

The Cartan Fixed Point Theorems

Lawrence C. Paulson

February 6, 2026

Abstract

The Cartan fixed point theorems concern the group of holomorphic automorphisms on a connected open set of \mathbb{C}^n . Ciolli et al. [1] have formalised the one-dimensional case of these theorems in HOL Light. This entry contains their proofs, ported to Isabelle/HOL. Thus it addresses the authors remark that “it would be important to write a formal proof in a language that can be read by both humans and machines.”

Contents

0.1 First Cartan Theorem 1
theory *Cartan*
imports *HOL-Complex-Analysis.Complex-Analysis*

begin

0.1 First Cartan Theorem

Ported from HOL Light. See Gianni Ciolli, Graziano Gentili, Marco Maggesi. A Certified Proof of the Cartan Fixed Point Theorems. *J Automated Reasoning* (2011) 47:319–336 DOI 10.1007/s10817-010-9198-6

lemma *deriv-left-inverse*:

assumes *f* holomorphic-on *S* **and** *g* holomorphic-on *T*
and open *S* **and** open *T*
and $f' S \subseteq T$
and [*simp*]: $\bigwedge z. z \in S \implies g (f z) = z$
and $w \in S$
shows $\text{deriv } f w * \text{deriv } g (f w) = 1$

proof –

have $\text{deriv } f w * \text{deriv } g (f w) = \text{deriv } g (f w) * \text{deriv } f w$
by (*simp add: algebra-simps*)

also have $\dots = \text{deriv } (g \circ f) w$

using *assms*

by (*metis analytic-on-imp-differentiable-at analytic-on-open deriv-chain image-subset-iff*)

also have $\dots = \text{deriv } id w$

apply (*rule complex-derivative-transform-within-open [where s=S]*)

apply (*rule assms holomorphic-on-compose-gen holomorphic-intros*)**+**

apply *simp*

done

also have $\dots = 1$

using *higher-deriv-id [of 1]* **by** *simp*

finally show *?thesis* .

qed

lemma *Cauchy-higher-deriv-bound*:

assumes *holf*: *f* holomorphic-on (ball *z* *r*)

and *contf*: *continuous-on* (*cball* *z* *r*) *f*
and $0 < r$ **and** $0 < n$
and *fin* : $\bigwedge w. w \in \text{ball } z \ r \implies f \ w \in \text{ball } y \ B0$
shows *norm* $((\text{deriv } \widehat{\widehat{n}}) \ f \ z) \leq (\text{fact } n) * B0 / r^{\widehat{n}}$
proof –
have $0 < B0$ **using** $\langle 0 < r \rangle$ *fin* [*of* *z*]
by (*metis* *ball-eq-empty* *ex-in-conv* *fin* *not-less*)
have *le-B0*: $\bigwedge w. \text{cmod } (w - z) \leq r \implies \text{cmod } (f \ w - y) \leq B0$
apply (*rule* *continuous-on-closure-norm-le* [*of* *ball* *z* *r* $\lambda w. f \ w - y$])
apply (*auto* *simp*: $\langle 0 < r \rangle$ *dist-norm* *norm-minus-commute*)
apply (*rule* *continuous-intros* *contf*)
using *fin* **apply** (*simp* *add*: *dist-commute* *dist-norm* *less-eq-real-def*)
done
have $(\text{deriv } \widehat{\widehat{n}}) \ f \ z = (\text{deriv } \widehat{\widehat{n}}) (\lambda w. f \ w) \ z - (\text{deriv } \widehat{\widehat{n}}) (\lambda w. y) \ z$
using $\langle 0 < n \rangle$ **by** *simp*
also **have** $\dots = (\text{deriv } \widehat{\widehat{n}}) (\lambda w. f \ w - y) \ z$
by (*rule* *higher-deriv-diff* [*OF* *holf*, *symmetric*]) (*auto* *simp*: $\langle 0 < r \rangle$ *holomor-*
phic-on-const)
finally **have** $(\text{deriv } \widehat{\widehat{n}}) \ f \ z = (\text{deriv } \widehat{\widehat{n}}) (\lambda w. f \ w - y) \ z$.
have *contf'*: *continuous-on* (*cball* *z* *r*) $(\lambda u. f \ u - y)$
by (*rule* *contf* *continuous-intros*)
have *holf'*: $(\lambda u. (f \ u - y))$ *holomorphic-on* (*ball* *z* *r*)
by (*simp* *add*: *holf* *holomorphic-on-diff* *holomorphic-on-const*)
define *a* **where** $a = (2 * \text{pi}) / (\text{fact } n)$
have $0 < a$ **by** (*simp* *add*: *a-def*)
have $B0 / r^{\widehat{(Suc \ n)} * 2 * \text{pi} * r} = a * ((\text{fact } n) * B0 / r^{\widehat{n}})$
using $\langle 0 < r \rangle$ **by** (*simp* *add*: *a-def* *divide-simps*)
have *der-dif*: $(\text{deriv } \widehat{\widehat{n}}) (\lambda w. f \ w - y) \ z = (\text{deriv } \widehat{\widehat{n}}) \ f \ z$
using $\langle 0 < r \rangle$ $\langle 0 < n \rangle$
by (*auto* *simp*: *higher-deriv-diff* [*OF* *holf* *holomorphic-on-const*])
have *norm* $((2 * \text{of-real } \text{pi} * i) / (\text{fact } n) * (\text{deriv } \widehat{\widehat{n}}) (\lambda w. f \ w - y) \ z)$
 $\leq (B0 / r^{\widehat{(Suc \ n)}}) * (2 * \text{pi} * r)$
apply (*rule* *has-contour-integral-bound-circlepath* [*of* $(\lambda u. (f \ u - y) / (u - z))^{\widehat{(Suc \ n)}} - z$])
using *Cauchy-has-contour-integral-higher-derivative-circlepath* [*OF* *contf'* *holf'*]
using $\langle 0 < B0 \rangle$ $\langle 0 < r \rangle$
apply (*auto* *simp*: *norm-divide* *norm-mult* *norm-power* *divide-simps* *le-B0*)
done
then **show** *?thesis*
using $\langle 0 < r \rangle$
by (*auto* *simp*: *norm-divide* *norm-mult* *norm-power* *field-simps* *der-dif* *le-B0*)
qed

lemma *higher-deriv-comp-lemma*:

assumes *s*: *open* *s* **and** *holf*: *f* *holomorphic-on* *s*
and $z \in s$
and *t*: *open* *t* **and** *holg*: *g* *holomorphic-on* *t*
and *fst*: $f' \ s \subseteq t$
and *n*: $i \leq n$

and $dfz: \text{deriv } f z = 1$ **and** $zero: \bigwedge i. [1 < i; i \leq n] \implies (\text{deriv } \overset{\sim}{\sim} i) f z = 0$
shows $(\text{deriv } \overset{\sim}{\sim} i) (g \circ f) z = (\text{deriv } \overset{\sim}{\sim} i) g (f z)$

using $n \text{ holg}$

proof (*induction* i *arbitrary*: g)

case 0 **then show** *?case* **by** *simp*

next

case ($Suc\ i$)

have $g \circ f$ *holomorphic-on* s **using** $Suc.prem\ s$ *holf*

using *fst* **by** (*simp* *add*: *holomorphic-on-compose-gen image-subset-iff*)

then have $1: \text{deriv } (g \circ f)$ *holomorphic-on* s

by (*simp* *add*: *holomorphic-deriv s*)

have $dg: \text{deriv } g$ *holomorphic-on* t

using $Suc.prem\ s$ **by** (*simp* *add*: $Suc.prem\ s(2)$ *holomorphic-deriv t*)

then have $\text{deriv } g$ *holomorphic-on* $f' s$

using *fst* **by** (*simp* *add*: *holomorphic-on-subset image-subset-iff*)

then have $dgf: (\text{deriv } g \circ f)$ *holomorphic-on* s

by (*simp* *add*: *holf holomorphic-on-compose*)

then have $2: (\lambda w. (\text{deriv } g \circ f) w * \text{deriv } f w)$ *holomorphic-on* s

by (*blast intro*: *holomorphic-intros holomorphic-on-compose holf s*)

have $(\text{deriv } \overset{\sim}{\sim} i) (\text{deriv } (g \circ f)) z = (\text{deriv } \overset{\sim}{\sim} i) (\lambda w. \text{deriv } g (f w) * \text{deriv } f w)$

z

apply (*rule higher-deriv-transform-within-open* [$OF\ 1\ 2$ [*unfolded o-def*] $s \langle z \in s \rangle$])

apply (*rule deriv-chain*)

using *holf Suc.prem\ s* **apply** (*auto simp*: *holomorphic-on-imp-differentiable-at s t*)

done

also have $\dots = (\sum_{j=0..i} \text{of-nat}(i\ \text{choose}\ j) * (\text{deriv } \overset{\sim}{\sim} j) (\lambda w. \text{deriv } g (f w))$

$z * (\text{deriv } \overset{\sim}{\sim} (i - j)) (\text{deriv } f) z)$

apply (*rule higher-deriv-mult* [$OF\ dg\ f$ [*unfolded o-def*] - $s \langle z \in s \rangle$])

by (*simp* *add*: *holf holomorphic-deriv s*)

also have $\dots = (\sum_{j=i..i} \text{of-nat}(i\ \text{choose}\ j) * (\text{deriv } \overset{\sim}{\sim} j) (\lambda w. \text{deriv } g (f w))$

$z * (\text{deriv } \overset{\sim}{\sim} Suc\ (i - j)) f z)$

proof -

have $*$: $(\text{deriv } \overset{\sim}{\sim} j) (\lambda w. \text{deriv } g (f w)) z = 0$ **if** $j < i$ **and** $nz: (\text{deriv } \overset{\sim}{\sim} (i - j)) (\text{deriv } f) z \neq 0$ **for** j

proof -

have $1 < Suc\ (i - j)$ $Suc\ (i - j) \leq n$

using $\langle j < i \rangle \langle Suc\ i \leq n \rangle$ **by** *auto*

then show *?thesis* **by** (*metis comp-def funpow.simps(2) funpow-swap1 zero nz*)

qed

then show *?thesis*

apply (*simp only*: *funpow-Suc-right o-def*)

apply (*rule comm-monoid-add-class.sum.mono-neutral-right, auto*)

done

qed

also have $\dots = (\text{deriv } \overset{\sim}{\sim} i) (\text{deriv } g) (f z)$

using $Suc.IH$ [$OF - dg$] $Suc.prem\ s$ **by** (*simp* *add*: dfz)

finally show ?case
 by (simp only: funpow-Suc-right o-def)
qed

lemma higher-deriv-comp-iter-lemma:

assumes s : open s **and** $holf$: f holomorphic-on s
and fss : $f' s \subseteq s$
and $z \in s$ **and** [simp]: $f z = z$
and n : $i \leq n$
and dfz : $deriv f z = 1$ **and** $zero$: $\bigwedge i. \llbracket 1 < i; i \leq n \rrbracket \implies (deriv \sim i) f z = 0$
shows $(deriv \sim i) (f \sim m) z = (deriv \sim i) f z$

proof –

have $holfm$: $(f \sim m)$ holomorphic-on s **for** m
apply (induction m , simp add: holomorphic-on-ident)
apply (simp only: funpow-Suc-right holomorphic-on-compose-gen [OF $holf$ -
 fss])

done

show ?thesis **using** n

proof (induction m)

case 0 **with** dfz **show** ?case

by (auto simp: zero)

next

case (Suc m)

have $(deriv \sim i) (f \sim m \circ f) z = (deriv \sim i) (f \sim m) (f z)$

using Suc.prem1 $holfm \langle z \in s \rangle dfz fss$ higher-deriv-comp-lemma $holf s zero$

by blast

also have $\dots = (deriv \sim i) f z$

by (simp add: Suc)

finally show ?case

by (simp only: funpow-Suc-right)

qed

qed

lemma higher-deriv-iter-top-lemma:

assumes s : open s **and** $holf$: f holomorphic-on s
and fss : $f' s \subseteq s$
and $z \in s$ **and** [simp]: $f z = z$
and dfz [simp]: $deriv f z = 1$
and n : $1 < n \wedge i. \llbracket 1 < i; i < n \rrbracket \implies (deriv \sim i) f z = 0$
shows $(deriv \sim n) (f \sim m) z = m * (deriv \sim n) f z$

using n

proof (induction n arbitrary: m)

case 0 **then show** ?case **by** simp

next

case (Suc n)

have [simp]: $(f \sim m) z = z$ **for** m

by (induction m) auto

have fms -sb: $(f \sim m)' s \subseteq s$ **for** m

```

apply (induction m)
using fss
apply force+
done
have hol $f$ m: (f $\sim$ m) holomorphic-on s for m
apply (induction m, simp add: holomorphic-on-ident)
apply (simp only: funpow-Suc-right holomorphic-on-compose-gen [OF hol $f$  -
fss])
done
then have hold $f$ m: deriv (f $\sim$ m) holomorphic-on s for m
by (simp add: holomorphic-deriv s)
have hold $f$ f $m$ : ( $\lambda$ z. deriv f ((f $\sim$ m) z)) holomorphic-on s for m
apply (rule holomorphic-on-compose-gen [where g=deriv f and t=s, unfolded
o-def])
using s  $\langle$ z  $\in$  s $\rangle$  hol $f$ m hol $f$ m $s$ -sb by (auto intro: holomorphic-intros)
have f-cd-w:  $\bigwedge$ w. w  $\in$  s  $\implies$  f field-differentiable at w
using hol $f$  holomorphic-on-imp-differentiable-at s by blast
have f-cd-mw:  $\bigwedge$ m w. w  $\in$  s  $\implies$  (f $\sim$ m) field-differentiable at w
using hol $f$ m holomorphic-on-imp-differentiable-at s by auto
have der-fm [simp]: deriv (f $\sim$ m) z = 1 for m
apply (induction m, simp add: deriv-ident)
apply (subst funpow-Suc-right)
apply (subst deriv-chain)
using  $\langle$ z  $\in$  s $\rangle$  hol $f$ m holomorphic-on-imp-differentiable-at s f-cd-w apply auto
done
note Suc(3) [simp]
note n-Suc = Suc
show ?case
proof (induction m)
case 0 with n-Suc show ?case
by (metis Zero-not-Suc funpow-simps-right(1) higher-deriv-id lambda-zero
nat-neq-iff of-nat-0)
next
case (Suc m)
have deriv-n $f$ f $m$ : (deriv  $\sim$ n) (deriv f o (f $\sim$ m)) z = (deriv  $\sim$ n) (deriv f
((f $\sim$ m) z))
apply (rule higher-deriv-comp-lemma [OF s hol $f$ m  $\langle$ z  $\in$  s $\rangle$  s - fms-sb order-refl])
using  $\langle$ z  $\in$  s $\rangle$  fss higher-deriv-comp-iter-lemma hol $f$  hol $f$  holomorphic-deriv s
apply auto
done
have deriv (f $\sim$ m o f) holomorphic-on s
by (metis funpow-Suc-right hold $f$ m)
moreover have ( $\lambda$ w. deriv f ((f $\sim$ m) w) * deriv (f $\sim$ m) w) holomorphic-on
s
by (rule holomorphic-on-mult [OF hold $f$ f $m$  hold $f$ m])
ultimately have (deriv  $\sim$ n) (deriv (f $\sim$ m o f)) z = (deriv  $\sim$ n) ( $\lambda$ w. deriv
f ((f $\sim$ m) w) * deriv (f $\sim$ m) w) z
apply (rule higher-deriv-transform-within-open [OF - - s  $\langle$ z  $\in$  s $\rangle$ ])
by (metis comp-funpow deriv-chain f-cd-mw f-cd-w fms-sb funpow-swap1 im-

```

age-subset-iff o-id
also have ... =
 $(\sum i=0..n. \text{of-nat}(n \text{ choose } i) * (\text{deriv } \sim i) (\lambda w. \text{deriv } f ((f \sim m) w)) z * (\text{deriv } \sim (n - i)) (\text{deriv } (f \sim m)) z)$
by (*rule higher-deriv-mult [OF holdffm holdfm s ⟨z ∈ s⟩]*)
also have ... = $(\sum i \in \{0, n\}. \text{of-nat}(n \text{ choose } i) * (\text{deriv } \sim i) (\lambda w. \text{deriv } f ((f \sim m) w)) z * (\text{deriv } \sim (n - i)) (\text{deriv } (f \sim m)) z)$
proof –
have *: $(\text{deriv } \sim i) (\lambda w. \text{deriv } f ((f \sim m) w)) z = 0$ **if** $i \leq n$ $0 < i$ $i \neq n$
and nz : $(\text{deriv } \sim (n - i)) (\text{deriv } (f \sim m)) z \neq 0$ **for** i
proof –
have *less*: $1 < \text{Suc } (n - i)$ **and** *le*: $\text{Suc } (n - i) \leq n$
using *that by auto*
have $(\text{deriv } \sim (\text{Suc } (n - i))) (f \sim m) z = (\text{deriv } \sim (\text{Suc } (n - i))) f z$
apply (*rule higher-deriv-comp-iter-lemma [OF s holf fss ⟨z ∈ s⟩ ⟨f z = z⟩ le dfz]*)
by *simp*
also have ... = 0
using $n\text{-Suc}(3)$ *less le le-imp-less-Suc* **by** *blast*
finally have $(\text{deriv } \sim (\text{Suc } (n - i))) (f \sim m) z = 0$.
then show *?thesis* **by** (*simp add: funpow-swap1 nz*)
qed
show *?thesis*
by (*rule comm-monoid-add-class.sum.mono-neutral-right*) (*auto simp: **)
qed
also have ... = $\text{of-nat } (\text{Suc } m) * (\text{deriv } \sim n) (\text{deriv } f) z$
apply (*subst Groups-Big.comm-monoid-add-class.sum.insert*)
apply (*simp-all add: deriv-nffm [unfolded o-def] of-nat-Suc [of 0] del: of-nat-Suc*)
using $n\text{-Suc}(2)$ *Suc*
apply (*auto simp del: funpow.simps simp: algebra-simps funpow-simps-right*)
done
finally have $(\text{deriv } \sim n) (\text{deriv } (f \sim m \circ f)) z = \text{of-nat } (\text{Suc } m) * (\text{deriv } \sim n) (\text{deriv } f) z$.
then show *?case*
apply (*simp only: funpow-Suc-right*)
apply (*simp add: o-def del: of-nat-Suc*)
done
qed
qed

Should be proved for n-dimensional vectors of complex numbers

theorem *first-Cartan-dim-1*:

assumes *holf*: f *holomorphic-on* s
and *open* s *connected* s *bounded* s
and *fss*: $f' s \subseteq s$
and $z \in s$ **and** [*simp*]: $f z = z$
and *dfz* [*simp*]: $\text{deriv } f z = 1$
and $w \in s$

```

    shows  $f w = w$ 
  proof -
    obtain  $c$  where  $0 < c$  and  $c: s \subseteq \text{ball } z \ c$ 
      using  $\langle \text{bounded } s \rangle \text{ bounded-subset-ballD}$  by blast
    obtain  $r$  where  $0 < r$  and  $r: \text{cball } z \ r \subseteq s$ 
      using  $\langle z \in s \rangle \text{ open-contains-cball } \langle \text{open } s \rangle$  by blast
    then have  $\text{bzc}: \text{ball } z \ r \subseteq s$  using  $\text{ball-subset-cball}$  by blast
    have  $\text{fms-sb}: (f \sim m) \ ' s \subseteq s$  for  $m$ 
      apply (induction  $m$ )
      using  $\text{fss}$  apply force+
    done
    have  $\text{holfm}: (f \sim m) \text{ holomorphic-on } s$  for  $m$ 
      apply (induction  $m$ , simp add:  $\text{holomorphic-on-ident}$ )
      apply (simp only:  $\text{funpow-Suc-right holomorphic-on-compose-gen [OF holm' - fss]}$ )
    done
    have  $*$ :  $(\text{deriv } \sim n) f z = (\text{deriv } \sim n) \text{ id } z$  for  $n$ 
    proof -
      consider  $n = 0 \mid n = 1 \mid 1 < n$  by arith
      then show ?thesis
    proof cases
      assume  $n = 0$  then show ?thesis by force
    next
      assume  $n = 1$  then show ?thesis by force
    next
      assume  $n1: n > 1$ 
      then have  $(\text{deriv } \sim n) f z = 0$ 
      proof (induction  $n$  rule:  $\text{less-induct}$ )
        case (less  $n$ )
        have  $\text{le}: \text{real } m * \text{cmod } ((\text{deriv } \sim n) f z) \leq \text{fact } n * c / r ^ n$  if  $m \neq 0$  for
           $m$ 
      proof -
        have  $\text{holfm}': (f \sim m) \text{ holomorphic-on } \text{ball } z \ r$ 
          using  $\text{holfm bzc holomorphic-on-subset}$  by blast
        then have  $\text{contfm}': \text{continuous-on } (\text{cball } z \ r) (f \sim m)$ 
          using  $\langle \text{cball } z \ r \subseteq s \rangle \text{ holm holomorphic-on-imp-continuous-on holomor- phic-on-subset}$  by blast
        have  $\text{real } m * \text{cmod } ((\text{deriv } \sim n) f z) = \text{cmod } (\text{real } m * (\text{deriv } \sim n) f z)$ 
          by (simp add:  $\text{norm-mult}$ )
        also have  $\dots = \text{cmod } ((\text{deriv } \sim n) (f \sim m) z)$ 
          apply (subst  $\text{higher-deriv-iter-top-lemma [OF } \langle \text{open } s \rangle \text{ holm' fss } \langle z \in s \rangle \langle f z = z \rangle \text{ dfz}]$ )
          using  $\text{less}$  apply auto
        done
        also have  $\dots \leq \text{fact } n * c / r ^ n$ 
          apply (rule  $\text{Cauchy-higher-deriv-bound [OF holfm' contfm' } \langle 0 < r \rangle$ ,
            where  $y=z$ )
          using  $\text{less.premis}$  apply  $\text{linarith}$ 
          using  $\text{fms-sb } c \ r \ \text{ball-subset-cball}$ 

```

```

    apply blast
  done
  finally show ?thesis .
qed
have cmod ((deriv  $\sim$  n) f z) = 0
proof (rule real-archimedian-rdiv-eq-0)
  show  $\bigwedge m. 0 < m \implies \text{real } m * \text{cmod } ((\text{deriv } \sim n) f z) \leq \text{fact } n * c / r$ 
 $\hat{n}$ 
    using  $\langle 0 < r \rangle \langle 0 < c \rangle$  le by (simp add: divide-simps)
  qed auto
  then show ?case by simp
qed
with n1 show ?thesis by simp
qed
qed
have f w = id w
  by (rule holomorphic-fun-eq-on-connected
      [OF holf holomorphic-on-id  $\langle \text{open } s \rangle \langle \text{connected } s \rangle * \langle z \in s \rangle \langle w \in s \rangle$ ])
also have ... = w by simp
finally show ?thesis .
qed

```

Second Cartan Theorem.

lemma *Cartan-is-linear*:

```

assumes holf: f holomorphic-on s
  and open s and connected s
  and 0  $\in$  s
  and ins:  $\bigwedge u z. [\text{norm } u = 1; z \in s] \implies u * z \in s$ 
  and feq:  $\bigwedge u z. [\text{norm } u = 1; z \in s] \implies f (u * z) = u * f z$ 
shows  $\exists c. \forall z \in s. f z = c * z$ 
proof -
  have [simp]: f 0 = 0
    using feq [of -1 0] assms by simp
  have uneq:  $u \hat{n} * (\text{deriv } \sim n) f (u * z) = u * (\text{deriv } \sim n) f z$ 
    if norm u = 1 z  $\in$  s for n u z
  proof -
    have holfw:  $(\lambda w. f (u * w))$  holomorphic-on s
      apply (rule holomorphic-on-compose-gen [OF - holf, unfolded o-def])
      using that ins by (auto simp: holomorphic-on-linear)
    have hol-d-fw:  $(\text{deriv } \sim n) (\lambda w. u * f w)$  holomorphic-on s for n
      by (rule holomorphic-higher-deriv holomorphic-intros holf assms)+
    have *:  $(\text{deriv } \sim n) (\lambda w. u * f w) z = u * (\text{deriv } \sim n) f z$  if z  $\in$  s for z
      using that
    proof (induction n arbitrary: z)
      case 0 then show ?case by simp
    next
      case (Suc n)
      have deriv  $((\text{deriv } \sim n) (\lambda w. u * f w)) z = \text{deriv } (\lambda w. u * (\text{deriv } \sim n) f w)$ 

```

z

```

    apply (rule complex-derivative-transform-within-open [OF hol-d-fuw])
    apply (auto intro!: holomorphic-higher-deriv holomorphic-intros assms Suc)
    done
  also have ... = u * deriv ((deriv  $\hat{\sim}$  n) f) z
    apply (rule deriv-cmult)
  using Suc <open s> holf holomorphic-higher-deriv holomorphic-on-imp-differentiable-at
by blast
  finally show ?case by simp
qed
have (deriv  $\hat{\sim}$  n) ( $\lambda w. f (u * w)$ ) z = u  $\hat{\sim}$  n * (deriv  $\hat{\sim}$  n) f (u * z)
  apply (rule higher-deriv-compose-linear [OF holf <open s> <open s>])
  apply (simp add: that)
  apply (simp add: ins that)
  done
moreover have (deriv  $\hat{\sim}$  n) ( $\lambda w. f (u * w)$ ) z = u * (deriv  $\hat{\sim}$  n) f z
  apply (subst higher-deriv-transform-within-open [OF holfuw, of  $\lambda w. u * f w$ ])
  apply (rule holomorphic-intros holf assms that)+
  apply blast
  using * <z  $\in$  s> apply blast
  done
ultimately show ?thesis by metis
qed
have dnf0: (deriv  $\hat{\sim}$  n) f 0 = 0 if len: 2  $\leq$  n for n
proof -
  have **: z = 0 if  $\wedge u::\text{complex. norm } u = 1 \implies u \hat{\sim} n * z = u * z$  for z
  proof -
    have  $\exists u::\text{complex. norm } u = 1 \wedge u \hat{\sim} n \neq u$ 
      using complex-not-root-unity [of n-1] len
      apply (simp add: algebra-simps le-diff-conv2, clarify)
      apply (rule-tac x=u in exI)
      apply (subst (asm) power-diff)
      apply auto
      done
    with that show ?thesis
      by auto
  qed
show ?thesis
  apply (rule **)
  using uneq [OF - <0  $\in$  s>]
  by force
qed
show ?thesis
  apply (rule-tac x = deriv f 0 in exI, clarify)
  apply (rule holomorphic-fun-eq-on-connected [OF holf - <open s> <connected s>
- <0  $\in$  s>])
  using dnf0 apply (auto simp: holomorphic-on-linear)
  done
qed

```

Should be proved for n-dimensional vectors of complex numbers

theorem *second-Cartan-dim-1*:

assumes *holf*: f holomorphic-on ball 0 r
and *holg*: g holomorphic-on ball 0 r
and [*simp*]: $f\ 0 = 0$ **and** [*simp*]: $g\ 0 = 0$
and *ballf*: $\bigwedge z. z \in \text{ball } 0\ r \implies f\ z \in \text{ball } 0\ r$
and *ballg*: $\bigwedge z. z \in \text{ball } 0\ r \implies g\ z \in \text{ball } 0\ r$
and *fg*: $\bigwedge z. z \in \text{ball } 0\ r \implies f\ (g\ z) = z$
and *gf*: $\bigwedge z. z \in \text{ball } 0\ r \implies g\ (f\ z) = z$
and $0 < r$
shows $\exists t. \forall z \in \text{ball } 0\ r. g\ z = \exp(i * \text{of-real } t) * z$

proof –

have *c-le-1*: $c \leq 1$
if $0 \leq c \wedge x. 0 \leq x \implies x < r \implies c * x < r$ **for** c

proof –

have *rst*: $\bigwedge r\ s\ t::\text{real}. 0 = r \vee s/r < t \vee r < 0 \vee \neg s < r * t$
by (*metis* (*no-types*) *mult-less-cancel-left-disj nonzero-mult-div-cancel-left times-divide-eq-right*)

{ **assume** $\neg r < c \wedge c * (c * (c * (c * r))) < 1$
then have $1 \leq c \implies (\exists r. \neg 1 < r \wedge \neg r < c)$
using $\langle 0 \leq c \rangle$ **by** (*metis* (*full-types*) *less-eq-real-def mult.right-neutral mult-left-mono not-less*)

then have $\neg 1 < c \vee \neg 1 \leq c$
by *linarith* }

moreover
{ **have** $\neg 0 \leq r / c \implies \neg 1 \leq c$
using $\langle 0 < r \rangle$ **by force**
then have $1 < c \implies \neg 1 \leq c$
using *rst* $\langle 0 < r \rangle$ **that**
by (*metis* *div-by-1 frac-less2 less-le-trans mult commute not-le order-refl pos-divide-le-eq zero-less-one*) }

ultimately show *?thesis*
by (*metis* (*no-types*) *linear not-less*)

qed

have *ugeq*: $u * g\ z = g\ (u * z)$ **if** *now*: $\text{norm } u = 1$ **and** $z: z \in \text{ball } 0\ r$ **for** $u\ z$

proof –

have [*simp*]: $u \neq 0$ **using** *that* **by** *auto*
have *hol1*: $(\lambda a. f\ (u * g\ a) / u)$ holomorphic-on ball 0 r
apply (*rule* *holomorphic-intros*)
apply (*rule* *holomorphic-on-compose-gen* [*OF* - *holf*, *unfolded o-def*])
apply (*rule* *holomorphic-intros holg*)
using *now ballg*
apply (*auto simp: dist-norm norm-mult holomorphic-on-const*)
done

have *cdf*: f field-differentiable at 0
using $\langle 0 < r \rangle$ *holf* *holomorphic-on-imp-differentiable-at* **by** *auto*

have *cdg*: g field-differentiable at 0
using $\langle 0 < r \rangle$ *holg* *holomorphic-on-imp-differentiable-at* **by** *auto*

have *cd-fug*: $(\lambda a. f\ (u * g\ a))$ field-differentiable at 0
apply (*rule* *field-differentiable-compose* [**where** $g=f$ **and** $f = \lambda a. (u * g\ a)$],

```

unfolded o-def])
  apply (rule derivative-intros)+
  using cdf cdg
  apply auto
  done
have deriv g 0 = deriv g (f 0)
  by simp
then have deriv f 0 * deriv g 0 = 1
  by (metis open-ball ⟨0 < r⟩ ballf centre-in-ball deriv-left-inverse gf holf holg
image-subsetI)
then have equ: deriv f 0 * deriv (λa. u * g a) 0 = u
  by (simp add: cdg deriv-cmult)
have der1: deriv (λa. f (u * g a) / u) 0 = 1
  apply (simp add: field-class.field-divide-inverse deriv-cmult-right [OF cd-fug])
  apply (subst deriv-chain [where g=f and f = λa. (u * g a), unfolded o-def])
  apply (rule derivative-intros cdf cdg | simp add: equ)+
  done
have fugeq: ∧w. w ∈ ball 0 r ⇒ f (u * g w) / u = w
  apply (rule first-Cartan-dim-1 [OF hol1, where z=0])
  apply (simp-all add: ⟨0 < r⟩)
  apply (auto simp: der1)
  using nou ballf ballg
  apply (simp add: dist-norm norm-mult norm-divide)
  done
have f(u * g z) = u * z
  by (metis ⟨u ≠ 0⟩ fugeq nonzero-mult-div-cancel-left z times-divide-eq-right)
also have ... = f (g (u * z))
  by (metis (no-types, lifting) fg mem-ball-0 mult-cancel-right2 norm-mult nou
z)
finally have f(u * g z) = f (g (u * z)) .
then have g (f (u * g z)) = g (f (g (u * z)))
  by simp
then show ?thesis
  apply (subst (asm) gf)
  apply (simp add: dist-norm norm-mult nou)
  using ballg mem-ball-0 z apply blast
  apply (subst (asm) gf)
  apply (simp add: dist-norm norm-mult nou)
  apply (metis ballg mem-ball-0 mult.left-neutral norm-mult nou z, simp)
  done
qed
obtain c where c: ∧z. z ∈ ball 0 r ⇒ g z = c * z
  apply (rule exE [OF Cartan-is-linear [OF holg]])
  apply (simp-all add: ⟨0 < r⟩ ugeq)
  apply (auto simp: dist-norm norm-mult)
  done
have gr2: g (f (r/2)) = c * f(r/2)
  apply (rule c) using ⟨0 < r⟩ ballf mem-ball-0 by force
then have norm c > 0

```

```

using ⟨0 < r⟩
by simp (metis ⟨f 0 = 0⟩ c dist-commute fg mem-ball mult-zero-left perfect-choose-dist)
then have [simp]: c ≠ 0 by auto
have xless: x < r * cmod c if 0 ≤ x x < r for x
proof –
  have x = norm (g (f (of-real x)))
  proof –
    have r > cmod (of-real x)
    by (simp add: that)
    then have complex-of-real x ∈ ball 0 r
    using mem-ball-0 by blast
    then show ?thesis
    using gf ⟨0 ≤ x⟩ by force
  qed
moreover have f (complex-of-real x) ∈ ball 0 r
  using ballf that by auto
ultimately show ?thesis
  by (metis ⟨0 < cmod c⟩ c mem-ball-0 mult.commute mult-less-cancel-left-pos norm-mult)
qed
have 11: 1 / norm c ≤ 1
  using xless by (intro c-le-1) (auto simp: divide-simps)
have  $[0 ≤ x; x < r] ⇒ cmod c * x < r$  for x
  using c [of x] ballg [of x] by (auto simp: norm-mult)
  then have norm c ≤ 1
  by (force intro: c-le-1)
moreover have 1 ≤ norm c
  using 11 by simp
ultimately have norm c = 1 by (rule antisym)
with complex-norm-eq-1-exp c show ?thesis
  by metis
qed
end

```

Bibliography

- [1] G. Cioli, G. Gentili, and M. Maggesi. A certified proof of the Cartan fixed point theorems. *J. Autom. Reason.*, 47(3):319–336, Oct. 2011.