

have $\text{line-integral-exists } F \{i\} g \text{ line-integral-exists } F \{j\} g \text{ finite } \{i, j\}$

using $\text{one-chain-i-integrals one-chain-j-integrals that by fastforce+}$

moreover have $\{i\} \cap \{j\} = \{\}$

by $(\text{simp add: } R2.i\text{-is-x-axis } R2.j\text{-is-y-axis})$

ultimately have $\text{line-integral } F \{i, j\} g = \text{line-integral } F \{i\} g + \text{line-integral } F \{j\} g$

by $(\text{metis insert-is-Un line-integral-sum-gen}(1))$

then show $k * \text{line-integral } F \{i, j\} g = k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g$

by $(\text{simp add: distrib-left})$

qed

have $k * \text{line-integral } F \{i, j\} g = k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g$

if $(k,g) \in \text{one-chain-typeI}$ **for** $k g$

proof –

have $\text{line-integral } F \{i, j\} g = \text{line-integral } F \{i\} g + \text{line-integral } F \{j\} g$

by $(\text{smt that disjoint-insert}(2) \text{ finite.emptyI finite.insertI } R2.i\text{-is-x-axis inf-bot-right insert-absorb insert-commute insert-is-Un } R2.j\text{-is-y-axis line-integral-sum-gen}(1) \text{ one-chain-i-integrals one-chain-j-integrals prod.case-eq-if singleton-inject snd-conv})$

then show $k * \text{line-integral } F \{i, j\} g = k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g$

by $(\text{simp add: distrib-left})$

qed

then show $?thesis$

using 0 **by** $(\text{smt sum.cong split-cong})$

qed

show $(\sum (k::\text{int}, g) \in \text{one-chain-typeI}. k * \text{line-integral } F \{i, j\} g) =$
 $(\sum (k, g) \in \text{one-chain-typeI}. k * \text{line-integral } F \{i\} g) + (\sum (k::\text{int}, g) \in \text{one-chain-typeI}. k * \text{line-integral } F \{j\} g) \wedge$
 $(\sum (k::\text{int}, g) \in \text{one-chain-typeII}. k * \text{line-integral } F \{i, j\} g) =$
 $(\sum (k, g) \in \text{one-chain-typeII}. k * \text{line-integral } F \{i\} g) + (\sum (k::\text{int}, g) \in \text{one-chain-typeII}. k * \text{line-integral } F \{j\} g)$

proof –

have $0: (\lambda x. (\text{case } x \text{ of } (k::\text{int}, g) \Rightarrow k * \text{line-integral } F \{i\} g) + (\text{case } x \text{ of } (k::\text{int}, g) \Rightarrow k * \text{line-integral } F \{j\} g)) =$
 $(\lambda x. (\text{case } x \text{ of } (k::\text{int}, g) \Rightarrow (k * \text{line-integral } F \{i\} g) + k * \text{line-integral } F \{j\} g))$

using $\text{comm-monoid-add-class.sum.distrib}$ **by** auto

then have $1: (\sum x \in \text{one-chain-typeI}. (\text{case } x \text{ of } (k::\text{int}, g) \Rightarrow k * \text{line-integral } F \{i\} g) + (\text{case } x \text{ of } (k::\text{int}, g) \Rightarrow k * \text{line-integral } F \{j\} g)) =$
 $(\sum x \in \text{one-chain-typeI}. (k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g))$

$F \{i\} g) + (\text{case } x \text{ of } (k::\text{int}, g) \Rightarrow k * \text{line-integral } F \{j\} g)) =$
 $(\sum_{x \in \text{one-chain-typeI}}. (\text{case } x \text{ of } (k::\text{int}, g) \Rightarrow (k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g)))$
by *presburger*
have $(\sum_{x \in \text{one-chain-typeII}}. (\text{case } x \text{ of } (k, g) \Rightarrow k * \text{line-integral } F \{i\} g) + (\text{case } x \text{ of } (k, g) \Rightarrow k * \text{line-integral } F \{j\} g)) =$
 $(\sum_{x \in \text{one-chain-typeII}}. (\text{case } x \text{ of } (k, g) \Rightarrow (k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g)))$
using *0* **by** *presburger*
then show *?thesis*
using *sum.distrib[of (\lambda(k, g). k * line-integral F {i} g) (\lambda(k, g). k * line-integral F {j} g) one-chain-typeI]*
*sum.distrib[of (\lambda(k, g). k * line-integral F {i} g) (\lambda(k, g). k * line-integral F {j} g) one-chain-typeII]*
line-integral-distrib 1
by *auto*
qed
qed
have *integral-dis: integral s (\lambda x. ?F-a' x - ?F-b' x) = integral s (\lambda x. ?F-a' x) + integral s (\lambda x. - ?F-b' x)*
proof –
have *(?F-a' has-integral integral (cubeImage twoCube) ?F-a') (cubeImage twoCube)*
if *twoCube \in two-chain-typeII* **for** *twoCube*
by *(simp add: analytically-valid-imp-part-deriv-integrable-on T2.F-anal-valid has-integral-integrable-integral that)*
then have $\bigwedge u. u \in (\text{cubeImage } \text{'two-chain-typeII}) \Longrightarrow (\text{?F-a' has-integral integral } u \text{ ?F-a'}) u$
by *auto*
then have *(?F-a' has-integral (\sum img \in cubeImage 'two-chain-typeII. integral img ?F-a')) s*
using *T2.valid-typeII-div unfolding gen-division-def*
by *(metis has-integral-Union)*
then have *F-a'-integrable:*
(?F-a' integrable-on s) **by** *auto*
have $\forall \text{twoCube} \in \text{two-chain-typeI}. (\text{?F-b' has-integral integral (cubeImage twoCube) ?F-b'}) (\text{cubeImage twoCube})$
using *analytically-valid-imp-part-deriv-integrable-on T1.F-anal-valid* **by** *blast*
then have $\bigwedge u. u \in (\text{cubeImage } \text{'two-chain-typeI}) \Longrightarrow (\text{?F-b' has-integral integral } u \text{ ?F-b'}) u$
by *auto*
then have *(?F-b' has-integral (\sum img \in cubeImage 'two-chain-typeI. integral img ?F-b')) s*
using *T1.valid-typeI-div unfolding gen-division-def*
by *(metis has-integral-Union)*
then show *?thesis*
by *(simp add: F-a'-integrable Henstock-Kurzweil-Integration.integral-diff has-integral-iff)*
qed
show *integral s (\lambda x. ?F-a' x - ?F-b' x) = one-chain-line-integral F {i, j}*

one-chain-typeI
using *one-chain-j-integrals integral-dis line-integral-dist typeI-regions-integral typeII-regions-integral*
by auto
show $\text{integral } s (\lambda x. ?F\text{-}a' x - ?F\text{-}b' x) = \text{one-chain-line-integral } F \{i, j\}$
one-chain-typeII
using *one-chain-i-integrals integral-dis line-integral-dist typeI-regions-integral typeII-regions-integral*
by auto
qed

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-typeII-one-chains-have-common-subdiv:
common-boundary-sudivision-exists one-chain-typeI one-chain-typeII
shows $\text{integral } s (\lambda x. \text{partial-vector-derivative } (\lambda x. (F x) \cdot j) i x - \text{partial-vector-derivative } (\lambda x. (F x) \cdot i) j x) = \text{one-chain-line-integral } F \{i, j\} \text{ one-chain-typeI}$
 $\text{integral } s (\lambda x. \text{partial-vector-derivative } (\lambda x. (F x) \cdot j) i x - \text{partial-vector-derivative } (\lambda x. (F x) \cdot i) j x) = \text{one-chain-line-integral } F \{i, j\} \text{ one-chain-typeII}$
using *GreenThm-typeI-typeII-divisible-region' only-vertical-division only-horizontal-division common-subdiv-imp-gen-common-subdiv[OF typeI-and-typeII-one-chains-have-common-subdiv]*
by auto

lemma *GreenThm-typeI-typeII-divisible-region-finite-holes:*

assumes *valid-cube-boundary: $\forall (k, \gamma) \in \text{boundary } C. \text{valid-path } \gamma$ 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: $s = \text{cubeImage } C$
shows $\text{integral } (\text{cubeImage } C) (\lambda x. \text{partial-vector-derivative } (\lambda x. F x \cdot j) i x - \text{partial-vector-derivative } (\lambda x. F x \cdot i) j x) =$
 $\text{one-chain-line-integral } F \{i, j\} (\text{boundary } C)$
using *GreenThm-typeI-typeII-divisible-region[OF only-vertical-division two-cube-boundary-is-boundary only-horizontal-division two-cube-boundary-is-boundary common-boundary-subdiv-exists-refl[OF assms(1)]] s-is-oneCube*
by auto

lemma *GreenThm-typeI-typeII-divisible-region-equivalent-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 \subseteq one-chain-typeI and

typeII-two-cubes-trace-vertical-boundaries:
two-chain-vertical-boundary two-chain-typeII \subseteq one-chain-typeII and
boundary-of-region-is-subset-of-partition-boundaries:
one-chain-typeI \subseteq two-chain-boundary two-chain-typeI
one-chain-typeII \subseteq two-chain-boundary two-chain-typeII and
typeI-and-typeII-one-chains-have-common-subdiv:
common-boundary-sudivision-exists one-chain-typeI one-chain-typeII
shows *integral s* $(\lambda x. \text{partial-vector-derivative } (\lambda x. (F x) \cdot j) i x - \text{partial-vector-derivative } (\lambda x. (F x) \cdot i) j x) = \text{one-chain-line-integral } F \{i, j\} \text{ one-chain-typeI}$
integral s $(\lambda x. \text{partial-vector-derivative } (\lambda x. (F x) \cdot j) i x - \text{partial-vector-derivative } (\lambda x. (F x) \cdot i) j x) = \text{one-chain-line-integral } F \{i, j\} \text{ one-chain-typeII}$
proof –
let $?F\text{-}b' = \text{partial-vector-derivative } (\lambda x. (F x) \cdot i) j$
let $?F\text{-}a' = \text{partial-vector-derivative } (\lambda x. (F x) \cdot j) i$
have *one-chain-i-integrals:*
one-chain-line-integral $F \{i\} \text{ one-chain-typeI} = \text{one-chain-line-integral } F \{i\}$
one-chain-typeII \wedge
 $(\forall (k, \gamma) \in \text{one-chain-typeI}. \text{line-integral-exists } F \{i\} \gamma) \wedge$
 $(\forall (k, \gamma) \in \text{one-chain-typeII}. \text{line-integral-exists } F \{i\} \gamma)$
proof (*intro conjI*)
have i : *boundary-chain one-chain-typeI*
using *two-chain-boundary-is-boundary-chain boundary-chain-def*
boundary-of-region-is-subset-of-partition-boundaries(1)
by *blast*
have i' : *boundary-chain one-chain-typeII*
using *two-chain-boundary-is-boundary-chain boundary-chain-def*
boundary-of-region-is-subset-of-partition-boundaries(2)
by *blast*
have $\bigwedge k \gamma. (k, \gamma) \in \text{one-chain-typeI} \implies \text{line-integral-exists } F \{i\} \gamma$
using *T1.typeI-cube-line-integral-exists-boundary assms*
by (*fastforce simp add: valid-two-chain-def*)
then show $ii: \forall (k, \gamma) \in \text{one-chain-typeI}. \text{line-integral-exists } F \{i\} \gamma$ **by** *auto*
have *finite* (*two-chain-boundary two-chain-typeI*)
unfolding *two-chain-boundary-def*
proof (*rule finite-UN-I*)
show *finite two-chain-typeI*
using *T1.valid-typeI-div finite-imageD gen-division-def valid-two-chain-def*
by *auto*
show $\bigwedge a. a \in \text{two-chain-typeI} \implies \text{finite } (\text{boundary } a)$
by (*simp add: boundary-def horizontal-boundary-def vertical-boundary-def*)
qed
then have *finite one-chain-typeI*
using *boundary-of-region-is-subset-of-partition-boundaries(1) finite-subset* **by**
fastforce
moreover have *finite* (*two-chain-boundary two-chain-typeII*)
unfolding *two-chain-boundary-def*
proof (*rule finite-UN-I*)
show *finite two-chain-typeII*
using *T2.valid-typeII-div finite-imageD gen-division-def valid-two-chain-def*

by *auto*
show $\bigwedge a. a \in \text{two-chain-typeII} \implies \text{finite (boundary } a)$
by (*simp add: boundary-def horizontal-boundary-def vertical-boundary-def*)
qed
then have *finite one-chain-typeII*
using *boundary-of-region-is-subset-of-partition-boundaries(2) finite-subset by*
fastforce
ultimately show *one-chain-line-integral F {i} one-chain-typeI = one-chain-line-integral*
F {i} one-chain-typeII
 $\forall (k, \gamma) \in \text{one-chain-typeII}. \text{line-integral-exists } F \{i\} \gamma$
using *ii common-subdivision-imp-eq-line-integral[OF typeI-and-typII-one-chains-have-common-subdiv*
i i' ii]
by *auto*
qed
have *one-chain-j-integrals:*
one-chain-line-integral F {j} one-chain-typeI = one-chain-line-integral F {j}
one-chain-typeII \wedge
 $(\forall (k, \gamma) \in \text{one-chain-typeI}. \text{line-integral-exists } F \{j\} \gamma) \wedge$
 $(\forall (k, \gamma) \in \text{one-chain-typeII}. \text{line-integral-exists } F \{j\} \gamma)$
proof (*intro conjI*)
have *i: boundary-chain one-chain-typeI and i': boundary-chain one-chain-typeII*
using *two-chain-boundary-is-boundary-chain boundary-of-region-is-subset-of-partition-boundaries*
unfolding *boundary-chain-def by blast+*
have *line-integral-exists F {j} γ if $(k, \gamma) \in \text{one-chain-typeII}$ for $k \gamma$*
proof –
have *F-is-continuous: $\forall \text{twoC} \in \text{two-chain-typeII}. \text{continuous-on (cubeImage$*
twoC) ($\lambda a. F(a) \cdot j$)
using *T2.F-anal-valid by(simp add: analytically-valid-def)*
show *line-integral-exists F {j} γ*
using *that T2.valid-typeII-div*
boundary-of-region-is-subset-of-partition-boundaries(2)
using *green-typeII-cube.line-integral-exists-on-typeII-Cube-boundaries' assms*
valid-two-chain-def
apply (*simp add: two-chain-boundary-def*)
by (*metis T2.typeII-cube-line-integral-exists-boundary case-prodD subset-iff*
that two-chain-boundary-def)
qed
then show *ii: $\forall (k, \gamma) \in \text{one-chain-typeII}. \text{line-integral-exists } F \{j\} \gamma$ by auto*
have *finite (two-chain-boundary two-chain-typeI)*
unfolding *two-chain-boundary-def*
proof (*rule finite-UN-I*)
show *finite two-chain-typeI*
using *T1.valid-typeI-div finite-imageD gen-division-def valid-two-chain-def*
by *auto*
show $\bigwedge a. a \in \text{two-chain-typeI} \implies \text{finite (boundary } a)$
by (*simp add: boundary-def horizontal-boundary-def vertical-boundary-def*)
qed
then have *iv: finite one-chain-typeI*
using *boundary-of-region-is-subset-of-partition-boundaries(1) finite-subset*

```

    by fastforce
  have finite (two-chain-boundary two-chain-typeII)
    unfolding two-chain-boundary-def
  proof (rule finite-UN-I)
    show finite two-chain-typeII
      using T2.valid-typeII-div finite-imageD gen-division-def valid-two-chain-def
  by auto
  show  $\bigwedge a. a \in \text{two-chain-typeII} \implies \text{finite (boundary } a)$ 
    by (simp add: boundary-def horizontal-boundary-def vertical-boundary-def)
  qed
  then have iv': finite one-chain-typeII
    using boundary-of-region-is-subset-of-partition-boundaries(2) finite-subset
    by fastforce
  have typeII-and-typI-one-chains-have-common-subdiv:
    common-boundary-sudivision-exists one-chain-typeII one-chain-typeI
  using typeI-and-typII-one-chains-have-common-subdiv
    common-boundary-sudivision-commutative
  by auto
  show one-chain-line-integral F {j} one-chain-typeI = one-chain-line-integral
  F {j} one-chain-typeII
     $\forall (k, \gamma) \in \text{one-chain-typeI}. \text{line-integral-exists } F \{j\} \gamma$ 
  using common-sudivision-imp-eq-line-integral[OF typeII-and-typI-one-chains-have-common-subdiv
    i' i ii iv' iv] ii
  by auto
  qed
  have typeI-regions-integral:
    integral s  $(\lambda x. - ?F-b' x) = \text{one-chain-line-integral } F \{i\} \text{ one-chain-typeI}$ 
  using T1.GreenThm-typeI-divisible-region-boundary gen-divisions(1)
    typeI-two-cubes-trace-horizontal-boundaries
    boundary-of-region-is-subset-of-partition-boundaries(1)
  by auto
  have typeII-regions-integral:
    integral s  $?F-a' = \text{one-chain-line-integral } F \{j\} \text{ one-chain-typeII}$ 
  using T2.GreenThm-typeII-divisible-region-boundary gen-divisions(2)
    typeII-two-cubes-trace-vertical-boundaries
    boundary-of-region-is-subset-of-partition-boundaries(2)
  by auto
  have line-integral-dist:
    one-chain-line-integral F {i, j} one-chain-typeI = one-chain-line-integral F {i}
  one-chain-typeI + one-chain-line-integral F {j} one-chain-typeI  $\wedge$ 
    one-chain-line-integral F {i, j} one-chain-typeII = one-chain-line-integral F
  {i} one-chain-typeII + one-chain-line-integral F {j} one-chain-typeII
  proof (simp add: one-chain-line-integral-def)
    have line-integral-distrib:
       $(\sum (k,g) \in \text{one-chain-typeI}. k * \text{line-integral } F \{i, j\} g) =$ 
       $(\sum (k,g) \in \text{one-chain-typeI}. k * \text{line-integral } F \{i\} g + k * \text{line-integral } F$ 
      {j} g)  $\wedge$ 
       $(\sum (k,g) \in \text{one-chain-typeII}. k * \text{line-integral } F \{i, j\} g) =$ 
       $(\sum (k,g) \in \text{one-chain-typeII}. k * \text{line-integral } F \{i\} g + k * \text{line-integral } F$ 

```

```

{j} g)
  proof -
    have 0: k * line-integral F {i, j} g = k * line-integral F {i} g + k *
line-integral F {j} g
    if (k,g) ∈ one-chain-typeII for k g
    proof -
      have line-integral F {i, j} g = line-integral F {i} g + line-integral F {j} g
      proof -
        have finite: finite {i, j} by auto
        have line-integral-all: ∀ i ∈ {i, j}. line-integral-exists F {i} g
        using one-chain-i-integrals one-chain-j-integrals that by auto
        show ?thesis
          using line-integral-sum-gen[OF finite] R2.i-is-x-axis R2.j-is-y-axis
line-integral-all by auto
      qed
    then show k * line-integral F {i, j} g = k * line-integral F {i} g + k *
line-integral F {j} g
    by (simp add: distrib-left)
  qed
  have k * line-integral F {i, j} g = k * line-integral F {i} g + k * line-integral
F {j} g
  if (k,g) ∈ one-chain-typeI for k g
  proof -
    have finite: finite {i, j} by auto
    have line-integral-all: ∀ i ∈ {i, j}. line-integral-exists F {i} g
    using one-chain-i-integrals one-chain-j-integrals that by auto
    have line-integral F {i, j} g = line-integral F {i} g + line-integral F {j} g
    using line-integral-sum-gen[OF finite] R2.i-is-x-axis R2.j-is-y-axis line-integral-all
by auto
    then show k * line-integral F {i, j} g = k * line-integral F {i} g + k *
line-integral F {j} g
    by (simp add: distrib-left)
  qed
  then show ?thesis
  using 0 by (smt sum.cong split-cong)
  qed
  show (∑ (k::int, g) ∈ one-chain-typeI. k * line-integral F {i, j} g) =
(∑ (k, g) ∈ one-chain-typeI. k * line-integral F {i} g) + (∑ (k::int,
g) ∈ one-chain-typeI. k * line-integral F {j} g) ∧
(∑ (k::int, g) ∈ one-chain-typeII. k * line-integral F {i, j} g) =
(∑ (k, g) ∈ one-chain-typeII. k * line-integral F {i} g) + (∑ (k::int,
g) ∈ one-chain-typeII. k * line-integral F {j} g)
  proof -
    have 0: (λx. (case x of (k::int, g) ⇒ k * line-integral F {i} g) + (case x of
(k::int, g) ⇒ k * line-integral F {j} g)) =
(λx. (case x of (k::int, g) ⇒ (k * line-integral F {i} g)
+ k * line-integral F {j} g))
    using comm-monoid-add-class.sum.distrib by auto
    then have 1: (∑ x ∈ one-chain-typeI. (case x of (k::int, g) ⇒ k * line-integral

```

$F \{i\} g) + (\text{case } x \text{ of } (k::\text{int}, g) \Rightarrow k * \text{line-integral } F \{j\} g)) =$
 $(\sum_{x \in \text{one-chain-typeI}}. (\text{case } x \text{ of } (k::\text{int}, g) \Rightarrow (k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g)))$
by *presburger*
have $(\sum_{x \in \text{one-chain-typeII}}. (\text{case } x \text{ of } (k, g) \Rightarrow k * \text{line-integral } F \{i\} g) + (\text{case } x \text{ of } (k, g) \Rightarrow k * \text{line-integral } F \{j\} g)) =$
 $(\sum_{x \in \text{one-chain-typeII}}. (\text{case } x \text{ of } (k, g) \Rightarrow (k * \text{line-integral } F \{i\} g + k * \text{line-integral } F \{j\} g)))$
using 0 **by** *presburger*
then show *?thesis*
using *sum.distrib[of (\lambda(k, g). k * line-integral F {i} g) (\lambda(k, g). k * line-integral F {j} g) one-chain-typeI]*
*sum.distrib[of (\lambda(k, g). k * line-integral F {i} g) (\lambda(k, g). k * line-integral F {j} g) one-chain-typeII]*
line-integral-distrib
1
by *auto*
qed
qed
have *integral-dis: integral s (\lambda x. ?F-a' x - ?F-b' x) = integral s (\lambda x. ?F-a' x) + integral s (\lambda x. - ?F-b' x)*
proof –
have *(?F-a' has-integral (\sum img \in cubeImage ' two-chain-typeII. integral img ?F-a')) s*
proof –
have *(?F-a' has-integral integral (cubeImage twoCube) ?F-a') (cubeImage twoCube)*
if *twoCube \in two-chain-typeII* **for** *twoCube*
by *(simp add: analytically-valid-imp-part-deriv-integrable-on T2.F-anal-valid has-integral-integrable-integral that)*
then have $\bigwedge u. u \in (\text{cubeImage ' two-chain-typeII}) \implies (\text{?F-a' has-integral integral } u \text{ ?F-a'}) u$
by *auto*
then show *?thesis*
using *gen-divisions(2) unfolding gen-division-def*
by *(metis has-integral-Union)*
qed
then have *F-a'-integrable:*
(?F-a' integrable-on s) **by** *auto*
have *(?F-b' has-integral (\sum img \in cubeImage ' two-chain-typeI. integral img ?F-b')) s*
proof –
have $\forall \text{twoCube} \in \text{two-chain-typeI}. (\text{?F-b' has-integral integral (cubeImage twoCube) ?F-b'}) (\text{cubeImage twoCube})$
by *(simp add: analytically-valid-imp-part-deriv-integrable-on T1.F-anal-valid has-integral-integrable-integral)*
then have $\bigwedge u. u \in (\text{cubeImage ' two-chain-typeI}) \implies (\text{?F-b' has-integral integral } u \text{ ?F-b'}) u$
by *auto*

```

    then show ?thesis
    using gen-divisions(1) unfolding gen-division-def
    by (metis has-integral-Union)
  qed
  then show ?thesis
  using F-a'-integrable Henstock-Kurzweil-Integration.integral-diff by auto
  qed
  show integral s (λx. ?F-a' x - ?F-b' x) = one-chain-line-integral F {i, j}
  one-chain-typeI
  using one-chain-j-integrals integral-dis line-integral-dist typeI-regions-integral
  typeII-regions-integral
  by auto
  show integral s (λx. ?F-a' x - ?F-b' x) = one-chain-line-integral F {i, j}
  one-chain-typeII
  using one-chain-i-integrals integral-dis line-integral-dist typeI-regions-integral
  typeII-regions-integral
  by auto
  qed

end
end
theory SymmetricR2Shapes
  imports Green
begin

context R2
begin

lemma valid-path-valid-swap:
  assumes valid-path (λx::real. ((f x)::real, (g x)::real))
  shows valid-path (prod.swap o (λx. (f x, g x)))
  unfolding o-def valid-path-def piecewise-C1-differentiable-on-def swap-simp
  proof (intro conjI)
    show continuous-on {0..1} (λx. (g x, f x))
    using assms
    using continuous-on-Pair continuous-on-componentwise[where f = (λx. (f x,
g x))]
    by (auto simp add: real-pair-basis valid-path-def piecewise-C1-differentiable-on-def)
    show ∃ S. finite S ∧ (λx. (g x, f x)) C1-differentiable-on {0..1} - S
    proof -
      obtain S where finite S and S: (λx. (f x, g x)) C1-differentiable-on {0..1}
      - S
      using assms
      by (auto simp add: real-pair-basis valid-path-def piecewise-C1-differentiable-on-def)
    have 0: f C1-differentiable-on {0..1} - S using S assms
    using C1-diff-components-2[of (1,0) (λx. (f x, g x))]
    by (auto simp add: real-pair-basis algebra-simps)
    have 1: g C1-differentiable-on {0..1} - S using S assms
    using C1-diff-components-2 [of (0,1), OF - S] real-pair-basis by fastforce
  end
end
end

```


have *: $(\lambda x. (g\ x, f\ x))$ *C1-differentiable-on* $\{0..1\} - S$
using *0 1 C1-differentiable-on-components* [where $f = (\lambda x. (g\ x, f\ x))$]
by (*auto simp add: real-pair-basis valid-path-def piecewise-C1-differentiable-on-def*)
then show ?thesis **using** *(finite S)* **by** *auto*
qed
qed

lemma *pair-fun-components*: $C = (\lambda x. (C\ x \cdot i, C\ x \cdot j))$
by (*simp add: i-is-x-axis inner-Pair-0 j-is-y-axis*)

lemma *swap-pair-fun*: $(\lambda y. \text{prod.swap}\ (C\ (y, 0))) = (\lambda x. (C\ (x, 0) \cdot j, C\ (x, 0) \cdot i))$
by (*simp add: prod.swap-def i-is-x-axis inner-Pair-0 j-is-y-axis*)

lemma *swap-pair-fun'*: $(\lambda y. \text{prod.swap}\ (C\ (y, 1))) = (\lambda x. (C\ (x, 1) \cdot j, C\ (x, 1) \cdot i))$
by (*simp add: prod.swap-def i-is-x-axis inner-Pair-0 j-is-y-axis*)

lemma *swap-pair-fun''*: $(\lambda y. \text{prod.swap}\ (C\ (0, y))) = (\lambda x. (C\ (0,x) \cdot j, C\ (0,x) \cdot i))$
by (*simp add: prod.swap-def i-is-x-axis inner-Pair-0 j-is-y-axis*)

lemma *swap-pair-fun'''*: $(\lambda y. \text{prod.swap}\ (C\ (1, y))) = (\lambda x. (C\ (1,x) \cdot j, C\ (1,x) \cdot i))$
by (*simp add: prod.swap-def i-is-x-axis inner-Pair-0 j-is-y-axis*)

lemma *swap-valid-boundaries*:

assumes $\forall (k,\gamma) \in \text{boundary}\ C. \text{valid-path}\ \gamma$
assumes $(k,\gamma) \in \text{boundary}\ (\text{prod.swap}\ o\ C\ o\ \text{prod.swap})$
shows *valid-path* γ
using *assms*
valid-path-valid-swap [of $\lambda x. (\lambda x. C\ (x, 0))\ x \cdot i\ \lambda x. (\lambda x. C\ (x, 0))\ x \cdot j$]
pair-fun-components [of $(\lambda x. C\ (x, 0))$]
pair-fun-components [of $(\lambda y. C\ (y, 0))$]
valid-path-valid-swap [of $\lambda x. (\lambda y. C\ (y, 1))\ x \cdot i\ \lambda x. (\lambda y. C\ (y, 1))\ x \cdot j$]
pair-fun-components [of $(\lambda y. C\ (y, 1))$]
pair-fun-components [of $(\lambda x. C\ (x, 1))$]
valid-path-valid-swap [of $\lambda x. (\lambda y. C\ (1,y))\ x \cdot i\ \lambda x. (\lambda y. C\ (1,y))\ x \cdot j$]
pair-fun-components [of $(\lambda y. C\ (1,y))$]
pair-fun-components [of $(\lambda x. C\ (1,x))$]
valid-path-valid-swap [of $\lambda x. (\lambda y. C\ (0,y))\ x \cdot i\ \lambda x. (\lambda y. C\ (0,y))\ x \cdot j$]
pair-fun-components [of $(\lambda y. C\ (0,y))$]
pair-fun-components [of $(\lambda x. C\ (0,x))$]
by (*auto simp add: boundary-def horizontal-boundary-def vertical-boundary-def o-def real-pair-basis swap-pair-fun swap-pair-fun' swap-pair-fun'' swap-pair-fun'''*)

lemma *prod-comp-eq*:

assumes $f = \text{prod.swap}\ o\ g$
shows $\text{prod.swap}\ o\ f = g$

```

using swap-comp-swap assms
by fastforce

lemma swap-typeI-is-typeII:
  assumes typeI-twoCube C
  shows typeII-twoCube (prod.swap o C o prod.swap)
proof (simp add: typeI-twoCube-def typeII-twoCube-def)
  obtain a b g1 g2 where C: a < b
    ( $\forall x \in \{a..b\}. g2\ x \leq g1\ x$ )
    cubeImage C =  $\{(x, y). x \in \{a..b\} \wedge y \in \{g2\ x..g1\ x\}\}$ 
    C =  $(\lambda(x, y). ((1 - x) * a + x * b, (1 - y) * g2\ ((1 - x) * a + x * b) + y$ 
  * g1  $((1 - x) * a + x * b)))$ 
    g1 piecewise-C1-differentiable-on  $\{a..b\}$ 
    g2 piecewise-C1-differentiable-on  $\{a..b\}$ 
    using typeI-cube-explicit-spec [OF assms]
    by blast
  show  $\exists a\ b. a < b \wedge$ 
    ( $\exists g1\ g2. (\forall x \in \{a..b\}. g2\ x \leq g1\ x) \wedge$ 
      prod.swap o C o prod.swap =
       $(\lambda(y, x). ((1 - y) * g2\ ((1 - x) * a + x * b) + y * g1\ ((1 - x) *$ 
a + x * b),  $(1 - x) * a + x * b)) \wedge$ 
      g1 piecewise-C1-differentiable-on  $\{a..b\} \wedge g2 piecewise-C1-differentiable-on
 $\{a..b\}$ )
    using C by (fastforce simp add: prod.swap-def o-def)
qed

lemma valid-cube-valid-swap:
  assumes valid-two-cube C
  shows valid-two-cube (prod.swap o C o prod.swap)
  using assms unfolding valid-two-cube-def boundary-def horizontal-boundary-def
vertical-boundary-def
  apply (auto simp: card-insert-if split: if-split-asm)
  apply (metis swap-swap)+
  done

lemma twoChainVertDiv-of-itself:
  assumes finite C
   $\forall (k, \gamma) \in (\text{two-chain-boundary } C). \text{valid-path } \gamma$ 
  shows only-vertical-division (two-chain-boundary C) C
proof (clarsimp simp add: only-vertical-division-def)
  show  $\exists \mathcal{V}\ \mathcal{H}. \text{finite } \mathcal{H} \wedge \text{finite } \mathcal{V} \wedge$ 
    ( $\forall x \in \mathcal{V}. \text{case } x \text{ of } (k, \gamma) \Rightarrow \exists x \in \text{two-chain-vertical-boundary } C. \text{case } x$ 
of  $(k', \gamma') \Rightarrow \exists a \in \{0..1\}. \exists b \in \{0..1\}. a \leq b \wedge \text{subpath } a\ b\ \gamma' = \gamma) \wedge$ 
    (common-sudiv-exists (two-chain-horizontal-boundary C)  $\mathcal{H} \vee$ 
    common-reparam-exists  $\mathcal{H}$  (two-chain-horizontal-boundary C))  $\wedge$ 
    boundary-chain  $\mathcal{H} \wedge \text{two-chain-boundary } C = \mathcal{V} \cup \mathcal{H} \wedge (\forall (k, \gamma) \in \mathcal{H}. \text{valid-path } \gamma)$ 
  proof (intro exI)
    let  $? \mathcal{H} = \text{two-chain-horizontal-boundary } C$$ 
```

have $0: \forall (k, \gamma) \in ?\mathcal{H}. \text{valid-path } \gamma$ **using** *assms(2)*
by (*auto simp add: two-chain-horizontal-boundary-def two-chain-boundary-def boundary-def*)
have $\bigwedge a b. (a, b) \in \text{two-chain-vertical-boundary } C \implies$
 $\exists x \in \text{two-chain-vertical-boundary } C. \text{case } x \text{ of } (k', \gamma') \Rightarrow \exists a \in \{0..1\}.$
 $\exists c \in \{0..1\}. a \leq c \wedge \text{subpath } a c \gamma' = b$
by (*metis (mono-tags, lifting) atLeastAtMost-iff case-prod-conv le-numeral-extra(1) order-refl subpath-trivial*)
moreover have *common-sudiv-exists ?H ?H*
using *gen-common-boundary-subdiv-exists-refl-twochain-boundary[OF 0 two-chain-horizontal-boundary-is-b boundary-def]*
by *auto*
moreover have *boundary-chain ?H*
using *two-chain-horizontal-boundary-is-boundary-chain* **by** *auto*
moreover have $\bigwedge a b. (a, b) \in \text{two-chain-boundary } C \implies (a, b) \notin ?\mathcal{H} \implies$
 $(a, b) \in \text{two-chain-vertical-boundary } C$
by (*auto simp add: two-chain-boundary-def two-chain-horizontal-boundary-def two-chain-vertical-boundary-def boundary-def*)
moreover have $\bigwedge a b. (a, b) \in \text{two-chain-vertical-boundary } C \implies (a, b) \in$
 $\text{two-chain-boundary } C$
 $\bigwedge a b. (a, b) \in ?\mathcal{H} \implies (a, b) \in \text{two-chain-boundary } C$
by (*auto simp add: two-chain-boundary-def two-chain-horizontal-boundary-def two-chain-vertical-boundary-def boundary-def*)
moreover have $\bigwedge a b. (a, b) \in ?\mathcal{H} \implies \text{valid-path } b$
using 0 **by** *blast*
ultimately show *finite ?H* \wedge
 $\text{finite } (\text{two-chain-vertical-boundary } C) \wedge$
 $(\forall x \in \text{two-chain-vertical-boundary } C.$
 $\text{case } x \text{ of } (k, \gamma) \Rightarrow \exists x \in \text{two-chain-vertical-boundary } C. \text{case } x \text{ of } (k',$
 $\gamma') \Rightarrow \exists a \in \{0..1\}. \exists b \in \{0..1\}. a \leq b \wedge \text{subpath } a b \gamma' = \gamma) \wedge$
 $(\text{common-sudiv-exists } ?\mathcal{H} ?\mathcal{H} \vee$
 $\text{common-reparam-exists } ?\mathcal{H} ?\mathcal{H}) \wedge$
 $\text{boundary-chain } ?\mathcal{H} \wedge \text{two-chain-boundary } C = \text{two-chain-vertical-boundary}$
 $C \cup ?\mathcal{H} \wedge (\forall (k, \gamma) \in ?\mathcal{H}. \text{valid-path } \gamma)$
by (*auto simp add: finite-two-chain-horizontal-boundary[OF assms(1)] finite-two-chain-vertical-boundary[assms(1)]*)
qed
qed

end

definition *x-coord* **where** $x\text{-coord} \equiv (\lambda t::\text{real}. t - 1/2)$

lemma *x-coord-smooth*: *x-coord* *C1-differentiable-on* $\{a..b\}$
by (*simp add: x-coord-def*)

lemma *x-coord-bounds*:

assumes $(0::\text{real}) \leq x \leq 1$
shows $-1/2 \leq x\text{-coord } x \wedge x\text{-coord } x \leq 1/2$

```

using assms by(auto simp add: x-coord-def)

lemma x-coord-img: x-coord ‘ {0::real}..1 } = { -1/2 .. 1/2 }
  by (auto simp add: x-coord-def image-def algebra-simps)

lemma x-coord-back-img: finite ({0..1} ∩ x-coord - ‘ {x::real})
  by (simp add: finite-vimageI inj-on-def x-coord-def)

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 ≤ x
    x ≤ 1
    0 ≤ y
    y ≤ 1
  shows 0 ≤ rot-x x y ∧ rot-x x y ≤ 1
proof -
  have i: ∧ a::real. a ≤ 0 ⇒ 0 ≤ y ⇒ y ≤ 1 ⇒ -1/2 < a ⇒ (a * (1 -
2*y) ≤ 1/2)
  proof -
    have 0: ∧ a::real. a ≤ 0 ⇒ 0 ≤ y ⇒ y ≤ 1 ⇒ -1/2 < a ⇒ (-a ≤
1/2)
    by (sos (((A<0 * A<1) * R<1) + (R<1 * (R<1/4 * [2*a + 1]^2))))
    have 1: ∧ a. a ≤ 0 ⇒ 0 ≤ y ⇒ y ≤ 1 ⇒ -1/2 < a ⇒ (a * (1 - 2*y)
≤ -a)
    by (sos (((A<0 * A<1) * R<1) + (((A<=0 * (A<1 * R<1)) * (R<2/3 *
[1]^2)) + (((A<=0 * (A<=2 * R<1)) * (R<2/3 * [1]^2)) + ((A<=0 * (A<=2
* (A<0 * R<1))) * (R<2/3 * [1]^2)))))))
    show ∧ a::real. a ≤ 0 ⇒ 0 ≤ y ⇒ y ≤ 1 ⇒ -1/2 < a ⇒ (a * (1 -
2*y) ≤ 1/2) using 0 1 by force
  qed
  have *: (x * 2 + y * 4 ≤ 3 + x * (y * 4)) = ( (x-1) ≤ 1/2 + (x-1) * (y *
2) )
  by (sos (((A<0 * R<1) + ((A<=0 * R<1) * (R<2 * [1]^2)))) & (((A<0 *
R<1) + ((A<=0 * R<1) * (R<1/2 * [1]^2))))))
  show ?thesis
  using assms
  apply(auto simp add: algebra-simps divide-simps linorder-class.not-le)
  apply (sos (((A<0 * R<1) + (((A<=2 * (A<=3 * R<1)) * (R<1 * [1]^2))
+ (((A<=1 * R<1) * (R<1 * [1]^2)) + ((A<=0 * (A<=1 * R<1)) * (R<2 *
[1]^2)))))))
  apply (sos (((A<0 * R<1) + (((A<=2 * R<1) * (R<1 * [1]^2)) + (((A<=1
* (A<=3 * R<1)) * (R<1 * [1]^2)) + ((A<=0 * (A<=2 * R<1)) * (R<2 *
[1]^2)))))))
  using i[of (x::real) - 1] affine-ineq
  apply (fastforce simp: algebra-simps *)+
  done

```

qed

end

1 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* = $\text{sqrt } (1/4 - t * t)$

definition *circle-cube* **where**

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

lemma *circle-cube-nice*:

shows *circle-cube* = $(\lambda(x,y). (d * x\text{-coord } x, (2 * y - 1) * d * \text{circle-y } (x\text{-coord } x)))$

by (*auto simp add: circle-cube-def circle-y-def x-coord-def*)

definition *rot-circle-cube* **where**

rot-circle-cube = *prod.swap* \circ (*circle-cube*) \circ *prod.swap*

abbreviation *rot-y* *t1* *t2* $\equiv ((t1 - 1/2)/(2 * \text{circle-y } (x\text{-coord } (rot-x \ t1 \ t2))) + 1/2)::real)$

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

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

by (*auto simp add: x-coord-inv-def x-coord-def*)

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

by (*auto simp add: x-coord-inv-def x-coord-def*)

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

abbreviation *rot-x''* (*x::real*) (*y::real*) $\equiv (x\text{-coord-inv } ((2 * y - 1) * \text{circle-y } (x\text{-coord } x)))$

lemma *circle-y-bounds*:

assumes $-1/2 \leq (x::real) \wedge x \leq 1/2$

shows $0 \leq \text{circle-y } x \wedge \text{circle-y } x \leq 1/2$

unfolding *circle-y-def real-sqrt-ge-0-iff*
proof –
show $0 \leq 1/4 - x * x$
using *assms* **by** (*sos* ((($A < 0 * R < 1$) + (($A \leq 0 * (A \leq 1 * R < 1)$) * ($R < 1 * [1]^2$))))))
show $\text{sqrt}(1/4 - x * x) \leq 1/2$
apply (*rule real-le-lsqrt*)
using *assms* **apply**(*auto simp add: divide-simps algebra-simps*)
by (*sos* ((($A < 0 * R < 1$) + (($A \leq 0 * (A \leq 1 * R < 1)$) * ($R < 1 * [1]^2$))))))
qed

lemma *circle-y-x-coord-bounds*:
assumes $0 \leq (x::\text{real}) \ x \leq 1$
shows $0 \leq \text{circle-y}(x\text{-coord } x) \wedge \text{circle-y}(x\text{-coord } x) \leq 1/2$
using *circle-y-bounds[OF x-coord-bounds[OF assms]]* **by** *auto*

lemma *rot-x-ivl*:
assumes $(0::\text{real}) \leq x \ x \leq 1 \ 0 \leq y \ y \leq 1$
shows $0 \leq \text{rot-x}'' \ x \ y \wedge \text{rot-x}'' \ x \ y \leq 1$
proof
have $\bigwedge a::\text{real}. 0 \leq a \wedge a \leq 1/2 \implies 0 \leq 1/2 + (2 * y - 1) * a$ **using** *assms*
by (*sos* ((($A < 0 * R < 1$) + ((($A \leq 4 * R < 1$) * ($R < 1/2 * [1]^2$)) + ((($A \leq 1 * (A \leq 5 * R < 1)$) * ($R < 1 * [1]^2$)) + (($A \leq 0 * (A \leq 4 * R < 1)$) * ($R < 1 * [1]^2$)))))))))
then show $0 \leq \text{rot-x}'' \ x \ y$
using *assms circle-y-x-coord-bounds* **by**(*auto simp add: x-coord-inv-def*)
have $\bigwedge a::\text{real}. 0 \leq a \wedge a \leq 1/2 \implies 1/2 + (2 * y - 1) * a \leq 1$ **using** *assms*
by (*sos* ((($A < 0 * R < 1$) + ((($A \leq 5 * R < 1$) * ($R < 1/2 * [1]^2$)) + ((($A \leq 1 * (A \leq 4 * R < 1)$) * ($R < 1 * [1]^2$)) + (($A \leq 0 * (A \leq 5 * R < 1)$) * ($R < 1 * [1]^2$)))))))))
then show $\text{rot-x}'' \ x \ y \leq 1$
using *assms circle-y-x-coord-bounds* **by** (*auto simp add: x-coord-inv-def*)
qed

abbreviation *rot-y''* ($x::\text{real}$) ($y::\text{real}$) $\equiv (x\text{-coord } x)/(2 * (\text{circle-y}(x\text{-coord } (\text{rot-x}'' \ x \ y)))) + 1/2$

lemma *rot-y-ivl*:
assumes $(0::\text{real}) \leq x \ x \leq 1 \ 0 \leq y \ y \leq 1$
shows $0 \leq \text{rot-y}'' \ x \ y \wedge \text{rot-y}'' \ x \ y \leq 1$
proof
show $0 \leq \text{rot-y}'' \ x \ y$
proof(*cases* ($x\text{-coord } x < 0$)
case *True*
have $i: \bigwedge a \ b::\text{real}. a < 0 \implies 0 \leq a + b \implies (0 \leq a/(2*(b)) + 1/2)$
by(*auto simp add: algebra-simps divide-simps*)
have $*$: $(1/2 - x) \leq \text{sqrt}(x * x + (1/4 + (x * (y * 4) + x * (x * (y * (y * 4)))))) - (x + (x * (x * (y * 4)) + x * (y * (y * 4))))$
apply (*rule real-le-rsqrt*)

```

using assms apply (simp add: algebra-simps power2-eq-square mult-left-le-one-le)
by (sos ((( $A < 0 * R < 1$ ) + (( $A <= 0 * (A <= 1 * (A <= 2 * (A <= 3 * R < 1))$ )))
* ( $R < 4 * [1]^2$ ))))))
have rw:  $|x - x * x| = x - x * x$  using assms
by (sos (( $\&$  ((( $A < 0 * A < 1$ ) *  $R < 1$ ) + (( $A <= 0 * (A <= 1 * (A < 1 * R < 1))$ ) * ( $R < 1 * [1]^2$ )))))) & ((( $A < 0 * A < 1$ ) *  $R < 1$ ) + (( $A <= 0 * (A <= 1 * (A < 1 * R < 1))$ ) * ( $R < 1 * [1]^2$ ))))))
have  $0 \leq x\text{-coord } x + (\text{circle-}y (x\text{-coord } (\text{rot-}x'' x y)))$ 
using * apply (auto simp add: x-coord-inv-2)
by (auto simp add: circle-y-def algebra-simps rw x-coord-def)
then show ?thesis
using True i by blast
next
case False
have i:  $\bigwedge a b::\text{real. } 0 \leq a \implies 0 \leq b \implies (0 \leq a/(2*b) + 1/2)$ 
by (auto simp add: algebra-simps divide-simps)
have  $0 \leq \text{circle-}y (x\text{-coord } (x\text{-coord-inv } ((2 * y - 1) * \text{circle-}y (x\text{-coord } x))))$ 
proof -
have rw:  $|x - x * x| = x - x * x$  using assms
by (sos (( $\&$  ((( $A < 0 * A < 1$ ) *  $R < 1$ ) + (( $A <= 0 * (A <= 1 * (A < 1 * R < 1))$ ) * ( $R < 1 * [1]^2$ )))))) & ((( $A < 0 * A < 1$ ) *  $R < 1$ ) + (( $A <= 0 * (A <= 1 * (A < 1 * R < 1))$ ) * ( $R < 1 * [1]^2$ ))))))
have  $\bigwedge x. 0 \leq x \implies x \leq 1/2 \implies -1/2 \leq (2 * y - 1) * x$  using assms
by (sos ((( $A < 0 * R < 1$ ) + (( $A <= 4 * R < 1$ ) * ( $R < 1/2 * [1]^2$ ))) + ((( $A <= 1 * (A <= 5 * R < 1)$ ) * ( $R < 1 * [1]^2$ ))) + (( $A <= 0 * (A <= 4 * R < 1)$ ) * ( $R < 1 * [1]^2$ ))))))
then have  $-1/2 \leq (2 * y - 1) * \text{circle-}y (x\text{-coord } x)$ 
using circle-y-x-coord-bounds assms(1-2) by auto
moreover
have  $\bigwedge x. 0 \leq x \implies x \leq 1/2 \implies (2 * y - 1) * x \leq 1/2$  using assms
by (sos ((( $A < 0 * R < 1$ ) + (( $A <= 5 * R < 1$ ) * ( $R < 1/2 * [1]^2$ ))) + ((( $A <= 1 * (A <= 4 * R < 1)$ ) * ( $R < 1 * [1]^2$ ))) + (( $A <= 0 * (A <= 5 * R < 1)$ ) * ( $R < 1 * [1]^2$ ))))))
then have  $(2 * y - 1) * \text{circle-}y (x\text{-coord } x) \leq 1/2$ 
using circle-y-x-coord-bounds assms(1-2) by auto
ultimately show  $0 \leq \text{circle-}y (x\text{-coord } (x\text{-coord-inv } ((2 * y - 1) * \text{circle-}y (x\text{-coord } x))))$ 
by (simp add: circle-y-bounds(1) x-coord-inv-2)
qed
then show ?thesis
using False by auto
qed
have i:  $\bigwedge a b::\text{real. } a < 0 \implies 0 \leq b \implies (a/(2*(b)) + 1/2) \leq 1$ 
by (auto simp add: algebra-simps divide-simps)
show  $\text{rot-}y'' x y \leq 1$ 
proof (cases ( $x\text{-coord } x < 0$ ))
case True
have i:  $\bigwedge a b::\text{real. } a < 0 \implies 0 \leq b \implies (a/(2*(b)) + 1/2) \leq 1$ 
by (auto simp add: algebra-simps divide-simps)

```

```

have  $\bigwedge x. 0 \leq x \implies x \leq 1/2 \implies -1/2 \leq (2 * y - 1) * x$  using assms
  by (sos ((( $A < 0 * R < 1$ ) + ((( $A <= 4 * R < 1$ ) * ( $R < 1/2 * [1]^2$ )) + ((( $A <= 1$ 
* ( $A <= 5 * R < 1$ )) * ( $R < 1 * [1]^2$ )) + (( $A <= 0 * (A <= 4 * R < 1)$ ) * ( $R < 1 * [1]^2$ )))))))))
  then have  $-1/2 \leq (2 * y - 1) * \text{circle-}y \text{ (}x\text{-coord } x)$ 
    using circle-y-x-coord-bounds assms(1-2) by auto
  moreover have  $\bigwedge x. 0 \leq x \implies x \leq 1/2 \implies (2 * y - 1) * x \leq 1/2$  using
assms
    by (sos ((( $A < 0 * R < 1$ ) + ((( $A <= 5 * R < 1$ ) * ( $R < 1/2 * [1]^2$ )) + ((( $A <= 1$ 
* ( $A <= 4 * R < 1$ )) * ( $R < 1 * [1]^2$ )) + (( $A <= 0 * (A <= 5 * R < 1)$ ) * ( $R < 1 * [1]^2$ )))))))))
    then have  $(2 * y - 1) * \text{circle-}y \text{ (}x\text{-coord } x) \leq 1/2$ 
      using circle-y-x-coord-bounds assms(1-2) by auto
    ultimately have  $0 \leq \text{circle-}y \text{ (}x\text{-coord (}x\text{-coord-inv ((}2 * y - 1) * \text{circle-}y$ 
( $x\text{-coord } x)))$ 
      by (simp add: circle-y-bounds(1) x-coord-inv-2)
    then show ?thesis
      by (simp add: True i)
  next
  case False
  have  $i: \bigwedge a b::\text{real}. 0 \leq a \implies a \leq b \implies (a/(2*b) + 1/2) \leq 1$ 
    by (auto simp add: algebra-simps divide-simps)
  have  $(x - 1/2) * (x - 1/2) \leq (x * x + (1/4 + (x * (y * 4) + x * (x * (y * (y * 4)))))) - (x + (x * (x * (y * 4) + x * (y * (y * 4))))))$ 
    using assms False
    apply (auto simp add: x-coord-def)
  by (sos ((( $A < 0 * R < 1$ ) + ((( $A <= 0 * (A <= 1 * (A <= 2 * R < 1))$ ) * ( $R < 2 * [1]^2$ )) + (( $A <= 0 * (A <= 1 * (A <= 2 * (A <= 3 * R < 1))$ ) * ( $R < 2 * [1]^2$ )))))))))
  then have  $\text{sqrt} ((x - 1/2) * (x - 1/2)) \leq \text{sqrt} (x * x + (1/4 + (x * (y * 4) + x * (x * (y * (y * 4)))))) - (x + (x * (x * (y * 4) + x * (y * (y * 4))))))$ 
    using real-sqrt-le-mono by blast
  then have  $*$ :  $(x - 1/2) \leq \text{sqrt} (x * x + (1/4 + (x * (y * 4) + x * (x * (y * (y * 4)))))) - (x + (x * (x * (y * 4) + x * (y * (y * 4))))))$ 
    using assms False by (auto simp add: x-coord-def)
  have  $rw: |x - x * x| = x - x * x$  using assms
    by (sos (( $()$ ) & ((( $A < 0 * A < 1$ ) *  $R < 1$ ) + (( $A <= 0 * (A <= 1 * (A < 1 * R < 1))$ ) * ( $R < 1 * [1]^2$ ))) & ((( $A < 0 * A < 1$ ) *  $R < 1$ ) + (( $A <= 0 * (A <= 1 * (A < 1 * R < 1))$ ) * ( $R < 1 * [1]^2$ )))))))))
  have  $x\text{-coord } x \leq \text{circle-}y \text{ (}x\text{-coord (}x\text{-coord-inv ((}2 * y - 1) * \text{circle-}y \text{ (}x\text{-coord } x)))$ 
    using  $*$  unfolding x-coord-inv-2
    by (auto simp add: circle-y-def algebra-simps rw x-coord-def)
  then show ?thesis
    using False i by auto
qed
qed

```

lemma *circle-eq-rot-circle*:

assumes $0 \leq x \ x \leq 1 \ 0 \leq y \ y \leq 1$

shows ($\text{circle-cube } (x, y) = (\text{rot-circle-cube } (\text{rot-}y'' x y, \text{rot-}x'' x y))$)
proof
have rw : $|1/4 - x\text{-coord } x * x\text{-coord } x| = 1/4 - x\text{-coord } x * x\text{-coord } x$
apply(*rule abs-of-nonneg*)
using *assms mult-left-le by (auto simp add: x-coord-def divide-simps algebra-simps)*
show $\text{fst } (\text{circle-cube } (x, y) = \text{fst } (\text{rot-circle-cube } (\text{rot-}y'' x y, \text{rot-}x'' x y))$
using *assms d-gt-0*
apply(*simp add: circle-cube-nice rot-circle-cube-def x-coord-inv-2 circle-y-def algebra-simps rw*)
apply(*auto simp add: x-coord-def algebra-simps*)
by (*sos* ((($(A < 0 * A < 1) * ((A < 0 * A < 1) * R < 1)$) + ($([d]^2 * A = 0) + ((A <= 1 * (A <= 2 * (A <= 3 * R < 1))) * (R < 8 * [d]^2)$) + ($(A <= 1 * (A <= 2 * (A <= 3 * (A < 1 * R < 1))) * (R < 8 * [d]^2)$))))) & ((($(A < 0 * A < 1) * ((A < 0 * A < 1) * R < 1)$) + ($([d]^2 * A = 0) + ((A <= 0 * (A <= 2 * (A <= 3 * R < 1))) * (R < 8 * [d]^2)$) + ($(A <= 0 * (A <= 2 * (A <= 3 * (A < 1 * R < 1))) * (R < 8 * [d]^2)$)))))

show $\text{snd } (\text{circle-cube } (x, y) = \text{snd } (\text{rot-circle-cube } (\text{rot-}y'' x y, \text{rot-}x'' x y))$
using *assms*
by(*auto simp add: circle-cube-def rot-circle-cube-def x-coord-inv-def circle-y-def x-coord-def*)
qed

lemma *rot-circle-eq-circle*:

assumes $0 \leq x \leq 1 \ 0 \leq y \leq 1$
shows ($\text{rot-circle-cube } (x, y) = (\text{circle-cube } (\text{rot-}x'' y x, \text{rot-}y'' y x))$)
proof
show $\text{fst } (\text{rot-circle-cube } (x, y) = \text{fst } (\text{circle-cube } (\text{rot-}x'' y x, \text{rot-}y'' y x))$
using *assms*
by(*auto simp add: circle-cube-def rot-circle-cube-def x-coord-inv-def circle-y-def x-coord-def*)
have rw : $|1/4 - x\text{-coord } y * x\text{-coord } y| = 1/4 - x\text{-coord } y * x\text{-coord } y$
apply(*rule abs-of-nonneg*)
using *assms mult-left-le by (auto simp add: x-coord-def divide-simps algebra-simps)*
show $\text{snd } (\text{rot-circle-cube } (x, y) = \text{snd } (\text{circle-cube } (\text{rot-}x'' y x, \text{rot-}y'' y x))$
using *assms d-gt-0*
apply(*simp add: circle-cube-nice rot-circle-cube-def x-coord-inv-2 circle-y-def algebra-simps rw*)
apply(*auto simp add: x-coord-def algebra-simps*)
by (*sos* ((($(A < 0 * A < 1) * ((A < 0 * A < 1) * R < 1)$) + ($([d]^2 * A = 0) + ((A <= 0 * (A <= 1 * (A <= 3 * R < 1))) * (R < 8 * [d]^2)$) + ($(A <= 0 * (A <= 1 * (A <= 3 * (A < 1 * R < 1))) * (R < 8 * [d]^2)$))))) & ((($(A < 0 * A < 1) * ((A < 0 * A < 1) * R < 1)$) + ($([d]^2 * A = 0) + ((A <= 0 * (A <= 1 * (A <= 2 * R < 1))) * (R < 8 * [d]^2)$) + ($(A <= 0 * (A <= 1 * (A <= 2 * (A < 1 * R < 1))) * (R < 8 * [d]^2)$)))))

qed

lemma *rot-img-eq*:

assumes $0 < d$
shows ($\text{cubeImage } (\text{circle-cube }) = (\text{cubeImage } (\text{rot-circle-cube}))$)

```

apply(auto simp add: cubeImage-def image-def cbox-def real-pair-basis)
by (meson rot-y-ivl rot-x-ivl assms circle-eq-rot-circle rot-circle-eq-circle)+

lemma rot-circle-div-circle:
  assumes  $0 < (d::real)$ 
  shows gen-division (cubeImage circle-cube) (cubeImage ‘ {rot-circle-cube})
  using rot-img-eq[OF assms] by(auto simp add: gen-division-def)

lemma circle-cube-boundary-valid:
  assumes  $(k,\gamma)\in\text{boundary circle-cube}$ 
  shows valid-path  $\gamma$ 
proof –
  have f01: finite{0,1}
    by simp
  show ?thesis
    using assms
    unfolding boundary-def horizontal-boundary-def vertical-boundary-def circle-cube-def
valid-path-def piecewise-C1-differentiable-on-def
    by safe (rule derivative-intros continuous-intros f01 exI ballI conjI refl | force
simp add: field-simps)+
qed

lemma rot-circle-cube-boundary-valid:
  assumes  $(k,\gamma)\in\text{boundary rot-circle-cube}$ 
  shows valid-path  $\gamma$ 
  using assms swap-valid-boundaries circle-cube-boundary-valid
  by (fastforce simp add: rot-circle-cube-def)

lemma diff-divide-cancel:
  fixes  $z::real$  shows  $z \neq 0 \implies (a * z - a * (b * z)) / z = (a - a * b)$ 
  by (auto simp: field-simps)

lemma circle-cube-is-type-I:
  assumes  $0 < d$ 
  shows typeI-twoCube circle-cube
  unfolding typeI-twoCube-def
proof (intro exI conjI ballI)
  have f01: finite{-d/2,d/2}
    by simp
  show  $-d/2 < d/2$ 
    using assms by simp
  show  $(\lambda x. d * \text{sqrt}(1/4 - (x/d) * (x/d)))$  piecewise-C1-differentiable-on  $\{-d/2..d/2\}$ 
    using assms unfolding piecewise-C1-differentiable-on-def
    apply (intro exI conjI)
    apply (rule ballI refl f01 derivative-intros continuous-intros | simp)+
    apply (auto simp: field-simps)
    by sos
  show  $(\lambda x. -d * \text{sqrt}(1/4 - (x/d) * (x/d)))$  piecewise-C1-differentiable-on  $\{-$ 

```

```

d/2..d/2}
  using assms unfolding piecewise-C1-differentiable-on-def
  apply (intro exI conjI)
  apply (rule ballI refl f01 derivative-intros continuous-intros | simp)+
  apply (auto simp: field-simps)
  by sos
show - d * sqrt (1/4 - x / d * (x / d)) ≤ d * sqrt (1/4 - x / d * (x / d))
  if x ∈ {- d/2..d/2} for x
proof -
  have *: x^2 ≤ (d/2)^2
    using real-sqrt-le-iff that by fastforce
  show ?thesis
  apply (rule mult-right-mono)
  using assms * apply (simp-all add: divide-simps power2-eq-square)
  done
qed
qed (auto simp add: circle-cube-def divide-simps algebra-simps diff-divide-cancel)

```

```

lemma rot-circle-cube-is-type-II:
  shows typeII-twoCube rot-circle-cube
  using d-gt-0 swap-typeI-is-typeII circle-cube-is-type-I
  by (auto simp add: rot-circle-cube-def)

```

```

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

```

```

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

```

```

definition circle-right-edge where
  circle-right-edge = (1::int, λy. (d/2, 0))

```

```

definition circle-left-edge where
  circle-left-edge = (- 1::int, λy. (- (d/2), 0))

```

```

lemma circle-cube-boundary-explicit:
  boundary circle-cube = {circle-left-edge, circle-right-edge, circle-bot-edge, circle-top-edge}
  by (auto simp add: valid-two-cube-def boundary-def horizontal-boundary-def vertical-boundary-def
circle-cube-def
  circle-top-edge-def circle-bot-edge-def circle-cube-nice x-coord-def circle-y-def
circle-left-edge-def circle-right-edge-def)

```

```

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

```

```

definition rot-circle-left-edge where
  rot-circle-left-edge = (- 1::int, λt. (- 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*:

$boundary\ (rot-circle-cube) =$

$\{rot-circle-top-edge, rot-circle-bot-edge, rot-circle-right-edge, rot-circle-left-edge\}$

by (*auto simp add: rot-circle-cube-def valid-two-cube-def boundary-def horizontal-boundary-def vertical-boundary-def circle-cube-def*

rot-circle-right-edge-def rot-circle-left-edge-def x-coord-def circle-y-def rot-circle-top-edge-def rot-circle-bot-edge-def)

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

$vertical-boundary\ rot-circle-cube = \{rot-circle-right-edge, rot-circle-left-edge\}$

by (*auto simp add: rot-circle-cube-def valid-two-cube-def vertical-boundary-def circle-cube-def*

rot-circle-right-edge-def rot-circle-left-edge-def x-coord-def circle-y-def)

lemma *circ-left-edge-neq-top*:

$(-1::int, \lambda y::real. (-d/2, 0)) \neq (-1, \lambda x. ((x - 1/2) * d, d * sqrt(1/4 - (x - 1/2) * (x - 1/2))))$

by (*metis (no-types, lifting) add-diff-cancel-right' d-gt-0 mult.commute mult-cancel-left order-less-irrefl prod.inject*)

lemma *circle-cube-valid-two-cube: valid-two-cube (circle-cube)*

proof (*auto simp add: valid-two-cube-def boundary-def horizontal-boundary-def vertical-boundary-def circle-cube-def*)

have $iw: (-1::int, \lambda y::real. (-d/2, 0)) \neq (-1, \lambda x. ((x - 1/2) * d, d * sqrt(1/4 - (x - 1/2) * (x - 1/2))))$

using *d-gt-0* **apply** (*auto simp add: algebra-simps*)

by (*metis (no-types, hide-lams) add-diff-cancel-right' add-uminus-conv-diff cancel-comm-monoid-add-class.diff-cancel less-eq-real-def linorder-not-le mult.left-neutral prod.simps(1)*)

have $v: (1::int, \lambda y. (d/2, 0)) \neq (1, \lambda x. ((x - 1/2) * d, -(d * sqrt(1/4 - (x - 1/2) * (x - 1/2))))$

using *d-gt-0* **apply** (*auto simp add: algebra-simps*)

by (*metis (no-types, hide-lams) diff-0 equal-neg-zero mult-zero-left nonzero-mult-div-cancel-left order-less-irrefl prod.sel(1) times-divide-eq-right zero-neq-numeral*)

show $card\ \{(-1::int, \lambda y. (-d/2, 0)), (1, \lambda y. (d/2, 0)), (1, \lambda x. ((x - 1/2) * d, -(d * sqrt(1/4 - (x - 1/2) * (x - 1/2))))),$

$(-1, \lambda x. ((x - 1/2) * d, d * sqrt(1/4 - (x - 1/2) * (x - 1/2))))\} = 4$

using *iw v* **by** *auto*

qed

lemma *rot-circle-cube-valid-two-cube*:

shows *valid-two-cube rot-circle-cube*

using *valid-cube-valid-swap circle-cube-valid-two-cube d-gt-0 rot-circle-cube-def*

by force

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

lemma *circle-top-bot-edges-neq'* [simp]:
shows *circle-top-edge* \neq *circle-bot-edge*
by (simp add: *circle-top-edge-def circle-bot-edge-def*)

lemma *rot-circle-top-left-edges-neq* [simp]: *rot-circle-top-edge* \neq *rot-circle-left-edge*
apply (simp add: *rot-circle-left-edge-def rot-circle-top-edge-def x-coord-def*)
by (metis (mono-tags, hide-lams) *cancel-comm-monoid-add-class.diff-cancel d-gt-0 divide-eq-0-iff mult-zero-left order-less-irrefl prod.sel(2) zero-neq-numeral*)

lemma *rot-circle-bot-left-edges-neq* [simp]: *rot-circle-bot-edge* \neq *rot-circle-left-edge*
by (simp add: *rot-circle-left-edge-def rot-circle-bot-edge-def x-coord-def*)

lemma *rot-circle-top-right-edges-neq* [simp]: *rot-circle-top-edge* \neq *rot-circle-right-edge*
by (simp add: *rot-circle-right-edge-def rot-circle-top-edge-def x-coord-def*)

lemma *rot-circle-bot-right-edges-neq* [simp]: *rot-circle-bot-edge* \neq *rot-circle-right-edge*
apply (simp add: *rot-circle-right-edge-def rot-circle-bot-edge-def x-coord-def*)
by (metis (mono-tags, hide-lams) *cancel-comm-monoid-add-class.diff-cancel d-gt-0 divide-eq-0-iff mult-zero-left neg-0-equal-iff-equal order-less-irrefl prod.sel(2) zero-neq-numeral*)

lemma *rot-circle-right-top-edges-neq'* [simp]: *rot-circle-right-edge* \neq *rot-circle-left-edge*
by (simp add: *rot-circle-left-edge-def rot-circle-right-edge-def*)

lemma *rot-circle-left-bot-edges-neq* [simp]: *rot-circle-left-edge* \neq *rot-circle-top-edge*
apply (simp add: *rot-circle-top-edge-def rot-circle-left-edge-def*)
by (metis (no-types, hide-lams) *cancel-comm-monoid-add-class.diff-cancel d-gt-0 mult.commute mult-zero-right nonzero-mult-div-cancel-left order-less-irrefl prod.sel(2) times-divide-eq-right x-coord-def zero-neq-numeral*)

lemma *circle-right-top-edges-neq* [simp]: *circle-right-edge* \neq *circle-top-edge*

proof –

have *circle-right-edge* = (1, ($\lambda r::real. (d / 2, 0::real)$))
using *circle.circle-right-edge-def circle-axioms* by blast
then show ?thesis
using *circle.circle-top-edge-def circle-axioms* by auto

qed

lemma *circle-left-bot-edges-neq* [simp]: *circle-left-edge* \neq *circle-bot-edge*

proof –

have *circle-bot-edge* = (1, $\lambda r. (x\text{-coord } r * d, - d * \text{circle-y } (x\text{-coord } r))$)
using *circle.circle-bot-edge-def circle-axioms* by blast
then show ?thesis
by (simp add: *circle-left-edge-def*)

qed

lemma *circle-left-top-edges-neq* [simp]: *circle-left-edge* \neq *circle-top-edge*
proof –
 have $\exists r. ((r - 1 / 2) * d, d * \text{sqrt} (1 / 4 - (r - 1 / 2) * (r - 1 / 2))) \neq (- (d / 2), 0)$
 by (*metis circle-left-edge-neq-top*)
 then have $(\exists r. d * r \neq - (d / 2)) \vee (\exists r \text{ ra. } (x\text{-coord } (x\text{-coord-inv } r) * d, d * \text{circle-y } (x\text{-coord } (x\text{-coord-inv } r))) = (x\text{-coord } \text{ra} * d, d * \text{circle-y } (x\text{-coord } \text{ra})) \wedge d * \text{circle-y } r \neq 0)$
 by (*metis mult.commute*)
 then have $(\lambda r. (x\text{-coord } r * d, d * \text{circle-y } (x\text{-coord } r))) \neq (\lambda r. (- (d / 2), 0))$
 by (*metis mult.commute prod.simps(1) x-coord-inv-2*)
 then show ?thesis
 by (*simp add: circle-left-edge-def circle-top-edge-def*)
qed

lemma *circle-right-bot-edges-neq* [simp]: *circle-right-edge* \neq *circle-bot-edge*
 by (*smt Pair-inject circle-bot-edge-def d-gt-0 circle.circle-right-edge-def circle-axioms real-mult-le-cancel-iff1 x-coord-def*)

definition *circle-polar* where

circle-polar $t = ((d/2) * \cos (2 * \text{pi} * t), (d/2) * \sin (2 * \text{pi} * t))$

lemma *circle-polar-smooth*: (*circle-polar*) *C1-differentiable-on* $\{0..1\}$

proof –
 have *inj* $((*) (2 * \text{pi}))$
 by (*auto simp: inj-on-def*)
 then have $*$: $\bigwedge x. \text{finite } (\{0..1\} \cap (*) (2 * \text{pi}) - \{x\})$
 by (*simp add: finite-vimageI*)
 have $(\lambda t. ((d/2) * \cos (2 * \text{pi} * t), (d/2) * \sin (2 * \text{pi} * t)))$ *C1-differentiable-on* $\{0..1\}$
 by (*rule * derivative-intros*) +
 then show ?thesis
 apply (*rule eq-smooth-gen*)
 by (*simp add: circle-polar-def*)
qed

abbreviation *custom-arccos* $\equiv (\lambda x. (\text{if } -1 \leq x \wedge x \leq 1 \text{ then arccos } x \text{ else } (\text{if } x < -1 \text{ then } -x + \text{pi} \text{ else } 1 - x)))$

lemma *cont-custom-arccos*:

assumes $S \subseteq \{-1..1\}$

shows *continuous-on* S *custom-arccos*

proof –

have *continuous-on* $(\{-1..1\} \cup \{\})$ *custom-arccos*

by (*auto intro!: continuous-on-cases continuous-intros*)

with *assms* show ?thesis

using *continuous-on-subset* by *auto*

qed

```

lemma custom-arccos-has-deriv:
  assumes  $-1 < x < 1$ 
  shows DERIV custom-arccos  $x \rightarrow \text{inverse } (- \text{sqrt } (1 - x^2))$ 
proof -
  have  $x1: |x|^2 < 1^2$ 
    by (simp add: abs-less-iff abs-square-less-1 assms)
  show ?thesis
    apply (rule DERIV-inverse-function [where  $f = \text{cos}$  and  $a = -1$  and  $b = 1$ ])
      apply (rule  $x1$  derivative-eq-intros | simp add: sin-arccos) +
    using assms  $x1$  cont-custom-arccos [of  $\{-1 < .. < 1\}$ ]
      apply (auto simp: continuous-on-eq-continuous-at greaterThanLessThan-subseteq-atLeastAtMost-iff)
    done
qed

```

```

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 -
  let ? $\varphi = ((*) (1/\text{pi}) \circ \text{custom-arccos} \circ (\lambda t. 2 * x\text{-coord } (1 - t)))$ 
  let ?LHS1 = ( $\lambda x. (- (d * \text{sqrt } (1/4 - x\text{-coord } (1 - x) * x\text{-coord } (1 - x))),$ 
 $x\text{-coord } (1 - x) * d)$ )
  let ?RHS1 = ( $\lambda x. \text{if } x * 2 \leq 1 \text{ then } (d * \text{cos } (2 * \text{pi} * (2 * x/4 + 1/4))/2, d$ 
 $* \text{sin } (2 * \text{pi} * (2 * x/4 + 1/4))/2)$ 
     $\text{else } (d * \text{cos } (2 * \text{pi} * ((2 * x - 1)/4 + 1/2))/2, d * \text{sin } (2$ 
 $* \text{pi} * ((2 * x - 1)/4 + 1/2))/2) \circ ?\varphi$ )
  let ?LHS2 =  $\lambda x. ((x\text{-coord } (1 - x) * d, d * \text{sqrt } (1/4 - x\text{-coord } (1 - x) * x\text{-coord } (1 - x))))$ 
  let ?RHS2 = ( $\lambda x. \text{if } x * 2 \leq 1 \text{ then } (d * \text{cos } (2 * x * \text{pi}/2)/2, d * \text{sin } (2 * x$ 
 $* \text{pi}/2)/2) \text{ else } (d * \text{cos } (2 * \text{pi} * ((2 * x - 1)/4 + 1/4))/2, d * \text{sin } (2 * \text{pi} * ((2 * x - 1)/4 + 1/4))/2) \circ ?\varphi$ )

```

```

let ?LHS3 =  $\lambda x. (x\text{-coord } x * d, - (d * \text{sqrt } (1/4 - x\text{-coord } x * x\text{-coord } x)))$ 
let ?RHS3 = ( $\lambda x. \text{if } x * 2 \leq 1 \text{ then } (d * \cos (2 * \text{pi} * (2 * x/4 + 1/2)))/2, d$ 
*  $\sin (2 * \text{pi} * (2 * x/4 + 1/2))/2$ 
      else ( $d * \cos (2 * \text{pi} * ((2 * x - 1)/4 + 3/4))/2, d * \sin (2 * \text{pi}$ 
*  $((2 * x - 1)/4 + 3/4))/2$ ))
let ?LHS4 =  $\lambda x. (d * \text{sqrt } (1/4 - x\text{-coord } x * x\text{-coord } x), x\text{-coord } x * d)$ 
let ?RHS4 = ( $\lambda x. \text{if } x * 2 \leq 1 \text{ then } (d * \cos (2 * \text{pi} * (2 * x/4 + 3/4)))/2, d$ 
*  $\sin (2 * \text{pi} * (2 * x/4 + 3/4))/2$ 
      else ( $d * \cos ((2 * x - 1) * \text{pi}/2)/2, d * \sin ((2 * x - 1) *$ 
 $\text{pi}/2)/2$ ))
have phi-diff: ? $\varphi$  piecewise-C1-differentiable-on {0..1}
  unfolding piecewise-C1-differentiable-on-def
proof
  show continuous-on {0..1} ? $\varphi$ 
    unfolding x-coord-def
    by (intro continuous-on-compose cont-custom-arccos continuous-intros) auto
  have ? $\varphi$  C1-differentiable-on {0..1} - {0,1}
    unfolding x-coord-def piecewise-C1-differentiable-on-def C1-differentiable-on-def
  valid-path-def
  by (simp | rule has-vector-derivative-pair-within DERIV-image-chain derivative-eq-intros
  continuous-intros exI ballI conjI
    | force simp add: field-simps | sos)+
  then show  $\exists s. \text{finite } s \wedge ?\varphi \text{ C1-differentiable-on } \{0..1\} - s$ 
    by force
qed
have inj: inj ? $\varphi$ 
  apply (intro comp-inj-on inj-on-cases inj-on-arccos)
  apply (auto simp add: inj-on-def x-coord-def)
  using pi-ge-zero apply auto[1]
  apply (smt arccos)
  by (smt arccos-lbound)
then have fin:  $\bigwedge x. [0 \leq x; x \leq 1] \implies \text{finite } (? \varphi - \{x\})$ 
  by (simp add: finite-vimageI)
have ? $\varphi$  ' {0..1} = {0..1}
proof
  show ? $\varphi$  ' {0..1}  $\subseteq$  {0..1}
    by (auto simp add: x-coord-def divide-simps arccos-lbound arccos-bounded)
  have arccos (1 - 2 * ((1 - cos (x * pi))/2)) = x * pi if 0  $\leq$  x x  $\leq$  1 for x
    using that by (simp add: field-simps arccos-cos)
  then show {0..1}  $\subseteq$  ? $\varphi$  ' {0..1}
    apply (auto simp: x-coord-def o-def image-def)
    by (rule-tac x=(1 - cos (x * pi))/2 in bexI) auto
qed
then have bij-phi: bij-betw ? $\varphi$  {0..1} {0..1}
  unfolding bij-betw-def using inj inj-on-subset by blast
have phi01: ? $\varphi$  - ' {0..1}  $\subseteq$  {0..1}
  by (auto simp add: subset-iff x-coord-def divide-simps)
  {
    fix x::real assume x: 0  $\leq$  x x  $\leq$  1
  }

```



```

then have i:  $-1 \leq (2 * x - 1) (2 * x - 1) \leq 1$  by auto
have ii:  $\cos(\pi / 2 + \arccos(1 - x * 2)) = -\sin(\arccos(1 - x * 2))$ 
  using minus-sin-cos-eq[symmetric, where ?x =  $\arccos(1 - x * 2)$ ]
  by (auto simp add: add.commute)
have  $2 * \sqrt{x - x * x} = \sqrt{4 * x - 4 * x * x}$ 
  by (metis mult.assoc real-sqrt-four real-sqrt-mult right-diff-distrib)
also have ... =  $\sqrt{(1 - (2 * x - 1) * (2 * x - 1))}$ 
  by (simp add: algebra-simps)
finally have iii:  $2 * \sqrt{x - x * x} = \cos(\arcsin(2 * x - 1)) \wedge 2 * \sqrt{x - x * x} = \sin(\arccos(1 - 2 * x))$ 
  using arccos-minus[OF i] unfolding minus-diff-eq sin-pi-minus
  by (simp add: cos-arcsin i power2-eq-square sin-arccos)
then have 1: ?LHS1 x = ?RHS1 x
  using d-gt-0 i x apply (simp add: x-coord-def field-simps)
  apply (auto simp add: diff-divide-distrib add-divide-distrib right-diff-distrib
mult.commute ii)
  using cos-sin-eq[where ?x =  $-\arccos(1 - x * 2)$ ]
  by (simp add: cos-sin-eq[where ?x =  $-\arccos(1 - x * 2)$ ] right-diff-distrib)
have 2: ?LHS2 x = ?RHS2 x
  using x d-gt-0 iii by (auto simp add: x-coord-def diff-divide-distrib algebra-simps)
have a:  $\cos(\pi / 2 - \arccos(x * 2 - 1)) = \cos(\pi / 2 - \arccos(1 - x * 2))$ 
  by (smt arccos-minus arccos-cos2 arccos-lbound cos-arccos cos-ge-minus-one
cos-le-one i(1) i(2) pi-def pi-half)
have b:  $\cos(\arccos(1 - x * 2) + \pi * 3 / 2) = \cos((\pi / 2 - \arccos(x * 2 - 1)) + 2 * \pi)$ 
  using arccos-minus[OF i] by (auto simp add: mult.commute add.commute)
then have c: ... =  $\cos(\pi / 2 - \arccos(x * 2 - 1))$  using cos-periodic by
blast
have  $\cos(-\pi - \arccos(1 - x * 2)) = \cos(-(\pi + \arccos(1 - x * 2)))$ 
  by (auto simp add: minus-add[where b =  $\pi$  and a =  $\arccos(1 - x * 2)$ ,
symmetric])
moreover have ... =  $\cos((\pi + \arccos(1 - x * 2)))$ 
  using cos-minus by blast
moreover have ... =  $\cos(2 * \pi - \arccos(x * 2 - 1))$ 
  using arccos-minus[OF i] by (auto simp add: mult.commute add.commute)
moreover have ... =  $\cos(\arccos(x * 2 - 1))$ 
  using cos-2pi-minus by auto
ultimately have d:  $\cos(-\pi - \arccos(1 - x * 2)) = (x * 2 - 1)$ 
  using cos-arccos[OF i] mult.commute by metis
have cosm:  $\bigwedge x. \cos(x - \pi * 2) = \cos x$ 
  by (metis cos-periodic eq-diff-eq' mult.commute)
have 34: ?LHS3 x = (?RHS3  $\circ$  ? $\varphi$ ) x ?LHS4 x = (?RHS4  $\circ$  ? $\varphi$ ) x
  using d-gt-0 x a b c iii cos-periodic [of  $\pi / 2 - \arccos(x * 2 - 1)$ ]
  apply (auto simp add: x-coord-def algebra-simps diff-divide-distrib power2-eq-square)
  apply (auto simp add: sin-cos-eq cosm)
  using d apply (auto simp add: right-diff-distrib)
  by (smt cos-minus)
note 1 2 34

```

```

} note * = this
show ?P1 ?P2 ?P3 ?P4
  apply (auto simp add: subcube-def circle-bot-edge-def circle-top-edge-def
circle-polar-def reversepath-def
subpath-def joinpaths-def circle-y-def rot-circle-left-edge-def rot-circle-right-edge-def)
  unfolding reparam-def
  by (rule ballI exI conjI impI phi-diff bij-phi phi01 fin * | force simp add:
x-coord-def)+
qed

```

definition *circle-cube-boundary-to-polarcircle* **where**

```

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

```

lemma *circle-arcs-neq*:

```

assumes 0 ≤ k k ≤ 1 0 ≤ n n ≤ 1 n < k k + n < 1
shows subcube k m (1, circle-polar) ≠ subcube n q (1, circle-polar)
proof (simp add: subcube-def subpath-def circle-polar-def)
  have cos (2 * pi * k) ≠ cos(2 * pi * n)
  unfolding cos-eq
proof safe
  show False if 2 * pi * k = 2 * pi * n + 2 * m * pi m ∈ ℤ for m
  proof -
    have 2 * pi * (k - n) = 2 * m * pi
      using distrib-left that by (simp add: left-diff-distrib mult.commute)
    then have a: m = (k - n) by auto
    have [k - n] = 0
      using assms by (simp add: floor-eq-iff)
    then have k - n ∉ ℤ
      using assms by (auto simp only: frac-eq-0-iff[symmetric] frac-def)
    then show False using that a by auto
  qed
  show False if 2 * pi * k = - (2 * pi * n) + 2 * m * pi m ∈ ℤ for m
  proof -
    have 2 * pi * (k + n) = 2 * m * pi

```

```

    using that by (auto simp add: distrib-left)
  then have a:  $m = (k + n)$  by auto
  have  $\lfloor k + n \rfloor = 0$ 
    using assms by (simp add: floor-eq-iff)
  then have  $k + n \notin \mathbb{Z}$ 
    using Ints-def assms by force
  then show False using that a by auto
qed
qed
then have  $(\lambda x. (d * \cos (2 * \pi * ((m - k) * x + k))/2, d * \sin (2 * \pi * ((m - k) * x + k))/2)) 0 \neq (\lambda x. (d * \cos (2 * \pi * ((q - n) * x + n))/2, d * \sin (2 * \pi * ((q - n) * x + n))/2)) 0$ 
  using d-gt-0 by auto
  then show  $(\lambda x. (d * \cos (2 * \pi * ((m - k) * x + k))/2, d * \sin (2 * \pi * ((m - k) * x + k))/2)) \neq (\lambda x. (d * \cos (2 * \pi * ((q - n) * x + n))/2, d * \sin (2 * \pi * ((q - n) * x + n))/2))$ 
    by metis
qed

lemma circle-arcs-neq-2:
  assumes  $0 \leq k \leq 1$   $0 \leq n \leq 1$   $n < k$   $0 < n$  and kn12:  $1/2 < k + n$  and
   $k + n < 3/2$ 
  shows  $\text{subcube } k \ m \ (1, \text{circle-polar}) \neq \text{subcube } n \ q \ (1, \text{circle-polar})$ 
proof (simp add: subcube-def subpath-def circle-polar-def)
  have  $\sin (2 * \pi * k) \neq \sin (2 * \pi * n)$ 
  unfolding sin-eq
proof safe
  show False if  $m \in \mathbb{Z}$   $2 * \pi * k = 2 * \pi * n + 2 * m * \pi$  for  $m$ 
  proof -
    have  $2 * \pi * (k - n) = 2 * m * \pi$ 
      using that by (simp add: left-diff-distrib mult.commute)
    then have a:  $m = (k - n)$  by auto
    have  $\lfloor k - n \rfloor = 0$ 
      using assms by (simp add: floor-eq-iff)
    then have  $k - n \notin \mathbb{Z}$ 
      using assms by (auto simp only: frac-eq-0-iff[symmetric] frac-def )
    then show False using that a by auto
  qed
  show False if  $2 * \pi * k = - (2 * \pi * n) + (2 * m + 1) * \pi$   $m \in \mathbb{Z}$  for  $m$ 
  proof -
    have i:  $\bigwedge \pi. 0 < \pi \implies 2 * \pi * (k + n) = 2 * m * \pi + \pi \implies m = (k + n) - 1/2$ 
      by (sos (((((A<0 * A<1) * R<1) + ([1/2] * A=0))) & (((A<0 * A<1) * R<1) + ([~1/2] * A=0))))))
    have  $2 * \pi * (k + n) = 2 * m * \pi + \pi$ 
      using that by (simp add: algebra-simps)
    then have a:  $m = (k + n) - 1/2$  using i[OF pi-gt-zero] by fastforce
    have  $\lfloor k + n - 1/2 \rfloor = 0$ 
      using assms by (auto simp add: floor-eq-iff)
  qed

```

then have $k + n - 1/2 \notin \mathbb{Z}$
by (*metis Ints-cases add commute add.left-neutral add.diff-cancel-left'*
add.diff-eq kn12 floor-of-int of-int-0 order-less-irrefl)
then show *False* **using** *that a* **by** *auto*
qed
qed
then have $(\lambda x. (d * \cos (2 * \pi * ((m - k) * x + k))/2, d * \sin (2 * \pi * ((m - k) * x + k))/2)) 0 \neq (\lambda x. (d * \cos (2 * \pi * ((q - n) * x + n))/2, d * \sin (2 * \pi * ((q - n) * x + n))/2)) 0$
using *d-gt-0* **by** *auto*
then show $(\lambda x. (d * \cos (2 * \pi * ((m - k) * x + k))/2, d * \sin (2 * \pi * ((m - k) * x + k))/2)) \neq (\lambda x. (d * \cos (2 * \pi * ((q - n) * x + n))/2, d * \sin (2 * \pi * ((q - n) * x + n))/2))$
by *metis*
qed

lemma *circle-cube-is-only-horizontal-div-of-rot*:
shows *only-horizontal-division* (*boundary* (*circle-cube*)) {*rot-circle-cube*}
unfolding *only-horizontal-division-def*
proof (*rule exI [of - {}]*, *simp*, *intro conjI ballI*)
show *finite* (*boundary* *circle-cube*)
using *circle.circle-cube-boundary-explicit circle-axioms* **by** *auto*
show *boundary-chain* (*boundary* *circle-cube*)
by (*simp add: two-cube-boundary-is-boundary*)
show $\bigwedge x. x \in \text{boundary } \text{circle-cube} \implies \text{case } x \text{ of } (k, x) \implies \text{valid-path } x$
using *circle-cube-boundary-valid* **by** *blast*
let $?V = (\text{boundary } (\text{circle-cube}))$
let $?pi = \{\text{circle-left-edge}, \text{circle-right-edge}\}$
let $?pj = \{\text{rot-circle-top-edge}, \text{rot-circle-bot-edge}\}$
let $?f = \text{circle-cube-boundary-to-polarcircle}$
let $?one-chaini = \text{boundary } (\text{circle-cube}) - ?pi$
have *c: common-reparam-exists* $?V$ (*two-chain-vertical-boundary* {*rot-circle-cube*})
unfolding *common-reparam-exists-def*
proof (*intro exI conjI*)
let $?subdiv = \{(\text{subcube } 0 (1/4) (1, \text{circle-polar})),$
 $(\text{subcube } (1/4) (1/2) (1, \text{circle-polar})),$
 $(\text{subcube } (1/2) (3/4) (1, \text{circle-polar})),$
 $(\text{subcube } (3/4) 1 (1, \text{circle-polar}))\}$
show $(\forall (k, \gamma) \in ?subdiv. \gamma \text{ } C1\text{-differentiable-on } \{0..1\})$
using *subpath-smooth[OF circle-polar-smooth]* **by** (*auto simp add: subcube-def*)
have *1: finite* $?subdiv$ **by** *auto*
show *boundary-chain* $?subdiv$
by (*simp add: boundary-chain-def subcube-def*)
show *chain-reparam-chain'* (*boundary* (*circle-cube*) - $?pi$) $?subdiv$
unfolding *chain-reparam-chain'-def*
proof (*intro exI conjI impI*)
show $\bigcup (?f \text{ ' } ?one-chaini) = ?subdiv$
apply (*simp add: circle-cube-boundary-to-polarcircle-def circle-cube-boundary-explicit*)
using *circle-top-bot-edges-neq'* **by** *metis*

```

    let ?l = [subcube 0 (1/4) (1, circle-polar), subcube (1/4) (1/2) (1,
circle-polar)]
    have chain-reparam-weak-path (coeff-cube-to-path (circle-top-edge)) {subcube
0 (1/4) (1, circle-polar), subcube (1/4) (1/2) (1, circle-polar)}
    unfolding chain-reparam-weak-path-def
    proof (intro exI conjI)
    show valid-chain-list ?l
    by (auto simp add: subcube-def circle-top-edge-def x-coord-def circle-y-def
pathfinish-def pathstart-def
reversepath-def subpath-def joinpaths-def)
    show reparam (coeff-cube-to-path circle-top-edge) (rec-join ?l)
    using circ-top-edge-reparam-polar-circ-split by auto
    show distinct ?l
    apply simp
    apply (subst neq-commute)
    apply (simp add: circle-arcs-neq)
    done
qed auto
moreover have chain-reparam-weak-path (coeff-cube-to-path (circle-bot-edge))
{subcube (1/2) (3/4) (1, circle-polar), subcube (3/4) 1 (1, circle-polar)}
    unfolding chain-reparam-weak-path-def
    proof
    let ?l = [subcube (1/2) (3/4) (1, circle-polar), subcube (3/4) 1 (1,
circle-polar)]
    have a: valid-chain-list ?l
    by (auto simp add: subcube-def circle-top-edge-def x-coord-def circle-y-def
pathfinish-def pathstart-def
reversepath-def subpath-def joinpaths-def)
    have b: reparam (rec-join [circle-bot-edge]) (rec-join ?l)
    using circ-bot-edge-reparam-polar-circ-split by auto
    have c: subcube (3/4) 1 (1, circle-polar) ≠ subcube (1/2) (3/4) (1,
circle-polar)
    apply (rule circle-arcs-neq-2) using d-gt-0(1) by auto
    show set ?l = {subcube (1/2) (3/4) (1, circle-polar), subcube (3/4) 1 (1,
circle-polar)} ∧
distinct ?l ∧ reparam (coeff-cube-to-path (circle-bot-edge))
(rec-join ?l) ∧ valid-chain-list ?l ∧ ?l ≠ [] using a b c by auto
    qed
ultimately
    show (∀ cube ∈ ?one-chaini. chain-reparam-weak-path (rec-join [cube]) (?f
cube))
    by (auto simp add: circle-cube-boundary-to-polarcircle-def UNION-eq
circle-cube-boundary-explicit)
    show (∀ p ∈ ?one-chaini. ∀ p' ∈ ?one-chaini. p ≠ p' → ?f p ∩ ?f p' = {})
    using circle-arcs-neq circle-arcs-neq-2
    apply (auto simp add: circle-cube-boundary-to-polarcircle-def UNION-eq
circle-cube-boundary-explicit neq-commute d-gt-0)
    using circle-top-bot-edges-neq' d-gt-0 apply auto[1]
    apply (smt atLeastAtMost-iff divide-less-eq-1-pos zero-less-divide-1-iff)

```

```

    apply (smt atLeastAtMost-iff divide-less-eq-1-pos zero-less-divide-iff)
    apply (smt atLeastAtMost-iff divide-cancel-left divide-less-eq-1-pos field-sum-of-halves
zero-less-divide-1-iff)
  done
  show (∀ x ∈ ?one-chain1. finite (?f x))
  by (auto simp add: circle-cube-boundary-to-polarcircle-def circle-cube-boundary-explicit)
qed
show (∀ (k, γ) ∈ ?pi. point-path γ)
using d-gt-0 by (auto simp add: point-path-def circle-left-edge-def circle-right-edge-def)
let ?f = rot-circle-cube-boundary-to-polarcircle
let ?one-chain2.0 = two-chain-vertical-boundary {rot-circle-cube} - ?pj
  show chain-reparam-chain' (two-chain-vertical-boundary {rot-circle-cube} -
?pj) ?subdiv
  unfolding chain-reparam-chain'-def
proof (intro exI conjI)
  have rw: ?one-chain2.0 = {rot-circle-left-edge, rot-circle-right-edge}
  by (auto simp add: rot-circle-cube-vertical-boundary-explicit two-chain-vertical-boundary-def)
  show ∪ (?f ' ?one-chain2.0) = ?subdiv
    using rot-circle-right-top-edges-neq'
    by (auto simp add: rot-circle-cube-boundary-to-polarcircle-def rw)
  show (∀ cube ∈ ?one-chain2.0. chain-reparam-weak-path (rec-join [cube]) (?f
cube))
  proof (clarsimp simp add: rot-circle-cube-boundary-to-polarcircle-def rw, intro
conjI)
    let ?l = [subcube (1/4) (1/2) (1, circle-polar), subcube (1/2) (3/4) (1,
circle-polar)]
    show chain-reparam-weak-path (coeff-cube-to-path (rot-circle-left-edge))
{subcube (1/4) (1/2) (1, circle-polar), subcube (1/2) (3/4) (1, circle-polar)}
    unfolding chain-reparam-weak-path-def
  proof (intro exI conjI)
    show valid-chain-list ?l
    by (auto simp add: subcube-def pathfinish-def pathstart-def reversepath-def
subpath-def joinpaths-def)
    show reparam (coeff-cube-to-path rot-circle-left-edge) (rec-join ?l)
      using rot-circ-left-edge-reparam-polar-circ-split by auto
    show distinct ?l
    apply simp
    apply (subst neq-commute)
    apply (simp add: circle-arcs-neq)
  done
qed auto
  show chain-reparam-weak-path (coeff-cube-to-path (rot-circle-right-edge))
{subcube (3/4) 1 (1, circle-polar), subcube 0 (1/4) (1, circle-polar)}
  unfolding chain-reparam-weak-path-def
proof (intro exI conjI)
let ?l = [subcube (3/4) 1 (1, circle-polar), subcube 0 (1/4) (1, circle-polar)]
  show valid-chain-list ?l
  by (auto simp add: circle-polar-def subcube-def pathfinish-def pathstart-def
reversepath-def subpath-def joinpaths-def)

```

```

show reparam (coeff-cube-to-path rot-circle-right-edge) (rec-join ?l)
  using rot-circ-right-edge-reparam-polar-circ-split by auto
show distinct ?l
  by (simp add: circle-arcs-neq)
qed auto
qed
show ( $\forall p \in ?one-chain2.0. \forall p' \in ?one-chain2.0. p \neq p' \longrightarrow ?f p \cap ?f p' = \{\}$ )
  using circle-arcs-neq circle-arcs-neq-2
apply (auto simp add: rot-circle-cube-boundary-to-polarcircle-def neq-commute)
apply (metis add.right-neutral divide-less-eq-1-pos dual-order.order-iff-strict
num.distinct(1) one-less-numeral-iff prod.sel(1) prod.sel(2) semiring-norm(68) subcube-def
zero-less-divide-1-iff zero-less-numeral)
apply (smt field-sum-of-halves)
done
show ( $\forall x \in ?one-chain2.0. finite (?f x)$ )
  by (auto simp add: rot-circle-cube-boundary-to-polarcircle-def UNION-eq rw)
qed
show ( $\forall (k, \gamma) \in ?pj. point-path \gamma$ )
  using d-gt-0 by (auto simp add: point-path-def rot-circle-top-edge-def rot-circle-bot-edge-def)
qed
then show common-sudiv-exists (two-chain-vertical-boundary {rot-circle-cube})
(boundary circle-cube)  $\vee$ 
  common-reparam-exists (boundary circle-cube) (two-chain-vertical-boundary
{rot-circle-cube})
  by blast
qed

```

lemma GreenThm-cirlce:

```

assumes  $\forall twoC \in \{circle-cube\}. analytically-valid (cubeImage twoC) (\lambda x. F x \cdot$ 
i) j
   $\forall twoC \in \{rot-circle-cube\}. analytically-valid (cubeImage twoC) (\lambda x. F x \cdot j)$  i
shows integral (cubeImage (circle-cube)) ( $\lambda x. partial-vector-derivative (\lambda x. F x$ 
 $\cdot j)$  i x - partial-vector-derivative ( $\lambda x. F x \cdot i$ ) j x) =
  one-chain-line-integral F {i, j} (boundary (circle-cube))
proof(rule green-typeI-typeII-chain.GreenThm-typeI-typeII-divisible-region-finite-holes[of
(cubeImage (circle-cube)) i j F {circle-cube} {rot-circle-cube},
  OF - - - circle-cube-is-only-horizontal-div-of-rot -], auto)
show  $\wedge a b. (a, b) \in boundary circle-cube \implies valid-path b$  using circle-cube-boundary-valid
by auto
show green-typeI-typeII-chain (cubeImage circle-cube) i j F {circle-cube} {rot-circle-cube}
using assms
proof(auto simp add: green-typeI-typeII-chain-def green-typeI-chain-def green-typeII-chain-def
green-typeI-chain-axioms-def green-typeII-chain-axioms-def
  intro!: circle-cube-is-type-I rot-circle-cube-is-type-II d-gt-0 R2-axioms)
show gen-division (cubeImage circle-cube) {cubeImage circle-cube} by (simp
add: gen-division-def)
show gen-division (cubeImage (circle-cube)) ({cubeImage rot-circle-cube})
  using rot-circle-div-circle d-gt-0 by auto
show valid-two-chain {rot-circle-cube} valid-two-chain {circle-cube}

```

```

    apply (auto simp add: valid-two-chain-def)
    using rot-circle-cube-valid-two-cube circle-cube-valid-two-cube assms(1) by
auto
  qed
  show only-vertical-division (boundary (circle-cube)) {circle-cube}
  using twoChainVertDiv-of-itself[of {circle-cube}]
  apply(simp add: two-chain-boundary-def)
  using circle-cube-boundary-valid
  by auto
qed
end
end

```

2 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 ≤ 0 then - a else a)
  by (simp add: abs-if dual-order.order-iff-strict)

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

definition diamond-y-gen :: real ⇒ real where
  diamond-y-gen ≡ λt. 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 ≤ 0 0 ≤ b
  shows diamond-y-gen piecewise-C1-differentiable-on {a..b}
  unfolding piecewise-C1-differentiable-on-def diamond-y-gen-def
proof (intro conjI exI)
  show continuous-on {a..b} (λt. 1 / 2 - |t|)
  by (intro continuous-intros)
  show finite{0}
  by simp
  show (λt. 1 / 2 - |t|) C1-differentiable-on {a..b} - {0}
  by (intro derivative-intros) auto
qed

```


lemma *diamond-cube-gen-boundary-valid*:
assumes $(k,\gamma)\in\text{boundary}$ (*diamond-cube-gen*)
shows *valid-path* γ
using *assms*
proof (*auto simp add: valid-path-def boundary-def horizontal-boundary-def vertical-boundary-def diamond-cube-gen-def*)
have *rw2*: $(\lambda x. \text{diamond-y-gen } (x\text{-coord } x)) = \text{diamond-y-gen } o \text{ x-coord}$ **by** *auto*
note [*derivative-intros*] = *C1-differentiable-on-pair pair-prod-smooth-pw-smooth scale-piecewise-C1-differentiable-on piecewise-C1-differentiable-neg piecewise-C1-differentiable-compose diamond-y-gen-valid*
show $(\lambda x. (d * x\text{-coord } x, - (d * \text{diamond-y-gen } (x\text{-coord } x))))$ *piecewise-C1-differentiable-on* $\{0..1\}$
apply (*auto intro!: derivative-intros*) +
apply (*auto simp add: x-coord-smooth rw2*)
by (*auto intro!: derivative-intros simp add: x-coord-img x-coord-back-img C1-differentiable-imp-piecewise[OF x-coord-smooth]*) +
show $(\lambda x. (d * x\text{-coord } x, d * \text{diamond-y-gen } (x\text{-coord } x)))$ *piecewise-C1-differentiable-on* $\{0..1\}$
apply (*auto intro!: derivative-intros*) +
apply (*auto simp add: x-coord-smooth rw2*)
by (*auto intro!: derivative-intros simp add: x-coord-img x-coord-back-img C1-differentiable-imp-piecewise[OF x-coord-smooth]*) +
qed

definition *diamond-x where*
 $\text{diamond-x} \equiv \lambda t. (t - 1/2) * d$

definition *diamond-y where*
 $\text{diamond-y} \equiv \lambda t. d/2 - |t|$

definition *diamond-cube where*
 $\text{diamond-cube} = (\lambda(x,y). (\text{diamond-x } x, (2 * y - 1) * (\text{diamond-y } (\text{diamond-x } x))))$

definition *rot-diamond-cube where*
 $\text{rot-diamond-cube} = \text{prod.swap } o (\text{diamond-cube}) o \text{prod.swap}$

lemma *diamond-eq-characterisations*:
shows $\text{diamond-cube } (x,y) = \text{diamond-cube-gen } (x,y)$
by (*auto simp add: diamond-cube-def diamond-cube-gen-def diamond-x-def x-coord-def diamond-y-def diamond-y-gen-def d-gt-0 field-simps mult-le-0-iff abs-if split: if-split-asm*)

lemma *diamond-eq-characterisations-fun*: $\text{diamond-cube} = \text{diamond-cube-gen}$
using *diamond-eq-characterisations* **by** *auto*

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

```

have f0: finite {0}
  by simp
show ?P ?Q
  unfolding piecewise-C1-differentiable-on-def diamond-y-def
  by (fastforce intro!: f0 continuous-intros derivative-intros)+
qed

```

```

lemma diamond-cube-boundary-valid:
  assumes (k,γ) ∈ boundary (diamond-cube)
  shows valid-path γ
  using diamond-cube-gen-boundary-valid assms d-gt-0
  by(simp add: diamond-eq-characterisations-fun)

```

```

lemma diamond-cube-is-type-I:
  shows typeI-twoCube (diamond-cube)
  unfolding typeI-twoCube-def
proof (intro exI conjI ballI)
  show -d/2 < d/2
    using d-gt-0 by auto
  show  $\bigwedge x. x \in \{-d/2..d/2\} \implies -\text{diamond-y } x \leq \text{diamond-y } x$ 
    using diamond-y-def by auto
  have f0: finite {0}
    by simp
  show diamond-y piecewise-C1-differentiable-on  $\{-d/2..d/2\}$ 
    ( $\lambda x. -\text{diamond-y } x$ ) piecewise-C1-differentiable-on  $\{-d/2..d/2\}$ 
    unfolding diamond-y-def piecewise-C1-differentiable-on-def
    by (rule conjI exI f0 continuous-intros derivative-intros | force)+
  show diamond-cube =
    ( $\lambda(x, y). ((1 - x) * (-d/2) + x * (d/2),$ 
       $(1 - y) * -\text{diamond-y } ((1 - x) * (-d/2) + x * (d/2)) +$ 
       $y * \text{diamond-y } ((1 - x) * (-d/2) + x * (d/2))))$ )
    by (auto simp: diamond-cube-def diamond-x-def diamond-y-def divide-simps
      algebra-simps)
qed

```

```

lemma diamond-cube-valid-two-cube:
  shows valid-two-cube (diamond-cube)
  apply (auto simp add: valid-two-cube-def boundary-def horizontal-boundary-def
    vertical-boundary-def diamond-cube-def
      diamond-x-def card-insert-if)
  apply (metis (no-types, hide-lams) cancel-comm-monoid-add-class.diff-cancel
    d-gt-0 mult.commute mult-2 mult-zero-right order-less-irrefl prod.inject field-sum-of-halves)
  apply (metis (no-types, hide-lams) add-diff-cancel-right' d-gt-0 mult-cancel-left
    mult-zero-right order-less-irrefl prod.inject)
  done

```

```

lemma rot-diamond-cube-boundary-valid:
  assumes (k,γ) ∈ boundary (rot-diamond-cube)
  shows valid-path γ

```

using *d-gt-0 swap-valid-boundaries diamond-cube-boundary-valid*
using *assms diamond-cube-is-type-I rot-diamond-cube-def* **by** *fastforce*

lemma *rot-diamond-cube-is-type-II:*
shows *typeII-twoCube (rot-diamond-cube)*
using *d-gt-0 swap-typeI-is-typeII diamond-cube-is-type-I*
by (*auto simp add: rot-diamond-cube-def*)

lemma *rot-diamond-cube-valid-two-cube: valid-two-cube (rot-diamond-cube)*
using *valid-cube-valid-swap diamond-cube-valid-two-cube d-gt-0 rot-diamond-cube-def*
by *force*

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:*
 $boundary\ (diamond-cube) =$
 $\{diamond-top-edges,$
 $\quad diamond-bot-edges,$
 $\quad (- 1::int, \lambda y. (diamond-x\ 0, (2 * y - 1) * diamond-y\ (diamond-x\ 0))),$
 $\quad (1::int, \lambda y. (diamond-x\ 1, (2 * y - 1) * diamond-y\ (diamond-x\ 1)))\}$
by (*auto simp only: diamond-top-edges-def diamond-bot-edges-def valid-two-cube-def*
boundary-def horizontal-boundary-def vertical-boundary-def diamond-cube-def Un-iff
algebra-simps)

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

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

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

lemma *diamond-edges-are-valid:*
 $valid-path\ (snd\ (diamond-top-left-edge))$
 $valid-path\ (snd\ (diamond-top-right-edge))$
 $valid-path\ (snd\ (diamond-bot-left-edge))$
 $valid-path\ (snd\ (diamond-bot-right-edge))$

by (*auto simp add: valid-path-def diamond-top-left-edge-def diamond-bot-right-edge-def diamond-bot-left-edge-def diamond-top-right-edge-def diamond-x-def*)

definition *diamond-cube-boundary-to-subdiv* **where**

diamond-cube-boundary-to-subdiv ($\gamma::(\text{int} \times (\text{real} \Rightarrow \text{real} \times \text{real}))$) \equiv
if ($\gamma = \text{diamond-top-edges}$) *then*
 $\{\text{diamond-top-left-edge}, \text{diamond-top-right-edge}\}$
else if ($\gamma = \text{diamond-bot-edges}$) *then*
 $\{\text{diamond-bot-left-edge}, \text{diamond-bot-right-edge}\}$
else $\{\}$

lemma *rot-diam-edge-1*:

$(1::\text{int}, \lambda x::\text{real}. ((x::\text{real}) * (2 * \text{diamond-y} (\text{diamond-x } 0)) - 1 * \text{diamond-y} (\text{diamond-x } 0), \text{diamond-x } 0)) =$
 $(1, \lambda x. (x * (2 * \text{diamond-y} (\text{diamond-x } 0)) - (\text{diamond-y} (\text{diamond-x } 0)), \text{diamond-x } 0))$
by (*auto simp add: algebra-simps*)

definition *diamond-left-edges* **where**

diamond-left-edges = $(- 1, \lambda y. (- \text{diamond-y} (\text{diamond-x } y), \text{diamond-x } y))$

definition *diamond-right-edges* **where**

diamond-right-edges = $(1, \lambda y. (\text{diamond-y} (\text{diamond-x } y), \text{diamond-x } y))$

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

boundary (*rot-diamond-cube*) = $\{(1::\text{int}, \lambda x::\text{real}. ((2 * x - 1) * \text{diamond-y} (\text{diamond-x } 0), \text{diamond-x } 0)),$
 $(- 1, \lambda x. ((2 * x - 1) * \text{diamond-y} (\text{diamond-x } 1), \text{diamond-x } 1)),$
 $\text{diamond-left-edges}, \text{diamond-right-edges}\}$

proof –

have *boundary* (*rot-diamond-cube*) = $\{(1::\text{int}, \lambda x::\text{real}. ((2 * x - 1) * \text{diamond-y} (\text{diamond-x } 0), \text{diamond-x } 0)),$
 $(- 1, \lambda x. ((2 * x - 1) * \text{diamond-y} (\text{diamond-x } 1), \text{diamond-x } 1)),$
 $(- 1, \lambda y. ((2 * 0 - 1) * \text{diamond-y} (\text{diamond-x } y), \text{diamond-x } y)),$
 $(1, \lambda y. ((2 * 1 - 1) * \text{diamond-y} (\text{diamond-x } y), \text{diamond-x } y))\}$

by (*auto simp only: valid-two-cube-def boundary-def horizontal-boundary-def vertical-boundary-def rot-diamond-cube-def diamond-cube-def o-def swap-simp Un-iff*)

then show *?thesis*

by (*auto simp add: diamond-left-edges-def diamond-right-edges-def*)

qed

lemma *rot-diamond-cube-vertical-boundary-explicit*:

vertical-boundary (*rot-diamond-cube*) = $\{\text{diamond-left-edges}, \text{diamond-right-edges}\}$

proof –

have *vertical-boundary* (*rot-diamond-cube*) = $\{(- 1::\text{int}, \lambda y. ((2 * 0 - 1) * \text{diamond-y} (\text{diamond-x } y), \text{diamond-x } y)),$
 $(1, \lambda y. ((2 * 1 - 1) * \text{diamond-y} (\text{diamond-x } y), \text{diamond-x } y))\}$

by (*auto simp only: valid-two-cube-def boundary-def horizontal-boundary-def vertical-boundary-def rot-diamond-cube-def diamond-cube-def o-def swap-simp Un-iff*)
then show *?thesis*
by (*auto simp add: diamond-left-edges-def diamond-right-edges-def*)
qed

definition *rot-diamond-cube-boundary-to-subdiv* **where**
rot-diamond-cube-boundary-to-subdiv ($\gamma :: (\text{int} \times (\text{real} \Rightarrow \text{real} \times \text{real}))$) \equiv
if ($\gamma = \text{diamond-left-edges}$) *then* {*diamond-bot-left-edge*, *diamond-top-left-edge*}
else if ($\gamma = \text{diamond-right-edges}$) *then* {*diamond-bot-right-edge*, *diamond-top-right-edge*}
else {}

definition *diamond-boundaries-reparam-map* **where**
diamond-boundaries-reparam-map $\equiv \text{id}$

lemma *diamond-boundaries-reparam-map-bij*:
bij (*diamond-boundaries-reparam-map*)
by (*auto simp add: bij-def full-SetCompr-eq[symmetric] diamond-boundaries-reparam-map-def*)

lemma *diamond-bot-edges-neq-diamond-top-edges*:
diamond-bot-edges \neq *diamond-top-edges*
by (*simp add: diamond-bot-edges-def diamond-top-edges-def*)

lemma *diamond-top-left-edge-neq-diamond-top-right-edge*:
diamond-top-left-edge \neq *diamond-top-right-edge*
apply (*simp add: diamond-top-left-edge-def diamond-top-right-edge-def diamond-x-def diamond-y-def*)
using *d-gt-0*
apply (*auto simp add: algebra-simps divide-simps*)
by (*metis (no-types, hide-lams) diff-zero div-0 divide-divide-eq-right order-less-irrefl prod.inject field-sum-of-halves*)

lemma *neqs1*:
shows ($\lambda x. (\text{diamond-x } x, \text{diamond-y } (\text{diamond-x } x)) \neq (\lambda x. (\text{diamond-x } x, -\text{diamond-y } (\text{diamond-x } x)))$)
and ($\lambda y. (-\text{diamond-y } (\text{diamond-x } y), \text{diamond-x } y) \neq (\lambda y. (\text{diamond-y } (\text{diamond-x } y), \text{diamond-x } y))$)
and ($\lambda x. (\text{diamond-x } (x/2 + 1/2), \text{diamond-x } (x/2 + 1/2) - d/2) \neq (\lambda x. (\text{diamond-x } (x/2), -\text{diamond-x } (x/2) - d/2))$)
and ($\lambda x. (\text{diamond-x } (x/2 + 1/2), d/2 - \text{diamond-x } (x/2 + 1/2)) \neq (\lambda x. (\text{diamond-x } (x/2), \text{diamond-x } (x/2) + d/2))$)
and ($\lambda x. (\text{diamond-x } (x/2), -\text{diamond-x } (x/2) - d/2) \neq (\lambda x. (\text{diamond-x } (x/2 + 1/2), \text{diamond-x } (x/2 + 1/2) - d/2))$)
and ($\lambda x. (\text{diamond-x } (x/2), \text{diamond-x } (x/2) + d/2) \neq (\lambda x. (\text{diamond-x } (x/2 + 1/2), d/2 - \text{diamond-x } (x/2 + 1/2))$)
using *d-gt-0* **by** (*auto simp: diamond-x-def diamond-y-def dest: fun-cong [where $x = 1/2$]*)

lemma *neqs2*:

shows $(\lambda x. (\text{diamond-x } x, \text{diamond-y } (\text{diamond-x } x))) \neq (\lambda x. ((2 * x - 1) * \text{diamond-y } (\text{diamond-x } 1), \text{diamond-x } 1))$
and $(\lambda x. (\text{diamond-x } x, - \text{diamond-y } (\text{diamond-x } x))) \neq (\lambda x. ((2 * x - 1) * \text{diamond-y } (\text{diamond-x } 0), \text{diamond-x } 0))$
using *d-gt-0* **by** (*auto simp: diamond-x-def diamond-y-def dest: fun-cong [where x = 1]*)

lemma *diamond-cube-is-only-horizontal-div-of-rot:*

shows *only-horizontal-division* (*boundary* (*diamond-cube*)) {*rot-diamond-cube*}

unfolding *only-horizontal-division-def*

proof (*rule exI [of - {}]*, *simp*, *intro conjI ballI*)

show *finite* (*boundary diamond-cube*)

by (*simp add: boundary-def horizontal-boundary-def vertical-boundary-def*)

show *boundary-chain* (*boundary diamond-cube*)

by (*simp add: two-cube-boundary-is-boundary*)

show $\bigwedge x. x \in \text{boundary diamond-cube} \implies \text{case } x \text{ of } (k, x) \implies \text{valid-path } x$

using *diamond-cube-boundary-valid* **by** *blast*

let *?V* = (*boundary* (*diamond-cube*))

have *0: finite ?V*

by (*auto simp add: boundary-def horizontal-boundary-def vertical-boundary-def*)

have *1: boundary-chain ?V* **using** *two-cube-boundary-is-boundary* **by** *auto*

let *?subdiv* = {*diamond-top-left-edge*, *diamond-top-right-edge*, *diamond-bot-left-edge*, *diamond-bot-right-edge*}

let *?pi* = $\{(1::\text{int}, \lambda x. ((2 * x - 1) * \text{diamond-y } (\text{diamond-x } 0), \text{diamond-x } 0)),$
 $(- 1, \lambda x. ((2 * x - 1) * \text{diamond-y } (\text{diamond-x } 1), \text{diamond-x } 1))\}$

let *?pj* = $\{(-1::\text{int}, \lambda y. (\text{diamond-x } 0, (2 * y - 1) * \text{diamond-y } (\text{diamond-x } 0))),$
 $(1, \lambda y. (\text{diamond-x } 1, (2 * y - 1) * \text{diamond-y } (\text{diamond-x } 1)))\}$

let *?f* = *diamond-cube-boundary-to-subdiv*

have *2: common-sudiv-exists* (*two-chain-vertical-boundary* {*rot-diamond-cube*})
?V

unfolding *common-sudiv-exists-def*

proof (*intro exI conjI*)

show *chain-subdiv-chain* (*boundary* (*diamond-cube*) - *?pj*) *?subdiv*

unfolding *chain-subdiv-chain-character*

proof (*intro exI conjI*)

have *1: (boundary* (*diamond-cube*)) - *?pj* = {*diamond-top-edges*, *diamond-bot-edges*}

apply (*auto simp add: diamond-cube-boundary-explicit diamond-x-def diamond-top-edges-def diamond-bot-edges-def*)

apply (*metis* (*no-types*, *hide-lams*) *abs-zero cancel-comm-monoid-add-class.diff-cancel diamond-x-def diamond-y-def diff-0 minus-diff-eq mult commute mult-zero-right prod inject neqs2*)

by (*metis* (*no-types*, *hide-lams*) *cancel-comm-monoid-add-class.diff-cancel d-gt-0 divide-eq-0-iff linorder-not-le mult commute mult-zero-right order-refl prod.sel(1) zero-neq-numeral*)

then show $\bigcup (\text{?f } \cdot (\text{boundary } (\text{diamond-cube}) - \text{?pj})) = \text{?subdiv}$

by (*auto simp add: diamond-top-edges-def diamond-bot-edges-def diamond-cube-boundary-to-subdiv-def*)

have *chain-subdiv-path* (*reversepath* $(\lambda x. (\text{diamond-x } x, \text{diamond-y } (\text{diamond-x } x))))$)

```

      {(- 1::int, λx. (diamond-x(x/2), diamond-x(x/2) + d/2)),
       (- 1::int, λx. (diamond-x(x/2 + 1/2), d/2 - diamond-x(x/2 +
1/2)))}
  proof -
    let ?F = λx. (diamond-x(x/2 + 1/2), d/2 - diamond-x(x/2 + 1/2))
    let ?G = λx. (diamond-x(x/2), diamond-x(x/2) + d/2)
    have dist: distinct [(-1, ?F), (-1, ?G)]
      using d-gt-0 by (auto simp: diamond-x-def diamond-y-def dest: fun-cong)
    have rj: rec-join [(-1, ?F), (-1, ?G)] = reversepath (λx. (diamond-x x,
diamond-y (diamond-x x)))
      using d-gt-0 by (auto simp add: subpath-def diamond-x-def diamond-y-def
joinpaths-def reversepath-def algebra-simps divide-simps)
    have pathstart ?F = pathfinish ?G
      using d-gt-0 by (auto simp add: subpath-def diamond-x-def diamond-y-def
pathfinish-def pathstart-def)
    then have val: valid-chain-list [(-1, ?F), (-1, ?G)]
      by (auto simp add: join-subpaths-middle)
    show ?thesis
      using chain-subdiv-path.I [OF dist rj val]
      by (simp add: insert-commute)
  qed
  moreover have chain-subdiv-path (λx. (diamond-x x, - diamond-y (diamond-x
x)))
    {(1::int, λx. (diamond-x(x/2), - diamond-x(x/2) - d/2)),
     (1::int, λx. (diamond-x(x/2 + 1/2), diamond-x(x/2 +
1/2) - d/2))}
  proof -
    let ?F = λx. (diamond-x(x/2), - diamond-x(x/2) - d/2)
    let ?G = λx. (diamond-x(x/2 + 1/2), diamond-x(x/2 + 1/2) - d/2)
    have dist: distinct [(1, ?F), (1, ?G)]
      using d-gt-0 by (auto simp: diamond-x-def diamond-y-def dest: fun-cong)
    have rj: rec-join [(1, ?F), (1, ?G)] = (λx. (diamond-x x, - diamond-y
(diamond-x x)))
      using d-gt-0 by (auto simp add: subpath-def diamond-x-def diamond-y-def
joinpaths-def algebra-simps divide-simps)
    have pathfinish ?F = pathstart ?G
      using d-gt-0 by (auto simp add: subpath-def diamond-x-def diamond-y-def
pathfinish-def pathstart-def)
    then have val: valid-chain-list [(1, ?F), (1, ?G)]
      by (auto simp add: join-subpaths-middle)
    show ?thesis
      using chain-subdiv-path.I [OF dist rj val] by simp
  qed
  ultimately show **:
    ∀(k::int, γ)∈boundary (diamond-cube) - ?pj.
      if k = (1::int) then chain-subdiv-path γ (?f (k::int, γ))
      else chain-subdiv-path (reversepath γ) (?f (k::int, γ))
    ∀p∈boundary (diamond-cube) - ?pj. ∀p'∈boundary (diamond-cube) - ?pj.
      p ≠ p' → ?f p ∩ ?f p' = {}

```

$\forall x \in \text{boundary } (\text{diamond-cube}) - ?pj. \text{finite } (?f x)$
by (*simp-all add: diamond-cube-boundary-to-subdiv-def UNION-eq 1 diamond-top-edges-def diamond-bot-edges-def diamond-bot-left-edge-def diamond-bot-right-edge-def diamond-top-left-edge-def diamond-top-right-edge-def neqs1*)
qed
have *: $\bigwedge f. \bigcup (f \text{ ' } \{\text{rot-diamond-cube}\}) = f \text{ (rot-diamond-cube)}$ **by** *auto*
show *chain-subdiv-chain (two-chain-vertical-boundary {rot-diamond-cube} - ?pi) ?subdiv*
unfolding *chain-subdiv-chain-character two-chain-vertical-boundary-def **
proof (*intro exI conjI*)
let *?f = rot-diamond-cube-boundary-to-subdiv*
have 1: (*vertical-boundary (rot-diamond-cube) - ?pi*) = {*diamond-left-edges, diamond-right-edges*}
apply (*auto simp add: rot-diamond-cube-vertical-boundary-explicit diamond-left-edges-def diamond-right-edges-def*)
apply (*metis (no-types, hide-lams) add.inverse-inverse add-diff-cancel-right' diff-numeral-special(11) mult.left-neutral mult.right-neutral prod.inject neqs1(2) uminus-add-conv-diff*)
by (*metis (no-types, hide-lams) diff-0 mult.left-neutral mult.minus-left mult-zero-right prod.inject neqs1(2)*)
show $\bigcup (?f \text{ ' } (\text{vertical-boundary } (\text{rot-diamond-cube}) - ?pi)) = ?subdiv$
apply (*simp add: rot-diamond-cube-boundary-to-subdiv-def 1 UNION-eq subpath-def*)
apply (*auto simp add: set-eq-iff diamond-right-edges-def diamond-left-edges-def*)
done
have *chain-subdiv-path (reversepath ($\lambda y. (- \text{diamond-y } (\text{diamond-x } y), \text{diamond-x } y))$)*

$$\{(1, \lambda x. (\text{diamond-x}(x/2), - \text{diamond-x}(x/2) - d/2)), (-1, \lambda x. (\text{diamond-x}(x/2), \text{diamond-x}(x/2) + d/2))\}$$
proof -
let *?F = $\lambda x. (\text{diamond-x}(x/2), - \text{diamond-x}(x/2) - d/2)$*
let *?G = $\lambda x. (\text{diamond-x}(x/2), \text{diamond-x}(x/2) + d/2)$*
have *dist: distinct [(-1, ?G), (1::int, ?F)]*
using *d-gt-0* **by** *simp*
have *rj: rec-join [(-1, ?G), (1::int, ?F)] = reversepath ($\lambda y. (- \text{diamond-y } (\text{diamond-x } y), \text{diamond-x } y)$)*
using *d-gt-0* **by** (*auto simp add: subpath-def diamond-x-def diamond-y-def joinpaths-def reversepath-def algebra-simps divide-simps*)
have *pathstart ?G = pathstart ?F*
using *d-gt-0* **by** (*auto simp add: subpath-def diamond-x-def diamond-y-def pathstart-def*)
then have *val: valid-chain-list [(-1, ?G), (1::int, ?F)]*
by (*auto simp add: join-subpaths-middle*)
show *?thesis*
using *chain-subdiv-path.I [OF dist rj val]* **by** (*simp add: insert-commute*)
qed
moreover have *chain-subdiv-path ($\lambda y. (\text{diamond-y } (\text{diamond-x } y), \text{diamond-x } y)$)*

$$\{(1, \lambda x. (\text{diamond-x}(x/2 + 1/2), \text{diamond-x}(x/2 + 1/2))\}$$


```

- d/2)),
      (-1,  $\lambda x. (\text{diamond-}x(x/2 + 1/2), d/2 - \text{diamond-}x(x/2$ 
+ 1/2))))}
proof -
  let ?F =  $\lambda x. (\text{diamond-}x(x/2 + 1/2), d/2 - \text{diamond-}x(x/2 + 1/2))$ 
  let ?G =  $\lambda x. (\text{diamond-}x(x/2 + 1/2), \text{diamond-}x(x/2 + 1/2) - d/2)$ 
  have dist: distinct [(1::int, ?G), (-1, ?F)]
    by simp
  have rj: rec-join [(1::int, ?G), (-1, ?F)] = ( $\lambda y. (\text{diamond-}y (\text{diamond-}x$ 
y), diamond-x y)
    using d-gt-0 by (auto simp add: subpath-def diamond-x-def diamond-y-def
joinpaths-def reversepath-def algebra-simps divide-simps)
  have pathfinish ?F = pathfinish ?G
    using d-gt-0 by (auto simp add: subpath-def diamond-x-def diamond-y-def
pathfinish-def pathstart-def)
  then have val: valid-chain-list [(1::int, ?G), (-1, ?F)]
    by (auto simp add: join-subpaths-middle)
  show ?thesis
    using chain-subdiv-path.I [OF dist rj val] by simp
qed
ultimately show  $\forall (k, \gamma) \in \text{vertical-boundary } (\text{rot-diamond-cube}) - ?pi.$ 
  if k = 1 then chain-subdiv-path  $\gamma$  (?f (k,  $\gamma$ ))
  else chain-subdiv-path (reversepath  $\gamma$ ) (?f (k,  $\gamma$ ))
 $\forall p \in \text{vertical-boundary } (\text{rot-diamond-cube}) - ?pi.$ 
   $\forall p' \in \text{vertical-boundary } (\text{rot-diamond-cube}) - ?pi.$ 
   $p \neq p' \longrightarrow ?f p \cap ?f p' = \{\}$ 
 $\forall x \in \text{vertical-boundary } (\text{rot-diamond-cube}) - ?pi. \text{finite } (?f x)$ 
by (auto simp add: rot-diamond-cube-boundary-to-subdiv-def 1 diamond-left-edges-def
diamond-right-edges-def diamond-bot-left-edge-def diamond-bot-right-edge-def
diamond-top-left-edge-def diamond-top-right-edge-def neqs1)
qed
show ( $\forall (k::\text{int}, \gamma) \in ?\text{subdiv}. \text{valid-path } \gamma$ )
  by (simp add: diamond-edges-are-valid prod.case-eq-if set-eq-iff)
show boundary-chain ?subdiv
  by (auto simp add: boundary-chain-def diamond-top-left-edge-def diamond-top-right-edge-def
diamond-bot-left-edge-def diamond-bot-right-edge-def)
  show ( $\forall (k, \gamma) \in ?pi. \text{point-path } \gamma$ )
    using d-gt-0 by (auto simp add: point-path-def diamond-x-def diamond-y-def)
  show ( $\forall (k, \gamma) \in ?pj. \text{point-path } \gamma$ )
    using d-gt-0 by (auto simp add: point-path-def diamond-x-def diamond-y-def)
qed
show common-sudiv-exists (two-chain-vertical-boundary {rot-diamond-cube}) (boundary
diamond-cube)  $\vee$ 
  common-reparam-exists (boundary diamond-cube) (two-chain-vertical-boundary
  {rot-diamond-cube})
  using 0 1 2 diamond-cube-boundary-valid by auto
qed

```

abbreviation $rot-y\ t1\ t2 \equiv (t1 - 1/2) / (2 * diamond-y-gen\ (x-coord\ (rot-x\ t1\ t2))) + 1/2$

lemma *rot-y-ivl*:

assumes $(0::real) \leq x\ x \leq 1\ 0 \leq y\ y \leq 1$
shows $0 \leq rot-y\ x\ y \wedge rot-y\ x\ y \leq 1$
using *assms*
apply (*auto simp add: x-coord-def diamond-y-gen-def algebra-simps divide-simps linorder-class.not-le abs-if*)
using *mult-nonneg-nonneg apply fastforce*
using *dual-order.order-iff-strict apply fastforce*
apply (*sos* ((($(A < 0 * A < 1) * R < 1$) + ($(A <= 4 * (A < 1 * R < 1)) * (R < 1/2 * [1]^2)$) + ($(A <= 3 * (A < 0 * R < 1)) * (R < 1/2 * [1]^2)$) + ($(A <= 0 * (A <= 2 * (A <= 3 * R < 1)) * (R < 1 * [1]^2)$)))) & ((($(A < 0 * A < 1) * R < 1$) + ($(A <= 4 * (A < 0 * R < 1)) * (R < 1/3 * [1]^2)$) + ($(A <= 4 * (A < 0 * R < 1)) * (R < 1/3 * [1]^2)$) + ($(A <= 3 * (A <= 4 * R < 1)) * (R < 1/3 * [1]^2)$))))))
using *assms(1) mult-left-le-one-le apply blast*
using *affine-ineq apply fastforce+*
done

lemma *diamond-gen-eq-rot-diamond*:

assumes $0 \leq x\ x \leq 1\ 0 \leq y\ y \leq 1$
shows $(diamond-cube-gen\ (x, y)) = (rot-diamond-cube\ (rot-y\ x\ y, rot-x\ x\ y))$
proof
show $snd\ (diamond-cube-gen\ (x, y)) = snd\ (rot-diamond-cube\ (rot-y\ x\ y, rot-x\ x\ y))$
apply (*simp only: rot-diamond-cube-def diamond-eq-characterisations-fun*)
by (*auto simp add: diamond-cube-gen-def x-coord-def diamond-y-gen-def algebra-simps divide-simps*)
show $fst\ (diamond-cube-gen\ (x, y)) = fst\ (rot-diamond-cube\ (rot-y\ x\ y, rot-x\ x\ y))$
using *assms*
apply (*auto simp add: diamond-cube-gen-def rot-diamond-cube-def diamond-eq-characterisations-fun*)
apply (*auto simp add: x-coord-def diamond-y-gen-def algebra-simps divide-simps abs-if split: if-split-asm*)
apply *sos+*
done
qed

lemma *rot-diamond-eq-diamond-gen*:

assumes $0 \leq x\ x \leq 1\ 0 \leq y\ y \leq 1$
shows $rot-diamond-cube\ (x, y) = diamond-cube-gen\ (rot-x\ y\ x, rot-y\ y\ x)$
proof
show $fst\ (rot-diamond-cube\ (x, y)) = fst\ (diamond-cube-gen\ (rot-x\ y\ x, rot-y\ y\ x))$
apply (*simp only: rot-diamond-cube-def diamond-eq-characterisations-fun*)
by (*auto simp add: diamond-cube-gen-def x-coord-def diamond-y-gen-def algebra-simps divide-simps*)

```

show snd (rot-diamond-cube (x, y)) = snd (diamond-cube-gen (rot-x y x, rot-y
y x))
  using assms
  apply(auto simp add: diamond-cube-gen-def rot-diamond-cube-def diamond-eq-characterisations-fun)
  apply(auto simp add: x-coord-def diamond-y-gen-def algebra-simps divide-simps)
abs-if split: if-split-asm)
  apply sos+
  done
qed

```

```

lemma rot-img-eq: cubeImage (diamond-cube-gen) = cubeImage (rot-diamond-cube)
proof(auto simp add: cubeImage-def image-def cbox-def real-pair-basis)
  show  $\exists a \geq 0. a \leq 1 \wedge (\exists b \geq 0. b \leq 1 \wedge \text{diamond-cube-gen } (x, y) = \text{rot-diamond-cube}$ 
(a, b))

```

```

  if  $0 \leq x \ x \leq 1 \ 0 \leq y \ y \leq 1 \ (a, b) = \text{diamond-cube-gen } (x, y)$ 
  for a b x y

```

```

proof (intro exI conjI)

```

```

  let ?a = rot-y x y

```

```

  let ?b = rot-x x y

```

```

  show  $0 \leq ?a \ ?a \leq 1$ 

```

```

    using rot-y-ivl that by blast+

```

```

  show  $0 \leq ?b \ ?b \leq 1$ 

```

```

    using rot-x-ivl that by blast+

```

```

  show diamond-cube-gen (x, y) = rot-diamond-cube (?a, ?b)

```

```

    using that d-gt-0 diamond-gen-eq-rot-diamond by auto

```

```

qed

```

```

show  $\exists a \geq 0. a \leq 1 \wedge (\exists b \geq 0. b \leq 1 \wedge \text{rot-diamond-cube } (x, y) = \text{diamond-cube-gen}$ 
(a, b))

```

```

  if  $0 \leq x \ x \leq 1 \ 0 \leq y \ y \leq 1 \ (a, b) = \text{rot-diamond-cube } (x, y)$  for a b x y

```

```

proof (intro exI conjI)

```

```

  let ?a = rot-x y x

```

```

  let ?b = rot-y y x

```

```

  show  $0 \leq ?a \ ?a \leq 1$ 

```

```

    using rot-x-ivl that by blast+

```

```

  show  $0 \leq ?b \ ?b \leq 1$ 

```

```

    using rot-y-ivl that by blast+

```

```

  show rot-diamond-cube (x, y) = diamond-cube-gen (?a, ?b)

```

```

    using that d-gt-0 rot-diamond-eq-diamond-gen by auto

```

```

qed

```

```

qed

```

```

lemma rot-diamond-gen-div-diamond-gen:

```

```

  shows gen-division (cubeImage (diamond-cube-gen)) (cubeImage ‘ {rot-diamond-cube})

```

```

  using rot-img-eq by(auto simp add: gen-division-def)

```

```

lemma rot-diamond-gen-div-diamond:

```

```

  shows gen-division (cubeImage (diamond-cube)) (cubeImage ‘ {rot-diamond-cube})

```

```

  using rot-diamond-gen-div-diamond-gen

```

```

  by(simp only: diamond-eq-characterisations-fun)

```

lemma *GreenThm-diamond*:

assumes *analytically-valid* (*cubeImage* (*diamond-cube*)) ($\lambda x. F x \cdot i$) *j*
analytically-valid (*cubeImage* (*diamond-cube*)) ($\lambda x. F x \cdot j$) *i*

shows *integral* (*cubeImage* (*diamond-cube*)) ($\lambda x. \text{partial-vector-derivative}$ ($\lambda x. F x \cdot j$) *i* $x - \text{partial-vector-derivative}$ ($\lambda x. F x \cdot i$) *j* x) =
one-chain-line-integral *F* {*i*, *j*} (*boundary* (*diamond-cube*))

proof –

have *0*: $\forall (k, \gamma) \in \text{boundary}$ (*diamond-cube*). *valid-path* γ
using *diamond-cube-boundary-valid* *d-gt-0* **by** *auto*

have $\bigwedge \text{twoCube}. \text{twoCube} \in \{\text{diamond-cube}\} \implies \text{typeI-twoCube}$ *twoCube*
using *assms* *diamond-cube-is-type-I* **by** *auto*

moreover **have** *valid-two-chain* {*diamond-cube*}
using *assms*(1) *diamond-cube-valid-two-cube* *valid-two-chain-def* **by** *auto*

moreover **have** *gen-division* (*cubeImage* (*diamond-cube*)) (*cubeImage* ‘ {*diamond-cube*})
by (*simp* *add*: *gen-division-def*)

moreover **have** ($\forall \text{twoCube} \in \{\text{rot-diamond-cube}\}. \text{typeII-twoCube}$ *twoCube*)
using *assms* *rot-diamond-cube-is-type-II* **by** *auto*

moreover **have** *valid-two-chain* {*rot-diamond-cube*}
using *assms*(1) *rot-diamond-cube-valid-two-cube* *valid-two-chain-def* **by** *auto*

moreover **have** *gen-division* (*cubeImage* (*diamond-cube*)) (*cubeImage* ‘ {*rot-diamond-cube*})
using *rot-diamond-gen-div-diamond* **by** *auto*

moreover **have** *3*: *only-vertical-division* (*boundary* (*diamond-cube*)) {*diamond-cube*}
using *twoChainVertDiv-of-itself*[*of* {*diamond-cube*}] *diamond-cube-boundary-valid*

assms
by(*auto* *simp* *add*: *two-chain-boundary-def*)

moreover **have** *4*: $\forall \text{twoC} \in \{\text{diamond-cube}\}. \text{analytically-valid}$ (*cubeImage* *twoC*)
($\lambda x. F x \cdot i$) *j*
using *assms*
by *fastforce*

moreover **have** *5*: $\forall \text{twoC} \in \{\text{rot-diamond-cube}\}. \text{analytically-valid}$ (*cubeImage* *twoC*)
($\lambda x. F x \cdot j$) *i*
using *assms* *diamond-eq-characterisations-fun* *rot-img-eq* **by** *auto*

ultimately **show** *?thesis*
using *green-typeI-typeII-chain*. *GreenThm-typeI-typeII-divisible-region-finite-holes*[*of*
(*cubeImage* (*diamond-cube*)) *i* *j* *F* {*diamond-cube*} {*rot-diamond-cube*}, *OF - 0 3*
diamond-cube-is-only-horizontal-div-of-rot -]
R2-axioms

by(*auto* *simp* *add*: *green-typeI-typeII-chain-def* *green-typeI-chain-def* *green-typeII-chain-def*
green-typeI-chain-axioms-def *green-typeII-chain-axioms-def*)

qed
end
end