Young's Inequality for Increasing Functions

Lawrence C. Paulson

March 17, 2025

Abstract

Young's inequality states that

$$ab \le \int_0^a f(x)dx + \int_0^b f^{-1}(y)dy$$

where $a \ge 0$, $b \ge 0$ and f is strictly increasing and continuous. Its proof is formalised following the development by Cunningham and Grossman [1]. Their idea is to make the intuitive, geometric folklore proof rigorous by reasoning about step functions. The lack of the Riemann integral makes the development longer than one would like, but their argument is reproduced faithfully.

Contents

1 Young's Inequality for Increasing Functions **3**

1.1 Toward Young's inequality 3

1.2 Regular divisions 4

1.3 Special cases of Young's inequality 7

Acknowledgements The author was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council.

1 Young's Inequality for Increasing Functions

From the following paper: Cunningham, F., and Nathaniel Grossman. "On Young's Inequality." The American Mathematical Monthly 78, no. 7 (1971): 781–83. https://doi.org/10.2307/2318018

```
theory Youngs imports
  HOL−Analysis.Analysis

begin
```

1.1 Toward Young's inequality

```
lemma strict-mono-image-endpoints:
  fixes f :: 'a::linear-continuum-topology ⇒ 'b::linorder-topology
  assumes strict-mono-on {a..b} f and f: continuous-on {a..b} f and a ≤ b
  shows f ‘ {a..b} = {f a..f b}
proof
  show f ‘ {a..b} ⊆ {f a..f b}
    using assms(1) strict-mono-on-leD by fastforce
  show {f a..f b} ⊆ f ‘ {a..b}
    using assms IVT'[OF - - - f] by (force simp: Bex-def)
qed
```

Generalisations of the type of f are not obvious

```
lemma strict-mono-continuous-invD:
  fixes f :: real ⇒ real
  assumes sm: strict-mono-on {a..} f and contf: continuous-on {a..} f
    and fim: f ‘ {a..} = {f a..} and g: ⋀x. x ≥ a ⟹ g (f x) = x
  shows continuous-on {f a..} g
proof (clarsimp simp add: continuous-on-eq-continuous-within)
  fix y
  assume f a ≤ y
  then obtain u where u: y+1 = f u u ≥ a
    by (smt (verit, best) atLeast-iff fim imageE)
  have continuous-on {f a..y+1} g
  proof −
    obtain continuous-on {a..u} f  strict-mono-on {a..u} f
      using contf sm continuous-on-subset by (force simp add: strict-mono-on-def)
    moreover have continuous-on (f ‘ {a..u}) g
      using assms continuous-on-subset
      by (intro continuous-on-inv) fastforce+
    ultimately show ?thesis
      using strict-mono-image-endpoints [of - - f]
      by (simp add: strict-mono-image-endpoints u)
  qed
  then have ∗: continuous (at y within {f a..y+1}) g
    by (simp add: ‹f a ≤ y› continuous-on-imp-continuous-within)
  show continuous (at y within {f a..}) g
  proof (clarsimp simp add: continuous-within-topological Ball-def)
```

fix B
assume *open B* **and** $g\ y \in B$
with $*$ **obtain** A **where** A: *open A* $y \in A$ **and** $\bigwedge x.\ f\ a \leq x \wedge x \leq y+1 \Longrightarrow x \in A \longrightarrow g\ x \in B$
by (*force simp*: *continuous-within-topological*)
then have $\forall x \geq f\ a.\ x \in A \cap ball\ y\ 1 \longrightarrow g\ x \in B$
by (*smt* (*verit*, *ccfv-threshold*) *IntE dist-norm mem-ball real-norm-def*)
then show $\exists A.\ open\ A \wedge y \in A \wedge (\forall x \geq f\ a.\ x \in A \longrightarrow g\ x \in B)$
by (*metis Elementary-Metric-Spaces.open-ball Int-iff A centre-in-ball open-Int zero-less-one*)
qed
qed

1.2 Regular divisions

Our lack of the Riemann integral forces us to construct explicitly the step functions mentioned in the text.

definition $segment \equiv \lambda n\ k.\ \{real\ k\ /\ real\ n..(1 + k)\ /\ real\ n\}$

lemma *segment-nonempty*: $segment\ n\ k \neq \{\}$
by (*auto simp*: *segment-def divide-simps*)

lemma *segment-Suc*: $segment\ n\ `\ \{..<Suc\ k\} = insert\ \{k/n..(1 + real\ k)\ /\ n\}\ (segment\ n\ `\ \{..<k\})$
by (*simp add*: *segment-def lessThan-Suc*)

lemma *Union-segment-image*: $\bigcup\ (segment\ n\ `\ \{..<k\}) = (if\ k=0\ then\ \{\}\ else\ \{0..real\ k/real\ n\})$
proof (*induction k*)
case (*Suc k*)
then show *?case*
by (*simp add*: *divide-simps segment-Suc Un-commute ivl-disj-un-two-touch split*: *if-split-asm*)
qed (*auto simp*: *segment-def*)

definition $segments \equiv \lambda n.\ segment\ n\ `\ \{..<n\}$

lemma *card-segments* [*simp*]: $card\ (segments\ n) = n$
by (*simp add*: *segments-def segment-def card-image divide-right-mono inj-on-def*)

lemma *segments-0* [*simp*]: $segments\ 0 = \{\}$
by (*simp add*: *segments-def*)

lemma *Union-segments*: $\bigcup\ (segments\ n) = (if\ n=0\ then\ \{\}\ else\ \{0..1\})$
by (*simp add*: *segments-def Union-segment-image*)

definition $regular\text{-}division \equiv \lambda a\ b\ n.\ (image\ ((+)\ a \circ (*)\ (b-a)))\ `\ (segments\ n)$

lemma *translate-scale-01*:

assumes *a* ≤ *b*
shows (λ*x*. *a* + (*b* − *a*) ∗ *x*) ‘ {*0*..*1*} = {*a*..*b*::*real*}
using *closed-segment-real-eq* [*of a b*] *assms closed-segment-eq-real-ivl* **by** *auto*

lemma *finite-regular-division* [*simp*]: *finite* (*regular-division a b n*)
by (*simp add*: *regular-division-def segments-def*)

lemma *card-regular-division* [*simp*]:
assumes *a*<*b*
shows *card* (*regular-division a b n*) = *n*
proof −
have *inj-on* ((‘) ((+) *a* ∘ (∗) (*b* − *a*))) (*segments n*)
proof
fix *x y*
assume ((+) *a* ∘ (∗) (*b* − *a*)) ‘ *x* = ((+) *a* ∘ (∗) (*b* − *a*)) ‘ *y*
then have (+) (−*a*) ‘ ((+) *a* ∘ (∗) (*b* − *a*)) ‘ *x* = (+) (−*a*) ‘ ((+) *a* ∘ (∗) (*b* − *a*)) ‘ *y*
by *simp*
then have ((∗) (*b* − *a*)) ‘ *x* = ((∗) (*b* − *a*)) ‘ *y*
by (*simp add*: *image-comp*)
then have (∗) (*inverse*(*b* − *a*)) ‘ (∗) (*b* − *a*) ‘ *x* = (∗) (*inverse*(*b* − *a*)) ‘ (∗) (*b* − *a*) ‘ *y*
by *simp*
then show *x* = *y*
using *assms* **by** (*simp add*: *image-comp mult-ac*)
qed
then show *?thesis*
by (*metis card-image card-segments regular-division-def*)
qed

lemma *Union-regular-division*:
assumes *a* ≤ *b*
shows ⋃(*regular-division a b n*) = (*if n*=*0 then* {} *else* {*a*..*b*})
using *assms*
by (*auto simp*: *regular-division-def Union-segments translate-scale-01 simp flip*: *image-Union*)

lemma *regular-division-eqI*:
assumes *K*: *K* = {*a* + (*b*−*a*)∗(*real k* / *n*) .. *a* + (*b*−*a*)∗((*1* + *real k*) / *n*)}
and *a*<*b k* < *n*
shows *K* ∈ *regular-division a b n*
unfolding *regular-division-def segments-def image-comp*
proof
have *K* = (λ*x*. (*b*−*a*) ∗ *x* + *a*) ‘ {*real k* / *real n*..(*1* + *real k*) / *real n*}
using *K* ‹*a*<*b*› **by** (*simp add*: *image-affinity-atLeastAtMost divide-simps*)
then show *K* = ((‘) ((+) *a* ∘ (∗) (*b* − *a*)) ∘ *segment n*) *k*
by (*simp add*: *segment-def add.commute*)
qed (*use assms* **in** *auto*)

```
lemma regular-divisionE:
  assumes K ∈ regular-division a b n a<b
  obtains k where k<n K = {a + (b−a)∗(real k / n) .. a + (b−a)∗((1 + real k)
/ n)}
proof −
  have eq: (λx. a + (b − a) ∗ x) = (λx. a + x) ∘ (λx. (b − a) ∗ x)
    by (simp add: o-def)
  obtain k where k<n K = ((λx. a+x) ∘ (λx. (b−a) ∗ x)) ‘ {k/n .. (1 + real k)
/ n}
    using assms by (auto simp: regular-division-def segments-def segment-def)
  with that ‹a<b› show ?thesis
    unfolding image-comp [symmetric]  by auto
qed

lemma regular-division-division-of:
  assumes a < b n>0
  shows (regular-division a b n) division-of {a..b}
proof (rule division-ofI)
  show finite (regular-division a b n)
    by (simp add: regular-division-def segments-def)
  show §: ⋃ (regular-division a b n) = {a..b}
    using Union-regular-division assms by simp
  fix K
  assume K: K ∈ regular-division a b n
  then obtain k where Keq: K = {a + (b−a)∗(k/n) .. a + (b−a)∗((1 + real k)
/ n)}
    using ‹a<b› regular-divisionE by meson
  show K ⊆ {a..b}
    using K Union-regular-division ‹n>0› by (metis Union-upper §)
  show K ≠ {}
    using K by (auto simp: regular-division-def segment-nonempty segments-def)
  show ∃ a b. K = cbox a b
    by (metis K ‹a<b› box-real(2) regular-divisionE)
  fix K′
  assume K′: K′ ∈ regular-division a b n and K ≠ K′
  then obtain k′ where Keq′: K′ = {a + (b−a)∗(k′/n) .. a + (b−a)∗((1 + real
k′) / n)}
    using K ‹a<b› regular-divisionE by meson
  consider 1 + real k ≤ k′ | 1 + real k′ ≤ k
    using Keq Keq′ ‹K ≠ K′› by force
  then show interior K ∩ interior K′ = {}
  proof cases
    case 1
    then show ?thesis
      by (simp add: Keq Keq′ min-def max-def divide-right-mono assms)
  next
    case 2
    then have interior K′ ∩ interior K = {}
      by (simp add: Keq Keq′ min-def max-def divide-right-mono assms)
```

then show *?thesis*
by (*simp add*: *inf-commute*)
qed
qed

1.3 Special cases of Young's inequality

lemma *weighted-nesting-sum*:
fixes $g :: nat \Rightarrow 'a::comm\text{-}ring\text{-}1$
shows $(\sum k<n.\ (1 + of\text{-}nat\ k) * (g\ (Suc\ k) - g\ k)) = of\text{-}nat\ n * g\ n - (\sum i<n.\ g\ i)$
by (*induction n*) (*auto simp*: *algebra-simps*)

theorem *Youngs-exact*:
fixes $f :: real \Rightarrow real$
assumes *sm*: *strict-mono-on* $\{0..\}$ *f* **and** *cont*: *continuous-on* $\{0..\}$ *f* **and** *a*: $a \geq 0$
and *f*: $f\ 0 = 0\ f\ a = b$
and *g*: $\bigwedge x.\ [\![0 \leq x;\ x \leq a]\!] \Longrightarrow g\ (f\ x) = x$
shows $a*b = integral\ \{0..a\}\ f + integral\ \{0..b\}\ g$
proof (*cases* $a=0$)
case *False*
with ‹$a \geq 0$› **have** $a > 0$ **by** *linarith*
then have $b \geq 0$
by (*smt* (*verit*, *best*) *atLeast-iff f sm strict-mono-onD*)
have *sm-0a*: *strict-mono-on* $\{0..a\}$ *f*
by (*metis atLeastAtMost-iff atLeast-iff sm strict-mono-on-def*)
have *cont-0a*: *continuous-on* $\{0..a\}$ *f*
using *cont continuous-on-subset* **by** *fastforce*
with *sm-0a* **have** *continuous-on* $\{0..b\}$ *g*
by (*metis a atLeastAtMost-iff compact-Icc continuous-on-inv f g strict-mono-image-endpoints*)
then have *intgb-g*: *g integrable-on* $\{0..b\}$
using *integrable-continuous-interval* **by** *blast*
have *intgb-f*: *f integrable-on* $\{0..a\}$
using *cont-0a integrable-continuous-real* **by** *blast*

have *f-iff* [*simp*]: $f\ x < f\ y \longleftrightarrow x < y\ f\ x \leq f\ y \longleftrightarrow x \leq y$
if $x \geq 0\ y \geq 0$ **for** *x y*
using *that* **by** (*smt* (*verit*, *best*) *atLeast-iff sm strict-mono-onD*)+
have *fim*: $f\ `\ \{0..a\} = \{0..b\}$
by (*simp add*: ‹$a \geq 0$› *cont-0a strict-mono-image-endpoints strict-mono-on-def f*)
have *uniformly-continuous-on* $\{0..a\}$ *f*
using *compact-uniformly-continuous cont-0a* **by** *blast*
then obtain *del* **where** *del-gt0*: $\bigwedge e.\ e>0 \Longrightarrow del\ e > 0$
and *del*: $\bigwedge e\ x\ x'.\ [\![|x'-x| < del\ e;\ e>0;\ x \in \{0..a\};\ x' \in \{0..a\}]\!] \Longrightarrow |f\ x' - f\ x| < e$
unfolding *uniformly-continuous-on-def dist-real-def* **by** *metis*

have ∗: |*a* ∗ *b* − *integral* {*0..a*} *f* − *integral* {*0..b*} *g*| < *2*∗ε **if** ε > *0* **for** ε
proof −
define δ **where** δ = *min a* (*del* (ε/*a*)) / *2*
have δ > *0* δ ≤ *a*
using ‹*a* > *0*› ‹ε > *0*› *del-gt0* **by** (*auto simp*: δ*-def*)
define *n* **where** *n* ≡ *nat*⌊*a* / δ⌋
define *a-seg* **where** *a-seg* ≡ λ*u*::*real*. *u* ∗ *a*/*n*
have *n* > *0*
using ‹*a* > *0*› ‹δ > *0*› ‹δ ≤ *a*› **by** (*simp add*: *n-def*)
have *a-seg-ge-0* [*simp*]: *a-seg x* ≥ *0* ⟷ *x* ≥ *0*
and *a-seg-le-a* [*simp*]: *a-seg x* ≤ *a* ⟷ *x* ≤ *n* **for** *x*
using ‹*n* > *0*› ‹*a* > *0*› **by** (*auto simp*: *a-seg-def zero-le-mult-iff divide-simps*)
have *a-seg-le-iff* [*simp*]: *a-seg x* ≤ *a-seg y* ⟷ *x* ≤ *y*
and *a-seg-less-iff* [*simp*]: *a-seg x* < *a-seg y* ⟷ *x* < *y* **for** *x y*
using ‹*n* > *0*› ‹*a* > *0*› **by** (*auto simp*: *a-seg-def zero-le-mult-iff divide-simps*)
have *strict-mono a-seg*
by (*simp add*: *strict-mono-def*)
have *a-seg-eq-a-iff*: *a-seg x* = *a* ⟷ *x*=*n* **for** *x*
using ‹*0* < *n*› ‹*a* > *0*› **by** (*simp add*: *a-seg-def nonzero-divide-eq-eq*)
have *fa-eq-b*: *f* (*a-seg n*) = *b*
using *a-seg-eq-a-iff f* **by** *fastforce*

have *a*/*d* < *real-of-int* ⌊*a* ∗ *2* / *min a d*⌋ **if** *d*>*0* **for** *d*
by (*smt* (*verit*) ‹*0* < δ› ‹δ ≤ *a*› *add-divide-distrib divide-less-eq-1-pos floor-eq-iff that*)
then have *an-less-del*: *a*/*n* < *del* (ε/*a*)
using ‹*a* > *0*› ‹ε > *0*› *del-gt0* **by** (*simp add*: *n-def* δ*-def field-simps*)

define *lower* **where** *lower* ≡ λ*x*. *a-seg*⌊(*real n* ∗ *x*) / *a*⌋
define *f1* **where** *f1* ≡ *f* ∘ *lower*
have *f1-lower*: *f1 x* ≤ *f x* **if** *0* ≤ *x x* ≤ *a* **for** *x*
proof −
have *lower x* ≤ *x*
using ‹*n* > *0*› *floor-divide-lower* [*OF* ‹*a* > *0*›]
by (*auto simp*: *lower-def a-seg-def field-simps*)
moreover have *lower x* ≥ *0*
unfolding *lower-def* **using** ‹*n* > *0*› ‹*a* ≥ *0*› ‹*0* ≤ *x*› **by** *force*
ultimately show *?thesis*
using *sm strict-mono-on-leD* **by** (*fastforce simp add*: *f1-def*)
qed
define *upper* **where** *upper* ≡ λ*x*. *a-seg*⌈*real n* ∗ *x* / *a*⌉
define *f2* **where** *f2* ≡ *f* ∘ *upper*
have *f2-upper*: *f2 x* ≥ *f x* **if** *0* ≤ *x x* ≤ *a* **for** *x*
proof −
have *x* ≤ *upper x*
using ‹*n* > *0*› *ceiling-divide-upper* [*OF* ‹*a* > *0*›] **by** (*simp add*: *upper-def a-seg-def field-simps*)
then show *?thesis*
using *sm strict-mono-on-leD* ‹*0* ≤ *x*› **by** (*force simp*: *f2-def*)

qed
let *?𝒟* = *regular-division 0 a n*
have *div*: *?𝒟 division-of* {*0..a*}
using ‹$a > 0$› ‹$n > 0$› *regular-division-division-of zero-less-nat-eq* **by** *presburger*

have *int-f1-D*: (*f1 has-integral f*(*Inf K*) ∗ (*a*/*n*)) *K*
and *int-f2-D*: (*f2 has-integral f*(*Sup K*) ∗ (*a*/*n*)) *K* **and** *less*: |*f*(*Sup K*) − *f*(*Inf K*)| < *ε*/*a*
if *K*∈*?𝒟* **for** *K*
proof −
from *regular-divisionE* [*OF that*] ‹$a > 0$›
obtain *k* **where** *k*<*n* **and** *k*: *K* = {*a-seg*(*real k*)..*a-seg*(*Suc k*)}
by (*auto simp*: *a-seg-def mult.commute*)
define *u* **where** *u* ≡ *a-seg k*
define *v* **where** *v* ≡ *a-seg* (*Suc k*)
have $u < v$ $0 \le u$ $0 \le v$ $u \le a$ $v \le a$ **and** *Kuv*: *K* = {*u..v*}
using ‹$n > 0$› ‹$k < n$› ‹$a > 0$› **by** (*auto simp*: *k u-def v-def divide-simps*)
have *InfK*: *Inf K* = *u* **and** *SupK*: *Sup K* = *v*
using *Kuv* ‹$u < v$› **apply** *force*
using ‹$n > 0$› ‹$a > 0$› **by** (*auto simp*: *divide-right-mono k u-def v-def*)
have *f1*: *f1 x* = *f* (*Inf K*) **if** *x* ∈ *K* − {*v*} **for** *x*
proof −
have *x* ∈ {*u..*<*v*}
using *that Kuv atLeastLessThan-eq-atLeastAtMost-diff* **by** *blast*
then have ⌊*real-of-int n* ∗ *x* / *a*⌋ = *int k*
using ‹$n > 0$› ‹$a > 0$› **by** (*simp add*: *field-simps u-def v-def a-seg-def floor-eq-iff*)
then show *?thesis*
by (*simp add*: *InfK f1-def lower-def a-seg-def mult.commute u-def*)
qed
have ((*λx. f* (*Inf K*)) *has-integral* (*f* (*Inf K*) ∗ (*a*/*n*))) *K*
using *has-integral-const-real* [*of f* (*Inf K*) *u v*]
‹$n > 0$› ‹$a > 0$› **by** (*simp add*: *Kuv field-simps a-seg-def u-def v-def*)
then show (*f1 has-integral* (*f* (*Inf K*) ∗ (*a*/*n*))) *K*
using *has-integral-spike-finite-eq* [*of* {*v*} *K λx. f* (*Inf K*) *f1*] *f1* **by** *simp*
have *f2*: *f2 x* = *f* (*Sup K*) **if** *x* ∈ *K* − {*u*} **for** *x*
proof −
have *x* ∈ {*u*<*..v*}
using *that Kuv greaterThanAtMost-eq-atLeastAtMost-diff* **by** *blast*
then have ⌈*x* ∗ *real-of-int n* / *a*⌉ = *1* + *int k*
using ‹$n > 0$› ‹$a > 0$› **by** (*simp add*: *field-simps u-def v-def a-seg-def ceiling-eq-iff*)
then show *?thesis*
by (*simp add*: *mult.commute f2-def upper-def a-seg-def SupK v-def*)
qed
have ((*λx. f* (*Sup K*)) *has-integral* (*f* (*Sup K*) ∗ (*a*/*n*))) *K*
using ‹$n > 0$› ‹$a > 0$› *has-integral-const-real* [*of f* (*Sup K*) *u v*]
by (*simp add*: *Kuv field-simps u-def v-def a-seg-def*)

then show (*f2 has-integral* (*f* (*Sup K*) ∗ (*a*/*n*))) *K*
using *has-integral-spike-finite-eq* [*of* {*u*} *K* λ*x*. *f* (*Sup K*) *f2*] *f2* **by** *simp*
have |*v* − *u*| < *del* (ε/*a*)
using ‹*n* > *0*› ‹*a* > *0*› **by** (*simp add*: *v-def u-def a-seg-def field-simps an-less-del*)
then have |*f v* − *f u*| < ε/*a*
using ‹ε > *0*› ‹*a* > *0*› ‹*0* ≤ *u*› ‹*u* ≤ *a*› ‹*0* ≤ *v*› ‹*v* ≤ *a*›
by (*intro del*) *auto*
then show |*f*(*Sup K*) − *f*(*Inf K*)| < ε/*a*
using *InfK SupK* **by** *blast*
qed

have *int-21-D*: ((λ*x*. *f2 x* − *f1 x*) *has-integral* (*f*(*Sup K*) − *f*(*Inf K*)) ∗ (*a*/*n*)) *K* **if** *K*∈*?𝒟* **for** *K*
using *that has-integral-diff* [*OF int-f2-D int-f1-D*] **by** (*simp add*: *algebra-simps*)

have *D-ne*: *?𝒟* ≠ {}
by (*metis* ‹*0* < *a*› ‹*n* > *0*› *card-gt-0-iff card-regular-division*)
have *f12*: ((λ*x*. *f2 x* − *f1 x*) *has-integral* (∑*K*∈*?𝒟*. (*f*(*Sup K*) − *f*(*Inf K*)) ∗ (*a*/*n*))) {*0*..*a*}
by (*intro div int-21-D has-integral-combine-division*)
moreover have (∑*K*∈*?𝒟*. (*f*(*Sup K*) − *f*(*Inf K*)) ∗ (*a*/*n*)) < ε
proof −
have (∑*K*∈*?𝒟*. (*f*(*Sup K*) − *f*(*Inf K*)) ∗ (*a*/*n*)) ≤ (∑*K*∈*?𝒟*. |*f*(*Sup K*) − *f*(*Inf K*)| ∗ (*a*/*n*))
using ‹*n* > *0*› ‹*a* > *0*›
by (*smt* (*verit*) *divide-pos-pos of-nat-0-less-iff sum-mono zero-le-mult-iff*)
also have . . . < (∑*K*∈*?𝒟*. ε/*n*)
using ‹*n* > *0*› ‹*a* > *0*› *less*
by (*intro sum-strict-mono finite-regular-division D-ne*) (*simp add*: *field-simps*)
also have . . . = ε
using ‹*n* > *0*› ‹*a* > *0*› **by** *simp*
finally show *?thesis* **.**
qed
ultimately have *f2-near-f1*: *integral* {*0*..*a*} (λ*x*. *f2 x* − *f1 x*) < ε
by (*simp add*: *integral-unique*)

define *yidx* **where** *yidx* ≡ λ*y*. *LEAST k*. *y* < *f* (*a-seg* (*Suc k*))
have *fa-yidx-le*: *f* (*a-seg* (*yidx y*)) ≤ *y* **and** *yidx-gt*: *y* < *f* (*a-seg* (*Suc* (*yidx y*)))
if *y* ∈ {*0*..*b*} **for** *y*
proof −
obtain *x* **where** *x*: *f x* = *y x* ∈ {*0*..*a*}
using *Topological-Spaces.IVT′* [*OF* - - - *cont-0a*] *assms*
by (*metis* ‹*y* ∈ {*0*..*b*}› *atLeastAtMost-iff*)
define *k* **where** *k* ≡ *nat* ⌊*x*/*a* ∗ *n*⌋
have *x-lims*: *a-seg k* ≤ *x x* < *a-seg* (*Suc k*)
using ‹*n* > *0*› ‹*0* < *a*› *floor-divide-lower floor-divide-upper* [*of a x*∗*n*] *x*
by (*auto simp*: *k-def a-seg-def field-simps*)

with *that x* **obtain** *f-lims*: *f* (*a-seg k*) ≤ *y y* < *f* (*a-seg* (*Suc k*))
  using *strict-mono-onD* [*OF sm*] **by** *force*
then have *a-seg* (*yidx y*) ≤ *a-seg k*
  by (*simp add*: *Least-le* ‹*strict-mono a-seg*› *strict-mono-less-eq yidx-def*)
then have *f* (*a-seg* (*yidx y*)) ≤ *f* (*a-seg k*)
  using *strict-mono-onD* [*OF sm*] **by** *simp*
then show *f* (*a-seg* (*yidx y*)) ≤ *y*
  using *f-lims* **by** *linarith*
show *y* < *f* (*a-seg* (*Suc* (*yidx y*)))
  by (*metis LeastI f-lims*(*2*) *yidx-def*)
qed

have *yidx-equality*: *yidx y* = *k* **if** *y* ∈ {*0*..*b*} *y* ∈ {*f* (*a-seg k*)..<*f* (*a-seg* (*Suc k*))} **for** *y k*
proof (*rule antisym*)
  show *yidx y* ≤ *k*
    unfolding *yidx-def* **by** (*metis atLeastLessThan-iff that*(*2*) *Least-le*)
  have (*a-seg* (*real k*)) < *a-seg* (*1* + *real* (*yidx y*))
  using *yidx-gt* [*OF that*(*1*)] *that*(*2*) *strict-mono-onD* [*OF sm*] *order-le-less-trans* **by** *fastforce*
  then have *real k* < *1* + *real* (*yidx y*)
    by (*simp add*: ‹*strict-mono a-seg*› *strict-mono-less*)
  then show *k* ≤ *yidx y*
    by *simp*
qed
have *yidx b* = *n*
proof −
  have *a* < (*1* + *real n*) ∗ *a* / *real n*
    using ‹*0* < *n*› ‹*0* < *a*› **by** (*simp add*: *divide-simps*)
  then have *b* < *f* (*a-seg* (*1* + *real n*))
    using *f* ‹*a* ≥ *0*› *a-seg-def sm strict-mono-onD* **by** *fastforce*
  then show *?thesis*
    using ‹*0* ≤ *b*› **by** (*auto simp*: *f a-seg-def yidx-equality*)
qed
moreover have *yidx-less-n*: *yidx y* < *n* **if** *y* < *b* **for** *y*
  by (*metis* ‹*0* < *n*› *fa-eq-b gr0-conv-Suc less-Suc-eq-le that Least-le yidx-def*)
ultimately have *yidx-le-n*: *yidx y* ≤ *n* **if** *y* ≤ *b* **for** *y*
  by (*metis dual-order.order-iff-strict that*)

have *zero-to-b-eq*: {*0*..*b*} = (⋃*k*<*n*. {*f*(*a-seg k*)..*f*(*a-seg* (*Suc k*))}) (**is** *?lhs* = *?rhs*)
proof
  show *?lhs* ⊆ *?rhs*
  proof
    fix *y* **assume** *y*: *y* ∈ {*0*..*b*}
    have *fn*: *f* (*a-seg n*) = *b*
      using *a-seg-eq-a-iff* ‹*f a* = *b*› **by** *fastforce*
    show *y* ∈ *?rhs*
    proof (*cases y*=*b*)

```
        case True
        with fn ‹n>0› show ?thesis
          by (rule-tac a=n−1 in UN-I) auto
      next
        case False
        with y show ?thesis
          apply (simp add: subset-iff Bex-def)
            by (metis atLeastAtMost-iff of-nat-Suc order-le-less yidx-gt fa-yidx-le
yidx-less-n)
      qed
    qed
    show ?rhs ⊆ ?lhs
      apply clarsimp
    by (smt (verit, best) a-seg-ge-0 a-seg-le-a f f-iff(2) nat-less-real-le of-nat-0-le-iff)
  qed

  define g1 where g1 ≡ λy. if y=b then a else a-seg (Suc (yidx y))
  define g2 where g2 ≡ λy. if y=0 then 0 else a-seg (yidx y)
  have g1: g1 y ∈ {0..a} if y ∈ {0..b} for y
      using that ‹a > 0› yidx-less-n [of y] by (auto simp: g1-def a-seg-def di-
vide-simps)
  have g2: g2 y ∈ {0..a} if y ∈ {0..b} for y
   using that ‹a > 0› yidx-le-n [of y] by (simp add: g2-def a-seg-def divide-simps)

  have g2-le-g: g2 y ≤ g y if y ∈ {0..b} for y
  proof −
    have f (g2 y) ≤ y
      using ‹f 0 = 0› g2-def that fa-yidx-le by presburger
    then have f (g2 y) ≤ f (g y)
      using that g by (smt (verit, best) atLeastAtMost-iff fim image-iff)
    then show ?thesis
    by (smt (verit, best) atLeastAtMost-iff fim g g2 imageE sm-0a strict-mono-onD
that)
  qed
  have g-le-g1: g y ≤ g1 y if y ∈ {0..b} for y
  proof −
    have y ≤ f (g1 y)
      by (smt (verit, best) ‹f a = b› g1-def that yidx-gt)
    then have f (g y) ≤ f (g1 y)
      using that g by (smt (verit, best) atLeastAtMost-iff fim image-iff)
    then show ?thesis
      by (smt (verit, ccfv-threshold) atLeastAtMost-iff f-iff(1) g1 that)
  qed

  define DN where DN ≡ λK. nat ⌊Inf K ∗ real n / a⌋
  have [simp]: DN {a ∗ real k / n..a ∗ (1 + real k) / n} = k for k
    using ‹n > 0› ‹a > 0› by (simp add: DN-def divide-simps)
  have DN: bij-betw DN ?𝒟 {..<n}
  proof (intro bij-betw-imageI)
```

show *inj-on DN* (*regular-division 0 a n*)
proof
fix K K'
assume $K \in$ *regular-division 0 a n*
with ‹$a > 0$› **obtain** k **where** k: $K = \{a * (real\ k\ /\ n) .. a * (1 + real\ k) / n\}$
by (*force elim*: *regular-divisionE*)
assume $K' \in$ *regular-division 0 a n*
with ‹$a > 0$› **obtain** k' **where** k': $K' = \{a * (real\ k'\ /\ n) .. a * (1 + real\ k') / n\}$
by (*force elim*: *regular-divisionE*)
assume $DN\ K = DN\ K'$
then show $K = K'$ **by** (*simp add*: k k')
qed
have $\exists K \in$*regular-division 0 a n*. $k = nat\ \lfloor Inf\ K * real\ n\ /\ a \rfloor$ **if** $k < n$ **for** k
using ‹$n > 0$› ‹$a > 0$› *that*
by (*force simp*: *divide-simps intro*: *regular-division-eqI* [*OF refl*])
with ‹$a>0$› **show** *DN* ‘ *regular-division 0 a n* = $\{..<n\}$
by (*auto simp*: *DN-def bij-betw-def image-iff frac-le elim*!: *regular-divisionE*)
qed

have *int-f1*: (*f1 has-integral* ($\sum k<n.\ f(a\text{-}seg\ k)) * (a/n)$) $\{0..a\}$
proof −
have *a-seg* (*real* (*DN K*)) = *Inf K* **if** $K \in$ *?𝒟* **for** K
using *that* ‹$a>0$› **by** (*auto simp*: *DN-def field-simps a-seg-def elim*: *regular-divisionE*)
then have ($\sum K\in$*?𝒟*. $f(Inf\ K) * (a/n)) = (\sum k<n.\ (f(a\text{-}seg\ k)) * (a/n))$
by (*simp flip*: *sum.reindex-bij-betw* [*OF DN*])
moreover have (*f1 has-integral* ($\sum K\in$*?𝒟*. $f(Inf\ K) * (a/n))$) $\{0..a\}$
by (*intro div int-f1-D has-integral-combine-division*)
ultimately show *?thesis*
by (*metis sum-distrib-right*)
qed

The claim (*f2 has-integral* ($\sum k<n.\ f\ (a\text{-}seg\ (real\ (Suc\ k)))) * (a\ /\ real\ n)$) $\{0..a\}$ can similarly be proved

have *int-g1-D*: (*g1 has-integral a-seg* (*Suc k*) $*$ (f (*a-seg* (*Suc k*)) − f (*a-seg k*)))
$\{f(a\text{-}seg\ k)..f(a\text{-}seg\ (Suc\ k))\}$
and *int-g2-D*: (*g2 has-integral a-seg k* $*$ (f (*a-seg* (*Suc k*)) − f (*a-seg k*)))
$\{f(a\text{-}seg\ k)..f(a\text{-}seg\ (Suc\ k))\}$
if $k < n$ **for** k
proof −
define u **where** $u \equiv f$ (*a-seg k*)
define v **where** $v \equiv f$ (*a-seg* (*Suc k*))
obtain $u < v$ $0 \le u$ $0 \le v$
unfolding *u-def v-def assms*
by (*smt* (*verit*, *best*) *a-seg-ge-0 a-seg-le-iff f*(*1*) *f-iff*(*1*) *of-nat-0-le-iff of-nat-Suc*)

have $u \le b$ $v \le b$
  using ‹$k < n$› ‹$a \ge 0$› **by** (*simp-all add*: *u-def v-def flip*: ‹$f\ a = b$›)
have *yidx-eq*: $yidx\ x = k$ **if** $x \in \{u..<v\}$ **for** x
  using ‹$0 \le u$› ‹$v \le b$› *that u-def v-def yidx-equality* **by** *auto*

have $g1\ x = a\text{-}seg\ (Suc\ k)$ **if** $x \in \{u..<v\}$ **for** x
  using *that* ‹$v \le b$› **by** (*simp add*: *g1-def yidx-eq*)
moreover have $((\lambda x.\ a\text{-}seg\ (Suc\ k))\ has\text{-}integral\ (a\text{-}seg\ (Suc\ k) * (v-u)))\ \{u..v\}$
  using *has-integral-const-real* ‹$u < v$›
  by (*metis content-real-if less-eq-real-def mult.commute real-scaleR-def*)
ultimately show $(g1\ has\text{-}integral\ (a\text{-}seg\ (Suc\ k) * (v-u)))\ \{u..v\}$
  using *has-integral-spike-finite-eq* [*of* $\{v\}$ $\{u..v\}$ $\lambda x.\ a\text{-}seg\ (Suc\ k)\ g1$] **by** *simp*

have *g2*: $g2\ x = a\text{-}seg\ k$ **if** $x \in \{u<..<v\}$ **for** x
  using *that* ‹$0 \le u$› **by** (*simp add*: *g2-def yidx-eq*)
moreover have $((\lambda x.\ a\text{-}seg\ k)\ has\text{-}integral\ (a\text{-}seg\ k * (v-u)))\ \{u..v\}$
  using *has-integral-const-real* ‹$u < v$›
  by (*metis content-real-if less-eq-real-def mult.commute real-scaleR-def*)
ultimately show $(g2\ has\text{-}integral\ (a\text{-}seg\ k * (v-u)))\ \{u..v\}$
  using *has-integral-spike-finite-eq* [*of* $\{u,v\}$ $\{u..v\}$ $\lambda x.\ a\text{-}seg\ k\ g2$] **by** *simp*
qed

have *int-g1*: $(g1\ has\text{-}integral\ (\sum k<n.\ a\text{-}seg\ (Suc\ k) * (f\ (a\text{-}seg\ (Suc\ k)) - f\ (a\text{-}seg\ k))))\ \{0..b\}$
and *int-g2*: $(g2\ has\text{-}integral\ (\sum k<n.\ a\text{-}seg\ k * (f\ (a\text{-}seg\ (Suc\ k)) - f\ (a\text{-}seg\ k))))\ \{0..b\}$
  unfolding *zero-to-b-eq* **using** *int-g1-D int-g2-D*
by (*auto simp*: *min-def pairwise-def intro*!: *has-integral-UN negligible-atLeastAtMostI*)

have $(\sum k<n.\ a\text{-}seg\ (Suc\ k) * (f\ (a\text{-}seg\ (Suc\ k)) - f\ (a\text{-}seg\ k)))$
  $= (\sum k<n.\ (Suc\ k) * (f\ (a\text{-}seg\ (Suc\ k)) - f\ (a\text{-}seg\ k))) * (a/n)$
unfolding *a-seg-def sum-distrib-right sum-divide-distrib* **by** (*simp add*: *mult-ac*)
also have $\ldots = (n * f\ (a\text{-}seg\ n) - (\sum k<n.\ f\ (a\text{-}seg\ k))) * a\ /\ n$
  using *weighted-nesting-sum* [**where** $g = f\ o\ a\text{-}seg$] **by** *simp*
also have $\ldots = a * b - (\sum k<n.\ f\ (a\text{-}seg\ k)) * a\ /\ n$
  using ‹$n > 0$› **by** (*simp add*: *fa-eq-b field-simps*)
finally have *int-g1'*: $(g1\ has\text{-}integral\ a * b - (\sum k<n.\ f\ (a\text{-}seg\ k)) * a\ /\ n)\ \{0..b\}$
  using *int-g1* **by** *simp*

The claim $(g2\ has\text{-}integral\ a * b - (\sum k<n.\ f\ (a\text{-}seg\ (real\ (Suc\ k)))) * a\ /\ real\ n)\ \{0..b\}$ can similarly be proved.

have *a-seg-diff*: $a\text{-}seg\ (Suc\ k) - a\text{-}seg\ k = a/n$ **for** k
  by (*simp add*: *a-seg-def field-split-simps*)
have *f-a-seg-diff*: $|f\ (a\text{-}seg\ (Suc\ k)) - f\ (a\text{-}seg\ k)| < \varepsilon/a$ **if** $k<n$ **for** k
  using *that* ‹$a > 0$› *a-seg-diff an-less-del* ‹$\varepsilon > 0$›
  by (*intro del*) *auto*

have (($\lambda x.$ *g1 x* − *g2 x*) *has-integral* ($\sum$ *k<n.* (*f* (*a-seg* (*Suc k*)) − *f* (*a-seg k*)) ∗ (*a*/*n*))) {*0*..*b*}
using *has-integral-diff* [*OF int-g1 int-g2*] *a-seg-diff*
apply (*simp flip*: *sum-subtractf left-diff-distrib*)
apply (*simp add*: *field-simps*)
done
moreover have ($\sum$ *k<n.* (*f* (*a-seg* (*Suc k*)) − *f* (*a-seg k*)) ∗ (*a*/*n*)) < ε
proof −
have ($\sum$ *k<n.* (*f* (*a-seg* (*Suc k*)) − *f* (*a-seg k*)) ∗ (*a*/*n*))
≤ ($\sum$ *k<n.* |*f* (*a-seg* (*Suc k*)) − *f* (*a-seg k*)| ∗ (*a*/*n*))
by *simp*
also have ... < ($\sum$ *k<n.* (ε/*a*) ∗ (*a*/*n*))
proof (*rule sum-strict-mono*)
fix *k* **assume** *k* ∈ {..<*n*}
with ‹*n* > *0*› ‹*a* > *0*› *divide-strict-right-mono f-a-seg-diff pos-less-divide-eq*
show |*f* (*a-seg* (*Suc k*)) − *f* (*a-seg k*)| ∗ (*a*/*n*) < ε/*a* ∗ (*a*/*n*) **by** *fastforce*
qed (*use* ‹*n* > *0*› **in** *auto*)
also have ... = ε
using ‹*n* > *0*› ‹*a* > *0*› **by** *simp*
finally show *?thesis* **.**
qed
ultimately have *g2-near-g1*: *integral* {*0*..*b*} ($\lambda x.$ *g1 x* − *g2 x*) < ε
by (*simp add*: *integral-unique*)

have *ab1*: *integral* {*0*..*a*} *f1* + *integral* {*0*..*b*} *g1* = *a*∗*b*
using *int-f1 int-g1'* **by** (*simp add*: *integral-unique*)

have *integral* {*0*..*a*} ($\lambda x.$ *f x* − *f1 x*) ≤ *integral* {*0*..*a*} ($\lambda x.$ *f2 x* − *f1 x*)
proof (*rule integral-le*)
show ($\lambda x.$ *f x* − *f1 x*) *integrable-on* {*0*..*a*} ($\lambda x.$ *f2 x* − *f1 x*) *integrable-on* {*0*..*a*}
using *Henstock-Kurzweil-Integration.integrable-diff int-f1 intgb-f f12* **by** *blast+*
qed (*auto simp*: *f2-upper*)
with *f2-near-f1* **have** *integral* {*0*..*a*} ($\lambda x.$ *f x* − *f1 x*) < ε
by *simp*
moreover have *integral* {*0*..*a*} *f1* ≤ *integral* {*0*..*a*} *f*
by (*intro integral-le has-integral-integral intgb-f has-integral-integrable* [*OF int-f1*])
(*simp add*: *f1-lower*)
ultimately have *f-error*: |*integral* {*0*..*a*} *f* − *integral* {*0*..*a*} *f1*| < ε
using *Henstock-Kurzweil-Integration.integral-diff int-f1 intgb-f* **by** *fastforce*

have *integral* {*0*..*b*} ($\lambda x.$ *g1 x* − *g x*) ≤ *integral* {*0*..*b*} ($\lambda x.$ *g1 x* − *g2 x*)
proof (*rule integral-le*)
show ($\lambda x.$ *g1 x* − *g x*) *integrable-on* {*0*..*b*} ($\lambda x.$ *g1 x* − *g2 x*) *integrable-on* {*0*..*b*}
using *Henstock-Kurzweil-Integration.integrable-diff int-g1 int-g2 intgb-g* **by**

blast+
  qed (*auto simp*: *g2-le-g*)
  with *g2-near-g1* **have** *integral* {*0*..*b*} (λ*x*. *g1 x* − *g x*) < ε
   by *simp*
  moreover have *integral* {*0*..*b*} *g* ≤ *integral* {*0*..*b*} *g1*
    by (*intro integral-le has-integral-integral intgb-g has-integral-integrable* [*OF int-g1*])
    (*simp add*: *g-le-g1*)
  ultimately have *g-error*: |*integral* {*0*..*b*} *g1* − *integral* {*0*..*b*} *g*| < ε
   using *integral-diff int-g1 intgb-g* **by** *fastforce*
  show *?thesis*
   using *f-error g-error ab1* **by** *linarith*
 qed
 show *?thesis*
  using ∗ [*of* |*a* ∗ *b* − *integral* {*0*..*a*} *f* − *integral* {*0*..*b*} *g*| / *2*] **by** *fastforce*
qed (*use assms* **in** *force*)

corollary *Youngs-strict*:
 fixes *f* :: *real* ⇒ *real*
 assumes *sm*: *strict-mono-on* {*0*..} *f* **and** *cont*: *continuous-on* {*0*..} *f* **and** *a*>*0* *b*≥*0*
  and *f*: *f 0* = *0 f a* ≠ *b* **and** *fim*: *f* ‘ {*0*..} = {*0*..}
  and *g*: ⋀*x*. *0* ≤ *x* ⟹ *g* (*f x*) = *x*
 shows *a*∗*b* < *integral* {*0*..*a*} *f* + *integral* {*0*..*b*} *g*
proof −
 have *f-iff* [*simp*]: *f x* < *f y* ⟷ *x* < *y f x* ≤ *f y* ⟷ *x* ≤ *y*
  if *x* ≥ *0 y* ≥ *0* **for** *x y*
  using *that* **by** (*smt* (*verit*, *best*) *atLeast-iff sm strict-mono-onD*)+
 let *?b*′ = *f a*
 have *?b*′ ≥ *0*
  by (*smt* (*verit*, *best*) ‹*0* < *a*› *atLeast-iff f sm strict-mono-onD*)
 then have *sm-gx*: *strict-mono-on* {*0*..} *g*
  unfolding *strict-mono-on-def*
  by (*smt* (*verit*, *best*) *atLeast-iff f-iff*(*1*) *f-inv-into-f fim g inv-into-into*)
 show *?thesis*
 proof (*cases ?b*′ < *b*)
  case *True*
  have *gt-a*: *a* < *g y* **if** *y* ∈ {*?b*′<..*b*} **for** *y*
  proof −
   have *a* = *g ?b*′
    using ‹*a* > *0*› *g* **by** *force*
   also have . . . < *g y*
    using ‹*0* ≤ *?b*′› *sm-gx strict-mono-onD that* **by** *fastforce*
   finally show *?thesis* **.**
  qed
  have *continuous-on* {*0*..} *g*
   by (*metis cont f*(*1*) *fim g sm strict-mono-continuous-invD*)

then have *contg*: *continuous-on* {*?b′..b*} *g*
by (*meson Icc-subset-Ici-iff* ‹*0* ≤ *f a*› *continuous-on-subset*)
have *mono-on* {*0..*} *g*
by (*simp add*: *sm-gx strict-mono-on-imp-mono-on*)
then have *int-g0b*: *g integrable-on* {*0..b*}
by (*simp add*: *integrable-on-mono-on mono-on-subset*)
then have *int-gb′b*: *g integrable-on* {*?b′..b*}
by (*simp add*: ‹*0* ≤ *?b′*› *integrable-on-subinterval*)
have *a* ∗ (*b* − *?b′*) = *integral* {*?b′..b*} (λ*y*. *a*)
using *True* **by** *force*
also have ... < *integral* {*?b′..b*} *g*
using *contg True gt-a* **by** (*intro integral-less-real*) *auto*
finally have ∗: *a* ∗ (*b* − *?b′*) < *integral* {*?b′..b*} *g* **.**
have *a*∗*b* = *a* ∗ *?b′* + *a* ∗ (*b* − *?b′*)
by (*simp add*: *algebra-simps*)
also have ... = *integral* {*0..a*} *f* + *integral* {*0..?b′*} *g* + *a* ∗ (*b* − *?b′*)
using *Youngs-exact* ‹*a* > *0*› *cont* ‹*f 0* = *0*› *g sm* **by** *force*
also have ... < *integral* {*0..a*} *f* + *integral* {*0..?b′*} *g* + *integral* {*?b′..b*} *g*
by (*simp add*: ∗)
also have ... = *integral* {*0..a*} *f* + *integral* {*0..b*} *g*
by (*smt* (*verit*) *Henstock-Kurzweil-Integration.integral-combine True* ‹*0* ≤ *?b′*›
int-g0b)
finally show *?thesis* **.**
next
case *False*
with *f* **have** *b* < *?b′* **by** *force*
obtain *a′* **where** *f a′* = *b a′* ≥ *0*
using *fim* ‹*b* ≥ *0*› **by** *force*
then have *a′* < *a*
using ‹*b* < *f a*› ‹*a* > *0*› **by** *force*
have *gt-b*: *b* < *f x* **if** *x* ∈ {*a′*<..*a*} **for** *x*
using ‹*0* ≤ *a′*› ‹*f a′* = *b*› *that* **by** *fastforce*
have *int-f0a*: *f integrable-on* {*0..a*}
by (*simp add*: *integrable-on-mono-on mono-on-def*)
then have *int-fa′a*: *f integrable-on* {*a′..a*}
by (*simp add*: ‹*0* ≤ *a′*› *integrable-on-subinterval*)
have *cont-f′*: *continuous-on* {*a′..a*} *f*
by (*meson Icc-subset-Ici-iff* ‹*0* ≤ *a′*› *cont continuous-on-subset*)
have *b* ∗ (*a* − *a′*) = *integral* {*a′..a*} (λ*x*. *b*)
using ‹*a′* < *a*› **by** *simp*
also have ... < *integral* {*a′..a*} *f*
using *cont-f′* ‹*a′* < *a*› *gt-b* **by** (*intro integral-less-real*) *auto*
finally have ∗: *b* ∗ (*a* − *a′*) < *integral* {*a′..a*} *f* **.**
have *a*∗*b* = *a′* ∗ *b* + *b* ∗ (*a* − *a′*)
by (*simp add*: *algebra-simps*)
also have ... = *integral* {*0..a′*} *f* + *integral* {*0..b*} *g* + *b* ∗ (*a* − *a′*)
by (*simp add*: *Youngs-exact* ‹*0* ≤ *a′*› ‹*f a′* = *b*› *cont f g sm*)
also have ... < *integral* {*0..a′*} *f* + *integral* {*0..b*} *g* + *integral* {*a′..a*} *f*
by (*simp add*: ∗)

also have ... = *integral* {*0*..*a*} *f* + *integral* {*0*..*b*} *g*
by (*smt* (*verit*) *Henstock-Kurzweil-Integration.integral-combine* ‹*0* ≤ *a*′› ‹*a*′ < *a*› *int-f0a*)
finally show *?thesis* **.**
qed
qed

corollary *Youngs-inequality*:
fixes *f* :: *real* ⇒ *real*
assumes *sm*: *strict-mono-on* {*0*..} *f* **and** *cont*: *continuous-on* {*0*..} *f* **and** *a*≥*0* *b*≥*0*
and *f*: *f 0* = *0* **and** *fim*: *f* ‘ {*0*..} = {*0*..}
and *g*: ⋀*x*. *0* ≤ *x* ⟹ *g* (*f x*) = *x*
shows *a*∗*b* ≤ *integral* {*0*..*a*} *f* + *integral* {*0*..*b*} *g*
proof (*cases a*=*0*)
case *True*
have *g x* ≥ *0* **if** *x* ≥ *0* **for** *x*
by (*metis atLeast-iff fim g imageE that*)
then have *0* ≤ *integral* {*0*..*b*} *g*
by (*metis Henstock-Kurzweil-Integration.integral-nonneg atLeastAtMost-iff not-integrable-integral order-refl*)
then show *?thesis*
by (*simp add*: *True*)
next
case *False*
then show *?thesis*
by (*smt* (*verit*) *assms Youngs-exact Youngs-strict*)
qed

end

References

[1] F. Cunningham and N. Grossman. On Young's inequality. *The American Mathematical Monthly*, 78(7):781–783, 1971.