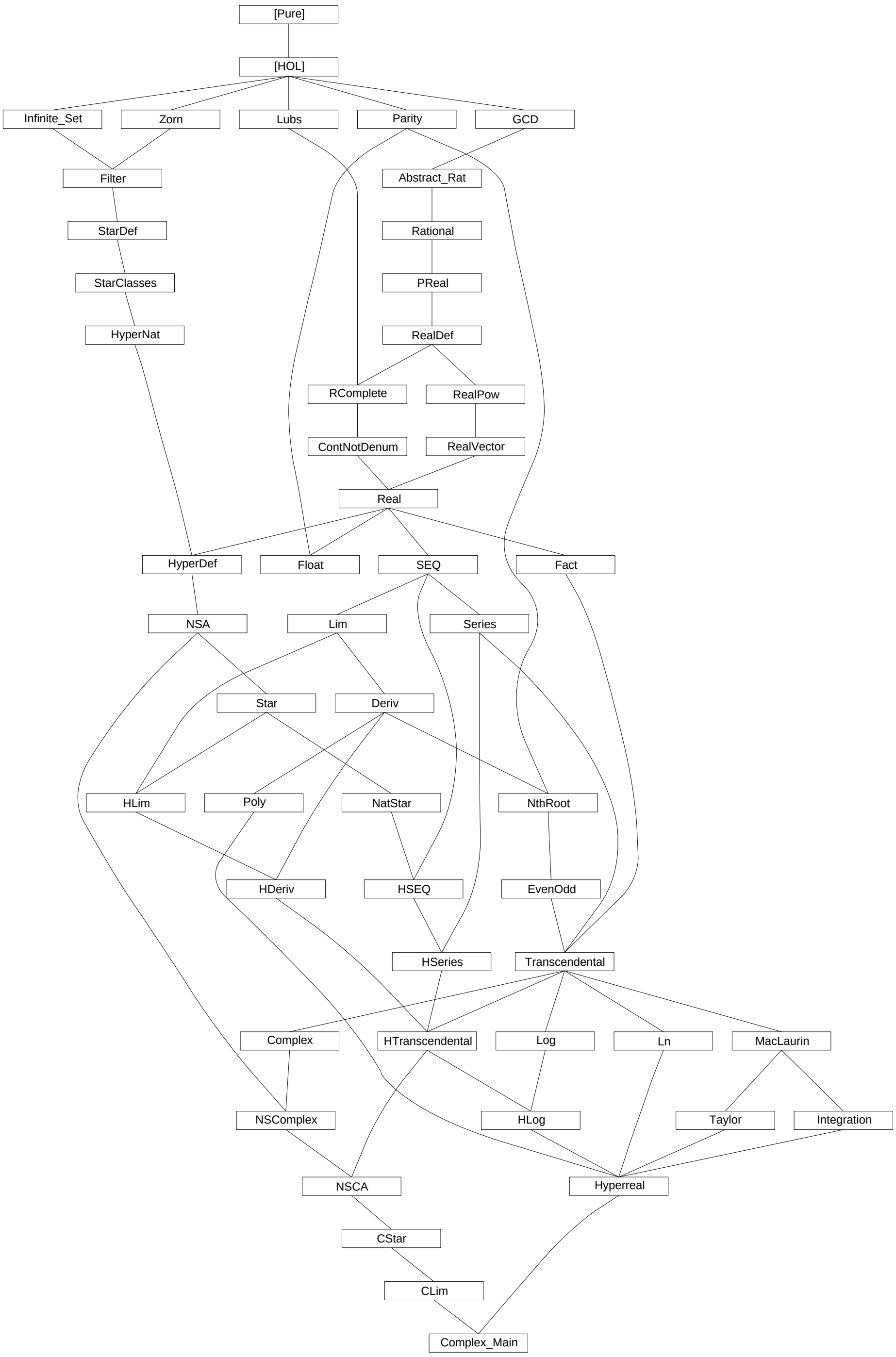
# Isabelle/HOL-Complex — Higher-Order Logic with Complex Numbers

November 22, 2007

## Contents


[Pure]
[HOL]
Infinite_Set
Zorn
Lubs
Parity
GCD
Filter
Abstract_Rat
StarDef
Rational
StarClasses
PReal
HyperNat
RealDef
RComplete
RealPow
ContNotDenum
RealVector
Real
HyperDef
Float
SEQ
Fact
NSA
Lim
Series
Star
Deriv
HLim
Poly
NatStar
NthRoot
HDeriv
HSEQ
EvenOdd
HSeries
Transcendental
Complex
HTranscendental
Log
Ln
MacLaurin
NSComplex
HLog
Taylor
Integration
NSCA
Hyperreal
CStar
CLim
Complex_Main

# 1 Lubs: Definitions of Upper Bounds and Least Upper Bounds

**theory** *Lubs*
**imports** *Main*
**begin**

Thanks to suggestions by James Margetson

**definition**
  setle :: ['a set, 'a::ord] => bool (**infixl** *<= 70) **where**
  S *<= x = (ALL y: S. y <= x)

**definition**
  setge :: ['a::ord, 'a set] => bool (**infixl** <=* 70) **where**
  x <=* S = (ALL y: S. x <= y)

**definition**
  leastP :: ['a =>bool,'a::ord] => bool **where**
  leastP P x = (P x & x <=* Collect P)

**definition**
  isUb :: ['a set, 'a set, 'a::ord] => bool **where**
  isUb R S x = (S *<= x & x: R)

**definition**
  isLub :: ['a set, 'a set, 'a::ord] => bool **where**
  isLub R S x = leastP (isUb R S) x

**definition**
  ubs :: ['a set, 'a::ord set] => 'a set **where**
  ubs R S = Collect (isUb R S)

## 1.1 Rules for the Relations *<= and <=*

**lemma** setleI: ALL y: S. y <= x ==> S *<= x
⟨proof⟩

**lemma** setleD: [| S *<= x; y: S |] ==> y <= x
⟨proof⟩

**lemma** setgeI: ALL y: S. x<= y ==> x <=* S
⟨proof⟩

**lemma** setgeD: [| x <=* S; y: S |] ==> x <= y
⟨proof⟩

## 1.2 Rules about the Operators *leastP*, *ub* and *lub*

**lemma** leastPD1: leastP P x ==> P x

⟨proof⟩

**lemma** *leastPD2*: leastP P x ==> x <=* Collect P
⟨proof⟩

**lemma** *leastPD3*: [| leastP P x; y: Collect P |] ==> x <= y
⟨proof⟩

**lemma** *isLubD1*: isLub R S x ==> S *<= x
⟨proof⟩

**lemma** *isLubD1a*: isLub R S x ==> x: R
⟨proof⟩

**lemma** *isLub-isUb*: isLub R S x ==> isUb R S x
⟨proof⟩

**lemma** *isLubD2*: [| isLub R S x; y : S |] ==> y <= x
⟨proof⟩

**lemma** *isLubD3*: isLub R S x ==> leastP(isUb R S) x
⟨proof⟩

**lemma** *isLubI1*: leastP(isUb R S) x ==> isLub R S x
⟨proof⟩

**lemma** *isLubI2*: [| isUb R S x; x <=* Collect (isUb R S) |] ==> isLub R S x
⟨proof⟩

**lemma** *isUbD*: [| isUb R S x; y : S |] ==> y <= x
⟨proof⟩

**lemma** *isUbD2*: isUb R S x ==> S *<= x
⟨proof⟩

**lemma** *isUbD2a*: isUb R S x ==> x: R
⟨proof⟩

**lemma** *isUbI*: [| S *<= x; x: R |] ==> isUb R S x
⟨proof⟩

**lemma** *isLub-le-isUb*: [| isLub R S x; isUb R S y |] ==> x <= y
⟨proof⟩

**lemma** *isLub-ubs*: isLub R S x ==> x <=* ubs R S
⟨proof⟩

**end**

# 2 GCD: The Greatest Common Divisor

**theory** *GCD*
**imports** *Main*
**begin**

See [?].

## 2.1 Specification of GCD on nats

**definition**
$is\text{-}gcd :: nat \Rightarrow nat \Rightarrow nat \Rightarrow bool$ **where** — *gcd* as a relation
$is\text{-}gcd\ p\ m\ n \longleftrightarrow p\ dvd\ m \wedge p\ dvd\ n \wedge$
$(\forall d.\ d\ dvd\ m \longrightarrow d\ dvd\ n \longrightarrow d\ dvd\ p)$

Uniqueness

**lemma** *is-gcd-unique*: $is\text{-}gcd\ m\ a\ b \Longrightarrow is\text{-}gcd\ n\ a\ b \Longrightarrow m = n$
$\langle proof \rangle$

Connection to divides relation

**lemma** *is-gcd-dvd*: $is\text{-}gcd\ m\ a\ b \Longrightarrow k\ dvd\ a \Longrightarrow k\ dvd\ b \Longrightarrow k\ dvd\ m$
$\langle proof \rangle$

Commutativity

**lemma** *is-gcd-commute*: $is\text{-}gcd\ k\ m\ n = is\text{-}gcd\ k\ n\ m$
$\langle proof \rangle$

## 2.2 GCD on nat by Euclid's algorithm

**fun**
$gcd :: nat \times nat => nat$
**where**
$gcd\ (m,\ n) = (if\ n = 0\ then\ m\ else\ gcd\ (n,\ m\ mod\ n))$

**lemma** *gcd-induct*:
**fixes** $m\ n :: nat$
**assumes** $\bigwedge m.\ P\ m\ 0$
**and** $\bigwedge m\ n.\ 0 < n \Longrightarrow P\ n\ (m\ mod\ n) \Longrightarrow P\ m\ n$
**shows** $P\ m\ n$
$\langle proof \rangle$

**lemma** *gcd-0* [*simp*]: $gcd\ (m,\ 0) = m$
$\langle proof \rangle$

**lemma** *gcd-0-left* [*simp*]: $gcd\ (0,\ m) = m$
$\langle proof \rangle$

**lemma** *gcd-non-0*: $n > 0 \Longrightarrow gcd\ (m,\ n) = gcd\ (n,\ m\ mod\ n)$

⟨*proof*⟩

**lemma** *gcd-1* [*simp*]: *gcd* (*m*, *Suc 0*) = *1*
⟨*proof*⟩

**declare** *gcd.simps* [*simp del*]

*gcd* (*m*, *n*) divides *m* and *n*. The conjunctions don't seem provable separately.

**lemma** *gcd-dvd1* [*iff*]: *gcd* (*m*, *n*) *dvd m*
**and** *gcd-dvd2* [*iff*]: *gcd* (*m*, *n*) *dvd n*
⟨*proof*⟩

Maximality: for all *m*, *n*, *k* naturals, if *k* divides *m* and *k* divides *n* then *k* divides *gcd* (*m*, *n*).

**lemma** *gcd-greatest*: *k dvd m* $\Longrightarrow$ *k dvd n* $\Longrightarrow$ *k dvd gcd* (*m*, *n*)
⟨*proof*⟩

Function gcd yields the Greatest Common Divisor.

**lemma** *is-gcd*: *is-gcd* (*gcd* (*m*, *n*)) *m n*
⟨*proof*⟩

## 2.3 Derived laws for GCD

**lemma** *gcd-greatest-iff* [*iff*]: *k dvd gcd* (*m*, *n*) $\longleftrightarrow$ *k dvd m* $\wedge$ *k dvd n*
⟨*proof*⟩

**lemma** *gcd-zero*: *gcd* (*m*, *n*) = *0* $\longleftrightarrow$ *m* = *0* $\wedge$ *n* = *0*
⟨*proof*⟩

**lemma** *gcd-commute*: *gcd* (*m*, *n*) = *gcd* (*n*, *m*)
⟨*proof*⟩

**lemma** *gcd-assoc*: *gcd* (*gcd* (*k*, *m*), *n*) = *gcd* (*k*, *gcd* (*m*, *n*))
⟨*proof*⟩

**lemma** *gcd-1-left* [*simp*]: *gcd* (*Suc 0*, *m*) = *1*
⟨*proof*⟩

Multiplication laws

**lemma** *gcd-mult-distrib2*: *k* * *gcd* (*m*, *n*) = *gcd* (*k* * *m*, *k* * *n*)
— [**?**, page 27]
⟨*proof*⟩

**lemma** *gcd-mult* [*simp*]: *gcd* (*k*, *k* * *n*) = *k*
⟨*proof*⟩

**lemma** *gcd-self* [*simp*]: $gcd\ (k,\ k) = k$
⟨*proof*⟩

**lemma** *relprime-dvd-mult*: $gcd\ (k,\ n) = 1 \implies k\ dvd\ m * n \implies k\ dvd\ m$
⟨*proof*⟩

**lemma** *relprime-dvd-mult-iff*: $gcd\ (k,\ n) = 1 \implies (k\ dvd\ m * n) = (k\ dvd\ m)$
⟨*proof*⟩

**lemma** *gcd-mult-cancel*: $gcd\ (k,\ n) = 1 \implies gcd\ (k * m,\ n) = gcd\ (m,\ n)$
⟨*proof*⟩

Addition laws

**lemma** *gcd-add1* [*simp*]: $gcd\ (m + n,\ n) = gcd\ (m,\ n)$
⟨*proof*⟩

**lemma** *gcd-add2* [*simp*]: $gcd\ (m,\ m + n) = gcd\ (m,\ n)$
⟨*proof*⟩

**lemma** *gcd-add2'* [*simp*]: $gcd\ (m,\ n + m) = gcd\ (m,\ n)$
⟨*proof*⟩

**lemma** *gcd-add-mult*: $gcd\ (m,\ k * m + n) = gcd\ (m,\ n)$
⟨*proof*⟩

**lemma** *gcd-dvd-prod*: $gcd\ (m,\ n)\ dvd\ m * n$
⟨*proof*⟩

Division by gcd yields rrelatively primes.

**lemma** *div-gcd-relprime*:
**assumes** *nz*: $a \neq 0 \vee b \neq 0$
**shows** $gcd\ (a\ div\ gcd(a,b),\ b\ div\ gcd(a,b)) = 1$
⟨*proof*⟩

## 2.4 LCM defined by GCD

**definition**
$lcm :: nat \times nat \Rightarrow nat$
**where**
$lcm = (\lambda(m,\ n).\ m * n\ div\ gcd\ (m,\ n))$

**lemma** *lcm-def*:
$lcm\ (m,\ n) = m * n\ div\ gcd\ (m,\ n)$
⟨*proof*⟩

**lemma** *prod-gcd-lcm*:
$m * n = gcd\ (m,\ n) * lcm\ (m,\ n)$

⟨*proof*⟩

**lemma** *lcm-0* [*simp*]: *lcm* (*m*, *0*) = *0*
⟨*proof*⟩

**lemma** *lcm-1* [*simp*]: *lcm* (*m*, *1*) = *m*
⟨*proof*⟩

**lemma** *lcm-0-left* [*simp*]: *lcm* (*0*, *n*) = *0*
⟨*proof*⟩

**lemma** *lcm-1-left* [*simp*]: *lcm* (*1*, *m*) = *m*
⟨*proof*⟩

**lemma** *dvd-pos*:
**fixes** *n m* :: *nat*
**assumes** *n* > *0* **and** *m dvd n*
**shows** *m* > *0*
⟨*proof*⟩

**lemma** *lcm-least*:
**assumes** *m dvd k* **and** *n dvd k*
**shows** *lcm* (*m*, *n*) *dvd k*
⟨*proof*⟩

**lemma** *lcm-dvd1* [*iff*]:
*m dvd lcm* (*m*, *n*)
⟨*proof*⟩

**lemma** *lcm-dvd2* [*iff*]:
*n dvd lcm* (*m*, *n*)
⟨*proof*⟩

## 2.5 GCD and LCM on integers

**definition**
*igcd* :: *int* ⇒ *int* ⇒ *int* **where**
*igcd i j* = *int* (*gcd* (*nat* (*abs i*), *nat* (*abs j*)))

**lemma** *igcd-dvd1* [*simp*]: *igcd i j dvd i*
⟨*proof*⟩

**lemma** *igcd-dvd2* [*simp*]: *igcd i j dvd j*
⟨*proof*⟩

**lemma** *igcd-pos*: *igcd i j* ≥ *0*
⟨*proof*⟩

**lemma** *igcd0* [*simp*]: (*igcd i j* = *0*) = (*i* = *0* ∧ *j* = *0*)

⟨*proof*⟩

**lemma** *igcd-commute*: $igcd\ i\ j = igcd\ j\ i$
⟨*proof*⟩

**lemma** *igcd-neg1* [*simp*]: $igcd\ (-\ i)\ j = igcd\ i\ j$
⟨*proof*⟩

**lemma** *igcd-neg2* [*simp*]: $igcd\ i\ (-\ j) = igcd\ i\ j$
⟨*proof*⟩

**lemma** *zrelprime-dvd-mult*: $igcd\ i\ j = 1 \Longrightarrow i\ dvd\ k * j \Longrightarrow i\ dvd\ k$
⟨*proof*⟩

**lemma** *int-nat-abs*: $int\ (nat\ (abs\ x)) = abs\ x$ ⟨*proof*⟩

**lemma** *igcd-greatest*:
**assumes** $k\ dvd\ m$ **and** $k\ dvd\ n$
**shows** $k\ dvd\ igcd\ m\ n$
⟨*proof*⟩

**lemma** *div-igcd-relprime*:
**assumes** *nz*: $a \neq 0 \vee b \neq 0$
**shows** $igcd\ (a\ div\ (igcd\ a\ b))\ (b\ div\ (igcd\ a\ b)) = 1$
⟨*proof*⟩

**definition** $ilcm = (\lambda i\ j.\ int\ (lcm(nat(abs\ i),nat(abs\ j))))$

**lemma** *dvd-ilcm-self1*[*simp*]: $i\ dvd\ ilcm\ i\ j$
⟨*proof*⟩

**lemma** *dvd-ilcm-self2*[*simp*]: $j\ dvd\ ilcm\ i\ j$
⟨*proof*⟩

**lemma** *dvd-imp-dvd-ilcm1*:
**assumes** $k\ dvd\ i$ **shows** $k\ dvd\ (ilcm\ i\ j)$
⟨*proof*⟩

**lemma** *dvd-imp-dvd-ilcm2*:
**assumes** $k\ dvd\ j$ **shows** $k\ dvd\ (ilcm\ i\ j)$
⟨*proof*⟩

**lemma** *zdvd-self-abs1*: $(d::int)\ dvd\ (abs\ d)$
⟨*proof*⟩

**lemma** *zdvd-self-abs2*: $(abs\ (d::int))\ dvd\ d$
⟨*proof*⟩

**lemma** *lcm-pos*:
  **assumes** *mpos*: $m > 0$
  **and** *npos*: $n>0$
  **shows** $lcm\ (m,n) > 0$
$\langle proof \rangle$

**lemma** *ilcm-pos*:
  **assumes** *anz*: $a \neq 0$
  **and** *bnz*: $b \neq 0$
  **shows** $0 < ilcm\ a\ b$
$\langle proof \rangle$

**end**

## 3 Abstract-Rat: Abstract rational numbers

**theory** *Abstract-Rat*
**imports** *GCD*
**begin**

**types** $Num = int \times int$

**abbreviation**
  $Num0\text{-}syn :: Num\ (0_N)$
**where** $0_N \equiv (0,\ 0)$

**abbreviation**
  $Numi\text{-}syn :: int \Rightarrow Num\ (\text{-}_N)$
**where** $i_N \equiv (i,\ 1)$

**definition**
  $isnormNum :: Num \Rightarrow bool$
**where**
  $isnormNum = (\lambda(a,b).\ (if\ a = 0\ then\ b = 0\ else\ b > 0 \wedge igcd\ a\ b = 1))$

**definition**
  $normNum :: Num \Rightarrow Num$
**where**
  $normNum = (\lambda(a,b).\ (if\ a{=}0 \vee b = 0\ then\ (0,0)\ else$
  $(let\ g = igcd\ a\ b$
   $in\ if\ b > 0\ then\ (a\ div\ g,\ b\ div\ g)\ else\ (-\ (a\ div\ g),\ -\ (b\ div\ g)))))$

**lemma** *normNum-isnormNum* [*simp*]: $isnormNum\ (normNum\ x)$
$\langle proof \rangle$

Arithmetic over Num

**definition**
$Nadd :: Num \Rightarrow Num \Rightarrow Num$ (**infixl** $+_N$ *60*)
**where**
$Nadd = (\lambda(a,b)\ (a',b').\ if\ a = 0 \lor b = 0\ then\ normNum(a',b')$
$else\ if\ a'=0 \lor b' = 0\ then\ normNum(a,b)$
$else\ normNum(a*b' + b*a',\ b*b'))$

**definition**
$Nmul :: Num \Rightarrow Num \Rightarrow Num$ (**infixl** $*_N$ *60*)
**where**
$Nmul = (\lambda(a,b)\ (a',b').\ let\ g = igcd\ (a*a')\ (b*b')$
$in\ (a*a'\ div\ g,\ b*b'\ div\ g))$

**definition**
$Nneg :: Num \Rightarrow Num$ $(\sim_N)$
**where**
$Nneg \equiv (\lambda(a,b).\ (-a,b))$

**definition**
$Nsub :: Num \Rightarrow Num \Rightarrow Num$ (**infixl** $-_N$ *60*)
**where**
$Nsub = (\lambda a\ b.\ a +_N \sim_N b)$

**definition**
$Ninv :: Num \Rightarrow Num$
**where**
$Ninv \equiv \lambda(a,b).\ if\ a < 0\ then\ (-b,\ |a|)\ else\ (b,a)$

**definition**
$Ndiv :: Num \Rightarrow Num \Rightarrow Num$ (**infixl** $\div_N$ *60*)
**where**
$Ndiv \equiv \lambda a\ b.\ a *_N Ninv\ b$

**lemma** *Nneg-normN*[*simp*]: $isnormNum\ x \Longrightarrow isnormNum\ (\sim_N x)$
⟨*proof*⟩
**lemma** *Nadd-normN*[*simp*]: $isnormNum\ (x +_N y)$
⟨*proof*⟩
**lemma** *Nsub-normN*[*simp*]: $[\![\ isnormNum\ y]\!] \Longrightarrow isnormNum\ (x -_N y)$
⟨*proof*⟩
**lemma** *Nmul-normN*[*simp*]: **assumes** *xn*:*isnormNum x* **and** *yn*: *isnormNum y*
**shows** $isnormNum\ (x *_N y)$
⟨*proof*⟩

**lemma** *Ninv-normN*[*simp*]: $isnormNum\ x \Longrightarrow isnormNum\ (Ninv\ x)$
⟨*proof*⟩

**lemma** *isnormNum-int*[*simp*]:
$isnormNum\ 0_N\ \ isnormNum\ (1::int)_N\ \ i \neq 0 \Longrightarrow isnormNum\ i_N$

⟨*proof*⟩

Relations over Num

**definition**
$Nlt0$:: $Num \Rightarrow bool$ ($0>_N$)
**where**
$Nlt0 = (\lambda(a,b).\ a < 0)$

**definition**
$Nle0$:: $Num \Rightarrow bool$ ($0\geq_N$)
**where**
$Nle0 = (\lambda(a,b).\ a \leq 0)$

**definition**
$Ngt0$:: $Num \Rightarrow bool$ ($0<_N$)
**where**
$Ngt0 = (\lambda(a,b).\ a > 0)$

**definition**
$Nge0$:: $Num \Rightarrow bool$ ($0\leq_N$)
**where**
$Nge0 = (\lambda(a,b).\ a \geq 0)$

**definition**
$Nlt$ :: $Num \Rightarrow Num \Rightarrow bool$ (**infix** $<_N$ *55*)
**where**
$Nlt = (\lambda a\ b.\ 0>_N (a -_N b))$

**definition**
$Nle$ :: $Num \Rightarrow Num \Rightarrow bool$ (**infix** $\leq_N$ *55*)
**where**
$Nle = (\lambda a\ b.\ 0\geq_N (a -_N b))$

**definition**
$INum = (\lambda(a,b).\ \textit{of-int}\ a\ /\ \textit{of-int}\ b)$

**lemma** *INum-int* [*simp*]: $INum\ i_N = ((\textit{of-int}\ i) ::'a::\textit{field})$ $INum\ 0_N = (0::'a::\textit{field})$
⟨*proof*⟩

**lemma** *isnormNum-unique*[*simp*]:
**assumes** *na*: *isnormNum x* **and** *nb*: *isnormNum y*
**shows** $((INum\ x\ ::'a::\{\textit{ring-char-0},\textit{field},\ \textit{division-by-zero}\}) = INum\ y) = (x = y)$ (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

**lemma** *isnormNum0*[*simp*]: $\textit{isnormNum}\ x \Longrightarrow (INum\ x = (0::'a::\{\textit{ring-char-0},\ \textit{field},\textit{division-by-zero}\})) = (x = 0_N)$
⟨*proof*⟩

**lemma** *of-int-div-aux*: $d$ ~= $0$ ==> ((*of-int* $x$)::'a::{*field*, *ring-char-0*}) / (*of-int* $d$) =
*of-int* ($x$ *div* $d$) + (*of-int* ($x$ *mod* $d$)) / ((*of-int* $d$)::'a)
⟨*proof*⟩

**lemma** *of-int-div*: ($d$::*int*) ~= $0$ ==> $d$ *dvd* $n$ ==>
(*of-int*($n$ *div* $d$)::'a::{*field*, *ring-char-0*}) = *of-int* $n$ / *of-int* $d$
⟨*proof*⟩

**lemma** *normNum*[*simp*]: *INum* (*normNum* $x$) = (*INum* $x$ :: 'a::{*ring-char-0*,*field*, *division-by-zero*})
⟨*proof*⟩

**lemma** *INum-normNum-iff* [*code*]: (*INum* $x$ ::'a::{*field*, *division-by-zero*, *ring-char-0*}) = *INum* $y$ $\longleftrightarrow$ *normNum* $x$ = *normNum* $y$ (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

**lemma** *Nadd*[*simp*]: *INum* ($x +_N y$) = *INum* $x$ + (*INum* $y$ :: 'a :: {*ring-char-0*,*division-by-zero*,*field*})
⟨*proof*⟩

**lemma** *Nmul*[*simp*]: *INum* ($x *_N y$) = *INum* $x$ * (*INum* $y$:: 'a :: {*ring-char-0*,*division-by-zero*,*field*})

⟨*proof*⟩

**lemma** *Nneg*[*simp*]: *INum* ($\sim_N$ $x$) = − (*INum* $x$ ::'a:: *field*)
⟨*proof*⟩

**lemma** *Nsub*[*simp*]: **shows** *INum* ($x -_N y$) = *INum* $x$ − (*INum* $y$:: 'a :: {*ring-char-0*,*division-by-zero*,*field*})
⟨*proof*⟩

**lemma** *Ninv*[*simp*]: *INum* (*Ninv* $x$) = ($1$::'a :: {*division-by-zero*,*field*}) / (*INum* $x$)
⟨*proof*⟩

**lemma** *Ndiv*[*simp*]: *INum* ($x \div_N y$) = *INum* $x$ / (*INum* $y$ ::'a :: {*ring-char-0*, *division-by-zero*,*field*}) ⟨*proof*⟩

**lemma** *Nlt0-iff* [*simp*]: **assumes** *nx*: *isnormNum* $x$
**shows** ((*INum* $x$ :: 'a :: {*ring-char-0*,*division-by-zero*,*ordered-field*})< $0$) = $0{>_N}$ $x$
⟨*proof*⟩

**lemma** *Nle0-iff* [*simp*]:**assumes** *nx*: *isnormNum* $x$
**shows** ((*INum* $x$ :: 'a :: {*ring-char-0*,*division-by-zero*,*ordered-field*}) $\leq$ $0$) = $0{\geq_N}$ $x$
⟨*proof*⟩

**lemma** *Ngt0-iff* [*simp*]:**assumes** *nx*: *isnormNum x* **shows** $((INum\ x :: 'a :: \{ring\text{-}char\text{-}0, division\text{-}by\text{-}zero, ordere$
$0) = 0 <_N x$
⟨*proof*⟩
**lemma** *Nge0-iff* [*simp*]:**assumes** *nx*: *isnormNum x*
**shows** $((INum\ x :: 'a :: \{ring\text{-}char\text{-}0, division\text{-}by\text{-}zero, ordered\text{-}field\}) \geq 0) = 0 \leq_N$
$x$
⟨*proof*⟩

**lemma** *Nlt-iff* [*simp*]: **assumes** *nx*: *isnormNum x* **and** *ny*: *isnormNum y*
**shows** $((INum\ x :: 'a :: \{ring\text{-}char\text{-}0, division\text{-}by\text{-}zero, ordered\text{-}field\}) < INum\ y)$
$= (x <_N y)$
⟨*proof*⟩

**lemma** *Nle-iff* [*simp*]: **assumes** *nx*: *isnormNum x* **and** *ny*: *isnormNum y*
**shows** $((INum\ x :: 'a :: \{ring\text{-}char\text{-}0, division\text{-}by\text{-}zero, ordered\text{-}field\}) \leq INum\ y)$
$= (x \leq_N y)$
⟨*proof*⟩

**lemma** *Nadd-commute*: $x +_N y = y +_N x$
⟨*proof*⟩

**lemma**[*simp*]: $(0,\ b) +_N y = normNum\ y$ $(a,\ 0) +_N y = normNum\ y$
$x +_N (0,\ b) = normNum\ x$ $x +_N (a,\ 0) = normNum\ x$
⟨*proof*⟩

**lemma** *normNum-nilpotent-aux*[*simp*]: **assumes** *nx*: *isnormNum x*
**shows** $normNum\ x = x$
⟨*proof*⟩

**lemma** *normNum-nilpotent*[*simp*]: $normNum\ (normNum\ x) = normNum\ x$
⟨*proof*⟩
**lemma** *normNum0*[*simp*]: $normNum\ (0,b) = 0_N$ $normNum\ (a,0) = 0_N$
⟨*proof*⟩
**lemma** *normNum-Nadd*: $normNum\ (x +_N y) = x +_N y$ ⟨*proof*⟩
**lemma** *Nadd-normNum1* [*simp*]: $normNum\ x +_N y = x +_N y$
⟨*proof*⟩
**lemma** *Nadd-normNum2* [*simp*]: $x +_N normNum\ y = x +_N y$
⟨*proof*⟩

**lemma** *Nadd-assoc*: $x +_N y +_N z = x +_N (y +_N z)$
⟨*proof*⟩

**lemma** *Nmul-commute*: $isnormNum\ x \Longrightarrow isnormNum\ y \Longrightarrow x *_N y = y *_N x$
⟨*proof*⟩

**lemma** *Nmul-assoc*: **assumes** *nx*: *isnormNum x* **and** *ny*:*isnormNum y* **and** *nz*:*isnormNum*
*z*
**shows** $x *_N y *_N z = x *_N (y *_N z)$
⟨*proof*⟩

**lemma** *Nsub0*: **assumes** $x$: *isnormNum* $x$ **and** $y$:*isnormNum* $y$ **shows** $(x -_N y = 0_N) = (x = y)$
$\langle proof \rangle$

**lemma** *Nmul0*[*simp*]: $c *_N 0_N = 0_N$ $0_N *_N c = 0_N$
$\langle proof \rangle$

**lemma** *Nmul-eq0*[*simp*]: **assumes** *nx*:*isnormNum* $x$ **and** *ny*: *isnormNum* $y$
**shows** $(x *_N y = 0_N) = (x = 0_N \vee y = 0_N)$
$\langle proof \rangle$
**lemma** *Nneg-Nneg*[*simp*]: $\sim_N (\sim_N c) = c$
$\langle proof \rangle$

**lemma** *Nmul1*[*simp*]:
*isnormNum* $c \Longrightarrow 1_N *_N c = c$
*isnormNum* $c \Longrightarrow c *_N 1_N = c$
$\langle proof \rangle$

**end**

# 4 Rational: Rational numbers

**theory** *Rational*
**imports** *Abstract-Rat*
**uses** (*rat-arith.ML*)
**begin**

## 4.1 Rational numbers

### 4.1.1 Equivalence of fractions

**definition**
*fraction* :: $(int \times int)$ *set* **where**
*fraction* $= \{x.\ snd\ x \neq 0\}$

**definition**
*ratrel* :: $((int \times int) \times (int \times int))$ *set* **where**
*ratrel* $= \{(x,y).\ snd\ x \neq 0 \wedge snd\ y \neq 0 \wedge fst\ x * snd\ y = fst\ y * snd\ x\}$

**lemma** *fraction-iff* [*simp*]: $(x \in fraction) = (snd\ x \neq 0)$
$\langle proof \rangle$

**lemma** *ratrel-iff* [*simp*]:
$((x,y) \in ratrel) =$
$(snd\ x \neq 0 \wedge snd\ y \neq 0 \wedge fst\ x * snd\ y = fst\ y * snd\ x)$
$\langle proof \rangle$

**lemma** *refl-ratrel*: *refl fraction ratrel*

⟨*proof*⟩

**lemma** *sym-ratrel*: *sym ratrel*
⟨*proof*⟩

**lemma** *trans-ratrel-lemma*:
  **assumes** *1*: $a * b' = a' * b$
  **assumes** *2*: $a' * b'' = a'' * b'$
  **assumes** *3*: $b' \neq (0{::}int)$
  **shows** $a * b'' = a'' * b$
⟨*proof*⟩

**lemma** *trans-ratrel*: *trans ratrel*
⟨*proof*⟩

**lemma** *equiv-ratrel*: *equiv fraction ratrel*
⟨*proof*⟩

**lemmas** *equiv-ratrel-iff* [*iff*] = *eq-equiv-class-iff* [*OF equiv-ratrel*]

**lemma** *equiv-ratrel-iff2*:
  $[\![snd\ x \neq 0;\ snd\ y \neq 0]\!]$
    $\Longrightarrow (ratrel\ ``\ \{x\} = ratrel\ ``\ \{y\}) = ((x,y) \in ratrel)$
⟨*proof*⟩

### 4.1.2 The type of rational numbers

**typedef** (*Rat*) *rat* = *fraction*//*ratrel*
⟨*proof*⟩

**lemma** *ratrel-in-Rat* [*simp*]: $snd\ x \neq 0 \Longrightarrow ratrel``\{x\} \in Rat$
⟨*proof*⟩

**declare** *Abs-Rat-inject* [*simp*]  *Abs-Rat-inverse* [*simp*]

**definition**
  *Fract* :: *int* ⇒ *int* ⇒ *rat* **where**
  [*code func del*]: $Fract\ a\ b = Abs\text{-}Rat\ (ratrel``\{(a,b)\})$

**lemma** *Fract-zero*:
  *Fract k 0* = *Fract l 0*
  ⟨*proof*⟩

**theorem** *Rat-cases* [*case-names Fract*, *cases type*: *rat*]:
    $(!!a\ b.\ q = Fract\ a\ b ==> b \neq 0 ==> C) ==> C$
  ⟨*proof*⟩

**theorem** *Rat-induct* [*case-names Fract*, *induct type*: *rat*]:

$(!!a\ b.\ b \neq 0 ==> P\ (Fract\ a\ b)) ==> P\ q$
⟨*proof*⟩

### 4.1.3 Congruence lemmas

**lemma** *add-congruent2*:
$(\lambda x\ y.\ ratrel``\{(fst\ x * snd\ y + fst\ y * snd\ x,\ snd\ x * snd\ y)\})$
*respects2 ratrel*
⟨*proof*⟩

**lemma** *minus-congruent*:
$(\lambda x.\ ratrel``\{(-\ fst\ x,\ snd\ x)\})$ *respects ratrel*
⟨*proof*⟩

**lemma** *mult-congruent2*:
$(\lambda x\ y.\ ratrel``\{(fst\ x * fst\ y,\ snd\ x * snd\ y)\})$ *respects2 ratrel*
⟨*proof*⟩

**lemma** *inverse-congruent*:
$(\lambda x.\ ratrel``\{if\ fst\ x{=}0\ then\ (0,1)\ else\ (snd\ x,\ fst\ x)\})$ *respects ratrel*
⟨*proof*⟩

**lemma** *le-congruent2*:
$(\lambda x\ y.\ \{(fst\ x * snd\ y)*(snd\ x * snd\ y) \leq (fst\ y * snd\ x)*(snd\ x * snd\ y)\})$
*respects2 ratrel*
⟨*proof*⟩

**lemmas** *UN-ratrel* = *UN-equiv-class* [*OF equiv-ratrel*]
**lemmas** *UN-ratrel2* = *UN-equiv-class2* [*OF equiv-ratrel equiv-ratrel*]

### 4.1.4 Standard operations on rational numbers

**instance** *rat* :: *zero*
*Zero-rat-def*: $0 == Fract\ 0\ 1$ ⟨*proof*⟩
**lemmas** [*code func del*] = *Zero-rat-def*

**instance** *rat* :: *one*
*One-rat-def*: $1 == Fract\ 1\ 1$ ⟨*proof*⟩
**lemmas** [*code func del*] = *One-rat-def*

**instance** *rat* :: *plus*
*add-rat-def*:
$q + r ==$
$Abs\text{-}Rat\ (\bigcup x \in Rep\text{-}Rat\ q.\ \bigcup y \in Rep\text{-}Rat\ r.$
$ratrel``\{(fst\ x * snd\ y + fst\ y * snd\ x,\ snd\ x * snd\ y)\})$ ⟨*proof*⟩
**lemmas** [*code func del*] = *add-rat-def*

**instance** *rat* :: *minus*
*minus-rat-def*:
$-\ q == Abs\text{-}Rat\ (\bigcup x \in Rep\text{-}Rat\ q.\ ratrel``\{(-\ fst\ x,\ snd\ x)\})$

*diff-rat-def*: $q - r == q + - (r::rat)$ ⟨*proof*⟩
**lemmas** [*code func del*] = *minus-rat-def diff-rat-def*

**instance** *rat* :: *times*
*mult-rat-def*:
$q * r ==$
*Abs-Rat* ($\bigcup x \in$ *Rep-Rat* $q$. $\bigcup y \in$ *Rep-Rat* $r$.
*ratrel*``{(*fst* $x$ * *fst* $y$, *snd* $x$ * *snd* $y$)}) ⟨*proof*⟩
**lemmas** [*code func del*] = *mult-rat-def*

**instance** *rat* :: *inverse*
*inverse-rat-def*:
*inverse* $q$ ==
*Abs-Rat* ($\bigcup x \in$ *Rep-Rat* $q$.
*ratrel*``{*if fst* $x$=0 *then* (0,1) *else* (*snd* $x$, *fst* $x$)})
*divide-rat-def*: $q$ / $r$ == $q$ * *inverse* ($r$::*rat*) ⟨*proof*⟩
**lemmas** [*code func del*] = *inverse-rat-def divide-rat-def*

**instance** *rat* :: *ord*
*le-rat-def*:
$q \leq r$ == *contents* ($\bigcup x \in$ *Rep-Rat* $q$. $\bigcup y \in$ *Rep-Rat* $r$.
{(*fst* $x$ * *snd* $y$)*(*snd* $x$ * *snd* $y$) $\leq$ (*fst* $y$ * *snd* $x$)*(*snd* $x$ * *snd* $y$)})
*less-rat-def*: ($z$ < ($w$::*rat*)) == ($z \leq w$ & $z \neq w$) ⟨*proof*⟩
**lemmas** [*code func del*] = *le-rat-def less-rat-def*

**instance** *rat* :: *abs*
*abs-rat-def*: $|q|$ == *if* $q$ < 0 *then* $-q$ *else* ($q$::*rat*) ⟨*proof*⟩

**instance** *rat* :: *sgn*
*sgn-rat-def*: *sgn*($q$::*rat*) == (*if* $q$=0 *then* 0 *else if* 0<$q$ *then* 1 *else* $-$ 1) ⟨*proof*⟩

**instance** *rat* :: *power* ⟨*proof*⟩

**primrec** (*rat*)
*rat-power-0*: $q$ ^ 0 = 1
*rat-power-Suc*: $q$ ^ (*Suc* $n$) = ($q$::*rat*) * ($q$ ^ $n$)

**theorem** *eq-rat*: $b \neq 0$ ==> $d \neq 0$ ==>
(*Fract* $a$ $b$ = *Fract* $c$ $d$) = ($a * d = c * b$)
⟨*proof*⟩

**theorem** *add-rat*: $b \neq 0$ ==> $d \neq 0$ ==>
*Fract* $a$ $b$ + *Fract* $c$ $d$ = *Fract* ($a * d + c * b$) ($b * d$)
⟨*proof*⟩

**theorem** *minus-rat*: $b \neq 0$ ==> $-$(*Fract* $a$ $b$) = *Fract* ($-a$) $b$
⟨*proof*⟩

**theorem** *diff-rat*: $b \neq 0$ ==> $d \neq 0$ ==>

$\quad Fract\ a\ b - Fract\ c\ d = Fract\ (a * d - c * b)\ (b * d)$
$\langle proof \rangle$

**theorem** *mult-rat*: $b \neq 0 ==> d \neq 0 ==>$
$Fract\ a\ b * Fract\ c\ d = Fract\ (a * c)\ (b * d)$
$\langle proof \rangle$

**theorem** *inverse-rat*: $a \neq 0 ==> b \neq 0 ==>$
$inverse\ (Fract\ a\ b) = Fract\ b\ a$
$\langle proof \rangle$

**theorem** *divide-rat*: $c \neq 0 ==> b \neq 0 ==> d \neq 0 ==>$
$Fract\ a\ b\ /\ Fract\ c\ d = Fract\ (a * d)\ (b * c)$
$\langle proof \rangle$

**theorem** *le-rat*: $b \neq 0 ==> d \neq 0 ==>$
$(Fract\ a\ b \leq Fract\ c\ d) = ((a * d) * (b * d) \leq (c * b) * (b * d))$
$\langle proof \rangle$

**theorem** *less-rat*: $b \neq 0 ==> d \neq 0 ==>$
$(Fract\ a\ b < Fract\ c\ d) = ((a * d) * (b * d) < (c * b) * (b * d))$
$\langle proof \rangle$

**theorem** *abs-rat*: $b \neq 0 ==> |Fract\ a\ b| = Fract\ |a|\ |b|$
$\langle proof \rangle$

### 4.1.5 The ordered field of rational numbers

**instance** *rat* :: *field*
$\langle proof \rangle$

**instance** *rat* :: *linorder*
$\langle proof \rangle$

**instance** *rat* :: *distrib-lattice*
$inf\ r\ s \equiv min\ r\ s$
$sup\ r\ s \equiv max\ r\ s$
$\langle proof \rangle$

**instance** *rat* :: *ordered-field*
$\langle proof \rangle$

**instance** *rat* :: *division-by-zero*
$\langle proof \rangle$

**instance** *rat* :: *recpower*
$\langle proof \rangle$

### 4.2 Various Other Results

**lemma** *minus-rat-cancel* [*simp*]: $b \neq 0 ==> \mathit{Fract}\ (-a)\ (-b) = \mathit{Fract}\ a\ b$
⟨*proof*⟩

**theorem** *Rat-induct-pos* [*case-names Fract*, *induct type*: *rat*]:
  **assumes** *step*: $!!a\ b.\ 0 < b ==> P\ (\mathit{Fract}\ a\ b)$
    **shows** $P\ q$
⟨*proof*⟩

**lemma** *zero-less-Fract-iff*:
    $0 < b ==> (0 < \mathit{Fract}\ a\ b) = (0 < a)$
⟨*proof*⟩

**lemma** *Fract-add-one*: $n \neq 0 ==> \mathit{Fract}\ (m + n)\ n = \mathit{Fract}\ m\ n + 1$
⟨*proof*⟩

**lemma** *of-nat-rat*: *of-nat* $k$ = *Fract* (*of-nat* $k$) *1*
⟨*proof*⟩

**lemma** *of-int-rat*: *of-int* $k$ = *Fract* $k$ *1*
⟨*proof*⟩

**lemma** *Fract-of-nat-eq*: *Fract* (*of-nat* $k$) *1* = *of-nat* $k$
⟨*proof*⟩

**lemma** *Fract-of-int-eq*: *Fract* $k$ *1* = *of-int* $k$
⟨*proof*⟩

**lemma** *Fract-of-int-quotient*: *Fract* $k$ $l$ = (*if* $l$ = *0* *then Fract 1 0 else of-int* $k$ / *of-int* $l$)
⟨*proof*⟩

### 4.3 Numerals and Arithmetic

**instance** *rat* :: *number*
  *rat-number-of-def*: (*number-of* $w$ :: *rat*) ≡ *of-int* $w$ ⟨*proof*⟩

**instance** *rat* :: *number-ring*
  ⟨*proof*⟩

⟨*ML*⟩

### 4.4 Embedding from Rationals to other Fields

**class** *field-char-0* = *field* + *ring-char-0*

**instance** *ordered-field* < *field-char-0* ⟨*proof*⟩

**definition**

$of\text{-}rat :: rat \Rightarrow 'a::field\text{-}char\text{-}0$
**where**
$[code\ func\ del]$: $of\text{-}rat\ q = contents\ (\bigcup(a,b) \in Rep\text{-}Rat\ q.\ \{of\text{-}int\ a\ /\ of\text{-}int\ b\})$

**lemma** *of-rat-congruent*:
$(\lambda(a,\ b).\ \{of\text{-}int\ a\ /\ of\text{-}int\ b::'a::field\text{-}char\text{-}0\})\ respects\ ratrel$
⟨*proof*⟩

**lemma** *of-rat-rat*:
$b \neq 0 \Longrightarrow of\text{-}rat\ (Fract\ a\ b) = of\text{-}int\ a\ /\ of\text{-}int\ b$
⟨*proof*⟩

**lemma** *of-rat-0* [*simp*]: $of\text{-}rat\ 0 = 0$
⟨*proof*⟩

**lemma** *of-rat-1* [*simp*]: $of\text{-}rat\ 1 = 1$
⟨*proof*⟩

**lemma** *of-rat-add*: $of\text{-}rat\ (a + b) = of\text{-}rat\ a + of\text{-}rat\ b$
⟨*proof*⟩

**lemma** *of-rat-minus*: $of\text{-}rat\ (-\ a) = -\ of\text{-}rat\ a$
⟨*proof*⟩

**lemma** *of-rat-diff*: $of\text{-}rat\ (a - b) = of\text{-}rat\ a - of\text{-}rat\ b$
⟨*proof*⟩

**lemma** *of-rat-mult*: $of\text{-}rat\ (a * b) = of\text{-}rat\ a * of\text{-}rat\ b$
⟨*proof*⟩

**lemma** *nonzero-of-rat-inverse*:
$a \neq 0 \Longrightarrow of\text{-}rat\ (inverse\ a) = inverse\ (of\text{-}rat\ a)$
⟨*proof*⟩

**lemma** *of-rat-inverse*:
$(of\text{-}rat\ (inverse\ a)::'a::\{field\text{-}char\text{-}0,division\text{-}by\text{-}zero\}) =$
$inverse\ (of\text{-}rat\ a)$
⟨*proof*⟩

**lemma** *nonzero-of-rat-divide*:
$b \neq 0 \Longrightarrow of\text{-}rat\ (a\ /\ b) = of\text{-}rat\ a\ /\ of\text{-}rat\ b$
⟨*proof*⟩

**lemma** *of-rat-divide*:
$(of\text{-}rat\ (a\ /\ b)::'a::\{field\text{-}char\text{-}0,division\text{-}by\text{-}zero\})$
$= of\text{-}rat\ a\ /\ of\text{-}rat\ b$
⟨*proof*⟩

**lemma** *of-rat-power*:

$(\textit{of-rat}\ (a\ \hat{}\ n)::'a::\{\textit{field-char-0},\textit{recpower}\}) = \textit{of-rat}\ a\ \hat{}\ n$
⟨*proof*⟩

**lemma** *of-rat-eq-iff* [*simp*]: $(\textit{of-rat}\ a = \textit{of-rat}\ b) = (a = b)$
⟨*proof*⟩

**lemmas** *of-rat-eq-0-iff* [*simp*] = *of-rat-eq-iff* [*of - 0*, *simplified*]

**lemma** *of-rat-eq-id* [*simp*]: $\textit{of-rat} = (\textit{id} :: \textit{rat} \Rightarrow \textit{rat})$
⟨*proof*⟩

Collapse nested embeddings

**lemma** *of-rat-of-nat-eq* [*simp*]: $\textit{of-rat}\ (\textit{of-nat}\ n) = \textit{of-nat}\ n$
⟨*proof*⟩

**lemma** *of-rat-of-int-eq* [*simp*]: $\textit{of-rat}\ (\textit{of-int}\ z) = \textit{of-int}\ z$
⟨*proof*⟩

**lemma** *of-rat-number-of-eq* [*simp*]:
$\textit{of-rat}\ (\textit{number-of}\ w) = (\textit{number-of}\ w :: 'a::\{\textit{number-ring},\textit{field-char-0}\})$
⟨*proof*⟩

**lemmas** *zero-rat* = *Zero-rat-def*
**lemmas** *one-rat* = *One-rat-def*

**abbreviation**
$\textit{rat-of-nat} :: \textit{nat} \Rightarrow \textit{rat}$
**where**
$\textit{rat-of-nat} \equiv \textit{of-nat}$

**abbreviation**
$\textit{rat-of-int} :: \textit{int} \Rightarrow \textit{rat}$
**where**
$\textit{rat-of-int} \equiv \textit{of-int}$

## 4.5 Implementation of rational numbers as pairs of integers

**definition**
$\textit{Rational} :: \textit{int} \times \textit{int} \Rightarrow \textit{rat}$
**where**
$\textit{Rational} = \textit{INum}$

**code-datatype** *Rational*

**lemma** *Rational-simp*:
$\textit{Rational}\ (k, l) = \textit{rat-of-int}\ k\ /\ \textit{rat-of-int}\ l$
⟨*proof*⟩

**lemma** *Rational-zero* [*simp*]: $\textit{Rational}\ 0_N = 0$

⟨*proof*⟩

**lemma** *Rational-lit* [*simp*]: *Rational* $i_N$ = *rat-of-int i*
⟨*proof*⟩

**lemma** *zero-rat-code* [*code*, *code unfold*]:
*0* = *Rational* $0_N$ ⟨*proof*⟩

**lemma** *zero-rat-code* [*code*, *code unfold*]:
*1* = *Rational* $1_N$ ⟨*proof*⟩

**lemma** [*code*, *code unfold*]:
*number-of k* = *rat-of-int* (*number-of k*)
⟨*proof*⟩

**definition**
[*code func del*]: *Fract′* (*b*::*bool*) *k l* = *Fract k l*

**lemma** [*code*]:
*Fract k l* = *Fract′* ($l \neq 0$) *k l*
⟨*proof*⟩

**lemma** [*code*]:
*Fract′ True k l* = (*if* $l \neq 0$ *then Rational* (*k*, *l*) *else Fract 1 0*)
⟨*proof*⟩

**lemma** [*code*]:
*of-rat* (*Rational* (*k*, *l*)) = (*if* $l \neq 0$ *then of-int k* / *of-int l else 0*)
⟨*proof*⟩

**instance** *rat* :: *eq* ⟨*proof*⟩

**lemma** *rat-eq-code* [*code*]: *Rational x* = *Rational y* ⟷ *normNum x* = *normNum y*
⟨*proof*⟩

**lemma** *rat-less-eq-code* [*code*]: *Rational x* ≤ *Rational y* ⟷ *normNum x* $\leq_N$ *normNum y*
⟨*proof*⟩

**lemma** *rat-less-code* [*code*]: *Rational x* < *Rational y* ⟷ *normNum x* $<_N$ *normNum y*
⟨*proof*⟩

**lemma** *rat-add-code* [*code*]: *Rational x* + *Rational y* = *Rational* ($x +_N y$)
⟨*proof*⟩

**lemma** *rat-mul-code* [*code*]: *Rational x* ∗ *Rational y* = *Rational* ($x *_N y$)
⟨*proof*⟩

**lemma** *rat-neg-code* [*code*]: $-$ *Rational* $x$ = *Rational* ($\sim_N$ $x$)
⟨*proof*⟩

**lemma** *rat-sub-code* [*code*]: *Rational* $x$ $-$ *Rational* $y$ = *Rational* ($x$ $-_N$ $y$)
⟨*proof*⟩

**lemma** *rat-inv-code* [*code*]: *inverse* (*Rational* $x$) = *Rational* (*Ninv* $x$)
⟨*proof*⟩

**lemma** *rat-div-code* [*code*]: *Rational* $x$ / *Rational* $y$ = *Rational* ($x$ $\div_N$ $y$)
⟨*proof*⟩

Setup for SML code generator

```
types-code
  rat ((int */ int))
attach (term-of) ⟨⟨
fun term-of-rat (p, q) =
  let
    val rT = Type (Rational.rat, [])
  in
    if q = 1 orelse p = 0 then HOLogic.mk-number rT p
    else Const (HOL.inverse-class.divide, rT --> rT --> rT) $
      HOLogic.mk-number rT p $ HOLogic.mk-number rT q
  end;
⟩⟩
attach (test) ⟨⟨
fun gen-rat i =
  let
    val p = random-range 0 i;
    val q = random-range 1 (i + 1);
    val g = Integer.gcd p q;
    val p' = p div g;
    val q' = q div g;
  in
    (if one-of [true, false] then p' else ~ p',
     if p' = 0 then 0 else q')
  end;
⟩⟩

consts-code
  Rational ((-))

consts-code
  of-int :: int ⇒ rat (⟨module⟩rat'-of'-int)
attach ⟨⟨
fun rat-of-int 0 = (0, 0)
  | rat-of-int i = (i, 1);
⟩⟩
```

**end**

# 5 PReal: Positive real numbers

**theory** *PReal*
**imports** *Rational*
**begin**

Could be generalized and moved to *Ring-and-Field*

**lemma** *add-eq-exists*: $\exists\, x.\ a{+}x = (b{::}rat)$
$\langle proof \rangle$

**definition**
$cut :: rat\ set \Rightarrow bool$ **where**
$cut\ A = (\{\} \subset A\ \&$
$A < \{r.\ 0 < r\}\ \&$
$(\forall\, y \in A.\ ((\forall\, z.\ 0{<}z\ \&\ z < y \dashrightarrow z \in A)\ \&\ (\exists\, u \in A.\ y < u))))$

**lemma** *cut-of-rat*:
**assumes** $q$: $0 < q$ **shows** $cut\ \{r{::}rat.\ 0 < r\ \&\ r < q\}$ (**is** $cut\ ?A$)
$\langle proof \rangle$

**typedef** $preal = \{A.\ cut\ A\}$
$\langle proof \rangle$

**instance** $preal :: \{ord,\ plus,\ minus,\ times,\ inverse,\ one\}$ $\langle proof \rangle$

**definition**
$preal\text{-}of\text{-}rat :: rat \Rightarrow preal$ **where**
$preal\text{-}of\text{-}rat\ q = Abs\text{-}preal\ \{x{::}rat.\ 0 < x\ \&\ x < q\}$

**definition**
$psup :: preal\ set \Rightarrow preal$ **where**
$psup\ P = Abs\text{-}preal\ (\bigcup X \in P.\ Rep\text{-}preal\ X)$

**definition**
$add\text{-}set :: [rat\ set, rat\ set] \Rightarrow rat\ set$ **where**
$add\text{-}set\ A\ B = \{w.\ \exists\, x \in A.\ \exists\, y \in B.\ w = x + y\}$

**definition**
$diff\text{-}set :: [rat\ set, rat\ set] \Rightarrow rat\ set$ **where**
$diff\text{-}set\ A\ B = \{w.\ \exists\, x.\ 0 < w\ \&\ 0 < x\ \&\ x \notin B\ \&\ x + w \in A\}$

**definition**
$mult\text{-}set :: [rat\ set, rat\ set] \Rightarrow rat\ set$ **where**
$mult\text{-}set\ A\ B = \{w.\ \exists\, x \in A.\ \exists\, y \in B.\ w = x * y\}$

**definition**
*inverse-set* :: *rat set* => *rat set* **where**
*inverse-set* $A = \{x.\ \exists y.\ 0 < x\ \&\ x < y\ \&\ inverse\ y \notin A\}$

**defs** (**overloaded**)

*preal-less-def*:
$R < S == Rep\text{-}preal\ R < Rep\text{-}preal\ S$

*preal-le-def*:
$R \leq S == Rep\text{-}preal\ R \subseteq Rep\text{-}preal\ S$

*preal-add-def*:
$R + S == Abs\text{-}preal\ (add\text{-}set\ (Rep\text{-}preal\ R)\ (Rep\text{-}preal\ S))$

*preal-diff-def*:
$R - S == Abs\text{-}preal\ (diff\text{-}set\ (Rep\text{-}preal\ R)\ (Rep\text{-}preal\ S))$

*preal-mult-def*:
$R * S == Abs\text{-}preal\ (mult\text{-}set\ (Rep\text{-}preal\ R)\ (Rep\text{-}preal\ S))$

*preal-inverse-def*:
$inverse\ R == Abs\text{-}preal\ (inverse\text{-}set\ (Rep\text{-}preal\ R))$

*preal-one-def*:
$1 == preal\text{-}of\text{-}rat\ 1$

Reduces equality on abstractions to equality on representatives

**declare** *Abs-preal-inject* [*simp*]
**declare** *Abs-preal-inverse* [*simp*]

**lemma** *rat-mem-preal*: $0 < q ==> \{r::rat.\ 0 < r\ \&\ r < q\} \in preal$
⟨*proof*⟩

**lemma** *preal-nonempty*: $A \in preal ==> \exists x\in A.\ 0 < x$
⟨*proof*⟩

**lemma** *preal-Ex-mem*: $A \in preal \Longrightarrow \exists x.\ x \in A$
⟨*proof*⟩

**lemma** *preal-imp-psubset-positives*: $A \in preal ==> A < \{r.\ 0 < r\}$
⟨*proof*⟩

**lemma** *preal-exists-bound*: $A \in preal ==> \exists x.\ 0 < x\ \&\ x \notin A$
⟨*proof*⟩

**lemma** *preal-exists-greater*: $[|\ A \in preal;\ y \in A\ |] ==> \exists u \in A.\ y < u$

⟨*proof*⟩

**lemma** *preal-downwards-closed*: $[| \ A \in preal; \ y \in A; \ 0 < z; \ z < y \ |] ==> z \in A$
⟨*proof*⟩

Relaxing the final premise

**lemma** *preal-downwards-closed'*:
$[| \ A \in preal; \ y \in A; \ 0 < z; \ z \leq y \ |] ==> z \in A$
⟨*proof*⟩

A positive fraction not in a positive real is an upper bound. Gleason p. 122 - Remark (1)

**lemma** *not-in-preal-ub*:
**assumes** $A$: $A \in preal$
**and** *notx*: $x \notin A$
**and** $y$: $y \in A$
**and** *pos*: $0 < x$
**shows** $y < x$
⟨*proof*⟩

preal lemmas instantiated to *Rep-preal X*

**lemma** *mem-Rep-preal-Ex*: $\exists x. \ x \in Rep\text{-}preal \ X$
⟨*proof*⟩

**lemma** *Rep-preal-exists-bound*: $\exists x>0. \ x \notin Rep\text{-}preal \ X$
⟨*proof*⟩

**lemmas** *not-in-Rep-preal-ub* = *not-in-preal-ub* [*OF Rep-preal*]

## 5.1 *preal-of-prat*: the Injection from prat to preal

**lemma** *rat-less-set-mem-preal*: $0 < y ==> \{u::rat. \ 0 < u \ \& \ u < y\} \in preal$
⟨*proof*⟩

**lemma** *rat-subset-imp-le*:
$[|\{u::rat. \ 0 < u \ \& \ u < x\} \subseteq \{u. \ 0 < u \ \& \ u < y\}; \ 0<x|] ==> x \leq y$
⟨*proof*⟩

**lemma** *rat-set-eq-imp-eq*:
$[|\{u::rat. \ 0 < u \ \& \ u < x\} = \{u. \ 0 < u \ \& \ u < y\};$
$0 < x; \ 0 < y|] ==> x = y$
⟨*proof*⟩

## 5.2 Properties of Ordering

**lemma** *preal-le-refl*: $w \leq (w::preal)$
⟨*proof*⟩

**lemma** *preal-le-trans*: $[| \ i \leq j; \ j \leq k \ |] ==> i \leq (k::preal)$

⟨*proof*⟩

**lemma** *preal-le-anti-sym*: $[| z \leq w; w \leq z |] ==> z = (w::preal)$
⟨*proof*⟩

**lemma** *preal-less-le*: $((w::preal) < z) = (w \leq z \ \& \ w \neq z)$
⟨*proof*⟩

**instance** *preal* :: *order*
⟨*proof*⟩

**lemma** *preal-imp-pos*: $[|A \in preal; r \in A|] ==> 0 < r$
⟨*proof*⟩

**lemma** *preal-le-linear*: $x <= y \mid y <= (x::preal)$
⟨*proof*⟩

**instance** *preal* :: *linorder*
⟨*proof*⟩

**instance** *preal* :: *distrib-lattice*
*inf* ≡ *min*
*sup* ≡ *max*
⟨*proof*⟩

## 5.3 Properties of Addition

**lemma** *preal-add-commute*: $(x::preal) + y = y + x$
⟨*proof*⟩

Lemmas for proving that addition of two positive reals gives a positive real

**lemma** *empty-psubset-nonempty*: $a \in A ==> \{\} \subset A$
⟨*proof*⟩

Part 1 of Dedekind sections definition

**lemma** *add-set-not-empty*:
$[|A \in preal; B \in preal|] ==> \{\} \subset \textit{add-set}\ A\ B$
⟨*proof*⟩

Part 2 of Dedekind sections definition. A structured version of this proof is *preal-not-mem-mult-set-Ex* below.

**lemma** *preal-not-mem-add-set-Ex*:
$[|A \in preal; B \in preal|] ==> \exists\, q>0.\ q \notin \textit{add-set}\ A\ B$
⟨*proof*⟩

**lemma** *add-set-not-rat-set*:
**assumes** $A$: $A \in preal$
**and** $B$: $B \in preal$

**shows** $add\text{-}set\ A\ B < \{r.\ 0 < r\}$
⟨proof⟩

Part 3 of Dedekind sections definition

**lemma** *add-set-lemma3*:
$[|A \in preal;\ B \in preal;\ u \in add\text{-}set\ A\ B;\ 0 < z;\ z < u|]$
$==> z \in add\text{-}set\ A\ B$
⟨proof⟩

Part 4 of Dedekind sections definition

**lemma** *add-set-lemma4*:
$[|A \in preal;\ B \in preal;\ y \in add\text{-}set\ A\ B|] ==> \exists\, u \in add\text{-}set\ A\ B.\ y < u$
⟨proof⟩

**lemma** *mem-add-set*:
$[|A \in preal;\ B \in preal|] ==> add\text{-}set\ A\ B \in preal$
⟨proof⟩

**lemma** *preal-add-assoc*: $((x::preal) + y) + z = x + (y + z)$
⟨proof⟩

**instance** *preal* :: *ab-semigroup-add*
⟨proof⟩

**lemma** *preal-add-left-commute*: $x + (y + z) = y + ((x + z)::preal)$
⟨proof⟩

Positive Real addition is an AC operator

**lemmas** *preal-add-ac* = *preal-add-assoc preal-add-commute preal-add-left-commute*

## 5.4 Properties of Multiplication

Proofs essentially same as for addition

**lemma** *preal-mult-commute*: $(x::preal) * y = y * x$
⟨proof⟩

Multiplication of two positive reals gives a positive real.

Lemmas for proving positive reals multiplication set in *preal*

Part 1 of Dedekind sections definition

**lemma** *mult-set-not-empty*:
$[|A \in preal;\ B \in preal|] ==> \{\} \subset mult\text{-}set\ A\ B$
⟨proof⟩

Part 2 of Dedekind sections definition

**lemma** *preal-not-mem-mult-set-Ex*:
**assumes** $A$: $A \in preal$

**and** $B$: $B \in preal$
**shows** $\exists\, q.\ 0 < q\ \&\ q \notin mult\text{-}set\ A\ B$
⟨*proof*⟩

**lemma** *mult-set-not-rat-set*:
**assumes** $A$: $A \in preal$
**and** $B$: $B \in preal$
**shows** $mult\text{-}set\ A\ B < \{r.\ 0 < r\}$
⟨*proof*⟩

Part 3 of Dedekind sections definition

**lemma** *mult-set-lemma3*:
$[|A \in preal;\ B \in preal;\ u \in mult\text{-}set\ A\ B;\ 0 < z;\ z < u|]$
$==> z \in mult\text{-}set\ A\ B$
⟨*proof*⟩

Part 4 of Dedekind sections definition

**lemma** *mult-set-lemma4*:
$[|A \in preal;\ B \in preal;\ y \in mult\text{-}set\ A\ B|] ==> \exists\, u \in mult\text{-}set\ A\ B.\ y < u$
⟨*proof*⟩

**lemma** *mem-mult-set*:
$[|A \in preal;\ B \in preal|] ==> mult\text{-}set\ A\ B \in preal$
⟨*proof*⟩

**lemma** *preal-mult-assoc*: $((x{::}preal) * y) * z = x * (y * z)$
⟨*proof*⟩

**instance** *preal* :: *ab-semigroup-mult*
⟨*proof*⟩

**lemma** *preal-mult-left-commute*: $x * (y * z) = y * ((x * z){::}preal)$
⟨*proof*⟩

Positive Real multiplication is an AC operator

**lemmas** *preal-mult-ac* =
*preal-mult-assoc preal-mult-commute preal-mult-left-commute*

Positive real 1 is the multiplicative identity element

**lemma** *preal-mult-1*: $(1{::}preal) * z = z$
⟨*proof*⟩

**instance** *preal* :: *comm-monoid-mult*
⟨*proof*⟩

**lemma** *preal-mult-1-right*: $z * (1{::}preal) = z$
⟨*proof*⟩

## 5.5 Distribution of Multiplication across Addition

**lemma** *mem-Rep-preal-add-iff*:
$(z \in \mathit{Rep\text{-}preal}(R{+}S)) = (\exists\, x \in \mathit{Rep\text{-}preal}\ R.\ \exists\, y \in \mathit{Rep\text{-}preal}\ S.\ z = x + y)$
⟨*proof*⟩

**lemma** *mem-Rep-preal-mult-iff*:
$(z \in \mathit{Rep\text{-}preal}(R{*}S)) = (\exists\, x \in \mathit{Rep\text{-}preal}\ R.\ \exists\, y \in \mathit{Rep\text{-}preal}\ S.\ z = x * y)$
⟨*proof*⟩

**lemma** *distrib-subset1*:
$\mathit{Rep\text{-}preal}\ (w * (x + y)) \subseteq \mathit{Rep\text{-}preal}\ (w * x + w * y)$
⟨*proof*⟩

**lemma** *preal-add-mult-distrib-mean*:
**assumes** $a$: $a \in \mathit{Rep\text{-}preal}\ w$
**and** $b$: $b \in \mathit{Rep\text{-}preal}\ w$
**and** $d$: $d \in \mathit{Rep\text{-}preal}\ x$
**and** $e$: $e \in \mathit{Rep\text{-}preal}\ y$
**shows** $\exists\, c \in \mathit{Rep\text{-}preal}\ w.\ a * d + b * e = c * (d + e)$
⟨*proof*⟩

**lemma** *distrib-subset2*:
$\mathit{Rep\text{-}preal}\ (w * x + w * y) \subseteq \mathit{Rep\text{-}preal}\ (w * (x + y))$
⟨*proof*⟩

**lemma** *preal-add-mult-distrib2*: $(w * ((x{::}\mathit{preal}) + y)) = (w * x) + (w * y)$
⟨*proof*⟩

**lemma** *preal-add-mult-distrib*: $(((x{::}\mathit{preal}) + y) * w) = (x * w) + (y * w)$
⟨*proof*⟩

**instance** *preal* :: *comm-semiring*
⟨*proof*⟩

## 5.6 Existence of Inverse, a Positive Real

**lemma** *mem-inv-set-ex*:
**assumes** $A$: $A \in \mathit{preal}$ **shows** $\exists\, x\ y.\ 0 < x\ \&\ x < y\ \&\ \mathit{inverse}\ y \notin A$
⟨*proof*⟩

Part 1 of Dedekind sections definition

**lemma** *inverse-set-not-empty*:
$A \in \mathit{preal} \Longrightarrow \{\} \subset \mathit{inverse\text{-}set}\ A$
⟨*proof*⟩

Part 2 of Dedekind sections definition

**lemma** *preal-not-mem-inverse-set-Ex*:
**assumes** $A$: $A \in \mathit{preal}$ **shows** $\exists\, q.\ 0 < q\ \&\ q \notin \mathit{inverse\text{-}set}\ A$
⟨*proof*⟩

**lemma** *inverse-set-not-rat-set*:
**assumes** $A$: $A \in preal$ **shows** $inverse\text{-}set\ A < \{r.\ 0 < r\}$
⟨*proof*⟩

Part 3 of Dedekind sections definition

**lemma** *inverse-set-lemma3*:
$[|A \in preal;\ u \in inverse\text{-}set\ A;\ 0 < z;\ z < u|]$
$==> z \in inverse\text{-}set\ A$
⟨*proof*⟩

Part 4 of Dedekind sections definition

**lemma** *inverse-set-lemma4*:
$[|A \in preal;\ y \in inverse\text{-}set\ A|] ==> \exists\, u \in inverse\text{-}set\ A.\ y < u$
⟨*proof*⟩

**lemma** *mem-inverse-set*:
$A \in preal ==> inverse\text{-}set\ A \in preal$
⟨*proof*⟩

## 5.7 Gleason's Lemma 9-3.4, page 122

**lemma** *Gleason9-34-exists*:
**assumes** $A$: $A \in preal$
**and** $\forall\, x \in A.\ x + u \in A$
**and** $0 \leq z$
**shows** $\exists\, b \in A.\ b + (of\text{-}int\ z) * u \in A$
⟨*proof*⟩

**lemma** *Gleason9-34-contra*:
**assumes** $A$: $A \in preal$
**shows** $[|\forall\, x \in A.\ x + u \in A;\ 0 < u;\ 0 < y;\ y \notin A|] ==> False$
⟨*proof*⟩

**lemma** *Gleason9-34*:
**assumes** $A$: $A \in preal$
**and** *upos*: $0 < u$
**shows** $\exists\, r \in A.\ r + u \notin A$
⟨*proof*⟩

## 5.8 Gleason's Lemma 9-3.6

**lemma** *lemma-gleason9-36*:
**assumes** $A$: $A \in preal$
**and** $x$: $1 < x$
**shows** $\exists\, r \in A.\ r*x \notin A$
⟨*proof*⟩

## 5.9 Existence of Inverse: Part 2

**lemma** *mem-Rep-preal-inverse-iff*:
$(z \in \textit{Rep-preal}(\textit{inverse } R)) =$
$(0 < z \wedge (\exists y.\ z < y \wedge \textit{inverse } y \notin \textit{Rep-preal } R))$
⟨*proof*⟩

**lemma** *Rep-preal-of-rat*:
$0 < q \Longrightarrow \textit{Rep-preal } (\textit{preal-of-rat } q) = \{x.\ 0 < x \wedge x < q\}$
⟨*proof*⟩

**lemma** *subset-inverse-mult-lemma*:
**assumes** *xpos*: $0 < x$ **and** *xless*: $x < 1$
**shows** $\exists r\ u\ y.\ 0 < r\ \&\ r < y\ \&\ \textit{inverse } y \notin \textit{Rep-preal } R\ \&$
$u \in \textit{Rep-preal } R\ \&\ x = r * u$
⟨*proof*⟩

**lemma** *subset-inverse-mult*:
$\textit{Rep-preal}(\textit{preal-of-rat } 1) \subseteq \textit{Rep-preal}(\textit{inverse } R * R)$
⟨*proof*⟩

**lemma** *inverse-mult-subset-lemma*:
**assumes** *rpos*: $0 < r$
**and** *rless*: $r < y$
**and** *notin*: $\textit{inverse } y \notin \textit{Rep-preal } R$
**and** *q*: $q \in \textit{Rep-preal } R$
**shows** $r*q < 1$
⟨*proof*⟩

**lemma** *inverse-mult-subset*:
$\textit{Rep-preal}(\textit{inverse } R * R) \subseteq \textit{Rep-preal}(\textit{preal-of-rat } 1)$
⟨*proof*⟩

**lemma** *preal-mult-inverse*: $\textit{inverse } R * R = (1::\textit{preal})$
⟨*proof*⟩

**lemma** *preal-mult-inverse-right*: $R * \textit{inverse } R = (1::\textit{preal})$
⟨*proof*⟩

Theorems needing *Gleason9-34*

**lemma** *Rep-preal-self-subset*: $\textit{Rep-preal } (R) \subseteq \textit{Rep-preal}(R + S)$
⟨*proof*⟩

**lemma** *Rep-preal-sum-not-subset*: $\sim \textit{Rep-preal } (R + S) \subseteq \textit{Rep-preal}(R)$
⟨*proof*⟩

**lemma** *Rep-preal-sum-not-eq*: $\textit{Rep-preal } (R + S) \neq \textit{Rep-preal}(R)$
⟨*proof*⟩

at last, Gleason prop. 9-3.5(iii) page 123

**lemma** *preal-self-less-add-left*: $(R::preal) < R + S$
⟨*proof*⟩

**lemma** *preal-self-less-add-right*: $(R::preal) < S + R$
⟨*proof*⟩

**lemma** *preal-not-eq-self*: $x \neq x + (y::preal)$
⟨*proof*⟩

## 5.10 Subtraction for Positive Reals

Gleason prop. 9-3.5(iv), page 123: proving $A < B \Longrightarrow \exists D.\ A + D = B$. We define the claimed $D$ and show that it is a positive real

Part 1 of Dedekind sections definition

**lemma** *diff-set-not-empty*:
$R < S ==> \{\} \subset$ *diff-set* (*Rep-preal* $S$) (*Rep-preal* $R$)
⟨*proof*⟩

Part 2 of Dedekind sections definition

**lemma** *diff-set-nonempty*:
$\exists q.\ 0 < q$ & $q \notin$ *diff-set* (*Rep-preal* $S$) (*Rep-preal* $R$)
⟨*proof*⟩

**lemma** *diff-set-not-rat-set*:
*diff-set* (*Rep-preal* $S$) (*Rep-preal* $R$) $< \{r.\ 0 < r\}$ (**is** *?lhs* $<$ *?rhs*)
⟨*proof*⟩

Part 3 of Dedekind sections definition

**lemma** *diff-set-lemma3*:
$[|R < S;\ u \in$ *diff-set* (*Rep-preal* $S$) (*Rep-preal* $R$); $0 < z;\ z < u|]$
$==> z \in$ *diff-set* (*Rep-preal* $S$) (*Rep-preal* $R$)
⟨*proof*⟩

Part 4 of Dedekind sections definition

**lemma** *diff-set-lemma4*:
$[|R < S;\ y \in$ *diff-set* (*Rep-preal* $S$) (*Rep-preal* $R$)$|]$
$==> \exists u \in$ *diff-set* (*Rep-preal* $S$) (*Rep-preal* $R$). $y < u$
⟨*proof*⟩

**lemma** *mem-diff-set*:
$R < S ==>$ *diff-set* (*Rep-preal* $S$) (*Rep-preal* $R$) $\in$ *preal*
⟨*proof*⟩

**lemma** *mem-Rep-preal-diff-iff*:
$R < S ==>$
($z \in$ *Rep-preal*$(S-R)$) =
($\exists x.\ 0 < x$ & $0 < z$ & $x \notin$ *Rep-preal* $R$ & $x + z \in$ *Rep-preal* $S$)

⟨*proof*⟩

proving that $R + D \le S$

**lemma** *less-add-left-lemma*:
  **assumes** *Rless*: $R < S$
    **and** *a*: $a \in \mathit{Rep\text{-}preal}\ R$
    **and** *cb*: $c + b \in \mathit{Rep\text{-}preal}\ S$
    **and** $c \notin \mathit{Rep\text{-}preal}\ R$
    **and** $0 < b$
    **and** $0 < c$
  **shows** $a + b \in \mathit{Rep\text{-}preal}\ S$
⟨*proof*⟩

**lemma** *less-add-left-le1*:
    $R < (S{::}\mathit{preal}) ==> R + (S{-}R) \le S$
⟨*proof*⟩

## 5.11 proving that $S \le R + D$ — trickier

**lemma** *lemma-sum-mem-Rep-preal-ex*:
    $x \in \mathit{Rep\text{-}preal}\ S ==> \exists\, e.\ 0 < e\ \&\ x + e \in \mathit{Rep\text{-}preal}\ S$
⟨*proof*⟩

**lemma** *less-add-left-lemma2*:
  **assumes** *Rless*: $R < S$
    **and** *x*: $x \in \mathit{Rep\text{-}preal}\ S$
    **and** *xnot*: $x \notin \mathit{Rep\text{-}preal}\ R$
  **shows** $\exists\, u\ v\ z.\ 0 < v\ \&\ 0 < z\ \&\ u \in \mathit{Rep\text{-}preal}\ R\ \&\ z \notin \mathit{Rep\text{-}preal}\ R\ \&$
        $z + v \in \mathit{Rep\text{-}preal}\ S\ \&\ x = u + v$
⟨*proof*⟩

**lemma** *less-add-left-le2*: $R < (S{::}\mathit{preal}) ==> S \le R + (S{-}R)$
⟨*proof*⟩

**lemma** *less-add-left*: $R < (S{::}\mathit{preal}) ==> R + (S{-}R) = S$
⟨*proof*⟩

**lemma** *less-add-left-Ex*: $R < (S{::}\mathit{preal}) ==> \exists\, D.\ R + D = S$
⟨*proof*⟩

**lemma** *preal-add-less2-mono1*: $R < (S{::}\mathit{preal}) ==> R + T < S + T$
⟨*proof*⟩

**lemma** *preal-add-less2-mono2*: $R < (S{::}\mathit{preal}) ==> T + R < T + S$
⟨*proof*⟩

**lemma** *preal-add-right-less-cancel*: $R + T < S + T ==> R < (S{::}\mathit{preal})$
⟨*proof*⟩

**lemma** *preal-add-left-less-cancel*: $T + R < T + S ==> R < (S::preal)$
⟨*proof*⟩

**lemma** *preal-add-less-cancel-right*: $((R::preal) + T < S + T) = (R < S)$
⟨*proof*⟩

**lemma** *preal-add-less-cancel-left*: $(T + (R::preal) < T + S) = (R < S)$
⟨*proof*⟩

**lemma** *preal-add-le-cancel-right*: $((R::preal) + T \leq S + T) = (R \leq S)$
⟨*proof*⟩

**lemma** *preal-add-le-cancel-left*: $(T + (R::preal) \leq T + S) = (R \leq S)$
⟨*proof*⟩

**lemma** *preal-add-less-mono*:
$[| x1 < y1; x2 < y2 |] ==> x1 + x2 < y1 + (y2::preal)$
⟨*proof*⟩

**lemma** *preal-add-right-cancel*: $(R::preal) + T = S + T ==> R = S$
⟨*proof*⟩

**lemma** *preal-add-left-cancel*: $C + A = C + B ==> A = (B::preal)$
⟨*proof*⟩

**lemma** *preal-add-left-cancel-iff*: $(C + A = C + B) = ((A::preal) = B)$
⟨*proof*⟩

**lemma** *preal-add-right-cancel-iff*: $(A + C = B + C) = ((A::preal) = B)$
⟨*proof*⟩

**lemmas** *preal-cancels* =
*preal-add-less-cancel-right preal-add-less-cancel-left*
*preal-add-le-cancel-right preal-add-le-cancel-left*
*preal-add-left-cancel-iff preal-add-right-cancel-iff*

**instance** *preal* :: *ordered-cancel-ab-semigroup-add*
⟨*proof*⟩

## 5.12 Completeness of type *preal*

Prove that supremum is a cut

Part 1 of Dedekind sections definition

**lemma** *preal-sup-set-not-empty*:
$P \neq \{\} ==> \{\} \subset (\bigcup X \in P.\ Rep\text{-}preal(X))$
⟨*proof*⟩

Part 2 of Dedekind sections definition

**lemma** *preal-sup-not-exists*:
$\forall X \in P.\ X \leq Y \ ==> \exists\, q.\ 0 < q\ \&\ q \notin (\bigcup X \in P.\ \mathit{Rep\text{-}preal}(X))$
⟨*proof*⟩

**lemma** *preal-sup-set-not-rat-set*:
$\forall X \in P.\ X \leq Y \ ==> (\bigcup X \in P.\ \mathit{Rep\text{-}preal}(X)) < \{r.\ 0 < r\}$
⟨*proof*⟩

Part 3 of Dedekind sections definition

**lemma** *preal-sup-set-lemma3*:
$[|P \neq \{\};\ \forall X \in P.\ X \leq Y;\ u \in (\bigcup X \in P.\ \mathit{Rep\text{-}preal}(X));\ 0 < z;\ z < u|]$
$==> z \in (\bigcup X \in P.\ \mathit{Rep\text{-}preal}(X))$
⟨*proof*⟩

Part 4 of Dedekind sections definition

**lemma** *preal-sup-set-lemma4*:
$[|P \neq \{\};\ \forall X \in P.\ X \leq Y;\ y \in (\bigcup X \in P.\ \mathit{Rep\text{-}preal}(X))\ |]$
$==> \exists\, u \in (\bigcup X \in P.\ \mathit{Rep\text{-}preal}(X)).\ y < u$
⟨*proof*⟩

**lemma** *preal-sup*:
$[|P \neq \{\};\ \forall X \in P.\ X \leq Y|] ==> (\bigcup X \in P.\ \mathit{Rep\text{-}preal}(X)) \in \mathit{preal}$
⟨*proof*⟩

**lemma** *preal-psup-le*:
$[|\ \forall X \in P.\ X \leq Y;\ \ x \in P\ |] ==> x \leq \mathit{psup}\ P$
⟨*proof*⟩

**lemma** *psup-le-ub*: $[|\ P \neq \{\};\ \forall X \in P.\ X \leq Y\ |] ==> \mathit{psup}\ P \leq Y$
⟨*proof*⟩

Supremum property

**lemma** *preal-complete*:
$[|\ P \neq \{\};\ \forall X \in P.\ X \leq Y\ |] ==> (\exists X \in P.\ Z < X) = (Z < \mathit{psup}\ P)$
⟨*proof*⟩

## 5.13 The Embedding from *rat* into *preal*

**lemma** *preal-of-rat-add-lemma1*:
$[|x < y + z;\ 0 < x;\ 0 < y|] ==> x * y * \mathit{inverse}\ (y + z) < (y{::}\mathit{rat})$
⟨*proof*⟩

**lemma** *preal-of-rat-add-lemma2*:
**assumes** $u < x + y$
**and** $0 < x$
**and** $0 < y$
**and** $0 < u$
**shows** $\exists\, v\ w{::}\mathit{rat}.\ w < y\ \&\ 0 < v\ \&\ v < x\ \&\ 0 < w\ \&\ u = v + w$
⟨*proof*⟩

**lemma** *preal-of-rat-add*:
$[| \ 0 < x; \ 0 < y|]$
$==> \textit{preal-of-rat} \ ((x::\textit{rat}) + y) = \textit{preal-of-rat} \ x + \textit{preal-of-rat} \ y$
⟨*proof*⟩

**lemma** *preal-of-rat-mult-lemma1*:
$[|x < y; \ 0 < x; \ 0 < z|] ==> x * z * \textit{inverse} \ y < (z::\textit{rat})$
⟨*proof*⟩

**lemma** *preal-of-rat-mult-lemma2*:
**assumes** *xless*: $x < y * z$
**and** *xpos*: $0 < x$
**and** *ypos*: $0 < y$
**shows** $x * z * \textit{inverse} \ y * \textit{inverse} \ z < (z::\textit{rat})$
⟨*proof*⟩

**lemma** *preal-of-rat-mult-lemma3*:
**assumes** *uless*: $u < x * y$
**and** $0 < x$
**and** $0 < y$
**and** $0 < u$
**shows** $\exists \, v \ w::\textit{rat}. \ v < x \ \& \ w < y \ \& \ 0 < v \ \& \ 0 < w \ \& \ u = v * w$
⟨*proof*⟩

**lemma** *preal-of-rat-mult*:
$[| \ 0 < x; \ 0 < y|]$
$==> \textit{preal-of-rat} \ ((x::\textit{rat}) * y) = \textit{preal-of-rat} \ x * \textit{preal-of-rat} \ y$
⟨*proof*⟩

**lemma** *preal-of-rat-less-iff*:
$[| \ 0 < x; \ 0 < y|] ==> (\textit{preal-of-rat} \ x < \textit{preal-of-rat} \ y) = (x < y)$
⟨*proof*⟩

**lemma** *preal-of-rat-le-iff*:
$[| \ 0 < x; \ 0 < y|] ==> (\textit{preal-of-rat} \ x \leq \textit{preal-of-rat} \ y) = (x \leq y)$
⟨*proof*⟩

**lemma** *preal-of-rat-eq-iff*:
$[| \ 0 < x; \ 0 < y|] ==> (\textit{preal-of-rat} \ x = \textit{preal-of-rat} \ y) = (x = y)$
⟨*proof*⟩

**end**

# 6 RealDef: Defining the Reals from the Positive Reals

**theory** *RealDef*
**imports** *PReal*
**uses** (*real-arith.ML*)
**begin**

**definition**
*realrel* :: ((*preal* $*$ *preal*) $*$ (*preal* $*$ *preal*)) *set* **where**
*realrel* = {*p*. ∃ *x1 y1 x2 y2*. *p* = ((*x1*,*y1*),(*x2*,*y2*)) & *x1*+*y2* = *x2*+*y1*}

**typedef** (*Real*) *real* = *UNIV*//*realrel*
⟨*proof*⟩

**definition**

*real-of-preal* :: *preal* => *real* **where**
*real-of-preal m* = *Abs-Real*(*realrel*``{(*m* + *1*, *1*)})

**instance** *real* :: *zero*
*real-zero-def*: *0* == *Abs-Real*(*realrel*``{(*1*, *1*)}) ⟨*proof*⟩
**lemmas** [*code func del*] = *real-zero-def*

**instance** *real* :: *one*
*real-one-def*: *1* == *Abs-Real*(*realrel*``{(*1* + *1*, *1*)}) ⟨*proof*⟩
**lemmas** [*code func del*] = *real-one-def*

**instance** *real* :: *plus*
*real-add-def*: *z* + *w* ==
*contents* (⋃(*x*,*y*) ∈ *Rep-Real*(*z*). ⋃(*u*,*v*) ∈ *Rep-Real*(*w*).
{ *Abs-Real*(*realrel*``{(*x*+*u*, *y*+*v*)}) }) ⟨*proof*⟩
**lemmas** [*code func del*] = *real-add-def*

**instance** *real* :: *minus*
*real-minus-def*: − *r* == *contents* (⋃(*x*,*y*) ∈ *Rep-Real*(*r*). { *Abs-Real*(*realrel*``{(*y*,*x*)})
})
*real-diff-def*: *r* − (*s*::*real*) == *r* + − *s* ⟨*proof*⟩
**lemmas** [*code func del*] = *real-minus-def real-diff-def*

**instance** *real* :: *times*
*real-mult-def*:
*z* $*$ *w* ==
*contents* (⋃(*x*,*y*) ∈ *Rep-Real*(*z*). ⋃(*u*,*v*) ∈ *Rep-Real*(*w*).
{ *Abs-Real*(*realrel*``{(*x*$*$*u* + *y*$*$*v*, *x*$*$*v* + *y*$*$*u*)}) }) ⟨*proof*⟩
**lemmas** [*code func del*] = *real-mult-def*

**instance** *real* :: *inverse*
*real-inverse-def*: *inverse* (*R*::*real*) == (*THE S*. (*R* = *0* & *S* = *0*) | *S* $*$ *R* = *1*)

*real-divide-def*: $R$ / ($S$::*real*) == $R$ ∗ *inverse* $S$ ⟨*proof*⟩
**lemmas** [*code func del*] = *real-inverse-def real-divide-def*

**instance** *real* :: *ord*
*real-le-def*: $z \leq (w::real)$ ==
$\exists x\ y\ u\ v.\ x{+}v \leq u{+}y\ \&\ (x,y) \in Rep\text{-}Real\ z\ \&\ (u,v) \in Rep\text{-}Real\ w$
*real-less-def*: $(x < (y::real))$ == $(x \leq y\ \&\ x \neq y)$ ⟨*proof*⟩
**lemmas** [*code func del*] = *real-le-def real-less-def*

**instance** *real* :: *abs*
*real-abs-def*: *abs* ($r$::*real*) == (*if* $r < 0$ *then* $-\ r$ *else* $r$) ⟨*proof*⟩

**instance** *real* :: *sgn*
*real-sgn-def*: *sgn* $x$ == (*if* $x{=}0$ *then* $0$ *else if* $0{<}x$ *then* $1$ *else* $-\ 1$) ⟨*proof*⟩

## 6.1 Equivalence relation over positive reals

**lemma** *preal-trans-lemma*:
**assumes** $x + y1 = x1 + y$
**and** $x + y2 = x2 + y$
**shows** $x1 + y2 = x2 + (y1::preal)$
⟨*proof*⟩

**lemma** *realrel-iff* [*simp*]: $(((x1,y1),(x2,y2)) \in realrel) = (x1 + y2 = x2 + y1)$
⟨*proof*⟩

**lemma** *equiv-realrel*: *equiv UNIV realrel*
⟨*proof*⟩

Reduces equality of equivalence classes to the *realrel* relation: $(realrel\ `` \{x\} = realrel\ `` \{y\}) = ((x,\ y) \in realrel)$

**lemmas** *equiv-realrel-iff* =
*eq-equiv-class-iff* [*OF equiv-realrel UNIV-I UNIV-I*]

**declare** *equiv-realrel-iff* [*simp*]

**lemma** *realrel-in-real* [*simp*]: $realrel``\{(x,y)\}$: *Real*
⟨*proof*⟩

**declare** *Abs-Real-inject* [*simp*]
**declare** *Abs-Real-inverse* [*simp*]

Case analysis on the representation of a real number as an equivalence class of pairs of positive reals.

**lemma** *eq-Abs-Real* [*case-names Abs-Real*, *cases type*: *real*]:
$(!!x\ y.\ z = Abs\text{-}Real(realrel``\{(x,y)\})$ ==> $P)$ ==> $P$
⟨*proof*⟩

## 6.2 Addition and Subtraction

**lemma** *real-add-congruent2-lemma*:
  $[|a + ba = aa + b;\ ab + bc = ac + bb|]$
  $==> a + ab + (ba + bc) = aa + ac + (b + (bb::preal))$
$\langle proof \rangle$

**lemma** *real-add*:
  $Abs\text{-}Real\ (realrel``\{(x,y)\}) + Abs\text{-}Real\ (realrel``\{(u,v)\}) =$
  $Abs\text{-}Real\ (realrel``\{(x+u,\ y+v)\})$
$\langle proof \rangle$

**lemma** *real-minus*: $-\ Abs\text{-}Real(realrel``\{(x,y)\}) = Abs\text{-}Real(realrel\ ``\ \{(y,x)\})$
$\langle proof \rangle$

**instance** *real* :: *ab-group-add*
$\langle proof \rangle$

## 6.3 Multiplication

**lemma** *real-mult-congruent2-lemma*:
  $!!(x1::preal).\ [|\ x1 + y2 = x2 + y1\ |] ==>$
    $x * x1 + y * y1 + (x * y2 + y * x2) =$
    $x * x2 + y * y2 + (x * y1 + y * x1)$
$\langle proof \rangle$

**lemma** *real-mult-congruent2*:
  $(\%p1\ p2.$
    $(\%(x1,y1).\ (\%(x2,y2).$
      $\{\ Abs\text{-}Real\ (realrel``\{(x1*x2 + y1*y2,\ x1*y2+y1*x2)\})\ \})\ p2)\ p1)$
  $respects2\ realrel$
$\langle proof \rangle$

**lemma** *real-mult*:
  $Abs\text{-}Real((realrel``\{(x1,y1)\})) * Abs\text{-}Real((realrel``\{(x2,y2)\})) =$
  $Abs\text{-}Real(realrel\ ``\ \{(x1*x2+y1*y2,x1*y2+y1*x2)\})$
$\langle proof \rangle$

**lemma** *real-mult-commute*: $(z::real) * w = w * z$
$\langle proof \rangle$

**lemma** *real-mult-assoc*: $((z1::real) * z2) * z3 = z1 * (z2 * z3)$
$\langle proof \rangle$

**lemma** *real-mult-1*: $(1::real) * z = z$
$\langle proof \rangle$

**lemma** *real-add-mult-distrib*: $((z1::real) + z2) * w = (z1 * w) + (z2 * w)$
$\langle proof \rangle$

one and zero are distinct

**lemma** *real-zero-not-eq-one*: $0 \neq (1::real)$
$\langle proof \rangle$

**instance** *real* :: *comm-ring-1*
$\langle proof \rangle$

## 6.4 Inverse and Division

**lemma** *real-zero-iff*: $Abs\text{-}Real\ (realrel\ ``\ \{(x,\ x)\}) = 0$
$\langle proof \rangle$

Instead of using an existential quantifier and constructing the inverse within the proof, we could define the inverse explicitly.

**lemma** *real-mult-inverse-left-ex*: $x \neq 0 ==> \exists\, y.\ y{*}x = (1::real)$
$\langle proof \rangle$

**lemma** *real-mult-inverse-left*: $x \neq 0 ==> inverse(x){*}x = (1::real)$
$\langle proof \rangle$

## 6.5 The Real Numbers form a Field

**instance** *real* :: *field*
$\langle proof \rangle$

Inverse of zero! Useful to simplify certain equations

**lemma** *INVERSE-ZERO*: $inverse\ 0 = (0::real)$
$\langle proof \rangle$

**instance** *real* :: *division-by-zero*
$\langle proof \rangle$

## 6.6 The $\leq$ Ordering

**lemma** *real-le-refl*: $w \leq (w::real)$
$\langle proof \rangle$

The arithmetic decision procedure is not set up for type preal. This lemma is currently unused, but it could simplify the proofs of the following two lemmas.

**lemma** *preal-eq-le-imp-le*:
  **assumes** *eq*: $a+b = c+d$ **and** *le*: $c \leq a$
  **shows** $b \leq (d::preal)$
$\langle proof \rangle$

**lemma** *real-le-lemma*:
  **assumes** *l*: $u1 + v2 \leq u2 + v1$
    **and** $x1 + v1 = u1 + y1$

**and** $x2 + v2 = u2 + y2$
**shows** $x1 + y2 \le x2 + (y1::preal)$
⟨*proof*⟩

**lemma** *real-le*:
$(Abs\text{-}Real(realrel``\{(x1,y1)\}) \le Abs\text{-}Real(realrel``\{(x2,y2)\})) =$
$(x1 + y2 \le x2 + y1)$
⟨*proof*⟩

**lemma** *real-le-anti-sym*: $[| z \le w; w \le z |] ==> z = (w::real)$
⟨*proof*⟩

**lemma** *real-trans-lemma*:
**assumes** $x + v \le u + y$
**and** $u + v' \le u' + v$
**and** $x2 + v2 = u2 + y2$
**shows** $x + v' \le u' + (y::preal)$
⟨*proof*⟩

**lemma** *real-le-trans*: $[| i \le j; j \le k |] ==> i \le (k::real)$
⟨*proof*⟩

**lemma** *real-less-le*: $((w::real) < z) = (w \le z \mathbin{\&} w \ne z)$
⟨*proof*⟩

**instance** *real* :: *order*
⟨*proof*⟩

**lemma** *real-le-linear*: $(z::real) \le w \mid w \le z$
⟨*proof*⟩

**instance** *real* :: *linorder*
⟨*proof*⟩

**lemma** *real-le-eq-diff*: $(x \le y) = (x-y \le (0::real))$
⟨*proof*⟩

**lemma** *real-add-left-mono*:
**assumes** *le*: $x \le y$ **shows** $z + x \le z + (y::real)$
⟨*proof*⟩

**lemma** *real-sum-gt-zero-less*: $(0 < S + (-W::real)) ==> (W < S)$
⟨*proof*⟩

**lemma** *real-less-sum-gt-zero*: $(W < S) ==> (0 < S + (-W::real))$

⟨*proof*⟩

**lemma** *real-mult-order*: $[| \ 0 < x; \ 0 < y \ |] ==> (0::real) < x * y$
⟨*proof*⟩

**lemma** *real-mult-less-mono2*: $[| \ (0::real) < z; \ x < y \ |] ==> z * x < z * y$
⟨*proof*⟩

**instance** *real* :: *distrib-lattice*
*inf* $x\ y \equiv$ *min* $x\ y$
*sup* $x\ y \equiv$ *max* $x\ y$
⟨*proof*⟩

## 6.7 The Reals Form an Ordered Field

**instance** *real* :: *ordered-field*
⟨*proof*⟩

**instance** *real* :: *lordered-ab-group-add* ⟨*proof*⟩

The function *real-of-preal* requires many proofs, but it seems to be essential for proving completeness of the reals from that of the positive reals.

**lemma** *real-of-preal-add*:
*real-of-preal* $((x::preal) + y) =$ *real-of-preal* $x$ + *real-of-preal* $y$
⟨*proof*⟩

**lemma** *real-of-preal-mult*:
*real-of-preal* $((x::preal) * y) =$ *real-of-preal* $x*$ *real-of-preal* $y$
⟨*proof*⟩

Gleason prop 9-4.4 p 127

**lemma** *real-of-preal-trichotomy*:
$\exists\, m.\ (x::real) =$ *real-of-preal* $m \mid x = 0 \mid x = -($*real-of-preal* $m)$
⟨*proof*⟩

**lemma** *real-of-preal-leD*:
*real-of-preal* $m1 \leq$ *real-of-preal* $m2 ==> m1 \leq m2$
⟨*proof*⟩

**lemma** *real-of-preal-lessI*: $m1 < m2 ==>$ *real-of-preal* $m1 <$ *real-of-preal* $m2$
⟨*proof*⟩

**lemma** *real-of-preal-lessD*:
*real-of-preal* $m1 <$ *real-of-preal* $m2 ==> m1 < m2$
⟨*proof*⟩

**lemma** *real-of-preal-less-iff* [*simp*]:
(*real-of-preal* $m1 <$ *real-of-preal* $m2) = (m1 < m2)$
⟨*proof*⟩

**lemma** *real-of-preal-le-iff*:
$(real\text{-}of\text{-}preal\ m1 \le real\text{-}of\text{-}preal\ m2) = (m1 \le m2)$
⟨*proof*⟩

**lemma** *real-of-preal-zero-less*: $0 < real\text{-}of\text{-}preal\ m$
⟨*proof*⟩

**lemma** *real-of-preal-minus-less-zero*: $-\ real\text{-}of\text{-}preal\ m < 0$
⟨*proof*⟩

**lemma** *real-of-preal-not-minus-gt-zero*: $\sim\ 0 < -\ real\text{-}of\text{-}preal\ m$
⟨*proof*⟩

## 6.8 Theorems About the Ordering

**lemma** *real-gt-zero-preal-Ex*: $(0 < x) = (\exists\, y.\ x = real\text{-}of\text{-}preal\ y)$
⟨*proof*⟩

**lemma** *real-gt-preal-preal-Ex*:
$real\text{-}of\text{-}preal\ z < x ==> \exists\, y.\ x = real\text{-}of\text{-}preal\ y$
⟨*proof*⟩

**lemma** *real-ge-preal-preal-Ex*:
$real\text{-}of\text{-}preal\ z \le x ==> \exists\, y.\ x = real\text{-}of\text{-}preal\ y$
⟨*proof*⟩

**lemma** *real-less-all-preal*: $y \le 0 ==> \forall\, x.\ y < real\text{-}of\text{-}preal\ x$
⟨*proof*⟩

**lemma** *real-less-all-real2*: $\sim\ 0 < y ==> \forall\, x.\ y < real\text{-}of\text{-}preal\ x$
⟨*proof*⟩

## 6.9 More Lemmas

**lemma** *real-mult-left-cancel*: $(c::real) \neq 0 ==> (c*a=c*b) = (a=b)$
⟨*proof*⟩

**lemma** *real-mult-right-cancel*: $(c::real) \neq 0 ==> (a*c=b*c) = (a=b)$
⟨*proof*⟩

**lemma** *real-mult-less-iff1* [*simp*]: $(0::real) < z ==> (x*z < y*z) = (x < y)$
⟨*proof*⟩

**lemma** *real-mult-le-cancel-iff1* [*simp*]: $(0::real) < z ==> (x*z \le y*z) = (x\le y)$
⟨*proof*⟩

**lemma** *real-mult-le-cancel-iff2* [*simp*]: $(0::real) < z ==> (z*x \le z*y) = (x\le y)$
⟨*proof*⟩

**lemma** *real-inverse-gt-one*: [| (*0*::*real*) < *x*; *x* < *1* |] ==> *1* < *inverse x*
⟨*proof*⟩

## 6.10 Embedding numbers into the Reals

**abbreviation**
*real-of-nat* :: *nat* ⇒ *real*
**where**
*real-of-nat* ≡ *of-nat*

**abbreviation**
*real-of-int* :: *int* ⇒ *real*
**where**
*real-of-int* ≡ *of-int*

**abbreviation**
*real-of-rat* :: *rat* ⇒ *real*
**where**
*real-of-rat* ≡ *of-rat*

**consts**

*real* :: *'a* => *real*

**defs (overloaded)**
*real-of-nat-def* [*code inline*]: *real* == *real-of-nat*
*real-of-int-def* [*code inline*]: *real* == *real-of-int*

**lemma** *real-eq-of-nat*: *real* = *of-nat*
⟨*proof*⟩

**lemma** *real-eq-of-int*: *real* = *of-int*
⟨*proof*⟩

**lemma** *real-of-int-zero* [*simp*]: *real* (*0*::*int*) = *0*
⟨*proof*⟩

**lemma** *real-of-one* [*simp*]: *real* (*1*::*int*) = (*1*::*real*)
⟨*proof*⟩

**lemma** *real-of-int-add* [*simp*]: *real*(*x* + *y*) = *real* (*x*::*int*) + *real y*
⟨*proof*⟩

**lemma** *real-of-int-minus* [*simp*]: *real*(−*x*) = −*real* (*x*::*int*)
⟨*proof*⟩

**lemma** *real-of-int-diff* [*simp*]: *real*(*x* − *y*) = *real* (*x*::*int*) − *real y*
⟨*proof*⟩

**lemma** *real-of-int-mult* [*simp*]: *real*($x * y$) = *real* ($x$::*int*) $*$ *real* $y$
⟨*proof*⟩

**lemma** *real-of-int-setsum* [*simp*]: *real* ((*SUM* $x$:$A$. $f$ $x$)::*int*) = (*SUM* $x$:$A$. *real*($f$ $x$))
⟨*proof*⟩

**lemma** *real-of-int-setprod* [*simp*]: *real* ((*PROD* $x$:$A$. $f$ $x$)::*int*) =
(*PROD* $x$:$A$. *real*($f$ $x$))
⟨*proof*⟩

**lemma** *real-of-int-zero-cancel* [*simp*]: (*real* $x$ = *0*) = ($x$ = (*0*::*int*))
⟨*proof*⟩

**lemma** *real-of-int-inject* [*iff*]: (*real* ($x$::*int*) = *real* $y$) = ($x$ = $y$)
⟨*proof*⟩

**lemma** *real-of-int-less-iff* [*iff*]: (*real* ($x$::*int*) < *real* $y$) = ($x$ < $y$)
⟨*proof*⟩

**lemma** *real-of-int-le-iff* [*simp*]: (*real* ($x$::*int*) ≤ *real* $y$) = ($x$ ≤ $y$)
⟨*proof*⟩

**lemma** *real-of-int-gt-zero-cancel-iff* [*simp*]: (*0* < *real* ($n$::*int*)) = (*0* < $n$)
⟨*proof*⟩

**lemma** *real-of-int-ge-zero-cancel-iff* [*simp*]: (*0* <= *real* ($n$::*int*)) = (*0* <= $n$)
⟨*proof*⟩

**lemma** *real-of-int-lt-zero-cancel-iff* [*simp*]: (*real* ($n$::*int*) < *0*) = ($n$ < *0*)
⟨*proof*⟩

**lemma** *real-of-int-le-zero-cancel-iff* [*simp*]: (*real* ($n$::*int*) <= *0*) = ($n$ <= *0*)
⟨*proof*⟩

**lemma** *real-of-int-abs* [*simp*]: *real* (*abs* $x$) = *abs*(*real* ($x$::*int*))
⟨*proof*⟩

**lemma** *int-less-real-le*: (($n$::*int*) < $m$) = (*real* $n$ + *1* <= *real* $m$)
⟨*proof*⟩

**lemma** *int-le-real-less*: (($n$::*int*) <= $m$) = (*real* $n$ < *real* $m$ + *1*)
⟨*proof*⟩

**lemma** *real-of-int-div-aux*: $d$ ~= *0* ==> (*real* ($x$::*int*)) / (*real* $d$) =
*real* ($x$ *div* $d$) + (*real* ($x$ *mod* $d$)) / (*real* $d$)
⟨*proof*⟩

**lemma** *real-of-int-div*: ($d$::*int*) ~= *0* ==> $d$ *dvd* $n$ ==>

$real(n\ div\ d) = real\ n\ /\ real\ d$
⟨*proof*⟩

**lemma** *real-of-int-div2*:
$0 <= real\ (n::int)\ /\ real\ (x) - real\ (n\ div\ x)$
⟨*proof*⟩

**lemma** *real-of-int-div3*:
$real\ (n::int)\ /\ real\ (x) - real\ (n\ div\ x) <= 1$
⟨*proof*⟩

**lemma** *real-of-int-div4*: $real\ (n\ div\ x) <= real\ (n::int)\ /\ real\ x$
⟨*proof*⟩

## 6.11 Embedding the Naturals into the Reals

**lemma** *real-of-nat-zero* [*simp*]: $real\ (0::nat) = 0$
⟨*proof*⟩

**lemma** *real-of-nat-one* [*simp*]: $real\ (Suc\ 0) = (1::real)$
⟨*proof*⟩

**lemma** *real-of-nat-add* [*simp*]: $real\ (m + n) = real\ (m::nat) + real\ n$
⟨*proof*⟩

**lemma** *real-of-nat-Suc*: $real\ (Suc\ n) = real\ n + (1::real)$
⟨*proof*⟩

**lemma** *real-of-nat-less-iff* [*iff*]:
$(real\ (n::nat) < real\ m) = (n < m)$
⟨*proof*⟩

**lemma** *real-of-nat-le-iff* [*iff*]: $(real\ (n::nat) \leq real\ m) = (n \leq m)$
⟨*proof*⟩

**lemma** *real-of-nat-ge-zero* [*iff*]: $0 \leq real\ (n::nat)$
⟨*proof*⟩

**lemma** *real-of-nat-Suc-gt-zero*: $0 < real\ (Suc\ n)$
⟨*proof*⟩

**lemma** *real-of-nat-mult* [*simp*]: $real\ (m * n) = real\ (m::nat) * real\ n$
⟨*proof*⟩

**lemma** *real-of-nat-setsum* [*simp*]: $real\ ((SUM\ x{:}A.\ f\ x)::nat) =$
$(SUM\ x{:}A.\ real(f\ x))$
⟨*proof*⟩

**lemma** *real-of-nat-setprod* [*simp*]: *real* ((*PROD x*:*A*. *f x*)::*nat*) =
  (*PROD x*:*A*. *real*(*f x*))
  ⟨*proof*⟩

**lemma** *real-of-card*: *real* (*card A*) = *setsum* (%*x*.*1*) *A*
  ⟨*proof*⟩

**lemma** *real-of-nat-inject* [*iff*]: (*real* (*n*::*nat*) = *real m*) = (*n* = *m*)
⟨*proof*⟩

**lemma** *real-of-nat-zero-iff* [*iff*]: (*real* (*n*::*nat*) = *0*) = (*n* = *0*)
⟨*proof*⟩

**lemma** *real-of-nat-diff*: $n \leq m$ ==> *real* (*m* − *n*) = *real* (*m*::*nat*) − *real n*
⟨*proof*⟩

**lemma** *real-of-nat-gt-zero-cancel-iff* [*simp*]: (*0* < *real* (*n*::*nat*)) = (*0* < *n*)
⟨*proof*⟩

**lemma** *real-of-nat-le-zero-cancel-iff* [*simp*]: (*real* (*n*::*nat*) ≤ *0*) = (*n* = *0*)
⟨*proof*⟩

**lemma** *not-real-of-nat-less-zero* [*simp*]: ~ *real* (*n*::*nat*) < *0*
⟨*proof*⟩

**lemma** *real-of-nat-ge-zero-cancel-iff* [*simp*]: (*0* ≤ *real* (*n*::*nat*))
⟨*proof*⟩

**lemma** *nat-less-real-le*: ((*n*::*nat*) < *m*) = (*real n* + *1* <= *real m*)
  ⟨*proof*⟩

**lemma** *nat-le-real-less*: ((*n*::*nat*) <= *m*) = (*real n* < *real m* + *1*)
  ⟨*proof*⟩

**lemma** *real-of-nat-div-aux*: *0* < *d* ==> (*real* (*x*::*nat*)) / (*real d*) =
  *real* (*x div d*) + (*real* (*x mod d*)) / (*real d*)
⟨*proof*⟩

**lemma** *real-of-nat-div*: *0* < (*d*::*nat*) ==> *d dvd n* ==>
  *real*(*n div d*) = *real n* / *real d*
  ⟨*proof*⟩

**lemma** *real-of-nat-div2*:
  *0* <= *real* (*n*::*nat*) / *real* (*x*) − *real* (*n div x*)
⟨*proof*⟩

**lemma** *real-of-nat-div3*:
  *real* (*n*::*nat*) / *real* (*x*) − *real* (*n div x*) <= *1*
⟨*proof*⟩

**lemma** *real-of-nat-div4*: *real* (*n div x*) <= *real* (*n*::*nat*) / *real x*
⟨*proof*⟩

**lemma** *real-of-int-real-of-nat*: *real* (*int n*) = *real n*
⟨*proof*⟩

**lemma** *real-of-int-of-nat-eq* [*simp*]: *real* (*of-nat n* :: *int*) = *real n*
⟨*proof*⟩

**lemma** *real-nat-eq-real* [*simp*]: *0* <= *x* ==> *real*(*nat x*) = *real x*
⟨*proof*⟩

## 6.12 Numerals and Arithmetic

**instance** *real* :: *number-ring*
*real-number-of-def*: *number-of w* ≡ *real-of-int w*
⟨*proof*⟩

**lemma** [*code*, *code unfold*]:
*number-of k* = *real-of-int* (*number-of k*)
⟨*proof*⟩

Collapse applications of *real* to *number-of*

**lemma** *real-number-of* [*simp*]: *real* (*number-of v* :: *int*) = *number-of v*
⟨*proof*⟩

**lemma** *real-of-nat-number-of* [*simp*]:
*real* (*number-of v* :: *nat*) =
(*if neg* (*number-of v* :: *int*) *then 0*
*else* (*number-of v* :: *real*))
⟨*proof*⟩

⟨*ML*⟩

## 6.13 Simprules combining x+y and 0: ARE THEY NEEDED?

Needed in this non-standard form by Hyperreal/Transcendental

**lemma** *real-0-le-divide-iff*:
((*0*::*real*) $\leq$ *x*/*y*) = ((*x* $\leq$ *0* | *0* $\leq$ *y*) & (*0* $\leq$ *x* | *y* $\leq$ *0*))
⟨*proof*⟩

**lemma** *real-add-minus-iff* [*simp*]: (*x* + − *a* = (*0*::*real*)) = (*x*=*a*)
⟨*proof*⟩

**lemma** *real-add-eq-0-iff*: (*x*+*y* = (*0*::*real*)) = (*y* = −*x*)
⟨*proof*⟩

**lemma** *real-add-less-0-iff*: $(x+y < (0::real)) = (y < -x)$
⟨*proof*⟩

**lemma** *real-0-less-add-iff*: $((0::real) < x+y) = (-x < y)$
⟨*proof*⟩

**lemma** *real-add-le-0-iff*: $(x+y \leq (0::real)) = (y \leq -x)$
⟨*proof*⟩

**lemma** *real-0-le-add-iff*: $((0::real) \leq x+y) = (-x \leq y)$
⟨*proof*⟩

### 6.13.1 Density of the Reals

**lemma** *real-lbound-gt-zero*:
$[| (0::real) < d1;\ 0 < d2\ |] ==> \exists e.\ 0 < e\ \&\ e < d1\ \&\ e < d2$
⟨*proof*⟩

Similar results are proved in *Ring-and-Field*

**lemma** *real-less-half-sum*: $x < y ==> x < (x+y) / (2::real)$
⟨*proof*⟩

**lemma** *real-gt-half-sum*: $x < y ==> (x+y)/(2::real) < y$
⟨*proof*⟩

## 6.14 Absolute Value Function for the Reals

**lemma** *abs-minus-add-cancel*: $abs(x + (-y)) = abs\ (y + (-(x::real)))$
⟨*proof*⟩

**lemma** *abs-le-interval-iff*: $(abs\ x \leq r) = (-r \leq x\ \&\ x \leq (r::real))$
⟨*proof*⟩

**lemma** *abs-add-one-gt-zero* [*simp*]: $(0::real) < 1 + abs(x)$
⟨*proof*⟩

**lemma** *abs-real-of-nat-cancel* [*simp*]: $abs\ (real\ x) = real\ (x::nat)$
⟨*proof*⟩

**lemma** *abs-add-one-not-less-self* [*simp*]: $\sim abs(x) + (1::real) < x$
⟨*proof*⟩

**lemma** *abs-sum-triangle-ineq*: $abs\ ((x::real) + y + (-l + -m)) \leq abs(x + -l) + abs(y + -m)$
⟨*proof*⟩

## 6.15 Implementation of rational real numbers as pairs of integers

**definition**
*Ratreal* :: *int* × *int* ⇒ *real*
**where**
*Ratreal* = *INum*

**code-datatype** *Ratreal*

**lemma** *Ratreal-simp*:
*Ratreal* (*k*, *l*) = *real-of-int k* / *real-of-int l*
⟨*proof*⟩

**lemma** *Ratreal-zero* [*simp*]: *Ratreal* $0_N$ = *0*
⟨*proof*⟩

**lemma** *Ratreal-lit* [*simp*]: *Ratreal* $i_N$ = *real-of-int i*
⟨*proof*⟩

**lemma** *zero-real-code* [*code*, *code unfold*]:
*0* = *Ratreal* $0_N$ ⟨*proof*⟩

**lemma** *one-real-code* [*code*, *code unfold*]:
*1* = *Ratreal* $1_N$ ⟨*proof*⟩

**instance** *real* :: *eq* ⟨*proof*⟩

**lemma** *real-eq-code* [*code*]: *Ratreal x* = *Ratreal y* ⟷ *normNum x* = *normNum y*
⟨*proof*⟩

**lemma** *real-less-eq-code* [*code*]: *Ratreal x* ≤ *Ratreal y* ⟷ *normNum x* $\leq_N$ *normNum y*
⟨*proof*⟩

**lemma** *real-less-code* [*code*]: *Ratreal x* < *Ratreal y* ⟷ *normNum x* $<_N$ *normNum y*
⟨*proof*⟩

**lemma** *real-add-code* [*code*]: *Ratreal x* + *Ratreal y* = *Ratreal* (*x* $+_N$ *y*)
⟨*proof*⟩

**lemma** *real-mul-code* [*code*]: *Ratreal x* ∗ *Ratreal y* = *Ratreal* (*x* $*_N$ *y*)
⟨*proof*⟩

**lemma** *real-neg-code* [*code*]: − *Ratreal x* = *Ratreal* ($\sim_N$ *x*)
⟨*proof*⟩

**lemma** *real-sub-code* [*code*]: *Ratreal x* − *Ratreal y* = *Ratreal* (*x* $-_N$ *y*)
⟨*proof*⟩

**lemma** *real-inv-code* [*code*]: *inverse* (*Ratreal x*) = *Ratreal* (*Ninv x*)
⟨*proof*⟩

**lemma** *real-div-code* [*code*]: *Ratreal x* / *Ratreal y* = *Ratreal* ($x \div_N y$)
⟨*proof*⟩

Setup for SML code generator

```
types-code
  real ((int */ int))
attach (term-of) ⟪
fun term-of-real (p, q) =
  let
    val rT = HOLogic.realT
  in
    if q = 1 orelse p = 0 then HOLogic.mk-number rT p
    else @{term op / :: real ⇒ real ⇒ real} $
      HOLogic.mk-number rT p $ HOLogic.mk-number rT q
  end;
⟫
attach (test) ⟪
fun gen-real i =
  let
    val p = random-range 0 i;
    val q = random-range 1 (i + 1);
    val g = Integer.gcd p q;
    val p' = p div g;
    val q' = q div g;
  in
    (if one-of [true, false] then p' else ~ p',
     if p' = 0 then 0 else q')
  end;
⟫

consts-code
  Ratreal ((-))

consts-code
  of-int :: int ⇒ real (⟨module⟩real'-of'-int)
attach ⟪
fun real-of-int 0 = (0, 0)
  | real-of-int i = (i, 1);
⟫

declare real-of-int-of-nat-eq [symmetric, code]

end
```

# 7 RComplete: Completeness of the Reals; Floor and Ceiling Functions

**theory** *RComplete*
**imports** *Lubs RealDef*
**begin**

**lemma** *real-sum-of-halves*: $x/2 + x/2 = (x::real)$
⟨*proof*⟩

## 7.1 Completeness of Positive Reals

Supremum property for the set of positive reals

Let $P$ be a non-empty set of positive reals, with an upper bound $y$. Then $P$ has a least upper bound (written $S$).

FIXME: Can the premise be weakened to $\forall x \in P.\ x \leq y$?

**lemma** *posreal-complete*:
**assumes** *positive-P*: $\forall x \in P.\ (0::real) < x$
**and** *not-empty-P*: $\exists x.\ x \in P$
**and** *upper-bound-Ex*: $\exists y.\ \forall x \in P.\ x<y$
**shows** $\exists S.\ \forall y.\ (\exists x \in P.\ y < x) = (y < S)$
⟨*proof*⟩

Completeness properties using *isUb*, *isLub* etc.

**lemma** *real-isLub-unique*: $[\mid$ *isLub* $R\ S\ x$; *isLub* $R\ S\ y \mid] ==> x = (y::real)$
⟨*proof*⟩

Completeness theorem for the positive reals (again).

**lemma** *posreals-complete*:
**assumes** *positive-S*: $\forall x \in S.\ 0 < x$
**and** *not-empty-S*: $\exists x.\ x \in S$
**and** *upper-bound-Ex*: $\exists u.$ *isUb* (*UNIV*::*real set*) $S\ u$
**shows** $\exists t.$ *isLub* (*UNIV*::*real set*) $S\ t$
⟨*proof*⟩

reals Completeness (again!)

**lemma** *reals-complete*:
**assumes** *notempty-S*: $\exists X.\ X \in S$
**and** *exists-Ub*: $\exists Y.$ *isUb* (*UNIV*::*real set*) $S\ Y$
**shows** $\exists t.$ *isLub* (*UNIV* :: *real set*) $S\ t$
⟨*proof*⟩

## 7.2 The Archimedean Property of the Reals

**theorem** *reals-Archimedean*:
**assumes** *x-pos*: $0 < x$

**shows** $\exists\, n.\ inverse\ (real\ (Suc\ n)) < x$
⟨*proof*⟩

There must be other proofs, e.g. *Suc* of the largest integer in the cut representing *x*.

**lemma** *reals-Archimedean2*: $\exists\, n.\ (x{::}real) < real\ (n{::}nat)$
⟨*proof*⟩

**lemma** *reals-Archimedean3*:
**assumes** *x-greater-zero*: $0 < x$
**shows** $\forall\,(y{::}real).\ \exists\,(n{::}nat).\ y < real\ n * x$
⟨*proof*⟩

**lemma** *reals-Archimedean6*:
$0 \le r ==> \exists\,(n{::}nat).\ real\ (n - 1) \le r\ \&\ r < real\ (n)$
⟨*proof*⟩

**lemma** *reals-Archimedean6a*: $0 \le r ==> \exists\, n.\ real\ (n) \le r\ \&\ r < real\ (Suc\ n)$
⟨*proof*⟩

**lemma** *reals-Archimedean-6b-int*:
$0 \le r ==> \exists\, n{::}int.\ real\ n \le r\ \&\ r < real\ (n+1)$
⟨*proof*⟩

**lemma** *reals-Archimedean-6c-int*:
$r < 0 ==> \exists\, n{::}int.\ real\ n \le r\ \&\ r < real\ (n+1)$
⟨*proof*⟩

## 7.3 Floor and Ceiling Functions from the Reals to the Integers

**definition**
*floor* :: *real* => *int* **where**
$floor\ r = (LEAST\ n{::}int.\ r < real\ (n+1))$

**definition**
*ceiling* :: *real* => *int* **where**
$ceiling\ r = -\ floor\ (-\ r)$

**notation** (*xsymbols*)
*floor* (⌊-⌋) **and**
*ceiling* (⌈-⌉)

**notation** (*HTML* **output**)
*floor* (⌊-⌋) **and**
*ceiling* (⌈-⌉)

**lemma** *number-of-less-real-of-int-iff* [*simp*]:

$((number\text{-}of\ n) < real\ (m::int)) = (number\text{-}of\ n < m)$
⟨proof⟩

**lemma** *number-of-less-real-of-int-iff2* [*simp*]:
$(real\ (m::int) < (number\text{-}of\ n)) = (m < number\text{-}of\ n)$
⟨proof⟩

**lemma** *number-of-le-real-of-int-iff* [*simp*]:
$((number\text{-}of\ n) \leq real\ (m::int)) = (number\text{-}of\ n \leq m)$
⟨proof⟩

**lemma** *number-of-le-real-of-int-iff2* [*simp*]:
$(real\ (m::int) \leq (number\text{-}of\ n)) = (m \leq number\text{-}of\ n)$
⟨proof⟩

**lemma** *floor-zero* [*simp*]: $floor\ 0 = 0$
⟨proof⟩

**lemma** *floor-real-of-nat-zero* [*simp*]: $floor\ (real\ (0::nat)) = 0$
⟨proof⟩

**lemma** *floor-real-of-nat* [*simp*]: $floor\ (real\ (n::nat)) = int\ n$
⟨proof⟩

**lemma** *floor-minus-real-of-nat* [*simp*]: $floor\ (-\ real\ (n::nat)) = -\ int\ n$
⟨proof⟩

**lemma** *floor-real-of-int* [*simp*]: $floor\ (real\ (n::int)) = n$
⟨proof⟩

**lemma** *floor-minus-real-of-int* [*simp*]: $floor\ (-\ real\ (n::int)) = -\ n$
⟨proof⟩

**lemma** *real-lb-ub-int*: $\exists\, n::int.\ real\ n \leq r\ \&\ r < real\ (n+1)$
⟨proof⟩

**lemma** *lemma-floor*:
**assumes** *a1*: $real\ m \leq r$ **and** *a2*: $r < real\ n + 1$
**shows** $m \leq (n::int)$
⟨proof⟩

**lemma** *real-of-int-floor-le* [*simp*]: $real\ (floor\ r) \leq r$
⟨proof⟩

**lemma** *floor-mono*: $x < y \Longrightarrow floor\ x \leq floor\ y$
⟨proof⟩

**lemma** *floor-mono2*: $x \leq y \Longrightarrow floor\ x \leq floor\ y$
⟨proof⟩

**lemma** *lemma-floor2*: $real\ n < real\ (x::int) + 1 ==> n \leq x$
⟨*proof*⟩

**lemma** *real-of-int-floor-cancel* [*simp*]:
$(real\ (floor\ x) = x) = (\exists\, n::int.\ x = real\ n)$
⟨*proof*⟩

**lemma** *floor-eq*: $[|\ real\ n < x;\ x < real\ n + 1\ |] ==> floor\ x = n$
⟨*proof*⟩

**lemma** *floor-eq2*: $[|\ real\ n \leq x;\ x < real\ n + 1\ |] ==> floor\ x = n$
⟨*proof*⟩

**lemma** *floor-eq3*: $[|\ real\ n < x;\ x < real\ (Suc\ n)\ |] ==> nat(floor\ x) = n$
⟨*proof*⟩

**lemma** *floor-eq4*: $[|\ real\ n \leq x;\ x < real\ (Suc\ n)\ |] ==> nat(floor\ x) = n$
⟨*proof*⟩

**lemma** *floor-number-of-eq* [*simp*]:
$floor(number\text{-}of\ n :: real) = (number\text{-}of\ n :: int)$
⟨*proof*⟩

**lemma** *floor-one* [*simp*]: $floor\ 1 = 1$
⟨*proof*⟩

**lemma** *real-of-int-floor-ge-diff-one* [*simp*]: $r - 1 \leq real(floor\ r)$
⟨*proof*⟩

**lemma** *real-of-int-floor-gt-diff-one* [*simp*]: $r - 1 < real(floor\ r)$
⟨*proof*⟩

**lemma** *real-of-int-floor-add-one-ge* [*simp*]: $r \leq real(floor\ r) + 1$
⟨*proof*⟩

**lemma** *real-of-int-floor-add-one-gt* [*simp*]: $r < real(floor\ r) + 1$
⟨*proof*⟩

**lemma** *le-floor*: $real\ a <= x ==> a <= floor\ x$
⟨*proof*⟩

**lemma** *real-le-floor*: $a <= floor\ x ==> real\ a <= x$
⟨*proof*⟩

**lemma** *le-floor-eq*: $(a <= floor\ x) = (real\ a <= x)$
⟨*proof*⟩

**lemma** *le-floor-eq-number-of* [*simp*]:

(*number-of* $n$ <= *floor* $x$) = (*number-of* $n$ <= $x$)
⟨*proof*⟩

**lemma** *le-floor-eq-zero* [*simp*]: (*0* <= *floor* $x$) = (*0* <= $x$)
⟨*proof*⟩

**lemma** *le-floor-eq-one* [*simp*]: (*1* <= *floor* $x$) = (*1* <= $x$)
⟨*proof*⟩

**lemma** *floor-less-eq*: (*floor* $x$ < $a$) = ($x$ < *real* $a$)
⟨*proof*⟩

**lemma** *floor-less-eq-number-of* [*simp*]:
(*floor* $x$ < *number-of* $n$) = ($x$ < *number-of* $n$)
⟨*proof*⟩

**lemma** *floor-less-eq-zero* [*simp*]: (*floor* $x$ < *0*) = ($x$ < *0*)
⟨*proof*⟩

**lemma** *floor-less-eq-one* [*simp*]: (*floor* $x$ < *1*) = ($x$ < *1*)
⟨*proof*⟩

**lemma** *less-floor-eq*: ($a$ < *floor* $x$) = (*real* $a$ + *1* <= $x$)
⟨*proof*⟩

**lemma** *less-floor-eq-number-of* [*simp*]:
(*number-of* $n$ < *floor* $x$) = (*number-of* $n$ + *1* <= $x$)
⟨*proof*⟩

**lemma** *less-floor-eq-zero* [*simp*]: (*0* < *floor* $x$) = (*1* <= $x$)
⟨*proof*⟩

**lemma** *less-floor-eq-one* [*simp*]: (*1* < *floor* $x$) = (*2* <= $x$)
⟨*proof*⟩

**lemma** *floor-le-eq*: (*floor* $x$ <= $a$) = ($x$ < *real* $a$ + *1*)
⟨*proof*⟩

**lemma** *floor-le-eq-number-of* [*simp*]:
(*floor* $x$ <= *number-of* $n$) = ($x$ < *number-of* $n$ + *1*)
⟨*proof*⟩

**lemma** *floor-le-eq-zero* [*simp*]: (*floor* $x$ <= *0*) = ($x$ < *1*)
⟨*proof*⟩

**lemma** *floor-le-eq-one* [*simp*]: (*floor* $x$ <= *1*) = ($x$ < *2*)
⟨*proof*⟩

**lemma** *floor-add* [*simp*]: *floor* ($x$ + *real* $a$) = *floor* $x$ + $a$

$\langle proof \rangle$

**lemma** *floor-add-number-of* [*simp*]:
$floor\ (x + number\text{-}of\ n) = floor\ x + number\text{-}of\ n$
$\langle proof \rangle$

**lemma** *floor-add-one* [*simp*]: $floor\ (x + 1) = floor\ x + 1$
$\langle proof \rangle$

**lemma** *floor-subtract* [*simp*]: $floor\ (x - real\ a) = floor\ x - a$
$\langle proof \rangle$

**lemma** *floor-subtract-number-of* [*simp*]: $floor\ (x - number\text{-}of\ n) =$
$floor\ x - number\text{-}of\ n$
$\langle proof \rangle$

**lemma** *floor-subtract-one* [*simp*]: $floor\ (x - 1) = floor\ x - 1$
$\langle proof \rangle$

**lemma** *ceiling-zero* [*simp*]: $ceiling\ 0 = 0$
$\langle proof \rangle$

**lemma** *ceiling-real-of-nat* [*simp*]: $ceiling\ (real\ (n::nat)) = int\ n$
$\langle proof \rangle$

**lemma** *ceiling-real-of-nat-zero* [*simp*]: $ceiling\ (real\ (0::nat)) = 0$
$\langle proof \rangle$

**lemma** *ceiling-floor* [*simp*]: $ceiling\ (real\ (floor\ r)) = floor\ r$
$\langle proof \rangle$

**lemma** *floor-ceiling* [*simp*]: $floor\ (real\ (ceiling\ r)) = ceiling\ r$
$\langle proof \rangle$

**lemma** *real-of-int-ceiling-ge* [*simp*]: $r \le real\ (ceiling\ r)$
$\langle proof \rangle$

**lemma** *ceiling-mono*: $x < y ==> ceiling\ x \le ceiling\ y$
$\langle proof \rangle$

**lemma** *ceiling-mono2*: $x \le y ==> ceiling\ x \le ceiling\ y$
$\langle proof \rangle$

**lemma** *real-of-int-ceiling-cancel* [*simp*]:
$(real\ (ceiling\ x) = x) = (\exists\, n::int.\ x = real\ n)$
$\langle proof \rangle$

**lemma** *ceiling-eq*: $[|\ real\ n < x;\ x < real\ n + 1\ |] ==> ceiling\ x = n + 1$
$\langle proof \rangle$

**lemma** *ceiling-eq2*: [| *real* $n < x$; $x \leq$ *real* $n + 1$ |] ==> *ceiling* $x = n + 1$
⟨*proof*⟩

**lemma** *ceiling-eq3*: [| *real* $n - 1 < x$; $x \leq$ *real* $n$ |] ==> *ceiling* $x = n$
⟨*proof*⟩

**lemma** *ceiling-real-of-int* [*simp*]: *ceiling* (*real* (*n*::*int*)) = *n*
⟨*proof*⟩

**lemma** *ceiling-number-of-eq* [*simp*]:
*ceiling* (*number-of n* :: *real*) = (*number-of n*)
⟨*proof*⟩

**lemma** *ceiling-one* [*simp*]: *ceiling 1* = *1*
⟨*proof*⟩

**lemma** *real-of-int-ceiling-diff-one-le* [*simp*]: *real* (*ceiling r*) $- 1 \leq r$
⟨*proof*⟩

**lemma** *real-of-int-ceiling-le-add-one* [*simp*]: *real* (*ceiling r*) $\leq r + 1$
⟨*proof*⟩

**lemma** *ceiling-le*: *x* <= *real a* ==> *ceiling x* <= *a*
⟨*proof*⟩

**lemma** *ceiling-le-real*: *ceiling x* <= *a* ==> *x* <= *real a*
⟨*proof*⟩

**lemma** *ceiling-le-eq*: (*ceiling x* <= *a*) = (*x* <= *real a*)
⟨*proof*⟩

**lemma** *ceiling-le-eq-number-of* [*simp*]:
(*ceiling x* <= *number-of n*) = (*x* <= *number-of n*)
⟨*proof*⟩

**lemma** *ceiling-le-zero-eq* [*simp*]: (*ceiling x* <= *0*) = (*x* <= *0*)
⟨*proof*⟩

**lemma** *ceiling-le-eq-one* [*simp*]: (*ceiling x* <= *1*) = (*x* <= *1*)
⟨*proof*⟩

**lemma** *less-ceiling-eq*: (*a* < *ceiling x*) = (*real a* < *x*)
⟨*proof*⟩

**lemma** *less-ceiling-eq-number-of* [*simp*]:
(*number-of n* < *ceiling x*) = (*number-of n* < *x*)
⟨*proof*⟩

**lemma** *less-ceiling-eq-zero* [*simp*]: $(0 < ceiling\ x) = (0 < x)$
⟨*proof*⟩

**lemma** *less-ceiling-eq-one* [*simp*]: $(1 < ceiling\ x) = (1 < x)$
⟨*proof*⟩

**lemma** *ceiling-less-eq*: $(ceiling\ x < a) = (x <= real\ a - 1)$
⟨*proof*⟩

**lemma** *ceiling-less-eq-number-of* [*simp*]:
$(ceiling\ x < number\text{-}of\ n) = (x <= number\text{-}of\ n - 1)$
⟨*proof*⟩

**lemma** *ceiling-less-eq-zero* [*simp*]: $(ceiling\ x < 0) = (x <= -1)$
⟨*proof*⟩

**lemma** *ceiling-less-eq-one* [*simp*]: $(ceiling\ x < 1) = (x <= 0)$
⟨*proof*⟩

**lemma** *le-ceiling-eq*: $(a <= ceiling\ x) = (real\ a - 1 < x)$
⟨*proof*⟩

**lemma** *le-ceiling-eq-number-of* [*simp*]:
$(number\text{-}of\ n <= ceiling\ x) = (number\text{-}of\ n - 1 < x)$
⟨*proof*⟩

**lemma** *le-ceiling-eq-zero* [*simp*]: $(0 <= ceiling\ x) = (-1 < x)$
⟨*proof*⟩

**lemma** *le-ceiling-eq-one* [*simp*]: $(1 <= ceiling\ x) = (0 < x)$
⟨*proof*⟩

**lemma** *ceiling-add* [*simp*]: $ceiling\ (x + real\ a) = ceiling\ x + a$
⟨*proof*⟩

**lemma** *ceiling-add-number-of* [*simp*]: $ceiling\ (x + number\text{-}of\ n) =$
$ceiling\ x + number\text{-}of\ n$
⟨*proof*⟩

**lemma** *ceiling-add-one* [*simp*]: $ceiling\ (x + 1) = ceiling\ x + 1$
⟨*proof*⟩

**lemma** *ceiling-subtract* [*simp*]: $ceiling\ (x - real\ a) = ceiling\ x - a$
⟨*proof*⟩

**lemma** *ceiling-subtract-number-of* [*simp*]: $ceiling\ (x - number\text{-}of\ n) =$
$ceiling\ x - number\text{-}of\ n$
⟨*proof*⟩

**lemma** *ceiling-subtract-one* [*simp*]: *ceiling* (*x* − *1*) = *ceiling x* − *1*
⟨*proof*⟩

## 7.4 Versions for the natural numbers

**definition**
*natfloor* :: *real* => *nat* **where**
*natfloor x* = *nat*(*floor x*)

**definition**
*natceiling* :: *real* => *nat* **where**
*natceiling x* = *nat*(*ceiling x*)

**lemma** *natfloor-zero* [*simp*]: *natfloor 0* = *0*
⟨*proof*⟩

**lemma** *natfloor-one* [*simp*]: *natfloor 1* = *1*
⟨*proof*⟩

**lemma** *zero-le-natfloor* [*simp*]: *0* <= *natfloor x*
⟨*proof*⟩

**lemma** *natfloor-number-of-eq* [*simp*]: *natfloor* (*number-of n*) = *number-of n*
⟨*proof*⟩

**lemma** *natfloor-real-of-nat* [*simp*]: *natfloor*(*real n*) = *n*
⟨*proof*⟩

**lemma** *real-natfloor-le*: *0* <= *x* ==> *real*(*natfloor x*) <= *x*
⟨*proof*⟩

**lemma** *natfloor-neg*: *x* <= *0* ==> *natfloor x* = *0*
⟨*proof*⟩

**lemma** *natfloor-mono*: *x* <= *y* ==> *natfloor x* <= *natfloor y*
⟨*proof*⟩

**lemma** *le-natfloor*: *real x* <= *a* ==> *x* <= *natfloor a*
⟨*proof*⟩

**lemma** *le-natfloor-eq*: *0* <= *x* ==> (*a* <= *natfloor x*) = (*real a* <= *x*)
⟨*proof*⟩

**lemma** *le-natfloor-eq-number-of* [*simp*]:
~ *neg*((*number-of n*)::*int*) ==> *0* <= *x* ==>
(*number-of n* <= *natfloor x*) = (*number-of n* <= *x*)
⟨*proof*⟩

**lemma** *le-natfloor-eq-one* [*simp*]: (*1* <= *natfloor x*) = (*1* <= *x*)

⟨*proof*⟩

**lemma** *natfloor-eq*: $real\ n <= x ==> x < real\ n + 1 ==> natfloor\ x = n$
⟨*proof*⟩

**lemma** *real-natfloor-add-one-gt*: $x < real(natfloor\ x) + 1$
⟨*proof*⟩

**lemma** *real-natfloor-gt-diff-one*: $x - 1 < real(natfloor\ x)$
⟨*proof*⟩

**lemma** *ge-natfloor-plus-one-imp-gt*: $natfloor\ z + 1 <= n ==> z < real\ n$
⟨*proof*⟩

**lemma** *natfloor-add* [*simp*]: $0 <= x ==> natfloor\ (x + real\ a) = natfloor\ x + a$
⟨*proof*⟩

**lemma** *natfloor-add-number-of* [*simp*]:
$\sim neg\ ((number\text{-}of\ n)::int) ==> 0 <= x ==>$
$natfloor\ (x + number\text{-}of\ n) = natfloor\ x + number\text{-}of\ n$
⟨*proof*⟩

**lemma** *natfloor-add-one*: $0 <= x ==> natfloor(x + 1) = natfloor\ x + 1$
⟨*proof*⟩

**lemma** *natfloor-subtract* [*simp*]: $real\ a <= x ==>$
$natfloor(x - real\ a) = natfloor\ x - a$
⟨*proof*⟩

**lemma** *natceiling-zero* [*simp*]: $natceiling\ 0 = 0$
⟨*proof*⟩

**lemma** *natceiling-one* [*simp*]: $natceiling\ 1 = 1$
⟨*proof*⟩

**lemma** *zero-le-natceiling* [*simp*]: $0 <= natceiling\ x$
⟨*proof*⟩

**lemma** *natceiling-number-of-eq* [*simp*]: $natceiling\ (number\text{-}of\ n) = number\text{-}of\ n$
⟨*proof*⟩

**lemma** *natceiling-real-of-nat* [*simp*]: $natceiling(real\ n) = n$
⟨*proof*⟩

**lemma** *real-natceiling-ge*: $x <= real(natceiling\ x)$
⟨*proof*⟩

**lemma** *natceiling-neg*: $x <= 0 ==> natceiling\ x = 0$
⟨*proof*⟩

**lemma** *natceiling-mono*: *x <= y ==> natceiling x <= natceiling y*
⟨*proof*⟩

**lemma** *natceiling-le*: *x <= real a ==> natceiling x <= a*
⟨*proof*⟩

**lemma** *natceiling-le-eq*: *0 <= x ==> (natceiling x <= a) = (x <= real a)*
⟨*proof*⟩

**lemma** *natceiling-le-eq-number-of* [*simp*]:
~ *neg((number-of n)::int) ==> 0 <= x ==>*
*(natceiling x <= number-of n) = (x <= number-of n)*
⟨*proof*⟩

**lemma** *natceiling-le-eq-one*: *(natceiling x <= 1) = (x <= 1)*
⟨*proof*⟩

**lemma** *natceiling-eq*: *real n < x ==> x <= real n + 1 ==> natceiling x = n + 1*
⟨*proof*⟩

**lemma** *natceiling-add* [*simp*]: *0 <= x ==>*
*natceiling (x + real a) = natceiling x + a*
⟨*proof*⟩

**lemma** *natceiling-add-number-of* [*simp*]:
~ *neg ((number-of n)::int) ==> 0 <= x ==>*
*natceiling (x + number-of n) = natceiling x + number-of n*
⟨*proof*⟩

**lemma** *natceiling-add-one*: *0 <= x ==> natceiling(x + 1) = natceiling x + 1*
⟨*proof*⟩

**lemma** *natceiling-subtract* [*simp*]: *real a <= x ==>*
*natceiling(x − real a) = natceiling x − a*
⟨*proof*⟩

**lemma** *natfloor-div-nat*: *1 <= x ==> y > 0 ==>*
*natfloor (x / real y) = natfloor x div y*
⟨*proof*⟩

**end**

# 8 ContNotDenum: Non-denumerability of the Continuum.

**theory** *ContNotDenum*
**imports** *RComplete*
**begin**

## 8.1 Abstract

The following document presents a proof that the Continuum is uncountable. It is formalised in the Isabelle/Isar theorem proving system.

*Theorem:* The Continuum $\mathbb{R}$ is not denumerable. In other words, there does not exist a function f:$\mathbb{N} \Rightarrow \mathbb{R}$ such that f is surjective.

*Outline:* An elegant informal proof of this result uses Cantor's Diagonalisation argument. The proof presented here is not this one. First we formalise some properties of closed intervals, then we prove the Nested Interval Property. This property relies on the completeness of the Real numbers and is the foundation for our argument. Informally it states that an intersection of countable closed intervals (where each successive interval is a subset of the last) is non-empty. We then assume a surjective function f:$\mathbb{N} \Rightarrow \mathbb{R}$ exists and find a real x such that x is not in the range of f by generating a sequence of closed intervals then using the NIP.

## 8.2 Closed Intervals

This section formalises some properties of closed intervals.

### 8.2.1 Definition

**definition**
  *closed-int* :: *real* $\Rightarrow$ *real* $\Rightarrow$ *real set* **where**
  *closed-int* $x\ y = \{z.\ x \leq z \wedge z \leq y\}$

### 8.2.2 Properties

**lemma** *closed-int-subset*:
  **assumes** *xy*: $x1 \geq x0$ $y1 \leq y0$
  **shows** *closed-int* $x1\ y1 \subseteq$ *closed-int* $x0\ y0$
⟨*proof*⟩

**lemma** *closed-int-least*:
  **assumes** *a*: $a \leq b$
  **shows** $a \in$ *closed-int* $a\ b \wedge (\forall x \in$ *closed-int* $a\ b.\ a \leq x)$
⟨*proof*⟩

**lemma** *closed-int-most*:

**assumes** $a$: $a \leq b$
**shows** $b \in$ *closed-int* $a$ $b$ $\wedge$ ($\forall x \in$ *closed-int* $a$ $b$. $x \leq b$)
⟨*proof*⟩

**lemma** *closed-not-empty*:
**shows** $a \leq b \Longrightarrow \exists x.\ x \in$ *closed-int* $a$ $b$
⟨*proof*⟩

**lemma** *closed-mem*:
**assumes** $a \leq c$ **and** $c \leq b$
**shows** $c \in$ *closed-int* $a$ $b$
⟨*proof*⟩

**lemma** *closed-subset*:
**assumes** *ac*: $a \leq b$ $c \leq d$
**assumes** *closed*: *closed-int* $a$ $b$ $\subseteq$ *closed-int* $c$ $d$
**shows** $b \geq c$
⟨*proof*⟩

## 8.3 Nested Interval Property

**theorem** *NIP*:
**fixes** $f$::*nat* $\Rightarrow$ *real set*
**assumes** *subset*: $\forall n.\ f$ (*Suc* $n$) $\subseteq f\ n$
**and** *closed*: $\forall n.\ \exists a\ b.\ f\ n =$ *closed-int* $a$ $b$ $\wedge$ $a \leq b$
**shows** $(\bigcap n.\ f\ n) \neq \{\}$
⟨*proof*⟩

## 8.4 Generating the intervals

### 8.4.1 Existence of non-singleton closed intervals

This lemma asserts that given any non-singleton closed interval (a,b) and any element c, there exists a closed interval that is a subset of (a,b) and that does not contain c and is a non-singleton itself.

**lemma** *closed-subset-ex*:
**fixes** $c$::*real*
**assumes** *alb*: $a < b$
**shows**
$\exists$ *ka kb*. *ka* $<$ *kb* $\wedge$ *closed-int ka kb* $\subseteq$ *closed-int* $a$ $b$ $\wedge$ $c \notin$ (*closed-int ka kb*)
⟨*proof*⟩

## 8.5 newInt: Interval generation

Given a function f:$\mathbb{N}\Rightarrow\mathbb{R}$, newInt (Suc n) f returns a closed interval such that *newInt* (*Suc* $n$) $f$ $\subseteq$ *newInt* $n$ $f$ and does not contain $f$ (*Suc* $n$). With the base case defined such that ($f$ *0*)$\notin$*newInt 0 f*.

### 8.5.1 Definition

**consts** *newInt* :: $nat \Rightarrow (nat \Rightarrow real) \Rightarrow (real\ set)$
**primrec**
$newInt\ 0\ f = closed\text{-}int\ (f\ 0 + 1)\ (f\ 0 + 2)$
$newInt\ (Suc\ n)\ f =$
$(SOME\ e.\ (\exists\ e1\ e2.$
$e1 < e2 \wedge$
$e = closed\text{-}int\ e1\ e2 \wedge$
$e \subseteq (newInt\ n\ f) \wedge$
$(f\ (Suc\ n)) \notin e)$
$)$

### 8.5.2 Properties

We now show that every application of newInt returns an appropriate interval.

**lemma** *newInt-ex*:
$\exists\ a\ b.\ a < b \wedge$
$newInt\ (Suc\ n)\ f = closed\text{-}int\ a\ b \wedge$
$newInt\ (Suc\ n)\ f \subseteq newInt\ n\ f \wedge$
$f\ (Suc\ n) \notin newInt\ (Suc\ n)\ f$
$\langle proof \rangle$

**lemma** *newInt-subset*:
$newInt\ (Suc\ n)\ f \subseteq newInt\ n\ f$
$\langle proof \rangle$

Another fundamental property is that no element in the range of f is in the intersection of all closed intervals generated by newInt.

**lemma** *newInt-inter*:
$\forall\ n.\ f\ n \notin (\bigcap n.\ newInt\ n\ f)$
$\langle proof \rangle$

**lemma** *newInt-notempty*:
$(\bigcap n.\ newInt\ n\ f) \neq \{\}$
$\langle proof \rangle$

## 8.6 Final Theorem

**theorem** *real-non-denum*:
**shows** $\neg\ (\exists f{::}nat{\Rightarrow}real.\ surj\ f)$
$\langle proof \rangle$

**end**

# 9 RealPow: Natural powers theory

**theory** *RealPow*
**imports** *RealDef*
**begin**

**declare** *abs-mult-self* [*simp*]

**instance** *real* :: *power* ⟨*proof*⟩

**primrec** (*realpow*)
*realpow-0*: $r \mathbin{\hat{}} 0 = 1$
*realpow-Suc*: $r \mathbin{\hat{}} (Suc\ n) = (r::real) * (r \mathbin{\hat{}} n)$

**instance** *real* :: *recpower*
⟨*proof*⟩

**lemma** *two-realpow-ge-one* [*simp*]: $(1::real) \leq 2 \mathbin{\hat{}} n$
⟨*proof*⟩

**lemma** *two-realpow-gt* [*simp*]: $real\ (n::nat) < 2 \mathbin{\hat{}} n$
⟨*proof*⟩

**lemma** *realpow-Suc-le-self*: $[\mid 0 \leq r;\ r \leq (1::real) \mid] ==> r \mathbin{\hat{}} Suc\ n \leq r$
⟨*proof*⟩

**lemma** *realpow-minus-mult* [*rule-format*]:
$0 < n \dashrightarrow (x::real) \mathbin{\hat{}} (n - 1) * x = x \mathbin{\hat{}} n$
⟨*proof*⟩

**lemma** *realpow-two-mult-inverse* [*simp*]:
$r \neq 0 ==> r * inverse\ r \mathbin{\hat{}} Suc\ (Suc\ 0) = inverse\ (r::real)$
⟨*proof*⟩

**lemma** *realpow-two-minus* [*simp*]: $(-x) \mathbin{\hat{}} Suc\ (Suc\ 0) = (x::real) \mathbin{\hat{}} Suc\ (Suc\ 0)$
⟨*proof*⟩

**lemma** *realpow-two-diff*:
$(x::real) \mathbin{\hat{}} Suc\ (Suc\ 0) - y \mathbin{\hat{}} Suc\ (Suc\ 0) = (x - y) * (x + y)$
⟨*proof*⟩

**lemma** *realpow-two-disj*:
$((x::real) \mathbin{\hat{}} Suc\ (Suc\ 0) = y \mathbin{\hat{}} Suc\ (Suc\ 0)) = (x = y \mid x = -y)$
⟨*proof*⟩

**lemma** *realpow-real-of-nat*: $real\ (m::nat) \mathbin{\hat{}} n = real\ (m \mathbin{\hat{}} n)$
⟨*proof*⟩

**lemma** *realpow-real-of-nat-two-pos* [*simp*] : $0 < real\ (Suc\ (Suc\ 0)\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *realpow-increasing*:
$[|(0::real) \leq x;\ 0 \leq y;\ x\ \hat{}\ Suc\ n \leq y\ \hat{}\ Suc\ n|] ==> x \leq y$
⟨*proof*⟩

## 9.1 Literal Arithmetic Involving Powers, Type *real*

**lemma** *real-of-int-power*: $real\ (x::int)\ \hat{}\ n = real\ (x\ \hat{}\ n)$
⟨*proof*⟩
**declare** *real-of-int-power* [*symmetric*, *simp*]

**lemma** *power-real-number-of*:
$(number\text{-}of\ v\ ::\ real)\ \hat{}\ n = real\ ((number\text{-}of\ v\ ::\ int)\ \hat{}\ n)$
⟨*proof*⟩

**declare** *power-real-number-of* [*of - number-of w*, *standard*, *simp*]

## 9.2 Properties of Squares

**lemma** *sum-squares-ge-zero*:
**fixes** $x\ y\ ::\ 'a::ordered\text{-}ring\text{-}strict$
**shows** $0 \leq x * x + y * y$
⟨*proof*⟩

**lemma** *not-sum-squares-lt-zero*:
**fixes** $x\ y\ ::\ 'a::ordered\text{-}ring\text{-}strict$
**shows** $\neg\ x * x + y * y < 0$
⟨*proof*⟩

**lemma** *sum-nonneg-eq-zero-iff*:
**fixes** $x\ y\ ::\ 'a::pordered\text{-}ab\text{-}group\text{-}add$
**assumes** $x$: $0 \leq x$ **and** $y$: $0 \leq y$
**shows** $(x + y = 0) = (x = 0 \wedge y = 0)$
⟨*proof*⟩

**lemma** *sum-squares-eq-zero-iff*:
**fixes** $x\ y\ ::\ 'a::ordered\text{-}ring\text{-}strict$
**shows** $(x * x + y * y = 0) = (x = 0 \wedge y = 0)$
⟨*proof*⟩

**lemma** *sum-squares-le-zero-iff*:
**fixes** $x\ y\ ::\ 'a::ordered\text{-}ring\text{-}strict$
**shows** $(x * x + y * y \leq 0) = (x = 0 \wedge y = 0)$
⟨*proof*⟩

**lemma** *sum-squares-gt-zero-iff*:

**fixes** $x\ y :: 'a::ordered\text{-}ring\text{-}strict$
**shows** $(0 < x * x + y * y) = (x \neq 0 \lor y \neq 0)$
⟨*proof*⟩

**lemma** *sum-power2-ge-zero*:
**fixes** $x\ y :: 'a::\{ordered\text{-}idom,recpower\}$
**shows** $0 \leq x^2 + y^2$
⟨*proof*⟩

**lemma** *not-sum-power2-lt-zero*:
**fixes** $x\ y :: 'a::\{ordered\text{-}idom,recpower\}$
**shows** $\neg\ x^2 + y^2 < 0$
⟨*proof*⟩

**lemma** *sum-power2-eq-zero-iff*:
**fixes** $x\ y :: 'a::\{ordered\text{-}idom,recpower\}$
**shows** $(x^2 + y^2 = 0) = (x = 0 \land y = 0)$
⟨*proof*⟩

**lemma** *sum-power2-le-zero-iff*:
**fixes** $x\ y :: 'a::\{ordered\text{-}idom,recpower\}$
**shows** $(x^2 + y^2 \leq 0) = (x = 0 \land y = 0)$
⟨*proof*⟩

**lemma** *sum-power2-gt-zero-iff*:
**fixes** $x\ y :: 'a::\{ordered\text{-}idom,recpower\}$
**shows** $(0 < x^2 + y^2) = (x \neq 0 \lor y \neq 0)$
⟨*proof*⟩

## 9.3 Squares of Reals

**lemma** *real-two-squares-add-zero-iff* [*simp*]:
$(x * x + y * y = 0) = ((x::real) = 0 \land y = 0)$
⟨*proof*⟩

**lemma** *real-sum-squares-cancel*: $x * x + y * y = 0 ==> x = (0::real)$
⟨*proof*⟩

**lemma** *real-sum-squares-cancel2*: $x * x + y * y = 0 ==> y = (0::real)$
⟨*proof*⟩

**lemma** *real-mult-self-sum-ge-zero*: $(0::real) \leq x*x + y*y$
⟨*proof*⟩

**lemma** *real-sum-squares-cancel-a*: $x * x = -(y * y) ==> x = (0::real)\ \&\ y=0$
⟨*proof*⟩

**lemma** *real-squared-diff-one-factored*: $x*x - (1::real) = (x + 1)*(x - 1)$
⟨*proof*⟩

**lemma** *real-mult-is-one* [*simp*]: $(x*x = (1::real)) = (x = 1 \mid x = -1)$
⟨*proof*⟩

**lemma** *real-sum-squares-not-zero*: $x \sim= 0 ==> x * x + y * y \sim= (0::real)$
⟨*proof*⟩

**lemma** *real-sum-squares-not-zero2*: $y \sim= 0 ==> x * x + y * y \sim= (0::real)$
⟨*proof*⟩

**lemma** *realpow-two-sum-zero-iff* [*simp*]:
$(x \mathbin{\hat{}} 2 + y \mathbin{\hat{}} 2 = (0::real)) = (x = 0 \;\&\; y = 0)$
⟨*proof*⟩

**lemma** *realpow-two-le-add-order* [*simp*]: $(0::real) \leq u \mathbin{\hat{}} 2 + v \mathbin{\hat{}} 2$
⟨*proof*⟩

**lemma** *realpow-two-le-add-order2* [*simp*]: $(0::real) \leq u \mathbin{\hat{}} 2 + v \mathbin{\hat{}} 2 + w \mathbin{\hat{}} 2$
⟨*proof*⟩

**lemma** *real-sum-square-gt-zero*: $x \sim= 0 ==> (0::real) < x * x + y * y$
⟨*proof*⟩

**lemma** *real-sum-square-gt-zero2*: $y \sim= 0 ==> (0::real) < x * x + y * y$
⟨*proof*⟩

**lemma** *real-minus-mult-self-le* [*simp*]: $-(u * u) \leq (x * (x::real))$
⟨*proof*⟩

**lemma** *realpow-square-minus-le* [*simp*]: $-(u \mathbin{\hat{}} 2) \leq (x::real) \mathbin{\hat{}} 2$
⟨*proof*⟩

**lemma** *real-sq-order*:
**fixes** $x::real$
**assumes** *xgt0*: $0 \leq x$ **and** *ygt0*: $0 \leq y$ **and** *sq*: $x\hat{}2 \leq y\hat{}2$
**shows** $x \leq y$
⟨*proof*⟩

## 9.4 Various Other Theorems

**lemma** *real-le-add-half-cancel*: $(x + y/2 \leq (y::real)) = (x \leq y\ /2)$
⟨*proof*⟩

**lemma** *real-minus-half-eq* [*simp*]: $(x::real) - x/2 = x/2$
⟨*proof*⟩

**lemma** *real-mult-inverse-cancel*:
$[|(0::real) < x;\ 0 < x1;\ x1 * y < x * u\ |]$

$==>$ *inverse* $x * y <$ *inverse* $x1 * u$
⟨*proof*⟩

**lemma** *real-mult-inverse-cancel2*:
$[|(0::real) < x; 0 < x1;\ x1 * y < x * u\ |] ==> y *$ *inverse* $x < u *$ *inverse* $x1$
⟨*proof*⟩

**lemma** *inverse-real-of-nat-gt-zero* [*simp*]: $0 <$ *inverse* (*real* (*Suc* $n$))
⟨*proof*⟩

**lemma** *inverse-real-of-nat-ge-zero* [*simp*]: $0 \leq$ *inverse* (*real* (*Suc* $n$))
⟨*proof*⟩

**lemma** *realpow-num-eq-if*: $(m::real)\ \hat{}\ n = ($*if* $n=0$ *then* $1$ *else* $m * m\ \hat{}\ (n - 1))$
⟨*proof*⟩

**end**

# 10 RealVector: Vector Spaces and Algebras over the Reals

**theory** *RealVector*
**imports** *RealPow*
**begin**

## 10.1 Locale for additive functions

**locale** *additive* =
  **fixes** $f$ :: $'a::ab\text{-}group\text{-}add \Rightarrow 'b::ab\text{-}group\text{-}add$
  **assumes** *add*: $f\ (x + y) = f\ x + f\ y$

**lemma** (**in** *additive*) *zero*: $f\ 0 = 0$
⟨*proof*⟩

**lemma** (**in** *additive*) *minus*: $f\ (-\ x) = -\ f\ x$
⟨*proof*⟩

**lemma** (**in** *additive*) *diff*: $f\ (x - y) = f\ x - f\ y$
⟨*proof*⟩

**lemma** (**in** *additive*) *setsum*: $f$ (*setsum* $g\ A$) = $(\sum x \in A.\ f\ (g\ x))$
⟨*proof*⟩

## 10.2 Real vector spaces

**class** *scaleR* = *type* +
  **fixes** *scaleR* :: $real \Rightarrow 'a \Rightarrow 'a$ (**infixr** $*_R$ 75)
**begin**

**abbreviation**
  *divideR* :: $'a \Rightarrow real \Rightarrow 'a$ (**infixl** $'/_R$ *70*)
**where**
  $x\ /_R\ r == scaleR\ (inverse\ r)\ x$

**end**

**instance** *real* :: *scaleR*
  *real-scaleR-def* [*simp*]: $scaleR\ a\ x \equiv a * x$ ⟨*proof*⟩

**class** *real-vector* = *scaleR* + *ab-group-add* +
  **assumes** *scaleR-right-distrib*: $scaleR\ a\ (x + y) = scaleR\ a\ x + scaleR\ a\ y$
  **and** *scaleR-left-distrib*: $scaleR\ (a + b)\ x = scaleR\ a\ x + scaleR\ b\ x$
  **and** *scaleR-scaleR* [*simp*]: $scaleR\ a\ (scaleR\ b\ x) = scaleR\ (a * b)\ x$
  **and** *scaleR-one* [*simp*]: $scaleR\ 1\ x = x$

**class** *real-algebra* = *real-vector* + *ring* +
  **assumes** *mult-scaleR-left* [*simp*]: $scaleR\ a\ x * y = scaleR\ a\ (x * y)$
  **and** *mult-scaleR-right* [*simp*]: $x * scaleR\ a\ y = scaleR\ a\ (x * y)$

**class** *real-algebra-1* = *real-algebra* + *ring-1*

**class** *real-div-algebra* = *real-algebra-1* + *division-ring*

**class** *real-field* = *real-div-algebra* + *field*

**instance** *real* :: *real-field*
⟨*proof*⟩

**lemma** *scaleR-left-commute*:
  **fixes** $x$ :: *'a::real-vector*
  **shows** $scaleR\ a\ (scaleR\ b\ x) = scaleR\ b\ (scaleR\ a\ x)$
⟨*proof*⟩

**interpretation** *scaleR-left*: *additive* [($\lambda a.\ scaleR\ a\ x$::*'a::real-vector*)]
⟨*proof*⟩

**interpretation** *scaleR-right*: *additive* [($\lambda x.\ scaleR\ a\ x$::*'a::real-vector*)]
⟨*proof*⟩

**lemmas** *scaleR-zero-left* [*simp*] = *scaleR-left.zero*

**lemmas** *scaleR-zero-right* [*simp*] = *scaleR-right.zero*

**lemmas** *scaleR-minus-left* [*simp*] = *scaleR-left.minus*

**lemmas** *scaleR-minus-right* [*simp*] = *scaleR-right.minus*

**lemmas** *scaleR-left-diff-distrib* = *scaleR-left.diff*

**lemmas** *scaleR-right-diff-distrib* = *scaleR-right.diff*

**lemma** *scaleR-eq-0-iff* [*simp*]:
**fixes** $x$ :: $'a$::*real-vector*
**shows** ($scaleR\ a\ x = 0$) = ($a = 0 \lor x = 0$)
⟨*proof*⟩

**lemma** *scaleR-left-imp-eq*:
**fixes** $x\ y$ :: $'a$::*real-vector*
**shows** $[\![a \neq 0;\ scaleR\ a\ x = scaleR\ a\ y]\!] \Longrightarrow x = y$
⟨*proof*⟩

**lemma** *scaleR-right-imp-eq*:
**fixes** $x\ y$ :: $'a$::*real-vector*
**shows** $[\![x \neq 0;\ scaleR\ a\ x = scaleR\ b\ x]\!] \Longrightarrow a = b$
⟨*proof*⟩

**lemma** *scaleR-cancel-left*:
**fixes** $x\ y$ :: $'a$::*real-vector*
**shows** ($scaleR\ a\ x = scaleR\ a\ y$) = ($x = y \lor a = 0$)
⟨*proof*⟩

**lemma** *scaleR-cancel-right*:
**fixes** $x\ y$ :: $'a$::*real-vector*
**shows** ($scaleR\ a\ x = scaleR\ b\ x$) = ($a = b \lor x = 0$)
⟨*proof*⟩

**lemma** *nonzero-inverse-scaleR-distrib*:
**fixes** $x$ :: $'a$::*real-div-algebra* **shows**
$[\![a \neq 0;\ x \neq 0]\!] \Longrightarrow inverse\ (scaleR\ a\ x) = scaleR\ (inverse\ a)\ (inverse\ x)$
⟨*proof*⟩

**lemma** *inverse-scaleR-distrib*:
**fixes** $x$ :: $'a$::{*real-div-algebra*,*division-by-zero*}
**shows** $inverse\ (scaleR\ a\ x) = scaleR\ (inverse\ a)\ (inverse\ x)$
⟨*proof*⟩

## 10.3 Embedding of the Reals into any *real-algebra-1*: *of-real*

**definition**
*of-real* :: *real* ⇒ $'a$::*real-algebra-1* **where**
*of-real* $r$ = *scaleR* $r$ *1*

**lemma** *scaleR-conv-of-real*: *scaleR* $r\ x$ = *of-real* $r * x$
⟨*proof*⟩

**lemma** *of-real-0* [*simp*]: *of-real* *0* = *0*

⟨*proof*⟩

**lemma** *of-real-1* [*simp*]: *of-real 1* = *1*
⟨*proof*⟩

**lemma** *of-real-add* [*simp*]: *of-real* (*x* + *y*) = *of-real x* + *of-real y*
⟨*proof*⟩

**lemma** *of-real-minus* [*simp*]: *of-real* (− *x*) = − *of-real x*
⟨*proof*⟩

**lemma** *of-real-diff* [*simp*]: *of-real* (*x* − *y*) = *of-real x* − *of-real y*
⟨*proof*⟩

**lemma** *of-real-mult* [*simp*]: *of-real* (*x* ∗ *y*) = *of-real x* ∗ *of-real y*
⟨*proof*⟩

**lemma** *nonzero-of-real-inverse*:
  *x* ≠ *0* ⟹ *of-real* (*inverse x*) =
   *inverse* (*of-real x* :: ′*a*::*real-div-algebra*)
⟨*proof*⟩

**lemma** *of-real-inverse* [*simp*]:
  *of-real* (*inverse x*) =
   *inverse* (*of-real x* :: ′*a*::{*real-div-algebra*,*division-by-zero*})
⟨*proof*⟩

**lemma** *nonzero-of-real-divide*:
  *y* ≠ *0* ⟹ *of-real* (*x* / *y*) =
   (*of-real x* / *of-real y* :: ′*a*::*real-field*)
⟨*proof*⟩

**lemma** *of-real-divide* [*simp*]:
  *of-real* (*x* / *y*) =
   (*of-real x* / *of-real y* :: ′*a*::{*real-field*,*division-by-zero*})
⟨*proof*⟩

**lemma** *of-real-power* [*simp*]:
  *of-real* (*x* ˆ *n*) = (*of-real x* :: ′*a*::{*real-algebra-1*,*recpower*}) ˆ *n*
⟨*proof*⟩

**lemma** *of-real-eq-iff* [*simp*]: (*of-real x* = *of-real y*) = (*x* = *y*)
⟨*proof*⟩

**lemmas** *of-real-eq-0-iff* [*simp*] = *of-real-eq-iff* [*of - 0*, *simplified*]

**lemma** *of-real-eq-id* [*simp*]: *of-real* = (*id* :: *real* ⇒ *real*)
⟨*proof*⟩

Collapse nested embeddings

**lemma** *of-real-of-nat-eq* [*simp*]: *of-real* (*of-nat n*) = *of-nat n*
⟨*proof*⟩

**lemma** *of-real-of-int-eq* [*simp*]: *of-real* (*of-int z*) = *of-int z*
⟨*proof*⟩

**lemma** *of-real-number-of-eq*:
*of-real* (*number-of w*) = (*number-of w* :: *'a*::{*number-ring*,*real-algebra-1*})
⟨*proof*⟩

Every real algebra has characteristic zero

**instance** *real-algebra-1* < *ring-char-0*
⟨*proof*⟩

## 10.4 The Set of Real Numbers

**definition**
*Reals* :: *'a*::*real-algebra-1 set* **where**
*Reals* ≡ *range of-real*

**notation** (*xsymbols*)
*Reals* (ℝ)

**lemma** *Reals-of-real* [*simp*]: *of-real r* ∈ *Reals*
⟨*proof*⟩

**lemma** *Reals-of-int* [*simp*]: *of-int z* ∈ *Reals*
⟨*proof*⟩

**lemma** *Reals-of-nat* [*simp*]: *of-nat n* ∈ *Reals*
⟨*proof*⟩

**lemma** *Reals-number-of* [*simp*]:
(*number-of w*::*'a*::{*number-ring*,*real-algebra-1*}) ∈ *Reals*
⟨*proof*⟩

**lemma** *Reals-0* [*simp*]: *0* ∈ *Reals*
⟨*proof*⟩

**lemma** *Reals-1* [*simp*]: *1* ∈ *Reals*
⟨*proof*⟩

**lemma** *Reals-add* [*simp*]: ⟦*a* ∈ *Reals*; *b* ∈ *Reals*⟧ ⟹ *a* + *b* ∈ *Reals*
⟨*proof*⟩

**lemma** *Reals-minus* [*simp*]: *a* ∈ *Reals* ⟹ − *a* ∈ *Reals*
⟨*proof*⟩

**lemma** *Reals-diff* [*simp*]: ⟦*a* ∈ *Reals*; *b* ∈ *Reals*⟧ ⟹ *a* − *b* ∈ *Reals*

⟨*proof*⟩

**lemma** *Reals-mult* [*simp*]: $[\![a \in Reals;\ b \in Reals]\!] \Longrightarrow a * b \in Reals$
⟨*proof*⟩

**lemma** *nonzero-Reals-inverse*:
  **fixes** $a$ :: $'a$::*real-div-algebra*
  **shows** $[\![a \in Reals;\ a \neq 0]\!] \Longrightarrow inverse\ a \in Reals$
⟨*proof*⟩

**lemma** *Reals-inverse* [*simp*]:
  **fixes** $a$ :: $'a$::{*real-div-algebra*,*division-by-zero*}
  **shows** $a \in Reals \Longrightarrow inverse\ a \in Reals$
⟨*proof*⟩

**lemma** *nonzero-Reals-divide*:
  **fixes** $a\ b$ :: $'a$::*real-field*
  **shows** $[\![a \in Reals;\ b \in Reals;\ b \neq 0]\!] \Longrightarrow a\ /\ b \in Reals$
⟨*proof*⟩

**lemma** *Reals-divide* [*simp*]:
  **fixes** $a\ b$ :: $'a$::{*real-field*,*division-by-zero*}
  **shows** $[\![a \in Reals;\ b \in Reals]\!] \Longrightarrow a\ /\ b \in Reals$
⟨*proof*⟩

**lemma** *Reals-power* [*simp*]:
  **fixes** $a$ :: $'a$::{*real-algebra-1*,*recpower*}
  **shows** $a \in Reals \Longrightarrow a\ \hat{}\ n \in Reals$
⟨*proof*⟩

**lemma** *Reals-cases* [*cases set*: *Reals*]:
  **assumes** $q \in \mathbb{R}$
  **obtains** (*of-real*) $r$ **where** $q = of\text{-}real\ r$
  ⟨*proof*⟩

**lemma** *Reals-induct* [*case-names of-real*, *induct set*: *Reals*]:
  $q \in \mathbb{R} \Longrightarrow (\bigwedge r.\ P\ (of\text{-}real\ r)) \Longrightarrow P\ q$
  ⟨*proof*⟩

## 10.5 Real normed vector spaces

**class** *norm* = *type* +
  **fixes** *norm* :: $'a \Rightarrow real$

**instance** *real* :: *norm*
  *real-norm-def* [*simp*]: $norm\ r \equiv |r|$ ⟨*proof*⟩

**class** *sgn-div-norm* = *scaleR* + *norm* + *sgn* +
  **assumes** *sgn-div-norm*: $sgn\ x = x\ /_R\ norm\ x$

**class** *real-normed-vector* = *real-vector* + *sgn-div-norm* +
**assumes** *norm-ge-zero* [*simp*]: $0 \leq norm\ x$
**and** *norm-eq-zero* [*simp*]: $norm\ x = 0 \longleftrightarrow x = 0$
**and** *norm-triangle-ineq*: $norm\ (x + y) \leq norm\ x + norm\ y$
**and** *norm-scaleR*: $norm\ (scaleR\ a\ x) = |a| * norm\ x$

**class** *real-normed-algebra* = *real-algebra* + *real-normed-vector* +
**assumes** *norm-mult-ineq*: $norm\ (x * y) \leq norm\ x * norm\ y$

**class** *real-normed-algebra-1* = *real-algebra-1* + *real-normed-algebra* +
**assumes** *norm-one* [*simp*]: $norm\ 1 = 1$

**class** *real-normed-div-algebra* = *real-div-algebra* + *real-normed-vector* +
**assumes** *norm-mult*: $norm\ (x * y) = norm\ x * norm\ y$

**class** *real-normed-field* = *real-field* + *real-normed-div-algebra*

**instance** *real-normed-div-algebra* < *real-normed-algebra-1*
⟨*proof*⟩

**instance** *real* :: *real-normed-field*
⟨*proof*⟩

**lemma** *norm-zero* [*simp*]: *norm* (*0*::'*a*::*real-normed-vector*) = *0*
⟨*proof*⟩

**lemma** *zero-less-norm-iff* [*simp*]:
**fixes** *x* :: '*a*::*real-normed-vector*
**shows** $(0 < norm\ x) = (x \neq 0)$
⟨*proof*⟩

**lemma** *norm-not-less-zero* [*simp*]:
**fixes** *x* :: '*a*::*real-normed-vector*
**shows** $\neg\ norm\ x < 0$
⟨*proof*⟩

**lemma** *norm-le-zero-iff* [*simp*]:
**fixes** *x* :: '*a*::*real-normed-vector*
**shows** $(norm\ x \leq 0) = (x = 0)$
⟨*proof*⟩

**lemma** *norm-minus-cancel* [*simp*]:
**fixes** *x* :: '*a*::*real-normed-vector*
**shows** $norm\ (- x) = norm\ x$
⟨*proof*⟩

**lemma** *norm-minus-commute*:
**fixes** *a b* :: '*a*::*real-normed-vector*

**shows** *norm* $(a - b)$ = *norm* $(b - a)$
⟨*proof*⟩

**lemma** *norm-triangle-ineq2*:
**fixes** $a\ b$ :: *'a::real-normed-vector*
**shows** *norm* $a$ − *norm* $b \leq$ *norm* $(a - b)$
⟨*proof*⟩

**lemma** *norm-triangle-ineq3*:
**fixes** $a\ b$ :: *'a::real-normed-vector*
**shows** |*norm* $a$ − *norm* $b$| $\leq$ *norm* $(a - b)$
⟨*proof*⟩

**lemma** *norm-triangle-ineq4*:
**fixes** $a\ b$ :: *'a::real-normed-vector*
**shows** *norm* $(a - b) \leq$ *norm* $a$ + *norm* $b$
⟨*proof*⟩

**lemma** *norm-diff-ineq*:
**fixes** $a\ b$ :: *'a::real-normed-vector*
**shows** *norm* $a$ − *norm* $b \leq$ *norm* $(a + b)$
⟨*proof*⟩

**lemma** *norm-diff-triangle-ineq*:
**fixes** $a\ b\ c\ d$ :: *'a::real-normed-vector*
**shows** *norm* $((a + b) - (c + d)) \leq$ *norm* $(a - c)$ + *norm* $(b - d)$
⟨*proof*⟩

**lemma** *abs-norm-cancel* [*simp*]:
**fixes** $a$ :: *'a::real-normed-vector*
**shows** |*norm* $a$| = *norm* $a$
⟨*proof*⟩

**lemma** *norm-add-less*:
**fixes** $x\ y$ :: *'a::real-normed-vector*
**shows** ⟦*norm* $x < r$; *norm* $y < s$⟧ $\Longrightarrow$ *norm* $(x + y) < r + s$
⟨*proof*⟩

**lemma** *norm-mult-less*:
**fixes** $x\ y$ :: *'a::real-normed-algebra*
**shows** ⟦*norm* $x < r$; *norm* $y < s$⟧ $\Longrightarrow$ *norm* $(x * y) < r * s$
⟨*proof*⟩

**lemma** *norm-of-real* [*simp*]:
*norm* (*of-real* $r$ :: *'a::real-normed-algebra-1*) = $|r|$
⟨*proof*⟩

**lemma** *norm-number-of* [*simp*]:
*norm* (*number-of* $w$::*'a::{number-ring,real-normed-algebra-1}*)

$= |$*number-of w*$|$
⟨*proof*⟩

**lemma** *norm-of-int* [*simp*]:
*norm* (*of-int z*::'*a*::*real-normed-algebra-1*) = |*of-int z*|
⟨*proof*⟩

**lemma** *norm-of-nat* [*simp*]:
*norm* (*of-nat n*::'*a*::*real-normed-algebra-1*) = *of-nat n*
⟨*proof*⟩

**lemma** *nonzero-norm-inverse*:
**fixes** *a* :: '*a*::*real-normed-div-algebra*
**shows** $a \neq 0 \Longrightarrow$ *norm* (*inverse a*) = *inverse* (*norm a*)
⟨*proof*⟩

**lemma** *norm-inverse*:
**fixes** *a* :: '*a*::{*real-normed-div-algebra*,*division-by-zero*}
**shows** *norm* (*inverse a*) = *inverse* (*norm a*)
⟨*proof*⟩

**lemma** *nonzero-norm-divide*:
**fixes** *a b* :: '*a*::*real-normed-field*
**shows** $b \neq 0 \Longrightarrow$ *norm* (*a* / *b*) = *norm a* / *norm b*
⟨*proof*⟩

**lemma** *norm-divide*:
**fixes** *a b* :: '*a*::{*real-normed-field*,*division-by-zero*}
**shows** *norm* (*a* / *b*) = *norm a* / *norm b*
⟨*proof*⟩

**lemma** *norm-power-ineq*:
**fixes** *x* :: '*a*::{*real-normed-algebra-1*,*recpower*}
**shows** *norm* (*x* ^ *n*) $\leq$ *norm x* ^ *n*
⟨*proof*⟩

**lemma** *norm-power*:
**fixes** *x* :: '*a*::{*real-normed-div-algebra*,*recpower*}
**shows** *norm* (*x* ^ *n*) = *norm x* ^ *n*
⟨*proof*⟩

## 10.6 Sign function

**lemma** *norm-sgn*:
*norm* (*sgn*(*x*::'*a*::*real-normed-vector*)) = (*if x* = *0 then 0 else 1*)
⟨*proof*⟩

**lemma** *sgn-zero* [*simp*]: *sgn*(*0*::'*a*::*real-normed-vector*) = *0*
⟨*proof*⟩

**lemma** *sgn-zero-iff*: $(sgn(x::'a::real\text{-}normed\text{-}vector) = 0) = (x = 0)$
⟨*proof*⟩

**lemma** *sgn-minus*: $sgn\ (-\ x) = -\ sgn(x::'a::real\text{-}normed\text{-}vector)$
⟨*proof*⟩

**lemma** *sgn-scaleR*:
$sgn\ (scaleR\ r\ x) = scaleR\ (sgn\ r)\ (sgn(x::'a::real\text{-}normed\text{-}vector))$
⟨*proof*⟩

**lemma** *sgn-one* [*simp*]: $sgn\ (1::'a::real\text{-}normed\text{-}algebra\text{-}1) = 1$
⟨*proof*⟩

**lemma** *sgn-of-real*:
$sgn\ (of\text{-}real\ r::'a::real\text{-}normed\text{-}algebra\text{-}1) = of\text{-}real\ (sgn\ r)$
⟨*proof*⟩

**lemma** *sgn-mult*:
**fixes** $x\ y\ ::\ 'a::real\text{-}normed\text{-}div\text{-}algebra$
**shows** $sgn\ (x * y) = sgn\ x * sgn\ y$
⟨*proof*⟩

**lemma** *real-sgn-eq*: $sgn\ (x::real) = x\ /\ |x|$
⟨*proof*⟩

**lemma** *real-sgn-pos*: $0 < (x::real) \Longrightarrow sgn\ x = 1$
⟨*proof*⟩

**lemma** *real-sgn-neg*: $(x::real) < 0 \Longrightarrow sgn\ x = -1$
⟨*proof*⟩

## 10.7 Bounded Linear and Bilinear Operators

**locale** *bounded-linear* = *additive* +
**constrains** $f\ ::\ 'a::real\text{-}normed\text{-}vector \Rightarrow 'b::real\text{-}normed\text{-}vector$
**assumes** *scaleR*: $f\ (scaleR\ r\ x) = scaleR\ r\ (f\ x)$
**assumes** *bounded*: $\exists\, K.\ \forall\, x.\ norm\ (f\ x) \le norm\ x * K$

**lemma** (**in** *bounded-linear*) *pos-bounded*:
$\exists\, K{>}0.\ \forall\, x.\ norm\ (f\ x) \le norm\ x * K$
⟨*proof*⟩

**lemma** (**in** *bounded-linear*) *nonneg-bounded*:
$\exists\, K{\ge}0.\ \forall\, x.\ norm\ (f\ x) \le norm\ x * K$
⟨*proof*⟩

**locale** *bounded-bilinear* =
**fixes** $prod\ ::\ ['a::real\text{-}normed\text{-}vector,\ 'b::real\text{-}normed\text{-}vector]$

$\Rightarrow$ *'c::real-normed-vector*
(**infixl** $**$ *70*)
**assumes** *add-left*: *prod* $(a + a')$ $b$ = *prod* $a$ $b$ + *prod* $a'$ $b$
**assumes** *add-right*: *prod* $a$ $(b + b')$ = *prod* $a$ $b$ + *prod* $a$ $b'$
**assumes** *scaleR-left*: *prod* (*scaleR* $r$ $a$) $b$ = *scaleR* $r$ (*prod* $a$ $b$)
**assumes** *scaleR-right*: *prod* $a$ (*scaleR* $r$ $b$) = *scaleR* $r$ (*prod* $a$ $b$)
**assumes** *bounded*: $\exists K. \forall a\ b.$ *norm* (*prod* $a$ $b$) $\leq$ *norm* $a$ $*$ *norm* $b$ $*$ $K$

**lemma** (**in** *bounded-bilinear*) *pos-bounded*:
$\exists K>0. \forall a\ b.$ *norm* $(a ** b) \leq$ *norm* $a$ $*$ *norm* $b$ $*$ $K$
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *nonneg-bounded*:
$\exists K\geq 0. \forall a\ b.$ *norm* $(a ** b) \leq$ *norm* $a$ $*$ *norm* $b$ $*$ $K$
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *additive-right*: *additive* ($\lambda b.$ *prod* $a$ $b$)
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *additive-left*: *additive* ($\lambda a.$ *prod* $a$ $b$)
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *zero-left*: *prod* $0$ $b$ = $0$
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *zero-right*: *prod* $a$ $0$ = $0$
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *minus-left*: *prod* $(-\ a)$ $b$ = $-$ *prod* $a$ $b$
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *minus-right*: *prod* $a$ $(-\ b)$ = $-$ *prod* $a$ $b$
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *diff-left*:
*prod* $(a - a')$ $b$ = *prod* $a$ $b$ $-$ *prod* $a'$ $b$
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *diff-right*:
*prod* $a$ $(b - b')$ = *prod* $a$ $b$ $-$ *prod* $a$ $b'$
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *bounded-linear-left*:
*bounded-linear* ($\lambda a.$ $a ** b$)
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *bounded-linear-right*:
*bounded-linear* ($\lambda b.$ $a ** b$)
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *prod-diff-prod*:
$(x ** y - a ** b) = (x - a) ** (y - b) + (x - a) ** b + a ** (y - b)$
⟨*proof*⟩

**interpretation** *mult*:
*bounded-bilinear* [$op * :: 'a \Rightarrow 'a \Rightarrow 'a$::*real-normed-algebra*]
⟨*proof*⟩

**interpretation** *mult-left*:
*bounded-linear* [($\lambda x$::'$a$::*real-normed-algebra*. $x * y$)]
⟨*proof*⟩

**interpretation** *mult-right*:
*bounded-linear* [($\lambda y$::'$a$::*real-normed-algebra*. $x * y$)]
⟨*proof*⟩

**interpretation** *divide*:
*bounded-linear* [($\lambda x$::'$a$::*real-normed-field*. $x / y$)]
⟨*proof*⟩

**interpretation** *scaleR*: *bounded-bilinear* [*scaleR*]
⟨*proof*⟩

**interpretation** *scaleR-left*: *bounded-linear* [$\lambda r$. *scaleR* $r$ $x$]
⟨*proof*⟩

**interpretation** *scaleR-right*: *bounded-linear* [$\lambda x$. *scaleR* $r$ $x$]
⟨*proof*⟩

**interpretation** *of-real*: *bounded-linear* [$\lambda r$. *of-real* $r$]
⟨*proof*⟩

**end**

**theory** *Real*
**imports** *ContNotDenum RealVector*
**begin**
**end**

# 11 Float: Floating Point Representation of the Reals

**theory** *Float*

**imports** *Real Parity*
**uses** ~~/*src*/*Tools*/*float.ML* (*float-arith.ML*)
**begin**

**definition**
*pow2* :: *int* ⇒ *real* **where**
$pow2\ a = (if\ (0 <= a)\ then\ (2\hat{}(nat\ a))\ else\ (inverse\ (2\hat{}(nat\ (-a)))))$

**definition**
*float* :: *int* ∗ *int* ⇒ *real* **where**
$float\ x = real\ (fst\ x) * pow2\ (snd\ x)$

**lemma** *pow2-0*[*simp*]: $pow2\ 0 = 1$
⟨*proof*⟩

**lemma** *pow2-1*[*simp*]: $pow2\ 1 = 2$
⟨*proof*⟩

**lemma** *pow2-neg*: $pow2\ x = inverse\ (pow2\ (-x))$
⟨*proof*⟩

**lemma** *pow2-add1*: $pow2\ (1 + a) = 2 * (pow2\ a)$
⟨*proof*⟩

**lemma** *pow2-add*: $pow2\ (a+b) = (pow2\ a) * (pow2\ b)$
⟨*proof*⟩

**lemma** $float\ (a,\ e) + float\ (b,\ e) = float\ (a + b,\ e)$
⟨*proof*⟩

**definition**
*int-of-real* :: *real* ⇒ *int* **where**
$int\text{-}of\text{-}real\ x = (SOME\ y.\ real\ y = x)$

**definition**
*real-is-int* :: *real* ⇒ *bool* **where**
$real\text{-}is\text{-}int\ x = (EX\ (u::int).\ x = real\ u)$

**lemma** *real-is-int-def2*: $real\text{-}is\text{-}int\ x = (x = real\ (int\text{-}of\text{-}real\ x))$
⟨*proof*⟩

**lemma** *float-transfer*: $real\text{-}is\text{-}int\ ((real\ a)*(pow2\ c)) \Longrightarrow float\ (a,\ b) = float\ (int\text{-}of\text{-}real\ ((real\ a)*(pow2\ c)),\ b - c)$
⟨*proof*⟩

**lemma** *pow2-int*: $pow2\ (int\ c) = (2::real)\hat{}c$
⟨*proof*⟩

**lemma** *float-transfer-nat*: $float\ (a,\ b) = float\ (a * 2\hat{}c,\ b - int\ c)$

⟨*proof*⟩

**lemma** *real-is-int-real*[*simp*]: *real-is-int* (*real* (*x*::*int*))
⟨*proof*⟩

**lemma** *int-of-real-real*[*simp*]: *int-of-real* (*real* *x*) = *x*
⟨*proof*⟩

**lemma** *real-int-of-real*[*simp*]: *real-is-int* *x* $\Longrightarrow$ *real* (*int-of-real* *x*) = *x*
⟨*proof*⟩

**lemma** *real-is-int-add-int-of-real*: *real-is-int* *a* $\Longrightarrow$ *real-is-int* *b* $\Longrightarrow$ (*int-of-real* (*a*+*b*)) = (*int-of-real* *a*) + (*int-of-real* *b*)
⟨*proof*⟩

**lemma** *real-is-int-add*[*simp*]: *real-is-int* *a* $\Longrightarrow$ *real-is-int* *b* $\Longrightarrow$ *real-is-int* (*a*+*b*)
⟨*proof*⟩

**lemma** *int-of-real-sub*: *real-is-int* *a* $\Longrightarrow$ *real-is-int* *b* $\Longrightarrow$ (*int-of-real* (*a*−*b*)) = (*int-of-real* *a*) − (*int-of-real* *b*)
⟨*proof*⟩

**lemma** *real-is-int-sub*[*simp*]: *real-is-int* *a* $\Longrightarrow$ *real-is-int* *b* $\Longrightarrow$ *real-is-int* (*a*−*b*)
⟨*proof*⟩

**lemma** *real-is-int-rep*: *real-is-int* *x* $\Longrightarrow$ *?!* (*a*::*int*). *real* *a* = *x*
⟨*proof*⟩

**lemma** *int-of-real-mult*:
  **assumes** *real-is-int* *a* *real-is-int* *b*
  **shows** (*int-of-real* (*a*∗*b*)) = (*int-of-real* *a*) ∗ (*int-of-real* *b*)
⟨*proof*⟩

**lemma** *real-is-int-mult*[*simp*]: *real-is-int* *a* $\Longrightarrow$ *real-is-int* *b* $\Longrightarrow$ *real-is-int* (*a*∗*b*)
⟨*proof*⟩

**lemma** *real-is-int-0*[*simp*]: *real-is-int* (*0*::*real*)
⟨*proof*⟩

**lemma** *real-is-int-1*[*simp*]: *real-is-int* (*1*::*real*)
⟨*proof*⟩

**lemma** *real-is-int-n1*: *real-is-int* (−*1*::*real*)
⟨*proof*⟩

**lemma** *real-is-int-number-of*[*simp*]: *real-is-int* ((*number-of* :: *int* ⇒ *real*) *x*)
⟨*proof*⟩

**lemma** *int-of-real-0*[*simp*]: *int-of-real* (*0*::*real*) = (*0*::*int*)

⟨*proof*⟩

**lemma** *int-of-real-1* [*simp*]: *int-of-real* (*1*::*real*) = (*1*::*int*)
⟨*proof*⟩

**lemma** *int-of-real-number-of* [*simp*]: *int-of-real* (*number-of b*) = *number-of b*
⟨*proof*⟩

**lemma** *float-transfer-even*: *even a* $\Longrightarrow$ *float* (*a*, *b*) = *float* (*a div 2*, *b+1*)
⟨*proof*⟩

**consts**
*norm-float* :: *int*∗*int* ⇒ *int*∗*int*

**lemma** *int-div-zdiv*: *int* (*a div b*) = (*int a*) *div* (*int b*)
⟨*proof*⟩

**lemma** *int-mod-zmod*: *int* (*a mod b*) = (*int a*) *mod* (*int b*)
⟨*proof*⟩

**lemma** *abs-div-2-less*: $a \neq 0 \Longrightarrow a \neq -1 \Longrightarrow$ *abs*((*a*::*int*) *div 2*) < *abs a*
⟨*proof*⟩

**lemma** *terminating-norm-float*: ∀ *a*. (*a*::*int*) ≠ *0* ∧ *even a* ⟶ *a* ≠ *0* ∧ |*a div 2*| < |*a*|
⟨*proof*⟩

**declare** [[*simp-depth-limit* = *2*]]
**recdef** *norm-float measure* (% (*a*,*b*). *nat* (*abs a*))
*norm-float* (*a*,*b*) = (*if* (*a* ≠ *0*) & (*even a*) *then norm-float* (*a div 2*, *b+1*) *else* (*if a*=*0 then* (*0*,*0*) *else* (*a*,*b*)))
(**hints** *simp*: *even-def terminating-norm-float*)
**declare** [[*simp-depth-limit* = *100*]]

**lemma** *norm-float*: *float x* = *float* (*norm-float x*)
⟨*proof*⟩

**lemma** *pow2-int*: *pow2* (*int n*) = *2*^*n*
⟨*proof*⟩

**lemma** *float-add-l0*: *float* (*0*, *e*) + *x* = *x*
⟨*proof*⟩

**lemma** *float-add-r0*: *x* + *float* (*0*, *e*) = *x*
⟨*proof*⟩

**lemma** *float-add*:
*float* (*a1*, *e1*) + *float* (*a2*, *e2*) =
(*if e1*<=*e2 then float* (*a1*+*a2*∗*2*^(*nat*(*e2*−*e1*)), *e1*)

*else float* $(a1{*}2\hat{}(nat\ (e1{-}e2)){+}a2,\ e2))$
⟨*proof*⟩

**lemma** *float-add-assoc1*:
$(x + float\ (y1,\ e1)) + float\ (y2,\ e2) = (float\ (y1,\ e1) + float\ (y2,\ e2)) + x$
⟨*proof*⟩

**lemma** *float-add-assoc2*:
$(float\ (y1,\ e1) + x) + float\ (y2,\ e2) = (float\ (y1,\ e1) + float\ (y2,\ e2)) + x$
⟨*proof*⟩

**lemma** *float-add-assoc3*:
$float\ (y1,\ e1) + (x + float\ (y2,\ e2)) = (float\ (y1,\ e1) + float\ (y2,\ e2)) + x$
⟨*proof*⟩

**lemma** *float-add-assoc4*:
$float\ (y1,\ e1) + (float\ (y2,\ e2) + x) = (float\ (y1,\ e1) + float\ (y2,\ e2)) + x$
⟨*proof*⟩

**lemma** *float-mult-l0*: $float\ (0,\ e) * x = float\ (0,\ 0)$
⟨*proof*⟩

**lemma** *float-mult-r0*: $x * float\ (0,\ e) = float\ (0,\ 0)$
⟨*proof*⟩

**definition**
$lbound :: real \Rightarrow real$
**where**
$lbound\ x = min\ 0\ x$

**definition**
$ubound :: real \Rightarrow real$
**where**
$ubound\ x = max\ 0\ x$

**lemma** *lbound*: $lbound\ x \leq x$
⟨*proof*⟩

**lemma** *ubound*: $x \leq ubound\ x$
⟨*proof*⟩

**lemma** *float-mult*:
$float\ (a1,\ e1) * float\ (a2,\ e2) =$
$(float\ (a1 * a2,\ e1 + e2))$
⟨*proof*⟩

**lemma** *float-minus*:
$-\ (float\ (a,b)) = float\ (-a,\ b)$
⟨*proof*⟩

**lemma** *zero-less-pow2*:
$0 < pow2\ x$
⟨*proof*⟩

**lemma** *zero-le-float*:
$(0 <= float\ (a,b)) = (0 <= a)$
⟨*proof*⟩

**lemma** *float-le-zero*:
$(float\ (a,b) <= 0) = (a <= 0)$
⟨*proof*⟩

**lemma** *float-abs*:
$abs\ (float\ (a,b)) = (if\ 0 <= a\ then\ (float\ (a,b))\ else\ (float\ (-a,b)))$
⟨*proof*⟩

**lemma** *float-zero*:
$float\ (0,\ b) = 0$
⟨*proof*⟩

**lemma** *float-pprt*:
$pprt\ (float\ (a,\ b)) = (if\ 0 <= a\ then\ (float\ (a,b))\ else\ (float\ (0,\ b)))$
⟨*proof*⟩

**lemma** *pprt-lbound*: $pprt\ (lbound\ x) = float\ (0,\ 0)$
⟨*proof*⟩

**lemma** *nprt-ubound*: $nprt\ (ubound\ x) = float\ (0,\ 0)$
⟨*proof*⟩

**lemma** *float-nprt*:
$nprt\ (float\ (a,\ b)) = (if\ 0 <= a\ then\ (float\ (0,b))\ else\ (float\ (a,\ b)))$
⟨*proof*⟩

**lemma** *norm-0-1*: $(0::\text{-}::number\text{-}ring) = Numeral0\ \&\ (1::\text{-}::number\text{-}ring) = Numeral1$
⟨*proof*⟩

**lemma** *add-left-zero*: $0 + a = (a::'a::comm\text{-}monoid\text{-}add)$
⟨*proof*⟩

**lemma** *add-right-zero*: $a + 0 = (a::'a::comm\text{-}monoid\text{-}add)$
⟨*proof*⟩

**lemma** *mult-left-one*: $1 * a = (a::'a::semiring\text{-}1)$
⟨*proof*⟩

**lemma** *mult-right-one*: $a * 1 = (a::'a::semiring\text{-}1)$
⟨*proof*⟩

**lemma** *int-pow-0*: $(a{::}int)\,\hat{}\,(Numeral0) = 1$
⟨*proof*⟩

**lemma** *int-pow-1*: $(a{::}int)\,\hat{}\,(Numeral1) = a$
⟨*proof*⟩

**lemma** *zero-eq-Numeral0-nring*: $(0{::}'a{::}number\text{-}ring) = Numeral0$
⟨*proof*⟩

**lemma** *one-eq-Numeral1-nring*: $(1{::}'a{::}number\text{-}ring) = Numeral1$
⟨*proof*⟩

**lemma** *zero-eq-Numeral0-nat*: $(0{::}nat) = Numeral0$
⟨*proof*⟩

**lemma** *one-eq-Numeral1-nat*: $(1{::}nat) = Numeral1$
⟨*proof*⟩

**lemma** *zpower-Pls*: $(z{::}int)\,\hat{}\,Numeral0 = Numeral1$
⟨*proof*⟩

**lemma** *zpower-Min*: $(z{::}int)\,\hat{}\,((-1){::}nat) = Numeral1$
⟨*proof*⟩

**lemma** *fst-cong*: $a{=}a' \Longrightarrow fst\ (a,b) = fst\ (a',b)$
⟨*proof*⟩

**lemma** *snd-cong*: $b{=}b' \Longrightarrow snd\ (a,b) = snd\ (a,b')$
⟨*proof*⟩

**lemma** *lift-bool*: $x \Longrightarrow x{=}True$
⟨*proof*⟩

**lemma** *nlift-bool*: $\sim x \Longrightarrow x{=}False$
⟨*proof*⟩

**lemma** *not-false-eq-true*: $(\sim\ False) = True$ ⟨*proof*⟩

**lemma** *not-true-eq-false*: $(\sim\ True) = False$ ⟨*proof*⟩

**lemmas** *binarith* =
*Pls-0-eq Min-1-eq*
*pred-Pls pred-Min pred-1 pred-0*
*succ-Pls succ-Min succ-1 succ-0*
*add-Pls add-Min add-BIT-0 add-BIT-10*
*add-BIT-11 minus-Pls minus-Min minus-1*
*minus-0 mult-Pls mult-Min mult-num1 mult-num0*
*add-Pls-right add-Min-right*

**lemma** *int-eq-number-of-eq*:
$(((\textit{number-of}\ v)::\textit{int})=(\textit{number-of}\ w)) = \textit{iszero}\ ((\textit{number-of}\ (v + \textit{uminus}\ w))::\textit{int})$
⟨*proof*⟩

**lemma** *int-iszero-number-of-Pls*: $\textit{iszero}\ (\textit{Numeral0}::\textit{int})$
⟨*proof*⟩

**lemma** *int-nonzero-number-of-Min*: $\sim(\textit{iszero}\ ((-1)::\textit{int}))$
⟨*proof*⟩

**lemma** *int-iszero-number-of-0*: $\textit{iszero}\ ((\textit{number-of}\ (w\ \textit{BIT}\ \textit{bit.B0}))::\textit{int}) = \textit{iszero}\ ((\textit{number-of}\ w)::\textit{int})$
⟨*proof*⟩

**lemma** *int-iszero-number-of-1*: $\neg\ \textit{iszero}\ ((\textit{number-of}\ (w\ \textit{BIT}\ \textit{bit.B1}))::\textit{int})$
⟨*proof*⟩

**lemma** *int-less-number-of-eq-neg*: $(((\textit{number-of}\ x)::\textit{int}) < \textit{number-of}\ y) = \textit{neg}\ ((\textit{number-of}\ (x + (\textit{uminus}\ y)))::\textit{int})$
⟨*proof*⟩

**lemma** *int-not-neg-number-of-Pls*: $\neg\ (\textit{neg}\ (\textit{Numeral0}::\textit{int}))$
⟨*proof*⟩

**lemma** *int-neg-number-of-Min*: $\textit{neg}\ (-1::\textit{int})$
⟨*proof*⟩

**lemma** *int-neg-number-of-BIT*: $\textit{neg}\ ((\textit{number-of}\ (w\ \textit{BIT}\ x))::\textit{int}) = \textit{neg}\ ((\textit{number-of}\ w)::\textit{int})$
⟨*proof*⟩

**lemma** *int-le-number-of-eq*: $(((\textit{number-of}\ x)::\textit{int}) \le \textit{number-of}\ y) = (\neg\ \textit{neg}\ ((\textit{number-of}\ (y + (\textit{uminus}\ x)))::\textit{int}))$
⟨*proof*⟩

**lemmas** *intarithrel* =
*int-eq-number-of-eq*
*lift-bool*[*OF int-iszero-number-of-Pls*] *nlift-bool*[*OF int-nonzero-number-of-Min*]
*int-iszero-number-of-0*
*lift-bool*[*OF int-iszero-number-of-1*] *int-less-number-of-eq-neg* *nlift-bool*[*OF int-not-neg-number-of-Pls*]
*lift-bool*[*OF int-neg-number-of-Min*]
*int-neg-number-of-BIT* *int-le-number-of-eq*

**lemma** *int-number-of-add-sym*: $((\textit{number-of}\ v)::\textit{int}) + \textit{number-of}\ w = \textit{number-of}\ (v + w)$
⟨*proof*⟩

**lemma** *int-number-of-diff-sym*: $((\textit{number-of}\ v)::\textit{int}) - \textit{number-of}\ w = \textit{number-of}$

$(v + (uminus\ w))$
⟨*proof*⟩

**lemma** *int-number-of-mult-sym*: $((number\text{-}of\ v)::int) * number\text{-}of\ w = number\text{-}of\ (v * w)$
⟨*proof*⟩

**lemma** *int-number-of-minus-sym*: $-\ ((number\text{-}of\ v)::int) = number\text{-}of\ (uminus\ v)$
⟨*proof*⟩

**lemmas** *intarith* = *int-number-of-add-sym int-number-of-minus-sym int-number-of-diff-sym int-number-of-mult-sym*

**lemmas** *natarith* = *add-nat-number-of diff-nat-number-of mult-nat-number-of eq-nat-number-of less-nat-number-of*

**lemmas** *powerarith* = *nat-number-of zpower-number-of-even*
*zpower-number-of-odd*[*simplified zero-eq-Numeral0-nring one-eq-Numeral1-nring*]
*zpower-Pls zpower-Min*

**lemmas** *floatarith*[*simplified norm-0-1*] = *float-add float-add-l0 float-add-r0 float-mult float-mult-l0 float-mult-r0*
*float-minus float-abs zero-le-float float-pprt float-nprt pprt-lbound nprt-ubound*

**lemmas** *arith* = *binarith intarith intarithrel natarith powerarith floatarith not-false-eq-true not-true-eq-false*

⟨*ML*⟩

**end**

# 12 SEQ: Sequences and Convergence

**theory** *SEQ*
**imports** *../Real/Real*
**begin**

**definition**
*Zseq* :: $[nat \Rightarrow {}'a::real\text{-}normed\text{-}vector] \Rightarrow bool$ **where**
— Standard definition of sequence converging to zero
$Zseq\ X = (\forall r>0.\ \exists no.\ \forall n\geq no.\ norm\ (X\ n) < r)$

**definition**
*LIMSEQ* :: [*nat* => *'a::real-normed-vector*, *'a*] => *bool*
(((-)/ ----> (-)) [*60*, *60*] *60*) **where**
— Standard definition of convergence of sequence
*X* ----> *L* = (∀ *r*. *0* < *r* --> (∃ *no*. ∀ *n*. *no* ≤ *n* --> *norm* (*X n* − *L*) <

$r$))

**definition**
*lim* :: (*nat* => *'a*::*real-normed-vector*) => *'a* **where**
— Standard definition of limit using choice operator
$lim\ X = (THE\ L.\ X \dashrightarrow L)$

**definition**
*convergent* :: (*nat* => *'a*::*real-normed-vector*) => *bool* **where**
— Standard definition of convergence
$convergent\ X = (\exists L.\ X \dashrightarrow L)$

**definition**
*Bseq* :: (*nat* => *'a*::*real-normed-vector*) => *bool* **where**
— Standard definition for bounded sequence
$Bseq\ X = (\exists K>0.\forall n.\ norm\ (X\ n) \le K)$

**definition**
*monoseq* :: (*nat*=>*real*)=>*bool* **where**
— Definition for monotonicity
$monoseq\ X = ((\forall m.\ \forall n\ge m.\ X\ m \le X\ n) \mid (\forall m.\ \forall n\ge m.\ X\ n \le X\ m))$

**definition**
*subseq* :: (*nat* => *nat*) => *bool* **where**
— Definition of subsequence
$subseq\ f = (\forall m.\ \forall n>m.\ (f\ m) < (f\ n))$

**definition**
*Cauchy* :: (*nat* => *'a*::*real-normed-vector*) => *bool* **where**
— Standard definition of the Cauchy condition
$Cauchy\ X = (\forall e>0.\ \exists M.\ \forall m \ge M.\ \forall n \ge M.\ norm\ (X\ m - X\ n) < e)$

## 12.1 Bounded Sequences

**lemma** *BseqI*: **assumes** $K$: $\bigwedge n.\ norm\ (X\ n) \le K$ **shows** *Bseq X*
⟨*proof*⟩

**lemma** *BseqD*: $Bseq\ X \Longrightarrow \exists K>0.\ \forall n.\ norm\ (X\ n) \le K$
⟨*proof*⟩

**lemma** *BseqE*: $[\![Bseq\ X;\ \bigwedge K.\ [\![0 < K;\ \forall n.\ norm\ (X\ n) \le K]\!] \Longrightarrow Q]\!] \Longrightarrow Q$
⟨*proof*⟩

**lemma** *BseqI2*: **assumes** $K$: $\forall n\ge N.\ norm\ (X\ n) \le K$ **shows** *Bseq X*
⟨*proof*⟩

**lemma** *Bseq-ignore-initial-segment*: $Bseq\ X \Longrightarrow Bseq\ (\lambda n.\ X\ (n + k))$
⟨*proof*⟩

**lemma** *Bseq-offset*: $Bseq\ (\lambda n.\ X\ (n + k)) \Longrightarrow Bseq\ X$
⟨*proof*⟩

## 12.2 Sequences That Converge to Zero

**lemma** *ZseqI*:
$(\bigwedge r.\ 0 < r \Longrightarrow \exists\, no.\ \forall\, n \geq no.\ norm\ (X\ n) < r) \Longrightarrow Zseq\ X$
⟨*proof*⟩

**lemma** *ZseqD*:
$[\![Zseq\ X;\ 0 < r]\!] \Longrightarrow \exists\, no.\ \forall\, n \geq no.\ norm\ (X\ n) < r$
⟨*proof*⟩

**lemma** *Zseq-zero*: $Zseq\ (\lambda n.\ 0)$
⟨*proof*⟩

**lemma** *Zseq-const-iff*: $Zseq\ (\lambda n.\ k) = (k = 0)$
⟨*proof*⟩

**lemma** *Zseq-norm-iff*: $Zseq\ (\lambda n.\ norm\ (X\ n)) = Zseq\ (\lambda n.\ X\ n)$
⟨*proof*⟩

**lemma** *Zseq-imp-Zseq*:
**assumes** $X$: $Zseq\ X$
**assumes** $Y$: $\bigwedge n.\ norm\ (Y\ n) \leq norm\ (X\ n) * K$
**shows** $Zseq\ (\lambda n.\ Y\ n)$
⟨*proof*⟩

**lemma** *Zseq-le*: $[\![Zseq\ Y;\ \forall\, n.\ norm\ (X\ n) \leq norm\ (Y\ n)]\!] \Longrightarrow Zseq\ X$
⟨*proof*⟩

**lemma** *Zseq-add*:
**assumes** $X$: $Zseq\ X$
**assumes** $Y$: $Zseq\ Y$
**shows** $Zseq\ (\lambda n.\ X\ n + Y\ n)$
⟨*proof*⟩

**lemma** *Zseq-minus*: $Zseq\ X \Longrightarrow Zseq\ (\lambda n.\ -\ X\ n)$
⟨*proof*⟩

**lemma** *Zseq-diff*: $[\![Zseq\ X;\ Zseq\ Y]\!] \Longrightarrow Zseq\ (\lambda n.\ X\ n\ -\ Y\ n)$
⟨*proof*⟩

**lemma** (**in** *bounded-linear*) *Zseq*:
**assumes** $X$: $Zseq\ X$
**shows** $Zseq\ (\lambda n.\ f\ (X\ n))$
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *Zseq*:

**assumes** $X$: *Zseq X*
**assumes** $Y$: *Zseq Y*
**shows** *Zseq* $(\lambda n.\ X\ n\ {*}{*}\ Y\ n)$
$\langle proof \rangle$

**lemma** (**in** *bounded-bilinear*) *Zseq-prod-Bseq*:
**assumes** $X$: *Zseq X*
**assumes** $Y$: *Bseq Y*
**shows** *Zseq* $(\lambda n.\ X\ n\ {*}{*}\ Y\ n)$
$\langle proof \rangle$

**lemma** (**in** *bounded-bilinear*) *Bseq-prod-Zseq*:
**assumes** $X$: *Bseq X*
**assumes** $Y$: *Zseq Y*
**shows** *Zseq* $(\lambda n.\ X\ n\ {*}{*}\ Y\ n)$
$\langle proof \rangle$

**lemma** (**in** *bounded-bilinear*) *Zseq-left*:
*Zseq X* $\Longrightarrow$ *Zseq* $(\lambda n.\ X\ n\ {*}{*}\ a)$
$\langle proof \rangle$

**lemma** (**in** *bounded-bilinear*) *Zseq-right*:
*Zseq X* $\Longrightarrow$ *Zseq* $(\lambda n.\ a\ {*}{*}\ X\ n)$
$\langle proof \rangle$

**lemmas** *Zseq-mult* = *mult.Zseq*
**lemmas** *Zseq-mult-right* = *mult.Zseq-right*
**lemmas** *Zseq-mult-left* = *mult.Zseq-left*

## 12.3 Limits of Sequences

**lemma** *LIMSEQ-iff*:
$(X\ {-}{-}{-}{-}{>}\ L) = (\forall r{>}0.\ \exists no.\ \forall n \geq no.\ norm\ (X\ n - L) < r)$
$\langle proof \rangle$

**lemma** *LIMSEQ-Zseq-iff*: $((\lambda n.\ X\ n)\ {-}{-}{-}{-}{>}\ L) = Zseq\ (\lambda n.\ X\ n - L)$
$\langle proof \rangle$

**lemma** *LIMSEQ-I*:
$(\bigwedge r.\ 0 < r \Longrightarrow \exists no.\ \forall n{\geq}no.\ norm\ (X\ n - L) < r) \Longrightarrow X\ {-}{-}{-}{-}{>}\ L$
$\langle proof \rangle$

**lemma** *LIMSEQ-D*:
$[\![X\ {-}{-}{-}{-}{>}\ L;\ 0 < r]\!] \Longrightarrow \exists no.\ \forall n{\geq}no.\ norm\ (X\ n - L) < r$
$\langle proof \rangle$

**lemma** *LIMSEQ-const*: $(\lambda n.\ k)\ {-}{-}{-}{-}{>}\ k$
$\langle proof \rangle$

**lemma** *LIMSEQ-const-iff*: $(\lambda n.\ k)$ $----\!>$ $l = (k = l)$
⟨*proof*⟩

**lemma** *LIMSEQ-norm*: $X$ $----\!>$ $a \Longrightarrow (\lambda n.\ norm\ (X\ n))$ $----\!>$ $norm\ a$
⟨*proof*⟩

**lemma** *LIMSEQ-ignore-initial-segment*:
$f$ $----\!>$ $a \Longrightarrow (\lambda n.\ f\ (n + k))$ $----\!>$ $a$
⟨*proof*⟩

**lemma** *LIMSEQ-offset*:
$(\lambda n.\ f\ (n + k))$ $----\!>$ $a \Longrightarrow f$ $----\!>$ $a$
⟨*proof*⟩

**lemma** *LIMSEQ-Suc*: $f$ $----\!>$ $l \Longrightarrow (\lambda n.\ f\ (Suc\ n))$ $----\!>$ $l$
⟨*proof*⟩

**lemma** *LIMSEQ-imp-Suc*: $(\lambda n.\ f\ (Suc\ n))$ $----\!>$ $l \Longrightarrow f$ $----\!>$ $l$
⟨*proof*⟩

**lemma** *LIMSEQ-Suc-iff*: $(\lambda n.\ f\ (Suc\ n))$ $----\!>$ $l = f$ $----\!>$ $l$
⟨*proof*⟩

**lemma** *add-diff-add*:
**fixes** $a\ b\ c\ d$ :: $'a$::*ab-group-add*
**shows** $(a + c) - (b + d) = (a - b) + (c - d)$
⟨*proof*⟩

**lemma** *minus-diff-minus*:
**fixes** $a\ b$ :: $'a$::*ab-group-add*
**shows** $(-\ a) - (-\ b) = -\ (a - b)$
⟨*proof*⟩

**lemma** *LIMSEQ-add*: $[\![X$ $----\!>$ $a;\ Y$ $----\!>$ $b]\!] \Longrightarrow (\lambda n.\ X\ n + Y\ n)$ $----\!>$ $a + b$
⟨*proof*⟩

**lemma** *LIMSEQ-minus*: $X$ $----\!>$ $a \Longrightarrow (\lambda n.\ -\ X\ n)$ $----\!>$ $-\ a$
⟨*proof*⟩

**lemma** *LIMSEQ-minus-cancel*: $(\lambda n.\ -\ X\ n)$ $----\!>$ $-\ a \Longrightarrow X$ $----\!>$ $a$
⟨*proof*⟩

**lemma** *LIMSEQ-diff*: $[\![X$ $----\!>$ $a;\ Y$ $----\!>$ $b]\!] \Longrightarrow (\lambda n.\ X\ n - Y\ n)$ $----\!>$ $a - b$
⟨*proof*⟩

**lemma** *LIMSEQ-unique*: $[\![X$ $----\!>$ $a;\ X$ $----\!>$ $b]\!] \Longrightarrow a = b$
⟨*proof*⟩

**lemma** (**in** *bounded-linear*) *LIMSEQ*:
$X ----> a \Longrightarrow (\lambda n.\ f\ (X\ n)) ----> f\ a$
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *LIMSEQ*:
$[\![X ----> a;\ Y ----> b]\!] \Longrightarrow (\lambda n.\ X\ n\ {*}{*}\ Y\ n) ----> a\ {*}{*}\ b$
⟨*proof*⟩

**lemma** *LIMSEQ-mult*:
**fixes** $a\ b$ :: $'a$::*real-normed-algebra*
**shows** $[|\ X ----> a;\ Y ----> b\ |] ==> (\%n.\ X\ n * Y\ n) ----> a * b$
⟨*proof*⟩

**lemma** *inverse-diff-inverse*:
$[\![(a::'a::division\text{-}ring) \neq 0;\ b \neq 0]\!]$
$\Longrightarrow inverse\ a - inverse\ b = - (inverse\ a * (a - b) * inverse\ b)$
⟨*proof*⟩

**lemma** *Bseq-inverse-lemma*:
**fixes** $x$ :: $'a$::*real-normed-div-algebra*
**shows** $[\![r \leq norm\ x;\ 0 < r]\!] \Longrightarrow norm\ (inverse\ x) \leq inverse\ r$
⟨*proof*⟩

**lemma** *Bseq-inverse*:
**fixes** $a$ :: $'a$::*real-normed-div-algebra*
**assumes** $X$: $X ----> a$
**assumes** $a$: $a \neq 0$
**shows** $Bseq\ (\lambda n.\ inverse\ (X\ n))$
⟨*proof*⟩

**lemma** *LIMSEQ-inverse-lemma*:
**fixes** $a$ :: $'a$::*real-normed-div-algebra*
**shows** $[\![X ----> a;\ a \neq 0;\ \forall n.\ X\ n \neq 0]\!]$
$\Longrightarrow (\lambda n.\ inverse\ (X\ n)) ----> inverse\ a$
⟨*proof*⟩

**lemma** *LIMSEQ-inverse*:
**fixes** $a$ :: $'a$::*real-normed-div-algebra*
**assumes** $X$: $X ----> a$
**assumes** $a$: $a \neq 0$
**shows** $(\lambda n.\ inverse\ (X\ n)) ----> inverse\ a$
⟨*proof*⟩

**lemma** *LIMSEQ-divide*:
**fixes** $a\ b$ :: $'a$::*real-normed-field*
**shows** $[\![X ----> a;\ Y ----> b;\ b \neq 0]\!] \Longrightarrow (\lambda n.\ X\ n\ /\ Y\ n) ----> a$
$/\ b$
⟨*proof*⟩

**lemma** *LIMSEQ-pow*:
**fixes** $a$ :: $'a$::{*real-normed-algebra*,*recpower*}
**shows** $X$ ——> $a \Longrightarrow (\lambda n.\ (X\ n)\ \hat{}\ m)$ ——> $a\ \hat{}\ m$
⟨*proof*⟩

**lemma** *LIMSEQ-setsum*:
**assumes** $n$: $\bigwedge n.\ n \in S \Longrightarrow X\ n$ ——> $L\ n$
**shows** $(\lambda m.\ \sum n{\in}S.\ X\ n\ m)$ ——> $(\sum n{\in}S.\ L\ n)$
⟨*proof*⟩

**lemma** *LIMSEQ-setprod*:
**fixes** $L$ :: $'a \Rightarrow 'b$::{*real-normed-algebra*,*comm-ring-1*}
**assumes** $n$: $\bigwedge n.\ n \in S \Longrightarrow X\ n$ ——> $L\ n$
**shows** $(\lambda m.\ \prod n{\in}S.\ X\ n\ m)$ ——> $(\prod n{\in}S.\ L\ n)$
⟨*proof*⟩

**lemma** *LIMSEQ-add-const*: $f$ ——> $a$ ==> (%$n$.($f\ n + b$)) ——> $a + b$
⟨*proof*⟩

**lemma** *LIMSEQ-add-minus*:
[| $X$ ——> $a$; $Y$ ——> $b$ |] ==> (%$n$. $X\ n + -Y\ n$) ——> $a + -b$
⟨*proof*⟩

**lemma** *LIMSEQ-diff-const*: $f$ ——> $a$ ==> (%$n$.($f\ n - b$)) ——> $a - b$
⟨*proof*⟩

**lemma** *LIMSEQ-diff-approach-zero*:
$g$ ——> $L$ ==> (%$x$. $f\ x - g\ x$) ——> $0$ ==>
$f$ ——> $L$
⟨*proof*⟩

**lemma** *LIMSEQ-diff-approach-zero2*:
$f$ ——> $L$ ==> (%$x$. $f\ x - g\ x$) ——> $0$ ==>
$g$ ——> $L$
⟨*proof*⟩

A sequence tends to zero iff its abs does

**lemma** *LIMSEQ-norm-zero*: (($\lambda n.$ *norm* ($X\ n$)) ——> $0$) = ($X$ ——> $0$)
⟨*proof*⟩

**lemma** *LIMSEQ-rabs-zero*: ((%$n$. $|f\ n|$) ——> $0$) = ($f$ ——> ($0$::*real*))
⟨*proof*⟩

**lemma** *LIMSEQ-imp-rabs*: $f$ ——> ($l$::*real*) ==> (%$n$. $|f\ n|$) ——> $|l|$
⟨*proof*⟩

An unbounded sequence's inverse tends to 0

**lemma** *LIMSEQ-inverse-zero*:
$\forall r$::*real*. $\exists N$. $\forall n \geq N$. $r < X\ n \Longrightarrow (\lambda n.\ inverse\ (X\ n))$ ----> *0*
⟨*proof*⟩

The sequence (*1*::′*a*) / $n$ tends to 0 as $n$ tends to infinity

**lemma** *LIMSEQ-inverse-real-of-nat*: (%$n$. *inverse*(*real*(*Suc n*))) ----> *0*
⟨*proof*⟩

The sequence $r$ + (*1*::′*a*) / $n$ tends to $r$ as $n$ tends to infinity is now easily proved

**lemma** *LIMSEQ-inverse-real-of-nat-add*:
(%$n$. $r$ + *inverse*(*real*(*Suc n*))) ----> $r$
⟨*proof*⟩

**lemma** *LIMSEQ-inverse-real-of-nat-add-minus*:
(%$n$. $r$ + −*inverse*(*real*(*Suc n*))) ----> $r$
⟨*proof*⟩

**lemma** *LIMSEQ-inverse-real-of-nat-add-minus-mult*:
(%$n$. $r$*( *1* + −*inverse*(*real*(*Suc n*)))) ----> $r$
⟨*proof*⟩

**lemma** *LIMSEQ-le-const*:
⟦$X$ ----> ($x$::*real*); $\exists N$. $\forall n \geq N$. $a \leq X\ n$⟧ $\Longrightarrow a \leq x$
⟨*proof*⟩

**lemma** *LIMSEQ-le-const2*:
⟦$X$ ----> ($x$::*real*); $\exists N$. $\forall n \geq N$. $X\ n \leq a$⟧ $\Longrightarrow x \leq a$
⟨*proof*⟩

**lemma** *LIMSEQ-le*:
⟦$X$ ----> $x$; $Y$ ----> $y$; $\exists N$. $\forall n \geq N$. $X\ n \leq Y\ n$⟧ $\Longrightarrow x \leq$ ($y$::*real*)
⟨*proof*⟩

## 12.4 Convergence

**lemma** *limI*: $X$ ----> $L$ ==> *lim* $X$ = $L$
⟨*proof*⟩

**lemma** *convergentD*: *convergent* $X$ ==> $\exists L$. ($X$ ----> $L$)
⟨*proof*⟩

**lemma** *convergentI*: ($X$ ----> $L$) ==> *convergent* $X$
⟨*proof*⟩

**lemma** *convergent-LIMSEQ-iff*: *convergent* $X$ = ($X$ ----> *lim* $X$)
⟨*proof*⟩

**lemma** *convergent-minus-iff*: (*convergent* $X$) = (*convergent* (%$n$. −($X\ n$)))

⟨*proof*⟩

## 12.5 Bounded Monotonic Sequences

Subsequence (alternative definition, (e.g. Hoskins)

**lemma** *subseq-Suc-iff*: $subseq\ f = (\forall n.\ (f\ n) < (f\ (Suc\ n)))$
⟨*proof*⟩

**lemma** *monoseq-Suc*:
$monoseq\ X = ((\forall n.\ X\ n \leq X\ (Suc\ n))$
$\mid (\forall n.\ X\ (Suc\ n) \leq X\ n))$
⟨*proof*⟩

**lemma** *monoI1*: $\forall m.\ \forall\ n \geq m.\ X\ m \leq X\ n ==> monoseq\ X$
⟨*proof*⟩

**lemma** *monoI2*: $\forall m.\ \forall\ n \geq m.\ X\ n \leq X\ m ==> monoseq\ X$
⟨*proof*⟩

**lemma** *mono-SucI1*: $\forall n.\ X\ n \leq X\ (Suc\ n) ==> monoseq\ X$
⟨*proof*⟩

**lemma** *mono-SucI2*: $\forall n.\ X\ (Suc\ n) \leq X\ n ==> monoseq\ X$
⟨*proof*⟩

Bounded Sequence

**lemma** *BseqD*: $Bseq\ X ==> \exists K.\ 0 < K\ \&\ (\forall n.\ norm\ (X\ n) \leq K)$
⟨*proof*⟩

**lemma** *BseqI*: $[\mid 0 < K; \forall n.\ norm\ (X\ n) \leq K\ \mid] ==> Bseq\ X$
⟨*proof*⟩

**lemma** *lemma-NBseq-def*:
$(\exists K > 0.\ \forall n.\ norm\ (X\ n) \leq K) =$
$(\exists N.\ \forall n.\ norm\ (X\ n) \leq real(Suc\ N))$
⟨*proof*⟩

alternative definition for Bseq

**lemma** *Bseq-iff*: $Bseq\ X = (\exists N.\ \forall n.\ norm\ (X\ n) \leq real(Suc\ N))$
⟨*proof*⟩

**lemma** *lemma-NBseq-def2*:
$(\exists K > 0.\ \forall n.\ norm\ (X\ n) \leq K) = (\exists N.\ \forall n.\ norm\ (X\ n) < real(Suc\ N))$
⟨*proof*⟩

**lemma** *Bseq-iff1a*: $Bseq\ X = (\exists N.\ \forall n.\ norm\ (X\ n) < real(Suc\ N))$
⟨*proof*⟩

### 12.5.1 Upper Bounds and Lubs of Bounded Sequences

**lemma** *Bseq-isUb*:
$!!(X::nat{=}{>}real).\ Bseq\ X\ {=}{=}{>}\ \exists\, U.\ isUb\ (UNIV::real\ set)\ \{x.\ \exists\, n.\ X\ n = x\}\ U$
⟨*proof*⟩

Use completeness of reals (supremum property) to show that any bounded sequence has a least upper bound

**lemma** *Bseq-isLub*:
$!!(X::nat{=}{>}real).\ Bseq\ X\ {=}{=}{>}$
$\exists\, U.\ isLub\ (UNIV::real\ set)\ \{x.\ \exists\, n.\ X\ n = x\}\ U$
⟨*proof*⟩

### 12.5.2 A Bounded and Monotonic Sequence Converges

**lemma** *lemma-converg1*:
$!!(X::nat{=}{>}real).\ [|\ \forall\, m.\ \forall\ n \geq m.\ X\ m \leq X\ n;$
$isLub\ (UNIV::real\ set)\ \{x.\ \exists\, n.\ X\ n = x\}\ (X\ ma)$
$|]\ {=}{=}{>}\ \forall\, n \geq ma.\ X\ n = X\ ma$
⟨*proof*⟩

The best of both worlds: Easier to prove this result as a standard theorem and then use equivalence to "transfer" it into the equivalent nonstandard form if needed!

**lemma** *Bmonoseq-LIMSEQ*: $\forall\, n.\ m \leq n\ {-}{-}{>}\ X\ n = X\ m\ {=}{=}{>}\ \exists\, L.\ (X\ {-}{-}{-}{-}{>}\ L)$
⟨*proof*⟩

**lemma** *lemma-converg2*:
$!!(X::nat{=}{>}real).$
$[|\ \forall\, m.\ X\ m\ {}^{\sim}{=}\ U;\ \ isLub\ UNIV\ \{x.\ \exists\, n.\ X\ n = x\}\ U\ |]\ {=}{=}{>}\ \forall\, m.\ X\ m < U$
⟨*proof*⟩

**lemma** *lemma-converg3*: $!!(X\ ::nat{=}{>}real).\ \forall\, m.\ X\ m \leq U\ {=}{=}{>}\ isUb\ UNIV\ \{x.\ \exists\, n.\ X\ n = x\}\ U$
⟨*proof*⟩

FIXME: $U - T < U$ is redundant

**lemma** *lemma-converg4*: $!!(X::nat{=}{>}\ real).$
$[|\ \forall\, m.\ X\ m\ {}^{\sim}{=}\ U;$
$isLub\ UNIV\ \{x.\ \exists\, n.\ X\ n = x\}\ U;$
$0 < T;$
$U + -\ T < U$
$|]\ {=}{=}{>}\ \exists\, m.\ U + -T < X\ m\ \&\ X\ m < U$
⟨*proof*⟩

A standard proof of the theorem for monotone increasing sequence

**lemma** *Bseq-mono-convergent*:
$[|\ Bseq\ X;\ \forall\, m.\ \forall\, n \geq m.\ X\ m \leq X\ n\ |]\ {=}{=}{>}\ convergent\ (X::nat{=}{>}real)$

⟨*proof*⟩

**lemma** *Bseq-minus-iff*: $Bseq\ (\%n.\ -(X\ n)) = Bseq\ X$
⟨*proof*⟩

Main monotonicity theorem

**lemma** *Bseq-monoseq-convergent*: $[|\ Bseq\ X;\ monoseq\ X\ |] ==> convergent\ X$
⟨*proof*⟩

### 12.5.3 A Few More Equivalence Theorems for Boundedness

alternative formulation for boundedness

**lemma** *Bseq-iff2*: $Bseq\ X = (\exists\, k > 0.\ \exists\, x.\ \forall\, n.\ norm\ (X(n) + -x) \leq k)$
⟨*proof*⟩

alternative formulation for boundedness

**lemma** *Bseq-iff3*: $Bseq\ X = (\exists\, k > 0.\ \exists\, N.\ \forall\, n.\ norm(X(n) + -X(N)) \leq k)$
⟨*proof*⟩

**lemma** *BseqI2*: $(\forall\, n.\ k \leq f\ n\ \&\ f\ n \leq (K::real)) ==> Bseq\ f$
⟨*proof*⟩

## 12.6 Cauchy Sequences

**lemma** *CauchyI*:
$(\bigwedge e.\ 0 < e \Longrightarrow \exists\, M.\ \forall\, m \geq M.\ \forall\, n \geq M.\ norm\ (X\ m - X\ n) < e) \Longrightarrow Cauchy\ X$
⟨*proof*⟩

**lemma** *CauchyD*:
$[\![Cauchy\ X;\ 0 < e]\!] \Longrightarrow \exists\, M.\ \forall\, m \geq M.\ \forall\, n \geq M.\ norm\ (X\ m - X\ n) < e$
⟨*proof*⟩

### 12.6.1 Cauchy Sequences are Bounded

A Cauchy sequence is bounded – this is the standard proof mechanization rather than the nonstandard proof

**lemma** *lemmaCauchy*: $\forall\, n \geq M.\ norm\ (X\ M - X\ n) < (1::real)$
$==> \ \forall\, n \geq M.\ norm\ (X\ n :: 'a::real\text{-}normed\text{-}vector) < 1 + norm\ (X\ M)$
⟨*proof*⟩

**lemma** *Cauchy-Bseq*: $Cauchy\ X ==> Bseq\ X$
⟨*proof*⟩

### 12.6.2 Cauchy Sequences are Convergent

**axclass** *banach* ⊆ *real-normed-vector*
*Cauchy-convergent*: $Cauchy\ X \Longrightarrow convergent\ X$

**theorem** *LIMSEQ-imp-Cauchy*:
**assumes** $X$: $X$ ----> $a$ **shows** *Cauchy* $X$
⟨*proof*⟩

**lemma** *convergent-Cauchy*: *convergent* $X \Longrightarrow$ *Cauchy* $X$
⟨*proof*⟩

Proof that Cauchy sequences converge based on the one from http://pirate.shu.edu/ wachsmut/ira/nu

If sequence $X$ is Cauchy, then its limit is the lub of $\{r.\ \exists N.\ \forall n \geq N.\ r < X\ n\}$

**lemma** *isUb-UNIV-I*: $(\bigwedge y.\ y \in S \Longrightarrow y \leq u) \Longrightarrow isUb\ UNIV\ S\ u$
⟨*proof*⟩

**lemma** *real-abs-diff-less-iff*:
$(|x - a| < (r::real)) = (a - r < x \wedge x < a + r)$
⟨*proof*⟩

**locale** (**open**) *real-Cauchy* =
**fixes** $X$ :: *nat* ⇒ *real*
**assumes** $X$: *Cauchy* $X$
**fixes** $S$ :: *real set*
**defines** *S-def*: $S \equiv \{x::real.\ \exists N.\ \forall n \geq N.\ x < X\ n\}$

**lemma** (**in** *real-Cauchy*) *mem-S*: $\forall n \geq N.\ x < X\ n \Longrightarrow x \in S$
⟨*proof*⟩

**lemma** (**in** *real-Cauchy*) *bound-isUb*:
**assumes** $N$: $\forall n \geq N.\ X\ n < x$
**shows** *isUb UNIV S x*
⟨*proof*⟩

**lemma** (**in** *real-Cauchy*) *isLub-ex*: $\exists u.$ *isLub UNIV S u*
⟨*proof*⟩

**lemma** (**in** *real-Cauchy*) *isLub-imp-LIMSEQ*:
**assumes** $x$: *isLub UNIV S x*
**shows** $X$ ----> $x$
⟨*proof*⟩

**lemma** (**in** *real-Cauchy*) *LIMSEQ-ex*: $\exists x.\ X$ ----> $x$
⟨*proof*⟩

**lemma** *real-Cauchy-convergent*:
**fixes** $X$ :: *nat* ⇒ *real*
**shows** *Cauchy* $X \Longrightarrow$ *convergent* $X$
⟨*proof*⟩

**instance** *real* :: *banach*

⟨proof⟩

**lemma** *Cauchy-convergent-iff*:
  **fixes** $X :: nat \Rightarrow 'a::banach$
  **shows** $Cauchy\ X = convergent\ X$
⟨proof⟩

## 12.7 Power Sequences

The sequence $x\ \hat{}\ n$ tends to 0 if $(0::'a) \leq x$ and $x < (1::'a)$. Proof will use (NS) Cauchy equivalence for convergence and also fact that bounded and monotonic sequence converges.

**lemma** *Bseq-realpow*: $[|\ 0 \leq (x::real);\ x \leq 1\ |] ==> Bseq\ (\%n.\ x\ \hat{}\ n)$
⟨proof⟩

**lemma** *monoseq-realpow*: $[|\ 0 \leq x;\ x \leq 1\ |] ==> monoseq\ (\%n.\ x\ \hat{}\ n)$
⟨proof⟩

**lemma** *convergent-realpow*:
  $[|\ 0 \leq (x::real);\ x \leq 1\ |] ==> convergent\ (\%n.\ x\ \hat{}\ n)$
⟨proof⟩

**lemma** *LIMSEQ-inverse-realpow-zero-lemma*:
  **fixes** $x :: real$
  **assumes** $x$: $0 \leq x$
  **shows** $real\ n * x + 1 \leq (x + 1)\ \hat{}\ n$
⟨proof⟩

**lemma** *LIMSEQ-inverse-realpow-zero*:
  $1 < (x::real) \Longrightarrow (\lambda n.\ inverse\ (x\ \hat{}\ n))\ ----> 0$
⟨proof⟩

**lemma** *LIMSEQ-realpow-zero*:
  $[\![0 \leq (x::real);\ x < 1]\!] \Longrightarrow (\lambda n.\ x\ \hat{}\ n)\ ----> 0$
⟨proof⟩

**lemma** *LIMSEQ-power-zero*:
  **fixes** $x :: 'a::\{real\text{-}normed\text{-}algebra\text{-}1,recpower\}$
  **shows** $norm\ x < 1 \Longrightarrow (\lambda n.\ x\ \hat{}\ n)\ ----> 0$
⟨proof⟩

**lemma** *LIMSEQ-divide-realpow-zero*:
  $1 < (x::real) ==> (\%n.\ a\ /\ (x\ \hat{}\ n))\ ----> 0$
⟨proof⟩

Limit of $c\ \hat{}\ n$ for $|c| < (1::'a)$

**lemma** *LIMSEQ-rabs-realpow-zero*: $|c| < (1::real) ==> (\%n.\ |c|\ \hat{}\ n)\ ----> 0$
⟨proof⟩

**lemma** *LIMSEQ-rabs-realpow-zero2*: |*c*| < (*1*::*real*) ==> (%*n*. *c* ^ *n*) −−−−> *0*
⟨*proof*⟩

**end**

# 13 Lim: Limits and Continuity

**theory** *Lim*
**imports** *SEQ*
**begin**

Standard Definitions

**definition**
  *LIM* :: ['*a*::*real-normed-vector* => '*b*::*real-normed-vector*, '*a*, '*b*] => *bool*
    (((-)/ −− (-)/ −−> (-)) [*60*, *0*, *60*] *60*) **where**
  *f* −− *a* −−> *L* =
    (∀ *r* > *0*. ∃ *s* > *0*. ∀ *x*. *x* ≠ *a* & *norm* (*x* − *a*) < *s*
      −−> *norm* (*f x* − *L*) < *r*)

**definition**
  *isCont* :: ['*a*::*real-normed-vector* => '*b*::*real-normed-vector*, '*a*] => *bool* **where**
  *isCont f a* = (*f* −− *a* −−> (*f a*))

**definition**
  *isUCont* :: ['*a*::*real-normed-vector* => '*b*::*real-normed-vector*] => *bool* **where**
  *isUCont f* = (∀ *r*>*0*. ∃ *s*>*0*. ∀ *x y*. *norm* (*x* − *y*) < *s* ⟶ *norm* (*f x* − *f y*) < *r*)

## 13.1 Limits of Functions

### 13.1.1 Purely standard proofs

**lemma** *LIM-eq*:
   *f* −− *a* −−> *L* =
   (∀ *r*>*0*.∃ *s*>*0*.∀ *x*. *x* ≠ *a* & *norm* (*x*−*a*) < *s* −−> *norm* (*f x* − *L*) < *r*)
⟨*proof*⟩

**lemma** *LIM-I*:
   (!!*r*. *0*<*r* ==> ∃ *s*>*0*.∀ *x*. *x* ≠ *a* & *norm* (*x*−*a*) < *s* −−> *norm* (*f x* − *L*) <
*r*)
    ==> *f* −− *a* −−> *L*
⟨*proof*⟩

**lemma** *LIM-D*:
   [| *f* −− *a* −−> *L*; *0*<*r* |]
    ==> ∃ *s*>*0*.∀ *x*. *x* ≠ *a* & *norm* (*x*−*a*) < *s* −−> *norm* (*f x* − *L*) < *r*
⟨*proof*⟩

**lemma** *LIM-offset*: $f$ $--$ $a$ $-->$ $L \Longrightarrow (\lambda x.\ f\ (x + k))$ $--$ $a - k$ $-->$ $L$
⟨*proof*⟩

**lemma** *LIM-offset-zero*: $f$ $--$ $a$ $-->$ $L \Longrightarrow (\lambda h.\ f\ (a + h))$ $--$ $0$ $-->$ $L$
⟨*proof*⟩

**lemma** *LIM-offset-zero-cancel*: $(\lambda h.\ f\ (a + h))$ $--$ $0$ $-->$ $L \Longrightarrow f$ $--$ $a$ $-->$ $L$
⟨*proof*⟩

**lemma** *LIM-const* [*simp*]: $(\%x.\ k)$ $--$ $x$ $-->$ $k$
⟨*proof*⟩

**lemma** *LIM-add*:
**fixes** $f\ g$ :: $'a$::*real-normed-vector* $\Rightarrow$ $'b$::*real-normed-vector*
**assumes** $f$: $f$ $--$ $a$ $-->$ $L$ **and** $g$: $g$ $--$ $a$ $-->$ $M$
**shows** $(\%x.\ f\,x + g(x))$ $--$ $a$ $-->$ $(L + M)$
⟨*proof*⟩

**lemma** *LIM-add-zero*:
$[\![f$ $--$ $a$ $-->$ $0;\ g$ $--$ $a$ $-->$ $0]\!] \Longrightarrow (\lambda x.\ f\,x + g\,x)$ $--$ $a$ $-->$ $0$
⟨*proof*⟩

**lemma** *minus-diff-minus*:
**fixes** $a\ b$ :: $'a$::*ab-group-add*
**shows** $(-\ a) - (-\ b) = -\ (a - b)$
⟨*proof*⟩

**lemma** *LIM-minus*: $f$ $--$ $a$ $-->$ $L$ $==>$ $(\%x.\ -f(x))$ $--$ $a$ $-->$ $-L$
⟨*proof*⟩

**lemma** *LIM-add-minus*:
[| $f$ $--$ $x$ $-->$ $l$; $g$ $--$ $x$ $-->$ $m$ |] $==>$ $(\%x.\ f(x) + -g(x))$ $--$ $x$ $-->$ $(l + -m)$
⟨*proof*⟩

**lemma** *LIM-diff*:
[| $f$ $--$ $x$ $-->$ $l$; $g$ $--$ $x$ $-->$ $m$ |] $==>$ $(\%x.\ f(x) - g(x))$ $--$ $x$ $-->$ $l-m$
⟨*proof*⟩

**lemma** *LIM-zero*: $f$ $--$ $a$ $-->$ $l \Longrightarrow (\lambda x.\ f\,x - l)$ $--$ $a$ $-->$ $0$
⟨*proof*⟩

**lemma** *LIM-zero-cancel*: $(\lambda x.\ f\,x - l)$ $--$ $a$ $-->$ $0 \Longrightarrow f$ $--$ $a$ $-->$ $l$
⟨*proof*⟩

**lemma** *LIM-zero-iff*: $(\lambda x.\ f\,x - l)$ $--$ $a$ $-->$ $0 = f$ $--$ $a$ $-->$ $l$
⟨*proof*⟩

**lemma** *LIM-imp-LIM*:
**assumes** $f$: $f\ -\!-\ a\ -\!-\!>\ l$
**assumes** *le*: $\bigwedge x.\ x \neq a \Longrightarrow norm\ (g\ x\ -\ m) \leq norm\ (f\ x\ -\ l)$
**shows** $g\ -\!-\ a\ -\!-\!>\ m$
⟨*proof*⟩

**lemma** *LIM-norm*: $f\ -\!-\ a\ -\!-\!>\ l \Longrightarrow (\lambda x.\ norm\ (f\ x))\ -\!-\ a\ -\!-\!>\ norm\ l$
⟨*proof*⟩

**lemma** *LIM-norm-zero*: $f\ -\!-\ a\ -\!-\!>\ 0 \Longrightarrow (\lambda x.\ norm\ (f\ x))\ -\!-\ a\ -\!-\!>\ 0$
⟨*proof*⟩

**lemma** *LIM-norm-zero-cancel*: $(\lambda x.\ norm\ (f\ x))\ -\!-\ a\ -\!-\!>\ 0 \Longrightarrow f\ -\!-\ a\ -\!-\!>\ 0$
⟨*proof*⟩

**lemma** *LIM-norm-zero-iff*: $(\lambda x.\ norm\ (f\ x))\ -\!-\ a\ -\!-\!>\ 0 = f\ -\!-\ a\ -\!-\!>\ 0$
⟨*proof*⟩

**lemma** *LIM-rabs*: $f\ -\!-\ a\ -\!-\!>\ (l{::}real) \Longrightarrow (\lambda x.\ |f\ x|)\ -\!-\ a\ -\!-\!>\ |l|$
⟨*proof*⟩

**lemma** *LIM-rabs-zero*: $f\ -\!-\ a\ -\!-\!>\ (0{::}real) \Longrightarrow (\lambda x.\ |f\ x|)\ -\!-\ a\ -\!-\!>\ 0$
⟨*proof*⟩

**lemma** *LIM-rabs-zero-cancel*: $(\lambda x.\ |f\ x|)\ -\!-\ a\ -\!-\!>\ (0{::}real) \Longrightarrow f\ -\!-\ a\ -\!-\!>\ 0$
⟨*proof*⟩

**lemma** *LIM-rabs-zero-iff*: $(\lambda x.\ |f\ x|)\ -\!-\ a\ -\!-\!>\ (0{::}real) = f\ -\!-\ a\ -\!-\!>\ 0$
⟨*proof*⟩

**lemma** *LIM-const-not-eq*:
**fixes** $a$ :: *'a::real-normed-algebra-1*
**shows** $k \neq L \Longrightarrow \neg\ (\lambda x.\ k)\ -\!-\ a\ -\!-\!>\ L$
⟨*proof*⟩

**lemmas** *LIM-not-zero* = *LIM-const-not-eq* [**where** $L = 0$]

**lemma** *LIM-const-eq*:
**fixes** $a$ :: *'a::real-normed-algebra-1*
**shows** $(\lambda x.\ k)\ -\!-\ a\ -\!-\!>\ L \Longrightarrow k = L$
⟨*proof*⟩

**lemma** *LIM-unique*:
**fixes** $a$ :: *'a::real-normed-algebra-1*
**shows** $[\![f\ -\!-\ a\ -\!-\!>\ L;\ f\ -\!-\ a\ -\!-\!>\ M]\!] \Longrightarrow L = M$
⟨*proof*⟩

**lemma** *LIM-ident* [*simp*]: $(\lambda x.\ x)$ −− $a$ −−> $a$
⟨*proof*⟩

Limits are equal for functions equal except at limit point

**lemma** *LIM-equal*:
  [| $\forall x.\ x \neq a$ −−> $(f\ x = g\ x)$ |] ==> $(f$ −− $a$ −−> $l) = (g$ −− $a$ −−> $l)$
⟨*proof*⟩

**lemma** *LIM-cong*:
  $[\![a = b; \bigwedge x.\ x \neq b \Longrightarrow f\ x = g\ x; l = m]\!]$
  $\Longrightarrow ((\lambda x.\ f\ x)$ −− $a$ −−> $l) = ((\lambda x.\ g\ x)$ −− $b$ −−> $m)$
⟨*proof*⟩

**lemma** *LIM-equal2*:
  **assumes** *1*: $0 < R$
  **assumes** *2*: $\bigwedge x.\ [\![x \neq a;\ norm\ (x - a) < R]\!] \Longrightarrow f\ x = g\ x$
  **shows** $g$ −− $a$ −−> $l \Longrightarrow f$ −− $a$ −−> $l$
⟨*proof*⟩

Two uses in Hyperreal/Transcendental.ML

**lemma** *LIM-trans*:
  [| $(\%x.\ f(x) + -g(x))$ −− $a$ −−> $0$;  $g$ −− $a$ −−> $l$ |] ==> $f$ −− $a$ −−> $l$
⟨*proof*⟩

**lemma** *LIM-compose*:
  **assumes** *g*: $g$ −− $l$ −−> $g\ l$
  **assumes** *f*: $f$ −− $a$ −−> $l$
  **shows** $(\lambda x.\ g\ (f\ x))$ −− $a$ −−> $g\ l$
⟨*proof*⟩

**lemma** *LIM-compose2*:
  **assumes** *f*: $f$ −− $a$ −−> $b$
  **assumes** *g*: $g$ −− $b$ −−> $c$
  **assumes** *inj*: $\exists\, d{>}0.\ \forall x.\ x \neq a \wedge norm\ (x - a) < d \longrightarrow f\ x \neq b$
  **shows** $(\lambda x.\ g\ (f\ x))$ −− $a$ −−> $c$
⟨*proof*⟩

**lemma** *LIM-o*: $[\![g$ −− $l$ −−> $g\ l;\ f$ −− $a$ −−> $l]\!] \Longrightarrow (g \circ f)$ −− $a$ −−> $g\ l$
⟨*proof*⟩

**lemma** *real-LIM-sandwich-zero*:
  **fixes** $f\ g$ :: $'a$::*real-normed-vector* $\Rightarrow$ *real*
  **assumes** *f*: $f$ −− $a$ −−> $0$
  **assumes** *1*: $\bigwedge x.\ x \neq a \Longrightarrow 0 \leq g\ x$
  **assumes** *2*: $\bigwedge x.\ x \neq a \Longrightarrow g\ x \leq f\ x$
  **shows** $g$ −− $a$ −−> $0$
⟨*proof*⟩

Bounded Linear Operators

**lemma** (**in** *bounded-linear*) *cont*: $f$ −− $a$ −−> $f$ $a$
⟨*proof*⟩

**lemma** (**in** *bounded-linear*) *LIM*:
$g$ −− $a$ −−> $l$ $\Longrightarrow$ ($\lambda x$. $f$ ($g$ $x$)) −− $a$ −−> $f$ $l$
⟨*proof*⟩

**lemma** (**in** *bounded-linear*) *LIM-zero*:
$g$ −− $a$ −−> *0* $\Longrightarrow$ ($\lambda x$. $f$ ($g$ $x$)) −− $a$ −−> *0*
⟨*proof*⟩

Bounded Bilinear Operators

**lemma** (**in** *bounded-bilinear*) *LIM-prod-zero*:
**assumes** $f$: $f$ −− $a$ −−> *0*
**assumes** $g$: $g$ −− $a$ −−> *0*
**shows** ($\lambda x$. $f$ $x$ ** $g$ $x$) −− $a$ −−> *0*
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *LIM-left-zero*:
$f$ −− $a$ −−> *0* $\Longrightarrow$ ($\lambda x$. $f$ $x$ ** $c$) −− $a$ −−> *0*
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *LIM-right-zero*:
$f$ −− $a$ −−> *0* $\Longrightarrow$ ($\lambda x$. $c$ ** $f$ $x$) −− $a$ −−> *0*
⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *LIM*:
⟦$f$ −− $a$ −−> $L$; $g$ −− $a$ −−> $M$⟧ $\Longrightarrow$ ($\lambda x$. $f$ $x$ ** $g$ $x$) −− $a$ −−> $L$ ** $M$
⟨*proof*⟩

**lemmas** *LIM-mult* = *mult.LIM*

**lemmas** *LIM-mult-zero* = *mult.LIM-prod-zero*

**lemmas** *LIM-mult-left-zero* = *mult.LIM-left-zero*

**lemmas** *LIM-mult-right-zero* = *mult.LIM-right-zero*

**lemmas** *LIM-scaleR* = *scaleR.LIM*

**lemmas** *LIM-of-real* = *of-real.LIM*

**lemma** *LIM-power*:
**fixes** $f$ :: $'a$::*real-normed-vector* $\Rightarrow$ $'b$::{*recpower*,*real-normed-algebra*}
**assumes** $f$: $f$ −− $a$ −−> $l$
**shows** ($\lambda x$. $f$ $x$ ^ $n$) −− $a$ −−> $l$ ^ $n$
⟨*proof*⟩

### 13.1.2 Derived theorems about *LIM*

**lemma** *LIM-inverse-lemma*:
**fixes** $x$ :: $'a$::*real-normed-div-algebra*
**assumes** $r$: $0 < r$
**assumes** $x$: *norm* $(x - 1) <$ *min* $(1/2)$ $(r/2)$
**shows** *norm* (*inverse* $x - 1$) $< r$
⟨*proof*⟩

**lemma** *LIM-inverse-fun*:
**assumes** $a$: $a \neq (0::'a$::*real-normed-div-algebra*)
**shows** *inverse* −− $a$ −−> *inverse* $a$
⟨*proof*⟩

**lemma** *LIM-inverse*:
**fixes** $L$ :: $'a$::*real-normed-div-algebra*
**shows** $[\![f$ −− $a$ −−> $L$; $L \neq 0]\!] \Longrightarrow (\lambda x.$ *inverse* $(f\ x))$ −− $a$ −−> *inverse* $L$
⟨*proof*⟩

## 13.2 Continuity

### 13.2.1 Purely standard proofs

**lemma** *LIM-isCont-iff*: $(f$ −− $a$ −−> $f\ a) = ((\lambda h.\ f\ (a + h))$ −− $0$ −−> $f\ a)$
⟨*proof*⟩

**lemma** *isCont-iff*: *isCont* $f\ x = (\lambda h.\ f\ (x + h))$ −− $0$ −−> $f\ x$
⟨*proof*⟩

**lemma** *isCont-ident* [*simp*]: *isCont* $(\lambda x.\ x)\ a$
⟨*proof*⟩

**lemma** *isCont-const* [*simp*]: *isCont* $(\lambda x.\ k)\ a$
⟨*proof*⟩

**lemma** *isCont-norm*: *isCont* $f\ a \Longrightarrow$ *isCont* $(\lambda x.$ *norm* $(f\ x))\ a$
⟨*proof*⟩

**lemma** *isCont-rabs*: *isCont* $f\ a \Longrightarrow$ *isCont* $(\lambda x.\ |f\ x :: real|)\ a$
⟨*proof*⟩

**lemma** *isCont-add*: $[\![$*isCont* $f\ a$; *isCont* $g\ a]\!] \Longrightarrow$ *isCont* $(\lambda x.\ f\ x + g\ x)\ a$
⟨*proof*⟩

**lemma** *isCont-minus*: *isCont* $f\ a \Longrightarrow$ *isCont* $(\lambda x.\ -\ f\ x)\ a$
⟨*proof*⟩

**lemma** *isCont-diff*: $[\![$*isCont* $f\ a$; *isCont* $g\ a]\!] \Longrightarrow$ *isCont* $(\lambda x.\ f\ x - g\ x)\ a$
⟨*proof*⟩

**lemma** *isCont-mult*:
  **fixes** $f\ g$ :: $'a$::*real-normed-vector* $\Rightarrow$ $'b$::*real-normed-algebra*
  **shows** $[\![$*isCont f a*; *isCont g a*$]\!] \Longrightarrow$ *isCont* $(\lambda x.\ f\ x * g\ x)\ a$
  ⟨*proof*⟩

**lemma** *isCont-inverse*:
  **fixes** $f$ :: $'a$::*real-normed-vector* $\Rightarrow$ $'b$::*real-normed-div-algebra*
  **shows** $[\![$*isCont f a*; $f\ a \neq 0]\!] \Longrightarrow$ *isCont* $(\lambda x.\ inverse\ (f\ x))\ a$
  ⟨*proof*⟩

**lemma** *isCont-LIM-compose*:
  $[\![$*isCont g l*; $f\ --\ a\ -->\ l]\!] \Longrightarrow (\lambda x.\ g\ (f\ x))\ --\ a\ -->\ g\ l$
  ⟨*proof*⟩

**lemma** *isCont-LIM-compose2*:
  **assumes** $f$ [*unfolded isCont-def*]: *isCont f a*
  **assumes** $g$: $g\ --\ f\ a\ -->\ l$
  **assumes** *inj*: $\exists\, d>0.\ \forall x.\ x \neq a \wedge norm\ (x - a) < d \longrightarrow f\ x \neq f\ a$
  **shows** $(\lambda x.\ g\ (f\ x))\ --\ a\ -->\ l$
⟨*proof*⟩

**lemma** *isCont-o2*: $[\![$*isCont f a*; *isCont g* $(f\ a)]\!] \Longrightarrow$ *isCont* $(\lambda x.\ g\ (f\ x))\ a$
  ⟨*proof*⟩

**lemma** *isCont-o*: $[\![$*isCont f a*; *isCont g* $(f\ a)]\!] \Longrightarrow$ *isCont* $(g\ o\ f)\ a$
  ⟨*proof*⟩

**lemma** (**in** *bounded-linear*) *isCont*: *isCont f a*
  ⟨*proof*⟩

**lemma** (**in** *bounded-bilinear*) *isCont*:
  $[\![$*isCont f a*; *isCont g a*$]\!] \Longrightarrow$ *isCont* $(\lambda x.\ f\ x\ **\ g\ x)\ a$
  ⟨*proof*⟩

**lemmas** *isCont-scaleR* = *scaleR.isCont*

**lemma** *isCont-of-real*:
  *isCont f a* $\Longrightarrow$ *isCont* $(\lambda x.\ of\text{-}real\ (f\ x))\ a$
  ⟨*proof*⟩

**lemma** *isCont-power*:
  **fixes** $f$ :: $'a$::*real-normed-vector* $\Rightarrow$ $'b$::{*recpower*,*real-normed-algebra*}
  **shows** *isCont f a* $\Longrightarrow$ *isCont* $(\lambda x.\ f\ x\ \hat{}\ n)\ a$
  ⟨*proof*⟩

**lemma** *isCont-abs* [*simp*]: *isCont abs* $(a$::*real*$)$
⟨*proof*⟩

## 13.3 Uniform Continuity

**lemma** *isUCont-isCont*: *isUCont f* ==> *isCont f x*
⟨*proof*⟩

**lemma** *isUCont-Cauchy*:
$[\![$*isUCont f*; *Cauchy X*$]\!] \Longrightarrow$ *Cauchy* ($\lambda n.\ f\ (X\ n)$)
⟨*proof*⟩

**lemma** (**in** *bounded-linear*) *isUCont*: *isUCont f*
⟨*proof*⟩

**lemma** (**in** *bounded-linear*) *Cauchy*: *Cauchy X* $\Longrightarrow$ *Cauchy* ($\lambda n.\ f\ (X\ n)$)
⟨*proof*⟩

## 13.4 Relation of LIM and LIMSEQ

**lemma** *LIMSEQ-SEQ-conv1*:
**fixes** *a* :: *'a::real-normed-vector*
**assumes** *X*: *X* −− *a* −−> *L*
**shows** $\forall S.\ (\forall n.\ S\ n \neq a) \wedge S$ −−−−> $a \longrightarrow (\lambda n.\ X\ (S\ n))$ −−−−> $L$
⟨*proof*⟩

**lemma** *LIMSEQ-SEQ-conv2*:
**fixes** *a* :: *real*
**assumes** $\forall S.\ (\forall n.\ S\ n \neq a) \wedge S$ −−−−> $a \longrightarrow (\lambda n.\ X\ (S\ n))$ −−−−> $L$
**shows** *X* −− *a* −−> *L*
⟨*proof*⟩

**lemma** *LIMSEQ-SEQ-conv*:
$(\forall S.\ (\forall n.\ S\ n \neq a) \wedge S$ −−−−> $(a::real) \longrightarrow (\lambda n.\ X\ (S\ n))$ −−−−> $L) =$
(*X* −− *a* −−> *L*)
⟨*proof*⟩

**end**

# 14 Deriv: Differentiation

**theory** *Deriv*
**imports** *Lim*
**begin**

Standard Definitions

**definition**
*deriv* :: [*'a::real-normed-field* ⇒ *'a*, *'a*, *'a*] ⇒ *bool*
— Differentiation: D is derivative of function f at x
((*DERIV* (-)/ (-)/ :> (-)) [*1000*, *1000*, *60*] *60*) **where**
*DERIV f x* :> *D* = ((%*h*. (*f*(*x* + *h*) − *f x*) / *h*) −− *0* −−> *D*)

**definition**
*differentiable* :: $['a::real\text{-}normed\text{-}field \Rightarrow 'a,\ 'a] \Rightarrow bool$
(**infixl** *differentiable 60*) **where**
$f$ *differentiable* $x = (\exists D.\ DERIV\ f\ x :> D)$

**consts**
*Bolzano-bisect* :: $[real*real{=}{>}bool,\ real,\ real,\ nat] => (real*real)$
**primrec**
*Bolzano-bisect P a b 0* = $(a,b)$
*Bolzano-bisect P a b (Suc n)* =
(*let* $(x,y)$ = *Bolzano-bisect P a b n*
*in if* $P(x,\ (x+y)/2)$ *then* $((x+y)/2,\ y)$
*else* $(x,\ (x+y)/2))$

## 14.1 Derivatives

**lemma** *DERIV-iff*: $(DERIV\ f\ x :> D) = ((\%h.\ (f(x + h) - f(x))/h) -\!- 0 -\!-\!> D)$
⟨*proof*⟩

**lemma** *DERIV-D*: $DERIV\ f\ x :> D ==> (\%h.\ (f(x + h) - f(x))/h) -\!- 0 -\!-\!> D$
⟨*proof*⟩

**lemma** *DERIV-const* [*simp*]: $DERIV\ (\lambda x.\ k)\ x :> 0$
⟨*proof*⟩

**lemma** *DERIV-ident* [*simp*]: $DERIV\ (\lambda x.\ x)\ x :> 1$
⟨*proof*⟩

**lemma** *add-diff-add*:
**fixes** $a\ b\ c\ d$ :: $'a::ab\text{-}group\text{-}add$
**shows** $(a + c) - (b + d) = (a - b) + (c - d)$
⟨*proof*⟩

**lemma** *DERIV-add*:
$[\![DERIV\ f\ x :> D;\ DERIV\ g\ x :> E]\!] \Longrightarrow DERIV\ (\lambda x.\ f\ x + g\ x)\ x :> D + E$
⟨*proof*⟩

**lemma** *DERIV-minus*:
$DERIV\ f\ x :> D \Longrightarrow DERIV\ (\lambda x.\ -\ f\ x)\ x :> -\ D$
⟨*proof*⟩

**lemma** *DERIV-diff*:
$[\![DERIV\ f\ x :> D;\ DERIV\ g\ x :> E]\!] \Longrightarrow DERIV\ (\lambda x.\ f\ x - g\ x)\ x :> D - E$
⟨*proof*⟩

**lemma** *DERIV-add-minus*:
$[\![DERIV\ f\ x :> D;\ DERIV\ g\ x :> E]\!] \Longrightarrow DERIV\ (\lambda x.\ f\ x + -\ g\ x)\ x :> D + -\ E$
$\langle proof \rangle$

**lemma** *DERIV-isCont*: $DERIV\ f\ x :> D \Longrightarrow isCont\ f\ x$
$\langle proof \rangle$

**lemma** *DERIV-mult-lemma*:
**fixes** $a\ b\ c\ d :: 'a::real\text{-}field$
**shows** $(a * b - c * d) / h = a * ((b - d) / h) + ((a - c) / h) * d$
$\langle proof \rangle$

**lemma** *DERIV-mult'*:
**assumes** $f$: $DERIV\ f\ x :> D$
**assumes** $g$: $DERIV\ g\ x :> E$
**shows** $DERIV\ (\lambda x.\ f\ x * g\ x)\ x :> f\ x * E + D * g\ x$
$\langle proof \rangle$

**lemma** *DERIV-mult*:
$[|\ DERIV\ f\ x :> Da;\ DERIV\ g\ x :> Db\ |]$
$==> DERIV\ (\%x.\ f\ x * g\ x)\ x :> (Da * g(x)) + (Db * f(x))$
$\langle proof \rangle$

**lemma** *DERIV-unique*:
$[|\ DERIV\ f\ x :> D;\ DERIV\ f\ x :> E\ |] ==> D = E$
$\langle proof \rangle$

Differentiation of finite sum

**lemma** *DERIV-sumr* [*rule-format* (*no-asm*)]:
$(\forall r.\ m \leq r\ \&\ r < (m + n) --> DERIV\ (\%x.\ f\ r\ x)\ x :> (f'\ r\ x))$
$--> DERIV\ (\%x.\ \sum n{=}m..{<}n::nat.\ f\ n\ x :: real)\ x :> (\sum r{=}m..{<}n.\ f'\ r\ x)$
$\langle proof \rangle$

Alternative definition for differentiability

**lemma** *DERIV-LIM-iff*:
$((\%h.\ (f(a + h) - f(a)) / h) --\ 0 --> D) =$
$((\%x.\ (f(x){-}f(a)) / (x{-}a)) --\ a --> D)$
$\langle proof \rangle$

**lemma** *DERIV-iff2*: $(DERIV\ f\ x :> D) = ((\%z.\ (f(z) - f(x)) / (z{-}x)) --\ x --> D)$
$\langle proof \rangle$

**lemma** *inverse-diff-inverse*:
$[\![(a::'a::division\text{-}ring) \neq 0;\ b \neq 0]\!]$
$\Longrightarrow inverse\ a - inverse\ b = -\ (inverse\ a * (a - b) * inverse\ b)$
$\langle proof \rangle$

**lemma** *DERIV-inverse-lemma*:
$[\![a \neq 0;\ b \neq (0::'a::real\text{-}normed\text{-}field)]\!]$
$\Longrightarrow (inverse\ a - inverse\ b)\ /\ h$
$= - (inverse\ a * ((a - b)\ /\ h) * inverse\ b)$
⟨*proof*⟩

**lemma** *DERIV-inverse′*:
**assumes** *der*: $DERIV\ f\ x :> D$
**assumes** *neq*: $f\ x \neq 0$
**shows** $DERIV\ (\lambda x.\ inverse\ (f\ x))\ x :> - (inverse\ (f\ x) * D * inverse\ (f\ x))$
(**is** $DERIV$ - - $:>\ ?E$)
⟨*proof*⟩

**lemma** *DERIV-divide*:
$[\![DERIV\ f\ x :> D;\ DERIV\ g\ x :> E;\ g\ x \neq 0]\!]$
$\Longrightarrow DERIV\ (\lambda x.\ f\ x\ /\ g\ x)\ x :> (D * g\ x - f\ x * E)\ /\ (g\ x * g\ x)$
⟨*proof*⟩

**lemma** *DERIV-power-Suc*:
**fixes** $f :: 'a \Rightarrow 'a::\{real\text{-}normed\text{-}field,recpower\}$
**assumes** $f$: $DERIV\ f\ x :> D$
**shows** $DERIV\ (\lambda x.\ f\ x\ \hat{}\ Suc\ n)\ x :> (1 + of\text{-}nat\ n) * (D * f\ x\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *DERIV-power*:
**fixes** $f :: 'a \Rightarrow 'a::\{real\text{-}normed\text{-}field,recpower\}$
**assumes** $f$: $DERIV\ f\ x :> D$
**shows** $DERIV\ (\lambda x.\ f\ x\ \hat{}\ n)\ x :> of\text{-}nat\ n * (D * f\ x\ \hat{}\ (n - Suc\ 0))$
⟨*proof*⟩

**lemma** *CARAT-DERIV*:
$(DERIV\ f\ x :> l) =$
$(\exists\, g.\ (\forall\, z.\ f\ z - f\ x = g\ z * (z{-}x))\ \&\ isCont\ g\ x\ \&\ g\ x = l)$
(**is** $?lhs = ?rhs$)
⟨*proof*⟩

**lemma** *DERIV-chain′*:
**assumes** $f$: $DERIV\ f\ x :> D$
**assumes** $g$: $DERIV\ g\ (f\ x) :> E$
**shows** $DERIV\ (\lambda x.\ g\ (f\ x))\ x :> E * D$
⟨*proof*⟩

**lemma** *DERIV-cmult*:
$DERIV\ f\ x\ :>\ D\ ==>\ DERIV\ (\%x.\ c * f\ x)\ x\ :>\ c*D$
⟨*proof*⟩

**lemma** *DERIV-chain*: $[|\ DERIV\ f\ (g\ x)\ :>\ Da;\ DERIV\ g\ x\ :>\ Db\ |]\ ==>\ DERIV\ (f\ o\ g)\ x\ :>\ Da * Db$
⟨*proof*⟩

**lemma** *DERIV-chain2*: $[|\ DERIV\ f\ (g\ x)\ :>\ Da;\ DERIV\ g\ x\ :>\ Db\ |]\ ==>\ DERIV\ (\%x.\ f\ (g\ x))\ x\ :>\ Da * Db$
⟨*proof*⟩

**lemma** *DERIV-cmult-Id* [*simp*]: $DERIV\ (op * c)\ x\ :>\ c$
⟨*proof*⟩

**lemma** *DERIV-pow*: $DERIV\ (\%x.\ x\ \hat{}\ n)\ x\ :>\ real\ n * (x\ \hat{}\ (n - Suc\ 0))$
⟨*proof*⟩

Power of -1

**lemma** *DERIV-inverse*:
**fixes** $x :: 'a::\{real\text{-}normed\text{-}field,recpower\}$
**shows** $x \neq 0\ ==>\ DERIV\ (\%x.\ inverse(x))\ x\ :>\ (-(inverse\ x\ \hat{}\ Suc\ (Suc\ 0)))$
⟨*proof*⟩

Derivative of inverse

**lemma** *DERIV-inverse-fun*:
**fixes** $x :: 'a::\{real\text{-}normed\text{-}field,recpower\}$
**shows** $[|\ DERIV\ f\ x\ :>\ d;\ f(x) \neq 0\ |]$
$==>\ DERIV\ (\%x.\ inverse(f\ x))\ x\ :>\ (-\ (d * inverse(f(x)\ \hat{}\ Suc\ (Suc\ 0))))$
⟨*proof*⟩

Derivative of quotient

**lemma** *DERIV-quotient*:
**fixes** $x :: 'a::\{real\text{-}normed\text{-}field,recpower\}$
**shows** $[|\ DERIV\ f\ x\ :>\ d;\ DERIV\ g\ x\ :>\ e;\ g(x) \neq 0\ |]$
$==>\ DERIV\ (\%y.\ f(y)\ /\ (g\ y))\ x\ :>\ (d*g(x) - (e*f(x)))\ /\ (g(x)\ \hat{}\ Suc\ (Suc\ 0))$
⟨*proof*⟩

## 14.2 Differentiability predicate

**lemma** *differentiableD*: $f\ differentiable\ x\ ==>\ \exists\, D.\ DERIV\ f\ x\ :>\ D$
⟨*proof*⟩

**lemma** *differentiableI*: $DERIV\ f\ x\ :>\ D\ ==>\ f\ differentiable\ x$

⟨*proof*⟩

**lemma** *differentiable-const*: $(\lambda z.\ a)$ *differentiable* $x$
  ⟨*proof*⟩

**lemma** *differentiable-sum*:
  **assumes** $f$ *differentiable* $x$
  **and** $g$ *differentiable* $x$
  **shows** $(\lambda x.\ f\ x + g\ x)$ *differentiable* $x$
⟨*proof*⟩

**lemma** *differentiable-diff*:
  **assumes** $f$ *differentiable* $x$
  **and** $g$ *differentiable* $x$
  **shows** $(\lambda x.\ f\ x - g\ x)$ *differentiable* $x$
⟨*proof*⟩

**lemma** *differentiable-mult*:
  **assumes** $f$ *differentiable* $x$
  **and** $g$ *differentiable* $x$
  **shows** $(\lambda x.\ f\ x * g\ x)$ *differentiable* $x$
⟨*proof*⟩

## 14.3 Nested Intervals and Bisection

Lemmas about nested intervals and proof by bisection (cf.Harrison). All considerably tidied by lcp.

**lemma** *lemma-f-mono-add* [*rule-format* (*no-asm*)]: $(\forall n.\ (f{::}nat{=}{>}real)\ n \le f\ (Suc\ n))\ {-}{-}{>}\ f\ m \le f(m + no)$
⟨*proof*⟩

**lemma** *f-inc-g-dec-Beq-f*: $[|\ \forall n.\ f(n) \le f(Suc\ n);$
  $\forall n.\ g(Suc\ n) \le g(n);$
  $\forall n.\ f(n) \le g(n)\ |]$
  $==>\ Bseq\ (f :: nat \Rightarrow real)$
⟨*proof*⟩

**lemma** *f-inc-g-dec-Beq-g*: $[|\ \forall n.\ f(n) \le f(Suc\ n);$
  $\forall n.\ g(Suc\ n) \le g(n);$
  $\forall n.\ f(n) \le g(n)\ |]$
  $==>\ Bseq\ (g :: nat \Rightarrow real)$
⟨*proof*⟩

**lemma** *f-inc-imp-le-lim*:
  **fixes** $f :: nat \Rightarrow real$
  **shows** $[\![\forall n.\ f\ n \le f\ (Suc\ n);\ convergent\ f]\!] \Longrightarrow f\ n \le lim\ f$
⟨*proof*⟩

**lemma** *lim-uminus*: $convergent\ g\ ==>\ lim\ (\%x.\ -\ g\ x) = -\ (lim\ g)$

⟨*proof*⟩

**lemma** *g-dec-imp-lim-le*:
  **fixes** $g :: nat \Rightarrow real$
  **shows** $[\![\forall n.\ g\ (Suc\ n) \leq g(n);\ convergent\ g]\!] \Longrightarrow lim\ g \leq g\ n$
⟨*proof*⟩

**lemma** *lemma-nest*: $[|\ \forall n.\ f(n) \leq f(Suc\ n);$
$\quad \forall n.\ g(Suc\ n) \leq g(n);$
$\quad \forall n.\ f(n) \leq g(n)\ |]$
$\quad ==> \exists l\ m :: real.\ l \leq m\ \&\ ((\forall n.\ f(n) \leq l)\ \&\ f\ ----> l)\ \&$
$\qquad ((\forall n.\ m \leq g(n))\ \&\ g\ ----> m)$
⟨*proof*⟩

**lemma** *lemma-nest-unique*: $[|\ \forall n.\ f(n) \leq f(Suc\ n);$
$\quad \forall n.\ g(Suc\ n) \leq g(n);$
$\quad \forall n.\ f(n) \leq g(n);$
$\quad (\%n.\ f(n) - g(n))\ ----> 0\ |]$
$\quad ==> \exists l{::}real.\ ((\forall n.\ f(n) \leq l)\ \&\ f\ ----> l)\ \&$
$\qquad ((\forall n.\ l \leq g(n))\ \&\ g\ ----> l)$
⟨*proof*⟩

The universal quantifiers below are required for the declaration of *Bolzano-nest-unique* below.

**lemma** *Bolzano-bisect-le*:
$a \leq b ==> \forall n.\ fst\ (Bolzano\text{-}bisect\ P\ a\ b\ n) \leq snd\ (Bolzano\text{-}bisect\ P\ a\ b\ n)$
⟨*proof*⟩

**lemma** *Bolzano-bisect-fst-le-Suc*: $a \leq b ==>$
$\forall n.\ fst(Bolzano\text{-}bisect\ P\ a\ b\ n) \leq fst\ (Bolzano\text{-}bisect\ P\ a\ b\ (Suc\ n))$
⟨*proof*⟩

**lemma** *Bolzano-bisect-Suc-le-snd*: $a \leq b ==>$
$\forall n.\ snd(Bolzano\text{-}bisect\ P\ a\ b\ (Suc\ n)) \leq snd\ (Bolzano\text{-}bisect\ P\ a\ b\ n)$
⟨*proof*⟩

**lemma** *eq-divide-2-times-iff*: $((x{::}real) = y\ /\ (2 * z)) = (2 * x = y/z)$
⟨*proof*⟩

**lemma** *Bolzano-bisect-diff*:
$\quad a \leq b ==>$
$\quad snd(Bolzano\text{-}bisect\ P\ a\ b\ n) - fst(Bolzano\text{-}bisect\ P\ a\ b\ n) =$
$\quad (b-a)\ /\ (2\ \hat{}\ n)$
⟨*proof*⟩

**lemmas** *Bolzano-nest-unique* =
  *lemma-nest-unique*
  [*OF Bolzano-bisect-fst-le-Suc Bolzano-bisect-Suc-le-snd Bolzano-bisect-le*]

**lemma** *not-P-Bolzano-bisect*:
  **assumes** $P$: $!!a\ b\ c.\ [|\ P(a,b);\ P(b,c);\ a \le b;\ b \le c|] ==> P(a,c)$
    **and** *notP*: $\sim P(a,b)$
    **and** *le*: $a \le b$
  **shows** $\sim P(fst(Bolzano\text{-}bisect\ P\ a\ b\ n),\ snd(Bolzano\text{-}bisect\ P\ a\ b\ n))$
⟨*proof*⟩

**lemma** *not-P-Bolzano-bisect'*:
  $[|\ \forall a\ b\ c.\ P(a,b)\ \&\ P(b,c)\ \&\ a \le b\ \&\ b \le c\ --> P(a,c);$
    $\sim P(a,b);\ \ a \le b\ |] ==>$
  $\forall n.\ \sim P(fst(Bolzano\text{-}bisect\ P\ a\ b\ n),\ snd(Bolzano\text{-}bisect\ P\ a\ b\ n))$
⟨*proof*⟩

**lemma** *lemma-BOLZANO*:
  $[|\ \forall a\ b\ c.\ P(a,b)\ \&\ P(b,c)\ \&\ a \le b\ \&\ b \le c\ --> P(a,c);$
    $\forall x.\ \exists d{::}real.\ 0 < d\ \&$
      $(\forall a\ b.\ a \le x\ \&\ x \le b\ \&\ (b-a) < d\ --> P(a,b));$
    $a \le b\ |]$
  $==> P(a,b)$
⟨*proof*⟩

**lemma** *lemma-BOLZANO2*: $((\forall a\ b\ c.\ (a \le b\ \&\ b \le c\ \&\ P(a,b)\ \&\ P(b,c))\ -->$
$P(a,c))\ \&$
  $(\forall x.\ \exists d{::}real.\ 0 < d\ \&$
      $(\forall a\ b.\ a \le x\ \&\ x \le b\ \&\ (b-a) < d\ --> P(a,b))))$
  $--> (\forall a\ b.\ a \le b\ --> P(a,b))$
⟨*proof*⟩

## 14.4 Intermediate Value Theorem

Prove Contrapositive by Bisection

**lemma** *IVT*: $[|\ f(a{::}real) \le (y{::}real);\ y \le f(b);$
    $a \le b;$
    $(\forall x.\ a \le x\ \&\ x \le b\ --> isCont\ f\ x)\ |]$
  $==> \exists x.\ a \le x\ \&\ x \le b\ \&\ f(x) = y$
⟨*proof*⟩

**lemma** *IVT2*: $[|\ f(b{::}real) \le (y{::}real);\ y \le f(a);$
    $a \le b;$
    $(\forall x.\ a \le x\ \&\ x \le b\ --> isCont\ f\ x)$
  $|] ==> \exists x.\ a \le x\ \&\ x \le b\ \&\ f(x) = y$
⟨*proof*⟩

**lemma** *IVT-objl*: $(f(a::real) \le (y::real)$ & $y \le f(b)$ & $a \le b$ &
$(\forall x.\ a \le x$ & $x \le b$ $-->$ $isCont\ f\ x))$
$-->$ $(\exists x.\ a \le x$ & $x \le b$ & $f(x) = y)$
⟨*proof*⟩

**lemma** *IVT2-objl*: $(f(b::real) \le (y::real)$ & $y \le f(a)$ & $a \le b$ &
$(\forall x.\ a \le x$ & $x \le b$ $-->$ $isCont\ f\ x))$
$-->$ $(\exists x.\ a \le x$ & $x \le b$ & $f(x) = y)$
⟨*proof*⟩

By bisection, function continuous on closed interval is bounded above

**lemma** *isCont-bounded*:
$[|\ a \le b; \forall x.\ a \le x$ & $x \le b$ $-->$ $isCont\ f\ x\ |]$
$==>$ $\exists M::real.\ \forall x::real.\ a \le x$ & $x \le b$ $-->$ $f(x) \le M$
⟨*proof*⟩

Refine the above to existence of least upper bound

**lemma** *lemma-reals-complete*: $((\exists x.\ x \in S)$ & $(\exists y.\ isUb\ UNIV\ S\ (y::real)))$ $-->$
$(\exists t.\ isLub\ UNIV\ S\ t)$
⟨*proof*⟩

**lemma** *isCont-has-Ub*: $[|\ a \le b; \forall x.\ a \le x$ & $x \le b$ $-->$ $isCont\ f\ x\ |]$
$==>$ $\exists M::real.\ (\forall x::real.\ a \le x$ & $x \le b$ $-->$ $f(x) \le M)$ &
$(\forall N.\ N < M$ $-->$ $(\exists x.\ a \le x$ & $x \le b$ & $N < f(x)))$
⟨*proof*⟩

Now show that it attains its upper bound

**lemma** *isCont-eq-Ub*:
**assumes** *le*: $a \le b$
**and** *con*: $\forall x::real.\ a \le x$ & $x \le b$ $-->$ $isCont\ f\ x$
**shows** $\exists M::real.\ (\forall x.\ a \le x$ & $x \le b$ $-->$ $f(x) \le M)$ &
$(\exists x.\ a \le x$ & $x \le b$ & $f(x) = M)$
⟨*proof*⟩

Same theorem for lower bound

**lemma** *isCont-eq-Lb*: $[|\ a \le b; \forall x.\ a \le x$ & $x \le b$ $-->$ $isCont\ f\ x\ |]$
$==>$ $\exists M::real.\ (\forall x::real.\ a \le x$ & $x \le b$ $-->$ $M \le f(x))$ &
$(\exists x.\ a \le x$ & $x \le b$ & $f(x) = M)$
⟨*proof*⟩

Another version.

**lemma** *isCont-Lb-Ub*: $[|a \le b; \forall x.\ a \le x$ & $x \le b$ $-->$ $isCont\ f\ x\ |]$
$==>$ $\exists L\ M::real.\ (\forall x::real.\ a \le x$ & $x \le b$ $-->$ $L \le f(x)$ & $f(x) \le M)$ &
$(\forall y.\ L \le y$ & $y \le M$ $-->$ $(\exists x.\ a \le x$ & $x \le b$ & $(f(x) = y)))$
⟨*proof*⟩

If $(0::'a) < f'\ x$ then $x$ is Locally Strictly Increasing At The Right

**lemma** *DERIV-left-inc*:

**fixes** $f$ :: *real* => *real*
**assumes** *der*: *DERIV f x* :> $l$
**and** $l$: $0 < l$
**shows** $\exists\, d > 0.\ \forall\, h > 0.\ h < d \dashrightarrow f(x) < f(x + h)$
⟨*proof*⟩

**lemma** *DERIV-left-dec*:
**fixes** $f$ :: *real* => *real*
**assumes** *der*: *DERIV f x* :> $l$
**and** $l$: $l < 0$
**shows** $\exists\, d > 0.\ \forall\, h > 0.\ h < d \dashrightarrow f(x) < f(x-h)$
⟨*proof*⟩

**lemma** *DERIV-local-max*:
**fixes** $f$ :: *real* => *real*
**assumes** *der*: *DERIV f x* :> $l$
**and** $d$: $0 < d$
**and** *le*: $\forall\, y.\ |x-y| < d \dashrightarrow f(y) \le f(x)$
**shows** $l = 0$
⟨*proof*⟩

Similar theorem for a local minimum

**lemma** *DERIV-local-min*:
**fixes** $f$ :: *real* => *real*
**shows** [| *DERIV f x* :> $l$; $0 < d$; $\forall\, y.\ |x-y| < d \dashrightarrow f(x) \le f(y)$ |] ==> $l = 0$
⟨*proof*⟩

In particular, if a function is locally flat

**lemma** *DERIV-local-const*:
**fixes** $f$ :: *real* => *real*
**shows** [| *DERIV f x* :> $l$; $0 < d$; $\forall\, y.\ |x-y| < d \dashrightarrow f(x) = f(y)$ |] ==> $l = 0$
⟨*proof*⟩

Lemma about introducing open ball in open interval

**lemma** *lemma-interval-lt*:
[| $a < x$; $x < b$ |]
==> $\exists\, d$::*real*. $0 < d$ & $(\forall\, y.\ |x-y| < d \dashrightarrow a < y$ & $y < b)$
⟨*proof*⟩

**lemma** *lemma-interval*: [| $a < x$; $x < b$ |] ==>
$\exists\, d$::*real*. $0 < d$ & $(\forall\, y.\ |x-y| < d \dashrightarrow a \le y$ & $y \le b)$
⟨*proof*⟩

Rolle's Theorem. If $f$ is defined and continuous on the closed interval $[a,b]$ and differentiable on the open interval $(a,b)$, and $f\ a = f\ b$, then there exists $x0 \in (a,b)$ such that $f'\ x0 = (0::'a)$

**theorem** *Rolle*:
**assumes** *lt*: $a < b$
**and** *eq*: $f(a) = f(b)$
**and** *con*: $\forall x.\ a \le x\ \&\ x \le b\ {-}{-}{>}\ isCont\ f\ x$
**and** *dif* [*rule-format*]: $\forall x.\ a < x\ \&\ x < b\ {-}{-}{>}\ f\ differentiable\ x$
**shows** $\exists z{::}real.\ a < z\ \&\ z < b\ \&\ DERIV\ f\ z :> 0$
⟨*proof*⟩

## 14.5 Mean Value Theorem

**lemma** *lemma-MVT*:
$f\ a - (f\ b - f\ a)/(b{-}a) * a = f\ b - (f\ b - f\ a)/(b{-}a) * (b{::}real)$
⟨*proof*⟩

**theorem** *MVT*:
**assumes** *lt*: $a < b$
**and** *con*: $\forall x.\ a \le x\ \&\ x \le b\ {-}{-}{>}\ isCont\ f\ x$
**and** *dif* [*rule-format*]: $\forall x.\ a < x\ \&\ x < b\ {-}{-}{>}\ f\ differentiable\ x$
**shows** $\exists l\ z{::}real.\ a < z\ \&\ z < b\ \&\ DERIV\ f\ z :> l\ \&$
$(f(b) - f(a) = (b{-}a) * l)$
⟨*proof*⟩

A function is constant if its derivative is 0 over an interval.

**lemma** *DERIV-isconst-end*:
**fixes** $f :: real => real$
**shows** $[|\ a < b;$
$\forall x.\ a \le x\ \&\ x \le b\ {-}{-}{>}\ isCont\ f\ x;$
$\forall x.\ a < x\ \&\ x < b\ {-}{-}{>}\ DERIV\ f\ x :> 0\ |]$
$==> f\ b = f\ a$
⟨*proof*⟩

**lemma** *DERIV-isconst1*:
**fixes** $f :: real => real$
**shows** $[|\ a < b;$
$\forall x.\ a \le x\ \&\ x \le b\ {-}{-}{>}\ isCont\ f\ x;$
$\forall x.\ a < x\ \&\ x < b\ {-}{-}{>}\ DERIV\ f\ x :> 0\ |]$
$==> \forall x.\ a \le x\ \&\ x \le b\ {-}{-}{>}\ f\ x = f\ a$
⟨*proof*⟩

**lemma** *DERIV-isconst2*:
**fixes** $f :: real => real$
**shows** $[|\ a < b;$
$\forall x.\ a \le x\ \&\ x \le b\ {-}{-}{>}\ isCont\ f\ x;$
$\forall x.\ a < x\ \&\ x < b\ {-}{-}{>}\ DERIV\ f\ x :> 0;$
$a \le x;\ x \le b\ |]$
$==> f\ x = f\ a$
⟨*proof*⟩

**lemma** *DERIV-isconst-all*:

**fixes** $f$ :: *real* => *real*
**shows** $\forall x.$ *DERIV* $f\ x$ :> $0$ ==> $f(x) = f(y)$
⟨*proof*⟩

**lemma** *DERIV-const-ratio-const*:
**fixes** $f$ :: *real* => *real*
**shows** $[|a \neq b; \forall x.$ *DERIV* $f\ x$ :> $k$ $|]$ ==> $(f(b) - f(a)) = (b-a) * k$
⟨*proof*⟩

**lemma** *DERIV-const-ratio-const2*:
**fixes** $f$ :: *real* => *real*
**shows** $[|a \neq b; \forall x.$ *DERIV* $f\ x$ :> $k$ $|]$ ==> $(f(b) - f(a))/(b-a) = k$
⟨*proof*⟩

**lemma** *real-average-minus-first* [*simp*]: $((a + b)\ /2 - a) = (b-a)/(2$::*real*$)$
⟨*proof*⟩

**lemma** *real-average-minus-second* [*simp*]: $((b + a)/2 - a) = (b-a)/(2$::*real*$)$
⟨*proof*⟩

Gallileo's "trick": average velocity = av. of end velocities

**lemma** *DERIV-const-average*:
**fixes** $v$ :: *real* => *real*
**assumes** *neq*: $a \neq (b$::*real*$)$
**and** *der*: $\forall x.$ *DERIV* $v\ x$ :> $k$
**shows** $v\ ((a + b)/2) = (v\ a + v\ b)/2$
⟨*proof*⟩

Dull lemma: an continuous injection on an interval must have a strict maximum at an end point, not in the middle.

**lemma** *lemma-isCont-inj*:
**fixes** $f$ :: *real* ⇒ *real*
**assumes** *d*: $0 < d$
**and** *inj* [*rule-format*]: $\forall z.\ |z-x| \leq d$ --> $g(f\ z) = z$
**and** *cont*: $\forall z.\ |z-x| \leq d$ --> *isCont* $f\ z$
**shows** $\exists z.\ |z-x| \leq d\ \&\ f\ x < f\ z$
⟨*proof*⟩

Similar version for lower bound.

**lemma** *lemma-isCont-inj2*:
**fixes** $f\ g$ :: *real* ⇒ *real*
**shows** $[|0 < d; \forall z.\ |z-x| \leq d$ --> $g(f\ z) = z;$
$\forall z.\ |z-x| \leq d$ --> *isCont* $f\ z$ $|]$
==> $\exists z.\ |z-x| \leq d\ \&\ f\ z < f\ x$
⟨*proof*⟩

Show there's an interval surrounding $f\ x$ in $f[[x - d, x + d]]$ .

**lemma** *isCont-inj-range*:

**fixes** $f$ :: *real* $\Rightarrow$ *real*
**assumes** $d$: $0 < d$
**and** *inj*: $\forall z.\ |z-x| \leq d$ --> $g(f\ z) = z$
**and** *cont*: $\forall z.\ |z-x| \leq d$ --> *isCont* $f\ z$
**shows** $\exists e>0.\ \forall y.\ |y - f\ x| \leq e$ --> $(\exists z.\ |z-x| \leq d\ \&\ f\ z = y)$
⟨*proof*⟩

Continuity of inverse function

**lemma** *isCont-inverse-function*:
**fixes** $f\ g$ :: *real* $\Rightarrow$ *real*
**assumes** $d$: $0 < d$
**and** *inj*: $\forall z.\ |z-x| \leq d$ --> $g(f\ z) = z$
**and** *cont*: $\forall z.\ |z-x| \leq d$ --> *isCont* $f\ z$
**shows** *isCont* $g\ (f\ x)$
⟨*proof*⟩

Derivative of inverse function

**lemma** *DERIV-inverse-function*:
**fixes** $f\ g$ :: *real* $\Rightarrow$ *real*
**assumes** *der*: *DERIV* $f\ (g\ x)$ :> $D$
**assumes** *neq*: $D \neq 0$
**assumes** $a$: $a < x$ **and** $b$: $x < b$
**assumes** *inj*: $\forall y.\ a < y \wedge y < b \longrightarrow f\ (g\ y) = y$
**assumes** *cont*: *isCont* $g\ x$
**shows** *DERIV* $g\ x$ :> *inverse* $D$
⟨*proof*⟩

**theorem** *GMVT*:
**fixes** $a\ b$ :: *real*
**assumes** *alb*: $a < b$
**and** *fc*: $\forall x.\ a \leq x \wedge x \leq b \longrightarrow$ *isCont* $f\ x$
**and** *fd*: $\forall x.\ a < x \wedge x < b \longrightarrow f$ *differentiable* $x$
**and** *gc*: $\forall x.\ a \leq x \wedge x \leq b \longrightarrow$ *isCont* $g\ x$
**and** *gd*: $\forall x.\ a < x \wedge x < b \longrightarrow g$ *differentiable* $x$
**shows** $\exists g'c\ f'c\ c.$ *DERIV* $g\ c$ :> $g'c \wedge$ *DERIV* $f\ c$ :> $f'c \wedge a < c \wedge c < b \wedge ((f\ b - f\ a) * g'c) = ((g\ b - g\ a) * f'c)$
⟨*proof*⟩

**lemma** *lemma-DERIV-subst*: [| *DERIV* $f\ x$ :> $D$; $D = E$ |] ==> *DERIV* $f\ x$ :> $E$
⟨*proof*⟩

**end**

# 15 NthRoot: Nth Roots of Real Numbers

**theory** *NthRoot*

**imports** *SEQ Parity Deriv*
**begin**

## 15.1 Existence of Nth Root

Existence follows from the Intermediate Value Theorem

**lemma** *realpow-pos-nth*:
  **assumes** $n$: $0 < n$
  **assumes** $a$: $0 < a$
  **shows** $\exists r>0.\ r \ \hat{}\ n = (a::real)$
$\langle proof \rangle$

**lemma** *realpow-pos-nth2*: $(0::real) < a \Longrightarrow \exists r>0.\ r \ \hat{}\ Suc\ n = a$
$\langle proof \rangle$

Uniqueness of nth positive root

**lemma** *realpow-pos-nth-unique*:
  $[\![0 < n;\ 0 < a]\!] \Longrightarrow \exists!r.\ 0 < r \wedge r \ \hat{}\ n = (a::real)$
$\langle proof \rangle$

## 15.2 Nth Root

We define roots of negative reals such that *root n* $(-\ x) = -$ *root n x*. This allows us to omit side conditions from many theorems.

**definition**
  *root* :: [*nat*, *real*] $\Rightarrow$ *real* **where**
  $root\ n\ x = (if\ 0 < x\ then\ (THE\ u.\ 0 < u \wedge u \ \hat{}\ n = x)\ else$
    $if\ x < 0\ then\ -\ (THE\ u.\ 0 < u \wedge u \ \hat{}\ n = -\ x)\ else\ 0)$

**lemma** *real-root-zero* [*simp*]: $root\ n\ 0 = 0$
$\langle proof \rangle$

**lemma** *real-root-minus*: $0 < n \Longrightarrow root\ n\ (-\ x) = -\ root\ n\ x$
$\langle proof \rangle$

**lemma** *real-root-gt-zero*: $[\![0 < n;\ 0 < x]\!] \Longrightarrow 0 < root\ n\ x$
$\langle proof \rangle$

**lemma** *real-root-pow-pos*:
  $[\![0 < n;\ 0 < x]\!] \Longrightarrow root\ n\ x \ \hat{}\ n = x$
$\langle proof \rangle$

**lemma** *real-root-pow-pos2* [*simp*]:
  $[\![0 < n;\ 0 \le x]\!] \Longrightarrow root\ n\ x \ \hat{}\ n = x$
$\langle proof \rangle$

**lemma** *odd-pos*: $odd\ (n::nat) \Longrightarrow 0 < n$

⟨*proof*⟩

**lemma** *odd-real-root-pow*: $odd\ n \Longrightarrow root\ n\ x\ \hat{}\ n = x$
⟨*proof*⟩

**lemma** *real-root-ge-zero*: $[\![0 < n;\ 0 \le x]\!] \Longrightarrow 0 \le root\ n\ x$
⟨*proof*⟩

**lemma** *real-root-power-cancel*: $[\![0 < n;\ 0 \le x]\!] \Longrightarrow root\ n\ (x\ \hat{}\ n) = x$
⟨*proof*⟩

**lemma** *odd-real-root-power-cancel*: $odd\ n \Longrightarrow root\ n\ (x\ \hat{}\ n) = x$
⟨*proof*⟩

**lemma** *real-root-pos-unique*:
$[\![0 < n;\ 0 \le y;\ y\ \hat{}\ n = x]\!] \Longrightarrow root\ n\ x = y$
⟨*proof*⟩

**lemma** *odd-real-root-unique*:
$[\![odd\ n;\ y\ \hat{}\ n = x]\!] \Longrightarrow root\ n\ x = y$
⟨*proof*⟩

**lemma** *real-root-one* [*simp*]: $0 < n \Longrightarrow root\ n\ 1 = 1$
⟨*proof*⟩

Root function is strictly monotonic, hence injective

**lemma** *real-root-less-mono-lemma*:
$[\![0 < n;\ 0 \le x;\ x < y]\!] \Longrightarrow root\ n\ x < root\ n\ y$
⟨*proof*⟩

**lemma** *real-root-less-mono*: $[\![0 < n;\ x < y]\!] \Longrightarrow root\ n\ x < root\ n\ y$
⟨*proof*⟩

**lemma** *real-root-le-mono*: $[\![0 < n;\ x \le y]\!] \Longrightarrow root\ n\ x \le root\ n\ y$
⟨*proof*⟩

**lemma** *real-root-less-iff* [*simp*]:
$0 < n \Longrightarrow (root\ n\ x < root\ n\ y) = (x < y)$
⟨*proof*⟩

**lemma** *real-root-le-iff* [*simp*]:
$0 < n \Longrightarrow (root\ n\ x \le root\ n\ y) = (x \le y)$
⟨*proof*⟩

**lemma** *real-root-eq-iff* [*simp*]:
$0 < n \Longrightarrow (root\ n\ x = root\ n\ y) = (x = y)$
⟨*proof*⟩

**lemmas** *real-root-gt-0-iff* [*simp*] = *real-root-less-iff* [**where** $x=0$, *simplified*]

**lemmas** *real-root-lt-0-iff* [*simp*] = *real-root-less-iff* [**where** $y=0$, *simplified*]
**lemmas** *real-root-ge-0-iff* [*simp*] = *real-root-le-iff* [**where** $x=0$, *simplified*]
**lemmas** *real-root-le-0-iff* [*simp*] = *real-root-le-iff* [**where** $y=0$, *simplified*]
**lemmas** *real-root-eq-0-iff* [*simp*] = *real-root-eq-iff* [**where** $y=0$, *simplified*]

**lemma** *real-root-gt-1-iff* [*simp*]: $0 < n \Longrightarrow (1 < root\ n\ y) = (1 < y)$
⟨*proof*⟩

**lemma** *real-root-lt-1-iff* [*simp*]: $0 < n \Longrightarrow (root\ n\ x < 1) = (x < 1)$
⟨*proof*⟩

**lemma** *real-root-ge-1-iff* [*simp*]: $0 < n \Longrightarrow (1 \leq root\ n\ y) = (1 \leq y)$
⟨*proof*⟩

**lemma** *real-root-le-1-iff* [*simp*]: $0 < n \Longrightarrow (root\ n\ x \leq 1) = (x \leq 1)$
⟨*proof*⟩

**lemma** *real-root-eq-1-iff* [*simp*]: $0 < n \Longrightarrow (root\ n\ x = 1) = (x = 1)$
⟨*proof*⟩

Roots of roots

**lemma** *real-root-Suc-0* [*simp*]: $root\ (Suc\ 0)\ x = x$
⟨*proof*⟩

**lemma** *real-root-pos-mult-exp*:
$[\![0 < m;\ 0 < n;\ 0 < x]\!] \Longrightarrow root\ (m * n)\ x = root\ m\ (root\ n\ x)$
⟨*proof*⟩

**lemma** *real-root-mult-exp*:
$[\![0 < m;\ 0 < n]\!] \Longrightarrow root\ (m * n)\ x = root\ m\ (root\ n\ x)$
⟨*proof*⟩

**lemma** *real-root-commute*:
$[\![0 < m;\ 0 < n]\!] \Longrightarrow root\ m\ (root\ n\ x) = root\ n\ (root\ m\ x)$
⟨*proof*⟩

Monotonicity in first argument

**lemma** *real-root-strict-decreasing*:
$[\![0 < n;\ n < N;\ 1 < x]\!] \Longrightarrow root\ N\ x < root\ n\ x$
⟨*proof*⟩

**lemma** *real-root-strict-increasing*:
$[\![0 < n;\ n < N;\ 0 < x;\ x < 1]\!] \Longrightarrow root\ n\ x < root\ N\ x$
⟨*proof*⟩

**lemma** *real-root-decreasing*:
$[\![0 < n;\ n < N;\ 1 \leq x]\!] \Longrightarrow root\ N\ x \leq root\ n\ x$
⟨*proof*⟩

**lemma** *real-root-increasing*:
$[\![0 < n; n < N; 0 \leq x; x \leq 1]\!] \Longrightarrow root\ n\ x \leq root\ N\ x$
⟨*proof*⟩

Roots of multiplication and division

**lemma** *real-root-mult-lemma*:
$[\![0 < n; 0 \leq x; 0 \leq y]\!] \Longrightarrow root\ n\ (x * y) = root\ n\ x * root\ n\ y$
⟨*proof*⟩

**lemma** *real-root-inverse-lemma*:
$[\![0 < n; 0 \leq x]\!] \Longrightarrow root\ n\ (inverse\ x) = inverse\ (root\ n\ x)$
⟨*proof*⟩

**lemma** *real-root-mult*:
**assumes** $n$: $0 < n$
**shows** $root\ n\ (x * y) = root\ n\ x * root\ n\ y$
⟨*proof*⟩

**lemma** *real-root-inverse*:
**assumes** $n$: $0 < n$
**shows** $root\ n\ (inverse\ x) = inverse\ (root\ n\ x)$
⟨*proof*⟩

**lemma** *real-root-divide*:
$0 < n \Longrightarrow root\ n\ (x\ /\ y) = root\ n\ x\ /\ root\ n\ y$
⟨*proof*⟩

**lemma** *real-root-power*:
$0 < n \Longrightarrow root\ n\ (x\ \hat{}\ k) = root\ n\ x\ \hat{}\ k$
⟨*proof*⟩

**lemma** *real-root-abs*: $0 < n \Longrightarrow root\ n\ |x| = |root\ n\ x|$
⟨*proof*⟩

Continuity and derivatives

**lemma** *isCont-root-pos*:
**assumes** $n$: $0 < n$
**assumes** $x$: $0 < x$
**shows** $isCont\ (root\ n)\ x$
⟨*proof*⟩

**lemma** *isCont-root-neg*:
$[\![0 < n; x < 0]\!] \Longrightarrow isCont\ (root\ n)\ x$
⟨*proof*⟩

**lemma** *isCont-root-zero*:
$0 < n \Longrightarrow isCont\ (root\ n)\ 0$
⟨*proof*⟩

**lemma** *isCont-real-root*: $0 < n \Longrightarrow isCont\ (root\ n)\ x$
⟨*proof*⟩

**lemma** *DERIV-real-root*:
**assumes** *n*: $0 < n$
**assumes** *x*: $0 < x$
**shows** $DERIV\ (root\ n)\ x :> inverse\ (real\ n * root\ n\ x\ \hat{}\ (n - Suc\ 0))$
⟨*proof*⟩

**lemma** *DERIV-odd-real-root*:
**assumes** *n*: *odd n*
**assumes** *x*: $x \neq 0$
**shows** $DERIV\ (root\ n)\ x :> inverse\ (real\ n * root\ n\ x\ \hat{}\ (n - Suc\ 0))$
⟨*proof*⟩

## 15.3 Square Root

**definition**
*sqrt* :: *real* ⇒ *real* **where**
$sqrt = root\ 2$

**lemma** *pos2*: $0 < (2::nat)$ ⟨*proof*⟩

**lemma** *real-sqrt-unique*: $[\![y^2 = x; 0 \leq y]\!] \Longrightarrow sqrt\ x = y$
⟨*proof*⟩

**lemma** *real-sqrt-abs* [*simp*]: $sqrt\ (x^2) = |x|$
⟨*proof*⟩

**lemma** *real-sqrt-pow2* [*simp*]: $0 \leq x \Longrightarrow (sqrt\ x)^2 = x$
⟨*proof*⟩

**lemma** *real-sqrt-pow2-iff* [*simp*]: $((sqrt\ x)^2 = x) = (0 \leq x)$
⟨*proof*⟩

**lemma** *real-sqrt-zero* [*simp*]: $sqrt\ 0 = 0$
⟨*proof*⟩

**lemma** *real-sqrt-one* [*simp*]: $sqrt\ 1 = 1$
⟨*proof*⟩

**lemma** *real-sqrt-minus*: $sqrt\ (-\ x) = -\ sqrt\ x$
⟨*proof*⟩

**lemma** *real-sqrt-mult*: $sqrt\ (x * y) = sqrt\ x * sqrt\ y$
⟨*proof*⟩

**lemma** *real-sqrt-inverse*: $sqrt\ (inverse\ x) = inverse\ (sqrt\ x)$
⟨*proof*⟩

**lemma** *real-sqrt-divide*: *sqrt* (*x* / *y*) = *sqrt x* / *sqrt y*
⟨*proof*⟩

**lemma** *real-sqrt-power*: *sqrt* (*x* ˆ *k*) = *sqrt x* ˆ *k*
⟨*proof*⟩

**lemma** *real-sqrt-gt-zero*: *0* < *x* ⟹ *0* < *sqrt x*
⟨*proof*⟩

**lemma** *real-sqrt-ge-zero*: *0* ≤ *x* ⟹ *0* ≤ *sqrt x*
⟨*proof*⟩

**lemma** *real-sqrt-less-mono*: *x* < *y* ⟹ *sqrt x* < *sqrt y*
⟨*proof*⟩

**lemma** *real-sqrt-le-mono*: *x* ≤ *y* ⟹ *sqrt x* ≤ *sqrt y*
⟨*proof*⟩

**lemma** *real-sqrt-less-iff* [*simp*]: (*sqrt x* < *sqrt y*) = (*x* < *y*)
⟨*proof*⟩

**lemma** *real-sqrt-le-iff* [*simp*]: (*sqrt x* ≤ *sqrt y*) = (*x* ≤ *y*)
⟨*proof*⟩

**lemma** *real-sqrt-eq-iff* [*simp*]: (*sqrt x* = *sqrt y*) = (*x* = *y*)
⟨*proof*⟩

**lemmas** *real-sqrt-gt-0-iff* [*simp*] = *real-sqrt-less-iff* [**where** *x*=*0*, *simplified*]
**lemmas** *real-sqrt-lt-0-iff* [*simp*] = *real-sqrt-less-iff* [**where** *y*=*0*, *simplified*]
**lemmas** *real-sqrt-ge-0-iff* [*simp*] = *real-sqrt-le-iff* [**where** *x*=*0*, *simplified*]
**lemmas** *real-sqrt-le-0-iff* [*simp*] = *real-sqrt-le-iff* [**where** *y*=*0*, *simplified*]
**lemmas** *real-sqrt-eq-0-iff* [*simp*] = *real-sqrt-eq-iff* [**where** *y*=*0*, *simplified*]

**lemmas** *real-sqrt-gt-1-iff* [*simp*] = *real-sqrt-less-iff* [**where** *x*=*1*, *simplified*]
**lemmas** *real-sqrt-lt-1-iff* [*simp*] = *real-sqrt-less-iff* [**where** *y*=*1*, *simplified*]
**lemmas** *real-sqrt-ge-1-iff* [*simp*] = *real-sqrt-le-iff* [**where** *x*=*1*, *simplified*]
**lemmas** *real-sqrt-le-1-iff* [*simp*] = *real-sqrt-le-iff* [**where** *y*=*1*, *simplified*]
**lemmas** *real-sqrt-eq-1-iff* [*simp*] = *real-sqrt-eq-iff* [**where** *y*=*1*, *simplified*]

**lemma** *isCont-real-sqrt*: *isCont sqrt x*
⟨*proof*⟩

**lemma** *DERIV-real-sqrt*:
*0* < *x* ⟹ *DERIV sqrt x* :> *inverse* (*sqrt x*) / *2*
⟨*proof*⟩

**lemma** *not-real-square-gt-zero* [*simp*]: (~ (*0*::*real*) < *x*∗*x*) = (*x* = *0*)
⟨*proof*⟩

**lemma** *real-sqrt-abs2* [*simp*]: $sqrt(x*x) = |x|$
⟨*proof*⟩

**lemma** *real-sqrt-pow2-gt-zero*: $0 < x ==> 0 < (sqrt\ x)^2$
⟨*proof*⟩

**lemma** *real-sqrt-not-eq-zero*: $0 < x ==> sqrt\ x \neq 0$
⟨*proof*⟩

**lemma** *real-inv-sqrt-pow2*: $0 < x ==> inverse\ (sqrt(x))\ \hat{}\ 2 = inverse\ x$
⟨*proof*⟩

**lemma** *real-sqrt-eq-zero-cancel*: $[|\ 0 \leq x;\ sqrt(x) = 0|] ==> x = 0$
⟨*proof*⟩

**lemma** *real-sqrt-ge-one*: $1 \leq x ==> 1 \leq sqrt\ x$
⟨*proof*⟩

**lemma** *real-sqrt-two-gt-zero* [*simp*]: $0 < sqrt\ 2$
⟨*proof*⟩

**lemma** *real-sqrt-two-ge-zero* [*simp*]: $0 \leq sqrt\ 2$
⟨*proof*⟩

**lemma** *real-sqrt-two-gt-one* [*simp*]: $1 < sqrt\ 2$
⟨*proof*⟩

**lemma** *sqrt-divide-self-eq*:
 **assumes** *nneg*: $0 \leq x$
 **shows** $sqrt\ x\ /\ x = inverse\ (sqrt\ x)$
⟨*proof*⟩

**lemma** *real-divide-square-eq* [*simp*]: $(((r::real) * a)\ /\ (r * r)) = a\ /\ r$
⟨*proof*⟩

**lemma** *lemma-real-divide-sqrt-less*: $0 < u ==> u\ /\ sqrt\ 2 < u$
⟨*proof*⟩

**lemma** *four-x-squared*:
 **fixes** $x::real$
 **shows** $4 * x^2 = (2 * x)^2$
⟨*proof*⟩

## 15.4 Square Root of Sum of Squares

**lemma** *real-sqrt-mult-self-sum-ge-zero* [*simp*]: $0 \leq sqrt(x*x + y*y)$
⟨*proof*⟩

**lemma** *real-sqrt-sum-squares-ge-zero* [*simp*]: $0 \leq sqrt\ (x^2 + y^2)$
⟨*proof*⟩

**declare** *real-sqrt-sum-squares-ge-zero* [*THEN abs-of-nonneg*, *simp*]

**lemma** *real-sqrt-sum-squares-mult-ge-zero* [*simp*]:
$0 \leq sqrt\ ((x^2 + y^2)*(xa^2 + ya^2))$
⟨*proof*⟩

**lemma** *real-sqrt-sum-squares-mult-squared-eq* [*simp*]:
$sqrt\ ((x^2 + y^2) * (xa^2 + ya^2))\ \hat{}\ 2 = (x^2 + y^2) * (xa^2 + ya^2)$
⟨*proof*⟩

**lemma** *real-sqrt-sum-squares-eq-cancel*: $sqrt\ (x^2 + y^2) = x \Longrightarrow y = 0$
⟨*proof*⟩

**lemma** *real-sqrt-sum-squares-eq-cancel2*: $sqrt\ (x^2 + y^2) = y \Longrightarrow x = 0$
⟨*proof*⟩

**lemma** *real-sqrt-sum-squares-ge1* [*simp*]: $x \leq sqrt\ (x^2 + y^2)$
⟨*proof*⟩

**lemma** *real-sqrt-sum-squares-ge2* [*simp*]: $y \leq sqrt\ (x^2 + y^2)$
⟨*proof*⟩

**lemma** *real-sqrt-ge-abs1* [*simp*]: $|x| \leq sqrt\ (x^2 + y^2)$
⟨*proof*⟩

**lemma** *real-sqrt-ge-abs2* [*simp*]: $|y| \leq sqrt\ (x^2 + y^2)$
⟨*proof*⟩

**lemma** *le-real-sqrt-sumsq* [*simp*]: $x \leq sqrt\ (x * x + y * y)$
⟨*proof*⟩

**lemma** *power2-sum*:
**fixes** $x\ y :: 'a::\{number\text{-}ring,recpower\}$
**shows** $(x + y)^2 = x^2 + y^2 + 2 * x * y$
⟨*proof*⟩

**lemma** *power2-diff*:
**fixes** $x\ y :: 'a::\{number\text{-}ring,recpower\}$
**shows** $(x - y)^2 = x^2 + y^2 - 2 * x * y$
⟨*proof*⟩

**lemma** *real-sqrt-sum-squares-triangle-ineq*:
$sqrt\ ((a + c)^2 + (b + d)^2) \leq sqrt\ (a^2 + b^2) + sqrt\ (c^2 + d^2)$
⟨*proof*⟩

**lemma** *real-sqrt-sum-squares-less*:

$[\![|x| < u \ / \ sqrt\ 2;\ |y| < u \ / \ sqrt\ 2]\!] \Longrightarrow sqrt\ (x^2 + y^2) < u$
$\langle proof \rangle$

Needed for the infinitely close relation over the nonstandard complex numbers

**lemma** *lemma-sqrt-hcomplex-capprox*:
$[|\ 0 < u;\ x < u/2;\ y < u/2;\ 0 \le x;\ 0 \le y\ |] ==> sqrt\ (x^2 + y^2) < u$
$\langle proof \rangle$

Legacy theorem names:

**lemmas** *real-root-pos2* = *real-root-power-cancel*
**lemmas** *real-root-pos-pos* = *real-root-gt-zero* [*THEN order-less-imp-le*]
**lemmas** *real-root-pos-pos-le* = *real-root-ge-zero*
**lemmas** *real-sqrt-mult-distrib* = *real-sqrt-mult*
**lemmas** *real-sqrt-mult-distrib2* = *real-sqrt-mult*
**lemmas** *real-sqrt-eq-zero-cancel-iff* = *real-sqrt-eq-0-iff*

**lemma** *real-root-pos*: $0 < x \Longrightarrow root\ (Suc\ n)\ (x\ \hat{}\ (Suc\ n)) = x$
$\langle proof \rangle$

**end**

# 16 Fact: Factorial Function

**theory** *Fact*
**imports** *../Real/Real*
**begin**

**consts** *fact* :: *nat => nat*
**primrec**
*fact-0*: $fact\ 0 = 1$
*fact-Suc*: $fact\ (Suc\ n) = (Suc\ n) * fact\ n$

**lemma** *fact-gt-zero* [*simp*]: $0 < fact\ n$
$\langle proof \rangle$

**lemma** *fact-not-eq-zero* [*simp*]: $fact\ n \neq 0$
$\langle proof \rangle$

**lemma** *real-of-nat-fact-not-zero* [*simp*]: $real\ (fact\ n) \neq 0$
$\langle proof \rangle$

**lemma** *real-of-nat-fact-gt-zero* [*simp*]: $0 < real(fact\ n)$
$\langle proof \rangle$

**lemma** *real-of-nat-fact-ge-zero* [*simp*]: $0 \leq real(fact\ n)$
⟨*proof*⟩

**lemma** *fact-ge-one* [*simp*]: $1 \leq fact\ n$
⟨*proof*⟩

**lemma** *fact-mono*: $m \leq n ==> fact\ m \leq fact\ n$
⟨*proof*⟩

Note that $fact\ 0 = fact\ 1$

**lemma** *fact-less-mono*: $[|\ 0 < m;\ m < n\ |] ==> fact\ m < fact\ n$
⟨*proof*⟩

**lemma** *inv-real-of-nat-fact-gt-zero* [*simp*]: $0 < inverse\ (real\ (fact\ n))$
⟨*proof*⟩

**lemma** *inv-real-of-nat-fact-ge-zero* [*simp*]: $0 \leq inverse\ (real\ (fact\ n))$
⟨*proof*⟩

**lemma** *fact-diff-Suc* [*rule-format*]:
$n < Suc\ m ==> fact\ (Suc\ m - n) = (Suc\ m - n) * fact\ (m - n)$
⟨*proof*⟩

**lemma** *fact-num0* [*simp*]: $fact\ 0 = 1$
⟨*proof*⟩

**lemma** *fact-num-eq-if*: $fact\ m = (if\ m{=}0\ then\ 1\ else\ m * fact\ (m - 1))$
⟨*proof*⟩

**lemma** *fact-add-num-eq-if*:
$fact\ (m + n) = (if\ m + n = 0\ then\ 1\ else\ (m + n) * fact\ (m + n - 1))$
⟨*proof*⟩

**lemma** *fact-add-num-eq-if2*:
$fact\ (m + n) = (if\ m = 0\ then\ fact\ n\ else\ (m + n) * fact\ ((m - 1) + n))$
⟨*proof*⟩

**end**

# 17 Series: Finite Summation and Infinite Series

**theory** *Series*
**imports** *SEQ*
**begin**

**definition**
*sums* :: $(nat \Rightarrow 'a{::}real\text{-}normed\text{-}vector) \Rightarrow 'a \Rightarrow bool$
(**infixr** *sums 80*) **where**

$f$ *sums* $s = (\%n.\ setsum\ f\ \{0..<n\})$ ----> $s$

**definition**
*summable* :: (*nat* ⇒ *'a::real-normed-vector*) ⇒ *bool* **where**
*summable* $f = (\exists\, s.\ f\ sums\ s)$

**definition**
*suminf* :: (*nat* ⇒ *'a::real-normed-vector*) ⇒ *'a* **where**
*suminf* $f = (THE\ s.\ f\ sums\ s)$

**syntax**
*-suminf* :: *idt* ⇒ *'a* ⇒ *'a* ($\sum$-. - [0, 10] 10)
**translations**
$\sum i.\ b$ == *CONST suminf* (%i. b)

**lemma** *sumr-diff-mult-const*:
$setsum\ f\ \{0..<n\} - (real\ n*r) = setsum\ (\%i.\ f\ i - r)\ \{0..<n::nat\}$
⟨*proof*⟩

**lemma** *real-setsum-nat-ivl-bounded*:
$(!!p.\ p < n \Longrightarrow f(p) \le K)$
$\Longrightarrow setsum\ f\ \{0..<n::nat\} \le real\ n * K$
⟨*proof*⟩

**lemma** *sumr-minus-one-realpow-zero* [*simp*]:
$(\sum i{=}0..<2*n.\ (-1)\ \hat{}\ Suc\ i) = (0::real)$
⟨*proof*⟩

**lemma** *sumr-one-lb-realpow-zero* [*simp*]:
$(\sum n{=}Suc\ 0..<n.\ f(n) * (0::real)\ \hat{}\ n) = 0$
⟨*proof*⟩

**lemma** *sumr-group*:
$(\sum m{=}0..<n::nat.\ setsum\ f\ \{m * k\ ..<\ m*k + k\}) = setsum\ f\ \{0\ ..<\ n * k\}$
⟨*proof*⟩

**lemma** *sumr-offset3*:
$setsum\ f\ \{0::nat..<n+k\} = (\sum m{=}0..<n.\ f\ (m+k)) + setsum\ f\ \{0..<k\}$
⟨*proof*⟩

**lemma** *sumr-offset*:
**fixes** $f$ :: *nat* ⇒ *'a::ab-group-add*
**shows** $(\sum m{=}0..<n.\ f(m+k)) = setsum\ f\ \{0..<n+k\} - setsum\ f\ \{0..<k\}$
⟨*proof*⟩

**lemma** *sumr-offset2*:

$\forall f.\ (\sum m{=}0..{<}n{::}nat.\ f(m{+}k){::}real) = setsum\, f\ \{0..{<}n{+}k\} - setsum\, f\ \{0..{<}k\}$
⟨proof⟩

**lemma** *sumr-offset4*:
$\forall n\ f.\ setsum\ f\ \{0{::}nat..{<}n{+}k\} = (\sum m{=}0..{<}n.\ f\ (m{+}k){::}real) + setsum\ f$
$\{0..{<}k\}$
⟨proof⟩

## 17.1 Infinite Sums, by the Properties of Limits

**lemma** *sums-summable*: $f\ sums\ l \Longrightarrow summable\ f$
⟨proof⟩

**lemma** *summable-sums*: $summable\ f \Longrightarrow f\ sums\ (suminf\ f)$
⟨proof⟩

**lemma** *summable-sumr-LIMSEQ-suminf*:
$summable\ f \Longrightarrow (\%n.\ setsum\ f\ \{0..{<}n\})\ {----}{>}\ (suminf\ f)$
⟨proof⟩

**lemma** *sums-unique*: $f\ sums\ s \Longrightarrow (s = suminf\ f)$
⟨proof⟩

**lemma** *sums-split-initial-segment*: $f\ sums\ s \Longrightarrow$
$(\%n.\ f(n + k))\ sums\ (s - (SUM\ i = 0..{<}\ k.\ f\ i))$
⟨proof⟩

**lemma** *summable-ignore-initial-segment*: $summable\ f \Longrightarrow$
$summable\ (\%n.\ f(n + k))$
⟨proof⟩

**lemma** *suminf-minus-initial-segment*: $summable\ f \Longrightarrow$
$suminf\ f = s \Longrightarrow suminf\ (\%n.\ f(n + k)) = s - (SUM\ i = 0..{<}\ k.\ f\ i)$
⟨proof⟩

**lemma** *suminf-split-initial-segment*: $summable\ f \Longrightarrow$
$suminf\ f = (SUM\ i = 0..{<}\ k.\ f\ i) + suminf\ (\%n.\ f(n + k))$
⟨proof⟩

**lemma** *series-zero*:
$(\forall m.\ n \le m \longrightarrow f(m) = 0) \Longrightarrow f\ sums\ (setsum\ f\ \{0..{<}n\})$
⟨proof⟩

**lemma** *sums-zero*: $(\lambda n.\ 0)\ sums\ 0$
⟨proof⟩

**lemma** *summable-zero*: $summable\ (\lambda n.\ 0)$
⟨proof⟩

**lemma** *suminf-zero*: *suminf* $(\lambda n.\ 0) = 0$
⟨*proof*⟩

**lemma** (**in** *bounded-linear*) *sums*:
$(\lambda n.\ X\ n)$ *sums* $a \Longrightarrow (\lambda n.\ f\ (X\ n))$ *sums* $(f\ a)$
⟨*proof*⟩

**lemma** (**in** *bounded-linear*) *summable*:
*summable* $(\lambda n.\ X\ n) \Longrightarrow$ *summable* $(\lambda n.\ f\ (X\ n))$
⟨*proof*⟩

**lemma** (**in** *bounded-linear*) *suminf*:
*summable* $(\lambda n.\ X\ n) \Longrightarrow f\ (\sum n.\ X\ n) = (\sum n.\ f\ (X\ n))$
⟨*proof*⟩

**lemma** *sums-mult*:
**fixes** $c$ :: $'a$::*real-normed-algebra*
**shows** $f$ *sums* $a \Longrightarrow (\lambda n.\ c * f\ n)$ *sums* $(c * a)$
⟨*proof*⟩

**lemma** *summable-mult*:
**fixes** $c$ :: $'a$::*real-normed-algebra*
**shows** *summable* $f \Longrightarrow$ *summable* $(\%n.\ c * f\ n)$
⟨*proof*⟩

**lemma** *suminf-mult*:
**fixes** $c$ :: $'a$::*real-normed-algebra*
**shows** *summable* $f \Longrightarrow$ *suminf* $(\lambda n.\ c * f\ n) = c *$ *suminf* $f$
⟨*proof*⟩

**lemma** *sums-mult2*:
**fixes** $c$ :: $'a$::*real-normed-algebra*
**shows** $f$ *sums* $a \Longrightarrow (\lambda n.\ f\ n * c)$ *sums* $(a * c)$
⟨*proof*⟩

**lemma** *summable-mult2*:
**fixes** $c$ :: $'a$::*real-normed-algebra*
**shows** *summable* $f \Longrightarrow$ *summable* $(\lambda n.\ f\ n * c)$
⟨*proof*⟩

**lemma** *suminf-mult2*:
**fixes** $c$ :: $'a$::*real-normed-algebra*
**shows** *summable* $f \Longrightarrow$ *suminf* $f * c = (\sum n.\ f\ n * c)$
⟨*proof*⟩

**lemma** *sums-divide*:
**fixes** $c$ :: $'a$::*real-normed-field*
**shows** $f$ *sums* $a \Longrightarrow (\lambda n.\ f\ n\ /\ c)$ *sums* $(a\ /\ c)$

⟨*proof*⟩

**lemma** *summable-divide*:
**fixes** $c$ :: $'a$::*real-normed-field*
**shows** *summable* $f \Longrightarrow$ *summable* ($\lambda n.\ f\ n\ /\ c$)
⟨*proof*⟩

**lemma** *suminf-divide*:
**fixes** $c$ :: $'a$::*real-normed-field*
**shows** *summable* $f \Longrightarrow$ *suminf* ($\lambda n.\ f\ n\ /\ c$) = *suminf* $f$ / $c$
⟨*proof*⟩

**lemma** *sums-add*: $[\![X$ *sums* $a$; $Y$ *sums* $b]\!] \Longrightarrow (\lambda n.\ X\ n + Y\ n)$ *sums* $(a + b)$
⟨*proof*⟩

**lemma** *summable-add*: $[\![$*summable* $X$; *summable* $Y]\!] \Longrightarrow$ *summable* ($\lambda n.\ X\ n + Y$
$n$)
⟨*proof*⟩

**lemma** *suminf-add*:
$[\![$*summable* $X$; *summable* $Y]\!] \Longrightarrow$ *suminf* $X$ + *suminf* $Y$ = $(\sum n.\ X\ n + Y\ n)$
⟨*proof*⟩

**lemma** *sums-diff*: $[\![X$ *sums* $a$; $Y$ *sums* $b]\!] \Longrightarrow (\lambda n.\ X\ n - Y\ n)$ *sums* $(a - b)$
⟨*proof*⟩

**lemma** *summable-diff*: $[\![$*summable* $X$; *summable* $Y]\!] \Longrightarrow$ *summable* ($\lambda n.\ X\ n - Y$
$n$)
⟨*proof*⟩

**lemma** *suminf-diff*:
$[\![$*summable* $X$; *summable* $Y]\!] \Longrightarrow$ *suminf* $X$ − *suminf* $Y$ = $(\sum n.\ X\ n - Y\ n)$
⟨*proof*⟩

**lemma** *sums-minus*: $X$ *sums* $a$ ==> ($\lambda n.\ -\ X\ n$) *sums* ($-\ a$)
⟨*proof*⟩

**lemma** *summable-minus*: *summable* $X \Longrightarrow$ *summable* ($\lambda n.\ -\ X\ n$)
⟨*proof*⟩

**lemma** *suminf-minus*: *summable* $X \Longrightarrow (\sum n.\ -\ X\ n) = -\ (\sum n.\ X\ n)$
⟨*proof*⟩

**lemma** *sums-group*:
[|*summable* $f$; $0 < k$ |] ==> (%n. *setsum* $f$ {n*k..<n*k+k}) *sums* (*suminf* $f$)
⟨*proof*⟩

A summable series of positive terms has limit that is at least as great as any partial sum.

**lemma** *series-pos-le*:
**fixes** $f :: nat \Rightarrow real$
**shows** $[\![summable\ f;\ \forall m \geq n.\ 0 \leq f\ m]\!] \Longrightarrow setsum\ f\ \{0..<n\} \leq suminf\ f$
⟨*proof*⟩

**lemma** *series-pos-less*:
**fixes** $f :: nat \Rightarrow real$
**shows** $[\![summable\ f;\ \forall m \geq n.\ 0 < f\ m]\!] \Longrightarrow setsum\ f\ \{0..<n\} < suminf\ f$
⟨*proof*⟩

**lemma** *suminf-gt-zero*:
**fixes** $f :: nat \Rightarrow real$
**shows** $[\![summable\ f;\ \forall n.\ 0 < f\ n]\!] \Longrightarrow 0 < suminf\ f$
⟨*proof*⟩

**lemma** *suminf-ge-zero*:
**fixes** $f :: nat \Rightarrow real$
**shows** $[\![summable\ f;\ \forall n.\ 0 \leq f\ n]\!] \Longrightarrow 0 \leq suminf\ f$
⟨*proof*⟩

**lemma** *sumr-pos-lt-pair*:
**fixes** $f :: nat \Rightarrow real$
**shows** $[\![summable\ f;$
$\forall d.\ 0 < f\ (k + (Suc(Suc\ 0) * d)) + f\ (k + ((Suc(Suc\ 0) * d) + 1))]\!]$
$\Longrightarrow setsum\ f\ \{0..<k\} < suminf\ f$
⟨*proof*⟩

Sum of a geometric progression.

**lemmas** *sumr-geometric* = *geometric-sum* [**where** *'a* = *real*]

**lemma** *geometric-sums*:
**fixes** $x :: 'a::\{real\text{-}normed\text{-}field,recpower\}$
**shows** $norm\ x < 1 \Longrightarrow (\lambda n.\ x\ \hat{}\ n)\ sums\ (1\ /\ (1 - x))$
⟨*proof*⟩

**lemma** *summable-geometric*:
**fixes** $x :: 'a::\{real\text{-}normed\text{-}field,recpower\}$
**shows** $norm\ x < 1 \Longrightarrow summable\ (\lambda n.\ x\ \hat{}\ n)$
⟨*proof*⟩

Cauchy-type criterion for convergence of series (c.f. Harrison)

**lemma** *summable-convergent-sumr-iff*:
$summable\ f = convergent\ (\%n.\ setsum\ f\ \{0..<n\})$
⟨*proof*⟩

**lemma** *summable-LIMSEQ-zero*: $summable\ f \Longrightarrow f\ ----> 0$
⟨*proof*⟩

**lemma** *summable-Cauchy*:

$summable\ (f::nat \Rightarrow 'a::banach) =$
$(\forall\, e > 0.\ \exists\, N.\ \forall\, m \geq N.\ \forall\, n.\ norm\ (setsum\ f\ \{m..<n\}) < e)$
$\langle proof \rangle$

Comparison test

**lemma** *norm-setsum*:
**fixes** $f :: 'a \Rightarrow 'b::real\text{-}normed\text{-}vector$
**shows** $norm\ (setsum\ f\ A) \leq (\sum i \in A.\ norm\ (f\ i))$
$\langle proof \rangle$

**lemma** *summable-comparison-test*:
**fixes** $f :: nat \Rightarrow 'a::banach$
**shows** $[\![\exists\, N.\ \forall\, n \geq N.\ norm\ (f\ n) \leq g\ n;\ summable\ g]\!] \Longrightarrow summable\ f$
$\langle proof \rangle$

**lemma** *summable-norm-comparison-test*:
**fixes** $f :: nat \Rightarrow 'a::banach$
**shows** $[\![\exists\, N.\ \forall\, n \geq N.\ norm\ (f\ n) \leq g\ n;\ summable\ g]\!]$
$\Longrightarrow summable\ (\lambda n.\ norm\ (f\ n))$
$\langle proof \rangle$

**lemma** *summable-rabs-comparison-test*:
**fixes** $f :: nat \Rightarrow real$
**shows** $[\![\exists\, N.\ \forall\, n \geq N.\ |f\ n| \leq g\ n;\ summable\ g]\!] \Longrightarrow summable\ (\lambda n.\ |f\ n|)$
$\langle proof \rangle$

Summability of geometric series for real algebras

**lemma** *complete-algebra-summable-geometric*:
**fixes** $x :: 'a::\{real\text{-}normed\text{-}algebra\text{-}1, banach, recpower\}$
**shows** $norm\ x < 1 \Longrightarrow summable\ (\lambda n.\ x\ \hat{}\ n)$
$\langle proof \rangle$

Limit comparison property for series (c.f. jrh)

**lemma** *summable-le*:
**fixes** $f\ g :: nat \Rightarrow real$
**shows** $[\![\forall\, n.\ f\ n \leq g\ n;\ summable\ f;\ summable\ g]\!] \Longrightarrow suminf\ f \leq suminf\ g$
$\langle proof \rangle$

**lemma** *summable-le2*:
**fixes** $f\ g :: nat \Rightarrow real$
**shows** $[\![\forall\, n.\ |f\ n| \leq g\ n;\ summable\ g]\!] \Longrightarrow summable\ f \wedge suminf\ f \leq suminf\ g$
$\langle proof \rangle$

**lemma** *suminf-0-le*:
**fixes** $f::nat \Rightarrow real$
**assumes** $gt0$: $\forall\, n.\ 0 \leq f\ n$ **and** $sm$: $summable\ f$
**shows** $0 \leq suminf\ f$
$\langle proof \rangle$

Absolute convergence imples normal convergence

**lemma** *summable-norm-cancel*:
 **fixes** $f$ :: $nat \Rightarrow 'a::banach$
 **shows** $summable\ (\lambda n.\ norm\ (f\ n)) \Longrightarrow summable\ f$
⟨*proof*⟩

**lemma** *summable-rabs-cancel*:
 **fixes** $f$ :: $nat \Rightarrow real$
 **shows** $summable\ (\lambda n.\ |f\ n|) \Longrightarrow summable\ f$
⟨*proof*⟩

Absolute convergence of series

**lemma** *summable-norm*:
 **fixes** $f$ :: $nat \Rightarrow 'a::banach$
 **shows** $summable\ (\lambda n.\ norm\ (f\ n)) \Longrightarrow norm\ (suminf\ f) \leq (\sum n.\ norm\ (f\ n))$
⟨*proof*⟩

**lemma** *summable-rabs*:
 **fixes** $f$ :: $nat \Rightarrow real$
 **shows** $summable\ (\lambda n.\ |f\ n|) \Longrightarrow |suminf\ f| \leq (\sum n.\ |f\ n|)$
⟨*proof*⟩

## 17.2 The Ratio Test

**lemma** *norm-ratiotest-lemma*:
 **fixes** $x\ y$ :: $'a::real\text{-}normed\text{-}vector$
 **shows** $[\![c \leq 0;\ norm\ x \leq c * norm\ y]\!] \Longrightarrow x = 0$
⟨*proof*⟩

**lemma** *rabs-ratiotest-lemma*: $[|\ c \leq 0;\ abs\ x \leq c * abs\ y\ |] ==> x = (0::real)$
⟨*proof*⟩

**lemma** *le-Suc-ex*: $(k::nat) \leq l ==> (\exists n.\ l = k + n)$
⟨*proof*⟩

**lemma** *le-Suc-ex-iff*: $((k::nat) \leq l) = (\exists n.\ l = k + n)$
⟨*proof*⟩

**lemma** *ratio-test-lemma2*:
 **fixes** $f$ :: $nat \Rightarrow 'a::banach$
 **shows** $[\![\forall n{\geq}N.\ norm\ (f\ (Suc\ n)) \leq c * norm\ (f\ n)]\!] \Longrightarrow 0 < c \lor summable\ f$
⟨*proof*⟩

**lemma** *ratio-test*:
 **fixes** $f$ :: $nat \Rightarrow 'a::banach$
 **shows** $[\![c < 1;\ \forall n{\geq}N.\ norm\ (f\ (Suc\ n)) \leq c * norm\ (f\ n)]\!] \Longrightarrow summable\ f$
⟨*proof*⟩

### 17.3 Cauchy Product Formula

**lemma** *setsum-triangle-reindex*:
  **fixes** $n :: nat$
  **shows** $(\sum (i,j) \in \{(i,j).\ i+j < n\}.\ f\ i\ j) = (\sum k=0..<n.\ \sum i=0..k.\ f\ i\ (k - i))$
$\langle proof \rangle$

**lemma** *Cauchy-product-sums*:
  **fixes** $a\ b :: nat \Rightarrow {}'a::\{real\text{-}normed\text{-}algebra,banach\}$
  **assumes** $a$: $summable\ (\lambda k.\ norm\ (a\ k))$
  **assumes** $b$: $summable\ (\lambda k.\ norm\ (b\ k))$
  **shows** $(\lambda k.\ \sum i=0..k.\ a\ i * b\ (k - i))\ sums\ ((\sum k.\ a\ k) * (\sum k.\ b\ k))$
$\langle proof \rangle$

**lemma** *Cauchy-product*:
  **fixes** $a\ b :: nat \Rightarrow {}'a::\{real\text{-}normed\text{-}algebra,banach\}$
  **assumes** $a$: $summable\ (\lambda k.\ norm\ (a\ k))$
  **assumes** $b$: $summable\ (\lambda k.\ norm\ (b\ k))$
  **shows** $(\sum k.\ a\ k) * (\sum k.\ b\ k) = (\sum k.\ \sum i=0..k.\ a\ i * b\ (k - i))$
$\langle proof \rangle$

**end**

## 18 EvenOdd: Even and Odd Numbers: Compatibility file for Parity

**theory** *EvenOdd*
**imports** *NthRoot*
**begin**

### 18.1 General Lemmas About Division

**lemma** *Suc-times-mod-eq*: $1<k ==> Suc\ (k * m)\ mod\ k = 1$
$\langle proof \rangle$

**declare** *Suc-times-mod-eq* [*of number-of w*, *standard*, *simp*]

**lemma** [*simp*]: $n\ div\ k \le (Suc\ n)\ div\ k$
$\langle proof \rangle$

**lemma** *Suc-n-div-2-gt-zero* [*simp*]: $(0::nat) < n ==> 0 < (n + 1)\ div\ 2$
$\langle proof \rangle$

**lemma** *div-2-gt-zero* [*simp*]: $(1::nat) < n ==> 0 < n\ div\ 2$
$\langle proof \rangle$

**lemma** *mod-mult-self3* [*simp*]: $(k*n + m)\ mod\ n = m\ mod\ (n::nat)$
$\langle proof \rangle$

**lemma** *mod-mult-self4* [*simp*]: $Suc\ (k*n + m)\ mod\ n = Suc\ m\ mod\ n$
⟨*proof*⟩

**lemma** *mod-Suc-eq-Suc-mod*: $Suc\ m\ mod\ n = Suc\ (m\ mod\ n)\ mod\ n$
⟨*proof*⟩

## 18.2 More Even/Odd Results

**lemma** *even-mult-two-ex*: $even(n) = (\exists\, m::nat.\ n = 2*m)$
⟨*proof*⟩

**lemma** *odd-Suc-mult-two-ex*: $odd(n) = (\exists\, m.\ n = Suc\ (2*m))$
⟨*proof*⟩

**lemma** *even-add* [*simp*]: $even(m + n::nat) = (even\ m = even\ n)$
⟨*proof*⟩

**lemma** *odd-add* [*simp*]: $odd(m + n::nat) = (odd\ m \neq odd\ n)$
⟨*proof*⟩

**lemma** *lemma-even-div2* [*simp*]: $even\ (n::nat) ==> (n + 1)\ div\ 2 = n\ div\ 2$
⟨*proof*⟩

**lemma** *lemma-not-even-div2* [*simp*]: $\sim even\ n ==> (n + 1)\ div\ 2 = Suc\ (n\ div\ 2)$
⟨*proof*⟩

**lemma** *even-num-iff*: $0 < n ==> even\ n = (\sim\ even(n - 1 :: nat))$
⟨*proof*⟩

**lemma** *even-even-mod-4-iff*: $even\ (n::nat) = even\ (n\ mod\ 4)$
⟨*proof*⟩

**lemma** *lemma-odd-mod-4-div-2*: $n\ mod\ 4 = (3::nat) ==> odd((n - 1)\ div\ 2)$
⟨*proof*⟩

**lemma** *lemma-even-mod-4-div-2*: $n\ mod\ 4 = (1::nat) ==> even\ ((n - 1)\ div\ 2)$
⟨*proof*⟩

**end**

# 19 Transcendental: Power Series, Transcendental Functions etc.

**theory** *Transcendental*
**imports** *NthRoot Fact Series EvenOdd Deriv*

**begin**

## 19.1 Properties of Power Series

**lemma** *lemma-realpow-diff*:
**fixes** $y :: {'a}{::}recpower$
**shows** $p \le n \Longrightarrow y \,\hat{}\, (Suc\ n - p) = (y \,\hat{}\, (n - p)) * y$
⟨*proof*⟩

**lemma** *lemma-realpow-diff-sumr*:
**fixes** $y :: {'a}{::}\{recpower,comm\text{-}semiring\text{-}0\}$ **shows**
$(\sum p{=}0..{<}Suc\ n.\ (x \,\hat{}\, p) * y \,\hat{}\, (Suc\ n - p)) =$
$y * (\sum p{=}0..{<}Suc\ n.\ (x \,\hat{}\, p) * y \,\hat{}\, (n - p))$
⟨*proof*⟩

**lemma** *lemma-realpow-diff-sumr2*:
**fixes** $y :: {'a}{::}\{recpower,comm\text{-}ring\}$ **shows**
$x \,\hat{}\, (Suc\ n) - y \,\hat{}\, (Suc\ n) =$
$(x - y) * (\sum p{=}0..{<}Suc\ n.\ (x \,\hat{}\, p) * y \,\hat{}\, (n - p))$
⟨*proof*⟩

**lemma** *lemma-realpow-rev-sumr*:
$(\sum p{=}0..{<}Suc\ n.\ (x \,\hat{}\, p) * (y \,\hat{}\, (n - p))) =$
$(\sum p{=}0..{<}Suc\ n.\ (x \,\hat{}\, (n - p)) * (y \,\hat{}\, p))$
⟨*proof*⟩

Power series has a 'circle' of convergence, i.e. if it sums for $x$, then it sums absolutely for $z$ with $|z| < |x|$.

**lemma** *powser-insidea*:
**fixes** $x\ z :: {'a}{::}\{real\text{-}normed\text{-}field,banach,recpower\}$
**assumes** $1$: $summable\ (\lambda n.\ f\ n * x \,\hat{}\, n)$
**assumes** $2$: $norm\ z < norm\ x$
**shows** $summable\ (\lambda n.\ norm\ (f\ n * z \,\hat{}\, n))$
⟨*proof*⟩

**lemma** *powser-inside*:
**fixes** $f :: nat \Rightarrow {'a}{::}\{real\text{-}normed\text{-}field,banach,recpower\}$ **shows**
$[|\ summable\ (\%n.\ f(n) * (x \,\hat{}\, n));\ norm\ z < norm\ x\ |]$
$==> summable\ (\%n.\ f(n) * (z \,\hat{}\, n))$
⟨*proof*⟩

## 19.2 Term-by-Term Differentiability of Power Series

**definition**
$diffs :: (nat => {'a}{::}ring\text{-}1) => nat => {'a}$ **where**
$diffs\ c = (\%n.\ of\text{-}nat\ (Suc\ n) * c(Suc\ n))$

Lemma about distributing negation over it

**lemma** *diffs-minus*: $diffs\ (\%n.\ - c\ n) = (\%n.\ - diffs\ c\ n)$

⟨*proof*⟩

Show that we can shift the terms down one

**lemma** *lemma-diffs*:
$(\sum n=0..<n.\ (diffs\ c)(n) * (x\ \hat{}\ n)) =$
$(\sum n=0..<n.\ of\text{-}nat\ n * c(n) * (x\ \hat{}\ (n - Suc\ 0))) +$
$(of\text{-}nat\ n * c(n) * x\ \hat{}\ (n - Suc\ 0))$
⟨*proof*⟩

**lemma** *lemma-diffs2*:
$(\sum n=0..<n.\ of\text{-}nat\ n * c(n) * (x\ \hat{}\ (n - Suc\ 0))) =$
$(\sum n=0..<n.\ (diffs\ c)(n) * (x\ \hat{}\ n)) -$
$(of\text{-}nat\ n * c(n) * x\ \hat{}\ (n - Suc\ 0))$
⟨*proof*⟩

**lemma** *diffs-equiv*:
$summable\ (\%n.\ (diffs\ c)(n) * (x\ \hat{}\ n)) ==>$
$(\%n.\ of\text{-}nat\ n * c(n) * (x\ \hat{}\ (n - Suc\ 0)))\ sums$
$(\sum n.\ (diffs\ c)(n) * (x\ \hat{}\ n))$
⟨*proof*⟩

**lemma** *lemma-termdiff1*:
**fixes** $z :: 'a :: \{recpower,comm\text{-}ring\}$ **shows**
$(\sum p=0..<m.\ (((z + h)\ \hat{}\ (m - p)) * (z\ \hat{}\ p)) - (z\ \hat{}\ m)) =$
$(\sum p=0..<m.\ (z\ \hat{}\ p) * (((z + h)\ \hat{}\ (m - p)) - (z\ \hat{}\ (m - p))))$
⟨*proof*⟩

**lemma** *less-add-one*: $m < n ==> (\exists\, d.\ n = m + d + Suc\ 0)$
⟨*proof*⟩

**lemma** *sumdiff*: $a + b - (c + d) = a - c + b - (d::real)$
⟨*proof*⟩

**lemma** *sumr-diff-mult-const2*:
$setsum\ f\ \{0..<n\} - of\text{-}nat\ n * (r::'a::ring\text{-}1) = (\sum i = 0..<n.\ f\ i - r)$
⟨*proof*⟩

**lemma** *lemma-termdiff2*:
**fixes** $h :: 'a :: \{recpower,field\}$
**assumes** $h$: $h \neq 0$ **shows**
$((z + h)\ \hat{}\ n - z\ \hat{}\ n) / h - of\text{-}nat\ n * z\ \hat{}\ (n - Suc\ 0) =$
$h * (\sum p=0..< n - Suc\ 0.\ \sum q=0..< n - Suc\ 0 - p.$
$(z + h)\ \hat{}\ q * z\ \hat{}\ (n - 2 - q))$ (**is** $?lhs = ?rhs$)
⟨*proof*⟩

**lemma** *real-setsum-nat-ivl-bounded2*:
**fixes** $K :: 'a::ordered\text{-}semidom$
**assumes** $f$: $\bigwedge p::nat.\ p < n \Longrightarrow f\ p \leq K$

**assumes** $K$: $0 \leq K$
**shows** *setsum f* $\{0..<n-k\} \leq$ *of-nat n* $* K$
⟨*proof*⟩

**lemma** *lemma-termdiff3*:
**fixes** $h\ z$ :: $'a$::{*real-normed-field*,*recpower*}
**assumes** $1$: $h \neq 0$
**assumes** $2$: *norm* $z \leq K$
**assumes** $3$: *norm* $(z + h) \leq K$
**shows** *norm* $(((z + h)\ \hat{}\ n - z\ \hat{}\ n) / h -$ *of-nat* $n * z\ \hat{}\ (n - Suc\ 0))$
$\leq$ *of-nat* $n *$ *of-nat* $(n - Suc\ 0) * K\ \hat{}\ (n - 2) *$ *norm* $h$
⟨*proof*⟩

**lemma** *lemma-termdiff4*:
**fixes** $f$ :: $'a$::{*real-normed-field*,*recpower*} $\Rightarrow$
$'b$::*real-normed-vector*
**assumes** $k$: $0 < (k$::*real*$)$
**assumes** *le*: $\bigwedge h.\ [\![h \neq 0;\ norm\ h < k]\!] \Longrightarrow norm\ (f\ h) \leq K * norm\ h$
**shows** $f$ −− $0$ −−> $0$
⟨*proof*⟩

**lemma** *lemma-termdiff5*:
**fixes** $g$ :: $'a$::{*recpower*,*real-normed-field*} $\Rightarrow$
*nat* $\Rightarrow$ $'b$::*banach*
**assumes** $k$: $0 < (k$::*real*$)$
**assumes** $f$: *summable f*
**assumes** *le*: $\bigwedge h\ n.\ [\![h \neq 0;\ norm\ h < k]\!] \Longrightarrow norm\ (g\ h\ n) \leq f\ n * norm\ h$
**shows** $(\lambda h.\ suminf\ (g\ h))$ −− $0$ −−> $0$
⟨*proof*⟩

FIXME: Long proofs

**lemma** *termdiffs-aux*:
**fixes** $x$ :: $'a$::{*recpower*,*real-normed-field*,*banach*}
**assumes** $1$: *summable* $(\lambda n.\ diffs\ (diffs\ c)\ n * K\ \hat{}\ n)$
**assumes** $2$: *norm* $x <$ *norm* $K$
**shows** $(\lambda h.\ \sum n.\ c\ n * (((x + h)\ \hat{}\ n - x\ \hat{}\ n) / h$
$-$ *of-nat* $n * x\ \hat{}\ (n - Suc\ 0)))$ −− $0$ −−> $0$
⟨*proof*⟩

**lemma** *termdiffs*:
**fixes** $K\ x$ :: $'a$::{*recpower*,*real-normed-field*,*banach*}
**assumes** $1$: *summable* $(\lambda n.\ c\ n * K\ \hat{}\ n)$
**assumes** $2$: *summable* $(\lambda n.\ (diffs\ c)\ n * K\ \hat{}\ n)$
**assumes** $3$: *summable* $(\lambda n.\ (diffs\ (diffs\ c))\ n * K\ \hat{}\ n)$
**assumes** $4$: *norm* $x <$ *norm* $K$
**shows** *DERIV* $(\lambda x.\ \sum n.\ c\ n * x\ \hat{}\ n)\ x$ :> $(\sum n.\ (diffs\ c)\ n * x\ \hat{}\ n)$
⟨*proof*⟩

## 19.3 Exponential Function

**definition**
*exp* :: $'a \Rightarrow 'a$::{*recpower*,*real-normed-field*,*banach*} **where**
*exp* $x = (\sum n.\ x\ \hat{}\ n\ /_R\ real\ (fact\ n))$

**definition**
*sin* :: *real* => *real* **where**
*sin* $x = (\sum n.$ (*if even*($n$) *then 0 else*
$(-1\ \hat{}\ ((n - Suc\ 0)\ div\ 2))/(real\ (fact\ n))) * x\ \hat{}\ n)$

**definition**
*cos* :: *real* => *real* **where**
*cos* $x = (\sum n.$ (*if even*($n$) *then* $(-1\ \hat{}\ (n\ div\ 2))/(real\ (fact\ n))$
*else* $0) * x\ \hat{}\ n)$

**lemma** *summable-exp-generic*:
**fixes** $x$ :: $'a$::{*real-normed-algebra-1*,*recpower*,*banach*}
**defines** *S-def*: $S \equiv \lambda n.\ x\ \hat{}\ n\ /_R\ real\ (fact\ n)$
**shows** *summable S*
⟨*proof*⟩

**lemma** *summable-norm-exp*:
**fixes** $x$ :: $'a$::{*real-normed-algebra-1*,*recpower*,*banach*}
**shows** *summable* $(\lambda n.\ norm\ (x\ \hat{}\ n\ /_R\ real\ (fact\ n)))$
⟨*proof*⟩

**lemma** *summable-exp*: *summable* (%$n$. *inverse* (*real* (*fact* $n$)) $* x\ \hat{}\ n$)
⟨*proof*⟩

**lemma** *summable-sin*:
*summable* (%$n$.
(*if even n then 0*
*else* $-1\ \hat{}\ ((n - Suc\ 0)\ div\ 2)/(real\ (fact\ n))) *$
$x\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *summable-cos*:
*summable* (%$n$.
(*if even n then*
$-1\ \hat{}\ (n\ div\ 2)/(real\ (fact\ n))$ *else* $0) * x\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *lemma-STAR-sin*:
(*if even n then 0*
*else* $-1\ \hat{}\ ((n - Suc\ 0)\ div\ 2)/(real\ (fact\ n))) * 0\ \hat{}\ n = 0$
⟨*proof*⟩

**lemma** *lemma-STAR-cos*:
$0 < n$ -->

$-1$ ^ $(n$ *div* $2)/($*real* $($*fact* $n)) * 0$ ^ $n = 0$
⟨*proof*⟩

**lemma** *lemma-STAR-cos1*:
$0 < n$ −−>
$(-1)$ ^ $(n$ *div* $2)/($*real* $($*fact* $n)) * 0$ ^ $n = 0$
⟨*proof*⟩

**lemma** *lemma-STAR-cos2*:
$(\sum n{=}1..{<}n.$ *if even* $n$ *then* $-1$ ^ $(n$ *div* $2)/($*real* $($*fact* $n)) * 0$ ^ $n$
*else* $0) = 0$
⟨*proof*⟩

**lemma** *exp-converges*: $(\lambda n.\ x$ ^ $n\ /_R$ *real* $($*fact* $n))$ *sums exp* $x$
⟨*proof*⟩

**lemma** *sin-converges*:
$(\%n.$ $($*if even* $n$ *then* $0$
*else* $-1$ ^ $((n - $ *Suc* $0)$ *div* $2)/($*real* $($*fact* $n))) *$
$x$ ^ $n)$ *sums* $sin(x)$
⟨*proof*⟩

**lemma** *cos-converges*:
$(\%n.$ $($*if even* $n$ *then*
$-1$ ^ $(n$ *div* $2)/($*real* $($*fact* $n))$
*else* $0) * x$ ^ $n)$ *sums* $cos(x)$
⟨*proof*⟩

## 19.4 Formal Derivatives of Exp, Sin, and Cos Series

**lemma** *exp-fdiffs*:
*diffs* $(\%n.$ *inverse*$($*real* $($*fact* $n))) = (\%n.$ *inverse*$($*real* $($*fact* $n)))$
⟨*proof*⟩

**lemma** *diffs-of-real*: *diffs* $(\lambda n.$ *of-real* $(f\ n)) = (\lambda n.$ *of-real* $($*diffs* $f\ n))$
⟨*proof*⟩

**lemma** *sin-fdiffs*:
*diffs*$(\%n.$ *if even* $n$ *then* $0$
*else* $-1$ ^ $((n -$ *Suc* $0)$ *div* $2)/($*real* $($*fact* $n)))$
$= (\%n.$ *if even* $n$ *then*
$-1$ ^ $(n$ *div* $2)/($*real* $($*fact* $n))$
*else* $0)$
⟨*proof*⟩

**lemma** *sin-fdiffs2*:
*diffs*$(\%n.$ *if even* $n$ *then* $0$
*else* $-1$ ^ $((n -$ *Suc* $0)$ *div* $2)/($*real* $($*fact* $n)))\ n$
$= ($*if even* $n$ *then*

$-1$ ^ $(n\ div\ 2)/(real\ (fact\ n))$
$else\ 0)$
⟨proof⟩

**lemma** *cos-fdiffs*:
$diffs(\%n.\ if\ even\ n\ then$
$-1$ ^ $(n\ div\ 2)/(real\ (fact\ n))\ else\ 0)$
$= (\%n.\ -\ (if\ even\ n\ then\ 0$
$else\ -1$ ^ $((n - Suc\ 0) div\ 2)/(real\ (fact\ n))))$
⟨proof⟩

**lemma** *cos-fdiffs2*:
$diffs(\%n.\ if\ even\ n\ then$
$-1$ ^ $(n\ div\ 2)/(real\ (fact\ n))\ else\ 0)\ n$
$= -\ (if\ even\ n\ then\ 0$
$else\ -1$ ^ $((n - Suc\ 0) div\ 2)/(real\ (fact\ n)))$
⟨proof⟩

Now at last we can get the derivatives of exp, sin and cos

**lemma** *lemma-sin-minus*:
$-\ sin\ x = (\sum n.\ -\ ((if\ even\ n\ then\ 0$
$else\ -1$ ^ $((n - Suc\ 0)\ div\ 2)/(real\ (fact\ n))) * x$ ^ $n))$
⟨proof⟩

**lemma** *lemma-exp-ext*: $exp = (\lambda x.\ \sum n.\ x$ ^ $n\ /_R\ real\ (fact\ n))$
⟨proof⟩

**lemma** *DERIV-exp* [*simp*]: $DERIV\ exp\ x :> exp(x)$
⟨proof⟩

**lemma** *lemma-sin-ext*:
$sin = (\%x.\ \sum n.$
$(if\ even\ n\ then\ 0$
$else\ -1$ ^ $((n - Suc\ 0)\ div\ 2)/(real\ (fact\ n))) *$
$x$ ^ $n)$
⟨proof⟩

**lemma** *lemma-cos-ext*:
$cos = (\%x.\ \sum n.$
$(if\ even\ n\ then\ -1$ ^ $(n\ div\ 2)/(real\ (fact\ n))\ else\ 0) *$
$x$ ^ $n)$
⟨proof⟩

**lemma** *DERIV-sin* [*simp*]: $DERIV\ sin\ x :> cos(x)$
⟨proof⟩

**lemma** *DERIV-cos* [*simp*]: $DERIV\ cos\ x :> -sin(x)$
⟨proof⟩

**lemma** *isCont-exp* [*simp*]: *isCont exp x*
⟨*proof*⟩

**lemma** *isCont-sin* [*simp*]: *isCont sin x*
⟨*proof*⟩

**lemma** *isCont-cos* [*simp*]: *isCont cos x*
⟨*proof*⟩

## 19.5 Properties of the Exponential Function

**lemma** *powser-zero*:
  **fixes** $f :: nat \Rightarrow 'a::\{real\text{-}normed\text{-}algebra\text{-}1,recpower\}$
  **shows** $(\sum n.\ f\ n * 0\ \hat{}\ n) = f\ 0$
⟨*proof*⟩

**lemma** *exp-zero* [*simp*]: $exp\ 0 = 1$
⟨*proof*⟩

**lemma** *setsum-head2*:
  $m \le n \Longrightarrow setsum\ f\ \{m..n\} = f\ m + setsum\ f\ \{Suc\ m..n\}$
⟨*proof*⟩

**lemma** *setsum-cl-ivl-Suc2*:
  $(\sum i{=}m..Suc\ n.\ f\ i) = (if\ Suc\ n < m\ then\ 0\ else\ f\ m + (\sum i{=}m..n.\ f\ (Suc\ i)))$
⟨*proof*⟩

**lemma** *exp-series-add*:
  **fixes** $x\ y :: 'a::\{real\text{-}field,recpower\}$
  **defines** *S-def*: $S \equiv \lambda x\ n.\ x\ \hat{}\ n\ /_R\ real\ (fact\ n)$
  **shows** $S\ (x + y)\ n = (\sum i{=}0..n.\ S\ x\ i * S\ y\ (n - i))$
⟨*proof*⟩

**lemma** *exp-add*: $exp\ (x + y) = exp\ x * exp\ y$
⟨*proof*⟩

**lemma** *exp-of-real*: $exp\ (of\text{-}real\ x) = of\text{-}real\ (exp\ x)$
⟨*proof*⟩

**lemma** *exp-ge-add-one-self-aux*: $0 \le (x::real) ==> (1 + x) \le exp(x)$
⟨*proof*⟩

**lemma** *exp-gt-one* [*simp*]: $0 < (x::real) ==> 1 < exp\ x$
⟨*proof*⟩

**lemma** *DERIV-exp-add-const*: $DERIV\ (\%x.\ exp\ (x + y))\ x :> exp(x + y)$
⟨*proof*⟩

**lemma** *DERIV-exp-minus* [*simp*]: *DERIV* (%*x*. *exp* $(-x)$) *x* :> $-$ *exp*$(-x)$
⟨*proof*⟩

**lemma** *DERIV-exp-exp-zero* [*simp*]: *DERIV* (%*x*. *exp* $(x + y) *$ *exp* $(- x)$) *x* :> *0*
⟨*proof*⟩

**lemma** *exp-add-mult-minus* [*simp*]: *exp*$(x + y)*$*exp*$(-x)$ = *exp*(*y*::*real*)
⟨*proof*⟩

**lemma** *exp-mult-minus* [*simp*]: *exp* $x *$ *exp*$(-x)$ = *1*
⟨*proof*⟩

**lemma** *exp-mult-minus2* [*simp*]: *exp*$(-x)*$*exp*$(x)$ = *1*
⟨*proof*⟩

**lemma** *exp-minus*: *exp*$(-x)$ = *inverse*(*exp*$(x)$)
⟨*proof*⟩

Proof: because every exponential can be seen as a square.

**lemma** *exp-ge-zero* [*simp*]: *0* $\leq$ *exp* (*x*::*real*)
⟨*proof*⟩

**lemma** *exp-not-eq-zero* [*simp*]: *exp* $x \neq$ *0*
⟨*proof*⟩

**lemma** *exp-gt-zero* [*simp*]: *0* < *exp* (*x*::*real*)
⟨*proof*⟩

**lemma** *inv-exp-gt-zero* [*simp*]: *0* < *inverse*(*exp* *x*::*real*)
⟨*proof*⟩

**lemma** *abs-exp-cancel* [*simp*]: |*exp* *x*::*real*| = *exp* *x*
⟨*proof*⟩

**lemma** *exp-real-of-nat-mult*: *exp*(*real* $n * x$) = *exp*$(x)$ ^ *n*
⟨*proof*⟩

**lemma** *exp-diff*: *exp*$(x - y)$ = *exp*$(x)$/(*exp* *y*)
⟨*proof*⟩

**lemma** *exp-less-mono*:
 **fixes** *x* *y* :: *real*
 **assumes** *xy*: $x < y$ **shows** *exp* *x* < *exp* *y*
⟨*proof*⟩

**lemma** *exp-less-cancel*: *exp* (*x*::*real*) < *exp* *y* ==> $x < y$

⟨*proof*⟩

**lemma** *exp-less-cancel-iff* [*iff*]: $(exp(x::real) < exp(y)) = (x < y)$
⟨*proof*⟩

**lemma** *exp-le-cancel-iff* [*iff*]: $(exp(x::real) \leq exp(y)) = (x \leq y)$
⟨*proof*⟩

**lemma** *exp-inj-iff* [*iff*]: $(exp\ (x::real) = exp\ y) = (x = y)$
⟨*proof*⟩

**lemma** *lemma-exp-total*: $1 \leq y ==> \exists x.\ 0 \leq x\ \&\ x \leq y - 1\ \&\ exp(x::real) = y$
⟨*proof*⟩

**lemma** *exp-total*: $0 < (y::real) ==> \exists x.\ exp\ x = y$
⟨*proof*⟩

## 19.6 Properties of the Logarithmic Function

**definition**
  $ln :: real => real$ **where**
  $ln\ x = (THE\ u.\ exp\ u = x)$

**lemma** *ln-exp* [*simp*]: $ln\ (exp\ x) = x$
⟨*proof*⟩

**lemma** *exp-ln* [*simp*]: $0 < x \Longrightarrow exp\ (ln\ x) = x$
⟨*proof*⟩

**lemma** *exp-ln-iff* [*simp*]: $(exp\ (ln\ x) = x) = (0 < x)$
⟨*proof*⟩

**lemma** *ln-mult*: $[|\ 0 < x;\ 0 < y\ |] ==> ln(x * y) = ln(x) + ln(y)$
⟨*proof*⟩

**lemma** *ln-inj-iff*[*simp*]: $[|\ 0 < x;\ 0 < y\ |] ==> (ln\ x = ln\ y) = (x = y)$
⟨*proof*⟩

**lemma** *ln-one*[*simp*]: $ln\ 1 = 0$
⟨*proof*⟩

**lemma** *ln-inverse*: $0 < x ==> ln(inverse\ x) = -\ ln\ x$
⟨*proof*⟩

**lemma** *ln-div*:
  $[|0 < x;\ 0 < y|] ==> ln(x/y) = ln\ x - ln\ y$
⟨*proof*⟩

**lemma** *ln-less-cancel-iff*[*simp*]: $[|\ 0 < x;\ 0 < y|] ==> (ln\ x < ln\ y) = (x < y)$

⟨*proof*⟩

**lemma** *ln-le-cancel-iff* [*simp*]: [| $0 < x$; $0 < y$|] ==> ($ln\ x \leq ln\ y$) = ($x \leq y$)
⟨*proof*⟩

**lemma** *ln-realpow*: $0 < x$ ==> $ln(x \hat{}\ n) = real\ n * ln(x)$
⟨*proof*⟩

**lemma** *ln-add-one-self-le-self* [*simp*]: $0 \leq x$ ==> $ln(1 + x) \leq x$
⟨*proof*⟩

**lemma** *ln-less-self* [*simp*]: $0 < x$ ==> $ln\ x < x$
⟨*proof*⟩

**lemma** *ln-ge-zero* [*simp*]:
  **assumes** *x*: $1 \leq x$ **shows** $0 \leq ln\ x$
⟨*proof*⟩

**lemma** *ln-ge-zero-imp-ge-one*:
  **assumes** *ln*: $0 \leq ln\ x$
    **and** *x*: $0 < x$
  **shows** $1 \leq x$
⟨*proof*⟩

**lemma** *ln-ge-zero-iff* [*simp*]: $0 < x$ ==> ($0 \leq ln\ x$) = ($1 \leq x$)
⟨*proof*⟩

**lemma** *ln-less-zero-iff* [*simp*]: $0 < x$ ==> ($ln\ x < 0$) = ($x < 1$)
⟨*proof*⟩

**lemma** *ln-gt-zero*:
  **assumes** *x*: $1 < x$ **shows** $0 < ln\ x$
⟨*proof*⟩

**lemma** *ln-gt-zero-imp-gt-one*:
  **assumes** *ln*: $0 < ln\ x$
    **and** *x*: $0 < x$
  **shows** $1 < x$
⟨*proof*⟩

**lemma** *ln-gt-zero-iff* [*simp*]: $0 < x$ ==> ($0 < ln\ x$) = ($1 < x$)
⟨*proof*⟩

**lemma** *ln-eq-zero-iff* [*simp*]: $0 < x$ ==> ($ln\ x = 0$) = ($x = 1$)
⟨*proof*⟩

**lemma** *ln-less-zero*: [| $0 < x$; $x < 1$ |] ==> $ln\ x < 0$
⟨*proof*⟩

**lemma** *exp-ln-eq*: $exp\ u = x ==> ln\ x = u$
⟨*proof*⟩

**lemma** *isCont-ln*: $0 < x \Longrightarrow isCont\ ln\ x$
⟨*proof*⟩

**lemma** *DERIV-ln*: $0 < x \Longrightarrow DERIV\ ln\ x :> inverse\ x$
⟨*proof*⟩

## 19.7 Basic Properties of the Trigonometric Functions

**lemma** *sin-zero* [*simp*]: $sin\ 0 = 0$
⟨*proof*⟩

**lemma** *cos-zero* [*simp*]: $cos\ 0 = 1$
⟨*proof*⟩

**lemma** *DERIV-sin-sin-mult* [*simp*]:
$DERIV\ (\%x.\ sin(x)*sin(x))\ x :> cos(x) * sin(x) + cos(x) * sin(x)$
⟨*proof*⟩

**lemma** *DERIV-sin-sin-mult2* [*simp*]:
$DERIV\ (\%x.\ sin(x)*sin(x))\ x :> 2 * cos(x) * sin(x)$
⟨*proof*⟩

**lemma** *DERIV-sin-realpow2* [*simp*]:
$DERIV\ (\%x.\ (sin\ x)^2)\ x :> cos(x) * sin(x) + cos(x) * sin(x)$
⟨*proof*⟩

**lemma** *DERIV-sin-realpow2a* [*simp*]:
$DERIV\ (\%x.\ (sin\ x)^2)\ x :> 2 * cos(x) * sin(x)$
⟨*proof*⟩

**lemma** *DERIV-cos-cos-mult* [*simp*]:
$DERIV\ (\%x.\ cos(x)*cos(x))\ x :> -sin(x) * cos(x) + -sin(x) * cos(x)$
⟨*proof*⟩

**lemma** *DERIV-cos-cos-mult2* [*simp*]:
$DERIV\ (\%x.\ cos(x)*cos(x))\ x :> -2 * cos(x) * sin(x)$
⟨*proof*⟩

**lemma** *DERIV-cos-realpow2* [*simp*]:
$DERIV\ (\%x.\ (cos\ x)^2)\ x :> -sin(x) * cos(x) + -sin(x) * cos(x)$
⟨*proof*⟩

**lemma** *DERIV-cos-realpow2a* [*simp*]:
$DERIV\ (\%x.\ (cos\ x)^2)\ x :> -2 * cos(x) * sin(x)$
⟨*proof*⟩

**lemma** *lemma-DERIV-subst*: $[| \; DERIV\, f\, x :> D;\; D = E \; |] ==> DERIV\, f\, x :> E$
⟨*proof*⟩

**lemma** *DERIV-cos-realpow2b*: $DERIV\ (\%x.\ (cos\ x)^2)\ x :> -(2 * cos(x) * sin(x))$
⟨*proof*⟩

**lemma** *DERIV-cos-cos-mult3* [*simp*]:
$DERIV\ (\%x.\ cos(x)*cos(x))\ x :> -(2 * cos(x) * sin(x))$
⟨*proof*⟩

**lemma** *DERIV-sin-circle-all*:
$\forall x.\ DERIV\ (\%x.\ (sin\ x)^2 + (cos\ x)^2)\ x :>$
$(2*cos(x)*sin(x) - 2*cos(x)*sin(x))$
⟨*proof*⟩

**lemma** *DERIV-sin-circle-all-zero* [*simp*]:
$\forall x.\ DERIV\ (\%x.\ (sin\ x)^2 + (cos\ x)^2)\ x :> 0$
⟨*proof*⟩

**lemma** *sin-cos-squared-add* [*simp*]: $((sin\ x)^2) + ((cos\ x)^2) = 1$
⟨*proof*⟩

**lemma** *sin-cos-squared-add2* [*simp*]: $((cos\ x)^2) + ((sin\ x)^2) = 1$
⟨*proof*⟩

**lemma** *sin-cos-squared-add3* [*simp*]: $cos\ x * cos\ x + sin\ x * sin\ x = 1$
⟨*proof*⟩

**lemma** *sin-squared-eq*: $(sin\ x)^2 = 1 - (cos\ x)^2$
⟨*proof*⟩

**lemma** *cos-squared-eq*: $(cos\ x)^2 = 1 - (sin\ x)^2$
⟨*proof*⟩

**lemma** *real-gt-one-ge-zero-add-less*: $[| \; 1 < x;\; 0 \le y \; |] ==> 1 < x + (y::real)$
⟨*proof*⟩

**lemma** *abs-sin-le-one* [*simp*]: $|sin\ x| \le 1$
⟨*proof*⟩

**lemma** *sin-ge-minus-one* [*simp*]: $-1 \le sin\ x$
⟨*proof*⟩

**lemma** *sin-le-one* [*simp*]: $sin\ x \le 1$
⟨*proof*⟩

**lemma** *abs-cos-le-one* [*simp*]: $|cos\ x| \le 1$

⟨*proof*⟩

**lemma** *cos-ge-minus-one* [*simp*]: $-1 \le cos\ x$
⟨*proof*⟩

**lemma** *cos-le-one* [*simp*]: $cos\ x \le 1$
⟨*proof*⟩

**lemma** *DERIV-fun-pow*: *DERIV g x :> m ==>*
*DERIV (%x. (g x) ^ n) x :> real n ∗ (g x) ^ (n − 1) ∗ m*
⟨*proof*⟩

**lemma** *DERIV-fun-exp*:
*DERIV g x :> m ==> DERIV (%x. exp(g x)) x :> exp(g x) ∗ m*
⟨*proof*⟩

**lemma** *DERIV-fun-sin*:
*DERIV g x :> m ==> DERIV (%x. sin(g x)) x :> cos(g x) ∗ m*
⟨*proof*⟩

**lemma** *DERIV-fun-cos*:
*DERIV g x :> m ==> DERIV (%x. cos(g x)) x :> −sin(g x) ∗ m*
⟨*proof*⟩

**lemmas** *DERIV-intros = DERIV-ident DERIV-const DERIV-cos DERIV-cmult*
*DERIV-sin DERIV-exp DERIV-inverse DERIV-pow*
*DERIV-add DERIV-diff DERIV-mult DERIV-minus*
*DERIV-inverse-fun DERIV-quotient DERIV-fun-pow*
*DERIV-fun-exp DERIV-fun-sin DERIV-fun-cos*

**lemma** *lemma-DERIV-sin-cos-add*:
*∀x.*
*DERIV (%x. (sin (x + y) − (sin x ∗ cos y + cos x ∗ sin y)) ^ 2 +*
*(cos (x + y) − (cos x ∗ cos y − sin x ∗ sin y)) ^ 2) x :> 0*
⟨*proof*⟩

**lemma** *sin-cos-add* [*simp*]:
*(sin (x + y) − (sin x ∗ cos y + cos x ∗ sin y)) ^ 2 +*
*(cos (x + y) − (cos x ∗ cos y − sin x ∗ sin y)) ^ 2 = 0*
⟨*proof*⟩

**lemma** *sin-add*: *sin (x + y) = sin x ∗ cos y + cos x ∗ sin y*
⟨*proof*⟩

**lemma** *cos-add*: *cos (x + y) = cos x ∗ cos y − sin x ∗ sin y*
⟨*proof*⟩

**lemma** *lemma-DERIV-sin-cos-minus*:

$\forall x.$ *DERIV* $(\%x.\ (sin(-x) + (sin\ x))\ \hat{}\ 2 + (cos(-x) - (cos\ x))\ \hat{}\ 2)\ x :> 0$
⟨*proof*⟩

**lemma** *sin-cos-minus* [*simp*]:
$(sin(-x) + (sin\ x))\ \hat{}\ 2 + (cos(-x) - (cos\ x))\ \hat{}\ 2 = 0$
⟨*proof*⟩

**lemma** *sin-minus* [*simp*]: $sin\ (-x) = -sin(x)$
⟨*proof*⟩

**lemma** *cos-minus* [*simp*]: $cos\ (-x) = cos(x)$
⟨*proof*⟩

**lemma** *sin-diff*: $sin\ (x - y) = sin\ x * cos\ y - cos\ x * sin\ y$
⟨*proof*⟩

**lemma** *sin-diff2*: $sin\ (x - y) = cos\ y * sin\ x - sin\ y * cos\ x$
⟨*proof*⟩

**lemma** *cos-diff*: $cos\ (x - y) = cos\ x * cos\ y + sin\ x * sin\ y$
⟨*proof*⟩

**lemma** *cos-diff2*: $cos\ (x - y) = cos\ y * cos\ x + sin\ y * sin\ x$
⟨*proof*⟩

**lemma** *sin-double* [*simp*]: $sin(2 * x) = 2 *\ sin\ x * cos\ x$
⟨*proof*⟩

**lemma** *cos-double*: $cos(2 * x) = ((cos\ x)^2) - ((sin\ x)^2)$
⟨*proof*⟩

## 19.8 The Constant Pi

**definition**
*pi* :: *real* **where**
$pi = 2 * (THE\ x.\ 0 \le (x{::}real)\ \&\ x \le 2\ \&\ cos\ x = 0)$

Show that there's a least positive $x$ with $cos\ x = 0$; hence define pi.

**lemma** *sin-paired*:
$(\%n.\ -1\ \hat{}\ n\ /(real\ (fact\ (2 * n + 1))) * x\ \hat{}\ (2 * n + 1))$
*sums* $sin\ x$
⟨*proof*⟩

**lemma** *sin-gt-zero*: $[|0 < x;\ x < 2\ |] ==> 0 < sin\ x$
⟨*proof*⟩

**lemma** *sin-gt-zero1*: $[|0 < x;\ x < 2\ |] ==> 0 < sin\ x$
⟨*proof*⟩

**lemma** *cos-double-less-one*: [| $0 < x$; $x < 2$ |] ==> $cos\ (2 * x) < 1$
⟨*proof*⟩

**lemma** *cos-paired*:
(%$n$. $-1$ ^ $n$ /(*real* (*fact* ($2 * n$))) $*$ $x$ ^ ($2 * n$)) *sums cos x*
⟨*proof*⟩

**declare** *zero-less-power* [*simp*]

**lemma** *fact-lemma*: *real* ($n$::*nat*) $* 4$ = *real* ($4 * n$)
⟨*proof*⟩

**lemma** *cos-two-less-zero* [*simp*]: $cos\ (2) < 0$
⟨*proof*⟩

**lemmas** *cos-two-neq-zero* [*simp*] = *cos-two-less-zero* [*THEN less-imp-neq*]
**lemmas** *cos-two-le-zero* [*simp*] = *cos-two-less-zero* [*THEN order-less-imp-le*]

**lemma** *cos-is-zero*: *EX*! $x$. $0 \leq x$ & $x \leq 2$ & $cos\ x = 0$
⟨*proof*⟩

**lemma** *pi-half*: $pi/2$ = (*THE* $x$. $0 \leq x$ & $x \leq 2$ & $cos\ x = 0$)
⟨*proof*⟩

**lemma** *cos-pi-half* [*simp*]: $cos\ (pi\ /\ 2) = 0$
⟨*proof*⟩

**lemma** *pi-half-gt-zero* [*simp*]: $0 < pi\ /\ 2$
⟨*proof*⟩

**lemmas** *pi-half-neq-zero* [*simp*] = *pi-half-gt-zero* [*THEN less-imp-neq*, *symmetric*]
**lemmas** *pi-half-ge-zero* [*simp*] = *pi-half-gt-zero* [*THEN order-less-imp-le*]

**lemma** *pi-half-less-two* [*simp*]: $pi\ /\ 2 < 2$
⟨*proof*⟩

**lemmas** *pi-half-neq-two* [*simp*] = *pi-half-less-two* [*THEN less-imp-neq*]
**lemmas** *pi-half-le-two* [*simp*] = *pi-half-less-two* [*THEN order-less-imp-le*]

**lemma** *pi-gt-zero* [*simp*]: $0 < pi$
⟨*proof*⟩

**lemma** *pi-ge-zero* [*simp*]: $0 \leq pi$
⟨*proof*⟩

**lemma** *pi-neq-zero* [*simp*]: $pi \neq 0$
⟨*proof*⟩

**lemma** *pi-not-less-zero* [*simp*]: $\neg\ pi < 0$
⟨*proof*⟩

**lemma** *minus-pi-half-less-zero* [*simp*]: $-(pi/2) < 0$
⟨*proof*⟩

**lemma** *sin-pi-half* [*simp*]: $sin(pi/2) = 1$
⟨*proof*⟩

**lemma** *cos-pi* [*simp*]: $cos\ pi = -1$
⟨*proof*⟩

**lemma** *sin-pi* [*simp*]: $sin\ pi = 0$
⟨*proof*⟩

**lemma** *sin-cos-eq*: $sin\ x = cos\ (pi/2 - x)$
⟨*proof*⟩
**declare** *sin-cos-eq* [*symmetric*, *simp*]

**lemma** *minus-sin-cos-eq*: $-sin\ x = cos\ (x + pi/2)$
⟨*proof*⟩
**declare** *minus-sin-cos-eq* [*symmetric*, *simp*]

**lemma** *cos-sin-eq*: $cos\ x = sin\ (pi/2 - x)$
⟨*proof*⟩
**declare** *cos-sin-eq* [*symmetric*, *simp*]

**lemma** *sin-periodic-pi* [*simp*]: $sin\ (x + pi) = -\ sin\ x$
⟨*proof*⟩

**lemma** *sin-periodic-pi2* [*simp*]: $sin\ (pi + x) = -\ sin\ x$
⟨*proof*⟩

**lemma** *cos-periodic-pi* [*simp*]: $cos\ (x + pi) = -\ cos\ x$
⟨*proof*⟩

**lemma** *sin-periodic* [*simp*]: $sin\ (x + 2{*}pi) = sin\ x$
⟨*proof*⟩

**lemma** *cos-periodic* [*simp*]: $cos\ (x + 2{*}pi) = cos\ x$
⟨*proof*⟩

**lemma** *cos-npi* [*simp*]: $cos\ (real\ n * pi) = -1\ \hat{}\ n$
⟨*proof*⟩

**lemma** *cos-npi2* [*simp*]: $cos\ (pi * real\ n) = -1\ \hat{}\ n$
⟨*proof*⟩

**lemma** *sin-npi* [*simp*]: $sin\ (real\ (n{::}nat) * pi) = 0$

⟨*proof*⟩

**lemma** *sin-npi2* [*simp*]: $sin\ (pi * real\ (n::nat)) = 0$
⟨*proof*⟩

**lemma** *cos-two-pi* [*simp*]: $cos\ (2 * pi) = 1$
⟨*proof*⟩

**lemma** *sin-two-pi* [*simp*]: $sin\ (2 * pi) = 0$
⟨*proof*⟩

**lemma** *sin-gt-zero2*: $[|\ 0 < x;\ x < pi/2\ |] ==> 0 < sin\ x$
⟨*proof*⟩

**lemma** *sin-less-zero*:
 **assumes** *lb*: $-\ pi/2 < x$ **and** $x < 0$ **shows** $sin\ x < 0$
⟨*proof*⟩

**lemma** *pi-less-4*: $pi < 4$
⟨*proof*⟩

**lemma** *cos-gt-zero*: $[|\ 0 < x;\ x < pi/2\ |] ==> 0 < cos\ x$
⟨*proof*⟩

**lemma** *cos-gt-zero-pi*: $[|\ -(pi/2) < x;\ x < pi/2\ |] ==> 0 < cos\ x$
⟨*proof*⟩

**lemma** *cos-ge-zero*: $[|\ -(pi/2) \leq x;\ x \leq pi/2\ |] ==> 0 \leq cos\ x$
⟨*proof*⟩

**lemma** *sin-gt-zero-pi*: $[|\ 0 < x;\ x < pi\ \ |] ==> 0 < sin\ x$
⟨*proof*⟩

**lemma** *sin-ge-zero*: $[|\ 0 \leq x;\ x \leq pi\ |] ==> 0 \leq sin\ x$
⟨*proof*⟩

**lemma** *cos-total*: $[|\ -1 \leq y;\ y \leq 1\ |] ==> EX!\ x.\ 0 \leq x\ \&\ x \leq pi\ \&\ (cos\ x = y)$
⟨*proof*⟩

**lemma** *sin-total*:
 $[|\ -1 \leq y;\ y \leq 1\ |] ==> EX!\ x.\ -(pi/2) \leq x\ \&\ x \leq pi/2\ \&\ (sin\ x = y)$
⟨*proof*⟩

**lemma** *reals-Archimedean4*:
 $[|\ 0 < y;\ 0 \leq x\ |] ==> \exists\, n.\ real\ n * y \leq x\ \&\ x < real\ (Suc\ n) * y$
⟨*proof*⟩

**lemma** *cos-zero-lemma*:
  $[|\ 0 \le x;\ cos\ x = 0\ |] ==>$
  $\exists\, n{::}nat.\ {}^{\sim}even\ n\ \&\ x = real\ n * (pi/2)$
⟨*proof*⟩

**lemma** *sin-zero-lemma*:
  $[|\ 0 \le x;\ sin\ x = 0\ |] ==>$
  $\exists\, n{::}nat.\ even\ n\ \&\ x = real\ n * (pi/2)$
⟨*proof*⟩

**lemma** *cos-zero-iff*:
  $(cos\ x = 0) =$
  $((\exists\, n{::}nat.\ {}^{\sim}even\ n\ \&\ (x = real\ n * (pi/2)))\ |$
  $(\exists\, n{::}nat.\ {}^{\sim}even\ n\ \&\ (x = -(real\ n * (pi/2)))))$
⟨*proof*⟩

**lemma** *sin-zero-iff*:
  $(sin\ x = 0) =$
  $((\exists\, n{::}nat.\ even\ n\ \&\ (x = real\ n * (pi/2)))\ |$
  $(\exists\, n{::}nat.\ even\ n\ \&\ (x = -(real\ n * (pi/2)))))$
⟨*proof*⟩

## 19.9 Tangent

**definition**
  *tan* :: *real* => *real* **where**
  $tan\ x = (sin\ x)/(cos\ x)$

**lemma** *tan-zero* [*simp*]: $tan\ 0 = 0$
⟨*proof*⟩

**lemma** *tan-pi* [*simp*]: $tan\ pi = 0$
⟨*proof*⟩

**lemma** *tan-npi* [*simp*]: $tan\ (real\ (n{::}nat) * pi) = 0$
⟨*proof*⟩

**lemma** *tan-minus* [*simp*]: $tan\ (-x) = -\ tan\ x$
⟨*proof*⟩

**lemma** *tan-periodic* [*simp*]: $tan\ (x + 2{*}pi) = tan\ x$
⟨*proof*⟩

**lemma** *lemma-tan-add1*:
  $[|\ cos\ x \neq 0;\ cos\ y \neq 0\ |]$
  $==> 1 - tan(x){*}tan(y) = cos\ (x + y)/(cos\ x * cos\ y)$
⟨*proof*⟩

**lemma** *add-tan-eq*:
$[| \; cos\ x \neq 0;\ cos\ y \neq 0 \; |]$
$==> tan\ x + tan\ y = sin(x + y)/(cos\ x * cos\ y)$
⟨*proof*⟩

**lemma** *tan-add*:
$[| \; cos\ x \neq 0;\ cos\ y \neq 0;\ cos\ (x + y) \neq 0 \; |]$
$==> tan(x + y) = (tan(x) + tan(y))/(1 - tan(x) * tan(y))$
⟨*proof*⟩

**lemma** *tan-double*:
$[| \; cos\ x \neq 0;\ cos\ (2 * x) \neq 0 \; |]$
$==> tan\ (2 * x) = (2 * tan\ x)/(1 - (tan(x)\ \hat{}\ 2))$
⟨*proof*⟩

**lemma** *tan-gt-zero*: $[| \; 0 < x;\ x < pi/2 \; |] ==> 0 < tan\ x$
⟨*proof*⟩

**lemma** *tan-less-zero*:
**assumes** *lb*: $- pi/2 < x$ **and** $x < 0$ **shows** $tan\ x < 0$
⟨*proof*⟩

**lemma** *lemma-DERIV-tan*:
$cos\ x \neq 0 ==> DERIV\ (\%x.\ sin(x)/cos(x))\ x :> inverse((cos\ x)^2)$
⟨*proof*⟩

**lemma** *DERIV-tan* [*simp*]: $cos\ x \neq 0 ==> DERIV\ tan\ x :> inverse((cos\ x)^2)$
⟨*proof*⟩

**lemma** *isCont-tan* [*simp*]: $cos\ x \neq 0 ==> isCont\ tan\ x$
⟨*proof*⟩

**lemma** *LIM-cos-div-sin* [*simp*]: $(\%x.\ cos(x)/sin(x))\ --\ pi/2\ --> 0$
⟨*proof*⟩

**lemma** *lemma-tan-total*: $0 < y ==> \exists x.\ 0 < x\ \&\ x < pi/2\ \&\ y < tan\ x$
⟨*proof*⟩

**lemma** *tan-total-pos*: $0 \leq y ==> \exists x.\ 0 \leq x\ \&\ x < pi/2\ \&\ tan\ x = y$
⟨*proof*⟩

**lemma** *lemma-tan-total1*: $\exists x.\ -(pi/2) < x\ \&\ x < (pi/2)\ \&\ tan\ x = y$
⟨*proof*⟩

**lemma** *tan-total*: $EX!\ x.\ -(pi/2) < x\ \&\ x < (pi/2)\ \&\ tan\ x = y$
⟨*proof*⟩

## 19.10 Inverse Trigonometric Functions

**definition**
*arcsin* :: *real* => *real* **where**
$arcsin\ y = (THE\ x.\ -(pi/2) \leq x\ \&\ x \leq pi/2\ \&\ sin\ x = y)$

**definition**
*arccos* :: *real* => *real* **where**
$arccos\ y = (THE\ x.\ 0 \leq x\ \&\ x \leq pi\ \&\ cos\ x = y)$

**definition**
*arctan* :: *real* => *real* **where**
$arctan\ y = (THE\ x.\ -(pi/2) < x\ \&\ x < pi/2\ \&\ tan\ x = y)$

**lemma** *arcsin*:
$[|\ -1 \leq y;\ y \leq 1\ |]$
$==> -(pi/2) \leq arcsin\ y\ \&$
$arcsin\ y \leq pi/2\ \&\ sin(arcsin\ y) = y$
⟨*proof*⟩

**lemma** *arcsin-pi*:
$[|\ -1 \leq y;\ y \leq 1\ |]$
$==> -(pi/2) \leq arcsin\ y\ \&\ arcsin\ y \leq pi\ \&\ sin(arcsin\ y) = y$
⟨*proof*⟩

**lemma** *sin-arcsin* [*simp*]: $[|\ -1 \leq y;\ y \leq 1\ |] ==> sin(arcsin\ y) = y$
⟨*proof*⟩

**lemma** *arcsin-bounded*:
$[|\ -1 \leq y;\ y \leq 1\ |] ==> -(pi/2) \leq arcsin\ y\ \&\ arcsin\ y \leq pi/2$
⟨*proof*⟩

**lemma** *arcsin-lbound*: $[|\ -1 \leq y;\ y \leq 1\ |] ==> -(pi/2) \leq arcsin\ y$
⟨*proof*⟩

**lemma** *arcsin-ubound*: $[|\ -1 \leq y;\ y \leq 1\ |] ==> arcsin\ y \leq pi/2$
⟨*proof*⟩

**lemma** *arcsin-lt-bounded*:
$[|\ -1 < y;\ y < 1\ |] ==> -(pi/2) < arcsin\ y\ \&\ arcsin\ y < pi/2$
⟨*proof*⟩

**lemma** *arcsin-sin*: $[|-(pi/2) \leq x;\ x \leq pi/2\ |] ==> arcsin(sin\ x) = x$
⟨*proof*⟩

**lemma** *arccos*:
$[|\ -1 \leq y;\ y \leq 1\ |]$
$==> 0 \leq arccos\ y\ \&\ arccos\ y \leq pi\ \&\ cos(arccos\ y) = y$
⟨*proof*⟩

**lemma** *cos-arccos* [*simp*]: $[| -1 \le y;\ y \le 1\ |] ==> cos(arccos\ y) = y$
⟨*proof*⟩

**lemma** *arccos-bounded*: $[| -1 \le y;\ y \le 1\ |] ==> 0 \le arccos\ y\ \&\ arccos\ y \le pi$
⟨*proof*⟩

**lemma** *arccos-lbound*: $[| -1 \le y;\ y \le 1\ |] ==> 0 \le arccos\ y$
⟨*proof*⟩

**lemma** *arccos-ubound*: $[| -1 \le y;\ y \le 1\ |] ==> arccos\ y \le pi$
⟨*proof*⟩

**lemma** *arccos-lt-bounded*:
$[| -1 < y;\ y < 1\ |]$
$==> 0 < arccos\ y\ \&\ arccos\ y < pi$
⟨*proof*⟩

**lemma** *arccos-cos*: $[|0 \le x;\ x \le pi\ |] ==> arccos(cos\ x) = x$
⟨*proof*⟩

**lemma** *arccos-cos2*: $[|x \le 0;\ -pi \le x\ |] ==> arccos(cos\ x) = -x$
⟨*proof*⟩

**lemma** *cos-arcsin*: $[\![-1 \le x;\ x \le 1]\!] \Longrightarrow cos\ (arcsin\ x) = sqrt\ (1 - x^2)$
⟨*proof*⟩

**lemma** *sin-arccos*: $[\![-1 \le x;\ x \le 1]\!] \Longrightarrow sin\ (arccos\ x) = sqrt\ (1 - x^2)$
⟨*proof*⟩

**lemma** *arctan* [*simp*]:
$-\ (pi/2) < arctan\ y\ \ \&\ arctan\ y < pi/2\ \&\ tan\ (arctan\ y) = y$
⟨*proof*⟩

**lemma** *tan-arctan*: $tan(arctan\ y) = y$
⟨*proof*⟩

**lemma** *arctan-bounded*: $-\ (pi/2) < arctan\ y\ \ \&\ arctan\ y < pi/2$
⟨*proof*⟩

**lemma** *arctan-lbound*: $-\ (pi/2) < arctan\ y$
⟨*proof*⟩

**lemma** *arctan-ubound*: $arctan\ y < pi/2$
⟨*proof*⟩

**lemma** *arctan-tan*:
$[|-(pi/2) < x;\ x < pi/2\ |] ==> arctan(tan\ x) = x$
⟨*proof*⟩

**lemma** *arctan-zero-zero* [*simp*]: $arctan\ 0 = 0$
⟨*proof*⟩

**lemma** *cos-arctan-not-zero* [*simp*]: $cos(arctan\ x) \neq 0$
⟨*proof*⟩

**lemma** *tan-sec*: $cos\ x \neq 0 ==> 1 + tan(x)\ \hat{}\ 2 = inverse(cos\ x)\ \hat{}\ 2$
⟨*proof*⟩

**lemma** *isCont-inverse-function2*:
**fixes** $f\ g :: real \Rightarrow real$ **shows**
$[\![a < x;\ x < b;$
$\forall z.\ a \leq z \wedge z \leq b \longrightarrow g\ (f\ z) = z;$
$\forall z.\ a \leq z \wedge z \leq b \longrightarrow isCont\ f\ z]\!]$
$\Longrightarrow isCont\ g\ (f\ x)$
⟨*proof*⟩

**lemma** *isCont-arcsin*: $[\![-1 < x;\ x < 1]\!] \Longrightarrow isCont\ arcsin\ x$
⟨*proof*⟩

**lemma** *isCont-arccos*: $[\![-1 < x;\ x < 1]\!] \Longrightarrow isCont\ arccos\ x$
⟨*proof*⟩

**lemma** *isCont-arctan*: $isCont\ arctan\ x$
⟨*proof*⟩

**lemma** *DERIV-arcsin*:
$[\![-1 < x;\ x < 1]\!] \Longrightarrow DERIV\ arcsin\ x :> inverse\ (sqrt\ (1 - x^2))$
⟨*proof*⟩

**lemma** *DERIV-arccos*:
$[\![-1 < x;\ x < 1]\!] \Longrightarrow DERIV\ arccos\ x :> inverse\ (-\ sqrt\ (1 - x^2))$
⟨*proof*⟩

**lemma** *DERIV-arctan*: $DERIV\ arctan\ x :> inverse\ (1 + x^2)$
⟨*proof*⟩

## 19.11 More Theorems about Sin and Cos

**lemma** *cos-45*: $cos\ (pi\ /\ 4) = sqrt\ 2\ /\ 2$
⟨*proof*⟩

**lemma** *cos-30*: $cos\ (pi\ /\ 6) = sqrt\ 3\ /\ 2$
⟨*proof*⟩

**lemma** *sin-45*: $sin\ (pi\ /\ 4) = sqrt\ 2\ /\ 2$
⟨*proof*⟩

**lemma** *sin-60*: $sin\ (pi\ /\ 3) = sqrt\ 3\ /\ 2$

⟨*proof*⟩

**lemma** *cos-60*: *cos (pi / 3) = 1 / 2*
⟨*proof*⟩

**lemma** *sin-30*: *sin (pi / 6) = 1 / 2*
⟨*proof*⟩

**lemma** *tan-30*: *tan (pi / 6) = 1 / sqrt 3*
⟨*proof*⟩

**lemma** *tan-45*: *tan (pi / 4) = 1*
⟨*proof*⟩

**lemma** *tan-60*: *tan (pi / 3) = sqrt 3*
⟨*proof*⟩

NEEDED??

**lemma** [*simp*]:
*sin (x + 1 / 2 * real (Suc m) * pi) =*
*cos (x + 1 / 2 * real (m) * pi)*
⟨*proof*⟩

NEEDED??

**lemma** [*simp*]:
*sin (x + real (Suc m) * pi / 2) =*
*cos (x + real (m) * pi / 2)*
⟨*proof*⟩

**lemma** *DERIV-sin-add* [*simp*]: *DERIV (%x. sin (x + k)) xa :> cos (xa + k)*
⟨*proof*⟩

**lemma** *sin-cos-npi* [*simp*]: *sin (real (Suc (2 * n)) * pi / 2) = (−1) ^ n*
⟨*proof*⟩

**lemma** *cos-2npi* [*simp*]: *cos (2 * real (n::nat) * pi) = 1*
⟨*proof*⟩

**lemma** *cos-3over2-pi* [*simp*]: *cos (3 / 2 * pi) = 0*
⟨*proof*⟩

**lemma** *sin-2npi* [*simp*]: *sin (2 * real (n::nat) * pi) = 0*
⟨*proof*⟩

**lemma** *sin-3over2-pi* [*simp*]: *sin (3 / 2 * pi) = − 1*
⟨*proof*⟩

**lemma** [*simp*]:

$cos(x + 1 / 2 * real(Suc\ m) * pi) = -sin\ (x + 1 / 2 * real\ m * pi)$
⟨*proof*⟩

**lemma** [*simp*]: $cos\ (x + real(Suc\ m) * pi / 2) = -sin\ (x + real\ m * pi / 2)$
⟨*proof*⟩

**lemma** *cos-pi-eq-zero* [*simp*]: $cos\ (pi * real\ (Suc\ (2 * m)) / 2) = 0$
⟨*proof*⟩

**lemma** *DERIV-cos-add* [*simp*]: $DERIV\ (\%x.\ cos\ (x + k))\ xa :> -\ sin\ (xa + k)$
⟨*proof*⟩

**lemma** *sin-zero-abs-cos-one*: $sin\ x = 0 ==> |cos\ x| = 1$
⟨*proof*⟩

**lemma** *exp-eq-one-iff* [*simp*]: $(exp\ (x::real) = 1) = (x = 0)$
⟨*proof*⟩

**lemma** *cos-one-sin-zero*: $cos\ x = 1 ==> sin\ x = 0$
⟨*proof*⟩

## 19.12 Existence of Polar Coordinates

**lemma** *cos-x-y-le-one*: $|x / sqrt\ (x^2 + y^2)| \leq 1$
⟨*proof*⟩

**lemma** *cos-arccos-abs*: $|y| \leq 1 \Longrightarrow cos\ (arccos\ y) = y$
⟨*proof*⟩

**lemma** *sin-arccos-abs*: $|y| \leq 1 \Longrightarrow sin\ (arccos\ y) = sqrt\ (1 - y^2)$
⟨*proof*⟩

**lemmas** *cos-arccos-lemma1* = *cos-arccos-abs* [*OF cos-x-y-le-one*]

**lemmas** *sin-arccos-lemma1* = *sin-arccos-abs* [*OF cos-x-y-le-one*]

**lemma** *polar-ex1*:
$0 < y ==> \exists\, r\ a.\ x = r * cos\ a\ \&\ y = r * sin\ a$
⟨*proof*⟩

**lemma** *polar-ex2*:
$y < 0 ==> \exists\, r\ a.\ x = r * cos\ a\ \&\ y = r * sin\ a$
⟨*proof*⟩

**lemma** *polar-Ex*: $\exists\, r\ a.\ x = r * cos\ a\ \&\ y = r * sin\ a$
⟨*proof*⟩

### 19.13 Theorems about Limits

**lemma** *isCont-inv-fun*:
**fixes** $f\ g$ :: $real \Rightarrow real$
**shows** $[\![\ 0 < d; \forall z. |z - x| \le d \ {-}{-}{>}\ g(f(z)) = z;$
$\forall z. |z - x| \le d \ {-}{-}{>}\ isCont\ f\ z\ ]\!]$
$==> isCont\ g\ (f\ x)$
⟨*proof*⟩

**lemma** *isCont-inv-fun-inv*:
**fixes** $f\ g$ :: $real \Rightarrow real$
**shows** $[\![\ 0 < d;$
$\forall z. |z - x| \le d \ {-}{-}{>}\ g(f(z)) = z;$
$\forall z. |z - x| \le d \ {-}{-}{>}\ isCont\ f\ z\ ]\!]$
$==> \exists e.\ 0 < e\ \&$
$(\forall y.\ 0 < |y - f(x)|\ \&\ |y - f(x)| < e \ {-}{-}{>}\ f(g(y)) = y)$
⟨*proof*⟩

Bartle/Sherbert: Introduction to Real Analysis, Theorem 4.2.9, p. 110

**lemma** *LIM-fun-gt-zero*:
$[\![\ f\ {-}{-}\ c\ {-}{-}{>}\ (l{::}real);\ 0 < l\ ]\!]$
$==> \exists r.\ 0 < r\ \&\ (\forall x{::}real.\ x \ne c\ \&\ |c - x| < r \ {-}{-}{>}\ 0 < f\ x)$
⟨*proof*⟩

**lemma** *LIM-fun-less-zero*:
$[\![\ f\ {-}{-}\ c\ {-}{-}{>}\ (l{::}real);\ l < 0\ ]\!]$
$==> \exists r.\ 0 < r\ \&\ (\forall x{::}real.\ x \ne c\ \&\ |c - x| < r \ {-}{-}{>}\ f\ x < 0)$
⟨*proof*⟩

**lemma** *LIM-fun-not-zero*:
$[\![\ f\ {-}{-}\ c\ {-}{-}{>}\ (l{::}real);\ l \ne 0\ ]\!]$
$==> \exists r.\ 0 < r\ \&\ (\forall x{::}real.\ x \ne c\ \&\ |c - x| < r \ {-}{-}{>}\ f\ x \ne 0)$
⟨*proof*⟩

**end**

## 20 Complex: Complex Numbers: Rectangular and Polar Representations

**theory** *Complex*
**imports** *../Hyperreal/Transcendental*
**begin**

**datatype** *complex* = *Complex real real*

**consts** *Re* :: *complex* ⇒ *real*
**primrec** *Re*: *Re* (*Complex x y*) = *x*

**consts** $Im :: complex \Rightarrow real$
**primrec** $Im$: $Im\ (Complex\ x\ y) = y$

**lemma** *complex-surj* [*simp*]: $Complex\ (Re\ z)\ (Im\ z) = z$
⟨*proof*⟩

**lemma** *complex-equality* [*intro?*]: $[\![Re\ x = Re\ y;\ Im\ x = Im\ y]\!] \Longrightarrow x = y$
⟨*proof*⟩

**lemma** *expand-complex-eq*: $(x = y) = (Re\ x = Re\ y \wedge Im\ x = Im\ y)$
⟨*proof*⟩

**lemmas** *complex-Re-Im-cancel-iff* = *expand-complex-eq*

## 20.1 Addition and Subtraction

**instance** *complex* :: *zero*
*complex-zero-def*:
$0 \equiv Complex\ 0\ 0$ ⟨*proof*⟩

**instance** *complex* :: *plus*
*complex-add-def*:
$x + y \equiv Complex\ (Re\ x + Re\ y)\ (Im\ x + Im\ y)$ ⟨*proof*⟩

**instance** *complex* :: *minus*
*complex-minus-def*:
$-\ x \equiv Complex\ (-\ Re\ x)\ (-\ Im\ x)$
*complex-diff-def*:
$x - y \equiv x + -\ y$ ⟨*proof*⟩

**lemma** *Complex-eq-0* [*simp*]: $(Complex\ a\ b = 0) = (a = 0 \wedge b = 0)$
⟨*proof*⟩

**lemma** *complex-Re-zero* [*simp*]: $Re\ 0 = 0$
⟨*proof*⟩

**lemma** *complex-Im-zero* [*simp*]: $Im\ 0 = 0$
⟨*proof*⟩

**lemma** *complex-add* [*simp*]:
$Complex\ a\ b + Complex\ c\ d = Complex\ (a + c)\ (b + d)$
⟨*proof*⟩

**lemma** *complex-Re-add* [*simp*]: $Re\ (x + y) = Re\ x + Re\ y$
⟨*proof*⟩

**lemma** *complex-Im-add* [*simp*]: $Im\ (x + y) = Im\ x + Im\ y$
⟨*proof*⟩

**lemma** *complex-minus* [*simp*]: $-$ (*Complex a b*) = *Complex* ($-$ *a*) ($-$ *b*)
⟨*proof*⟩

**lemma** *complex-Re-minus* [*simp*]: *Re* ($-$ *x*) = $-$ *Re x*
⟨*proof*⟩

**lemma** *complex-Im-minus* [*simp*]: *Im* ($-$ *x*) = $-$ *Im x*
⟨*proof*⟩

**lemma** *complex-diff* [*simp*]:
  *Complex a b* $-$ *Complex c d* = *Complex* (*a* $-$ *c*) (*b* $-$ *d*)
⟨*proof*⟩

**lemma** *complex-Re-diff* [*simp*]: *Re* (*x* $-$ *y*) = *Re x* $-$ *Re y*
⟨*proof*⟩

**lemma** *complex-Im-diff* [*simp*]: *Im* (*x* $-$ *y*) = *Im x* $-$ *Im y*
⟨*proof*⟩

**instance** *complex* :: *ab-group-add*
⟨*proof*⟩

## 20.2 Multiplication and Division

**instance** *complex* :: *one*
  *complex-one-def*:
    *1* ≡ *Complex 1 0* ⟨*proof*⟩

**instance** *complex* :: *times*
  *complex-mult-def*:
    *x* ∗ *y* ≡ *Complex* (*Re x* ∗ *Re y* $-$ *Im x* ∗ *Im y*) (*Re x* ∗ *Im y* + *Im x* ∗ *Re y*)
⟨*proof*⟩

**instance** *complex* :: *inverse*
  *complex-inverse-def*:
    *inverse x* ≡
     *Complex* (*Re x* / ((*Re x*)$^2$ + (*Im x*)$^2$)) ($-$ *Im x* / ((*Re x*)$^2$ + (*Im x*)$^2$))
  *complex-divide-def*:
    *x* / *y* ≡ *x* ∗ *inverse y* ⟨*proof*⟩

**lemma** *Complex-eq-1* [*simp*]: (*Complex a b* = *1*) = (*a* = *1* ∧ *b* = *0*)
⟨*proof*⟩

**lemma** *complex-Re-one* [*simp*]: *Re 1* = *1*
⟨*proof*⟩

**lemma** *complex-Im-one* [*simp*]: *Im 1* = *0*
⟨*proof*⟩

**lemma** *complex-mult* [*simp*]:
$Complex\ a\ b * Complex\ c\ d = Complex\ (a * c - b * d)\ (a * d + b * c)$
⟨*proof*⟩

**lemma** *complex-Re-mult* [*simp*]: $Re\ (x * y) = Re\ x * Re\ y - Im\ x * Im\ y$
⟨*proof*⟩

**lemma** *complex-Im-mult* [*simp*]: $Im\ (x * y) = Re\ x * Im\ y + Im\ x * Re\ y$
⟨*proof*⟩

**lemma** *complex-inverse* [*simp*]:
$inverse\ (Complex\ a\ b) = Complex\ (a\ /\ (a^2 + b^2))\ (-\ b\ /\ (a^2 + b^2))$
⟨*proof*⟩

**lemma** *complex-Re-inverse*:
$Re\ (inverse\ x) = Re\ x\ /\ ((Re\ x)^2 + (Im\ x)^2)$
⟨*proof*⟩

**lemma** *complex-Im-inverse*:
$Im\ (inverse\ x) = -\ Im\ x\ /\ ((Re\ x)^2 + (Im\ x)^2)$
⟨*proof*⟩

**instance** *complex* :: *field*
⟨*proof*⟩

**instance** *complex* :: *division-by-zero*
⟨*proof*⟩

## 20.3 Exponentiation

**instance** *complex* :: *power* ⟨*proof*⟩

**primrec**
*complexpow-0*: $z\ \hat{}\ 0 = 1$
*complexpow-Suc*: $z\ \hat{}\ (Suc\ n) = (z{::}complex) * (z\ \hat{}\ n)$

**instance** *complex* :: *recpower*
⟨*proof*⟩

## 20.4 Numerals and Arithmetic

**instance** *complex* :: *number*
*complex-number-of-def*:
*number-of w* ≡ *of-int w* ⟨*proof*⟩

**instance** *complex* :: *number-ring*
⟨*proof*⟩

**lemma** *complex-Re-of-nat* [*simp*]: $Re\ (of\text{-}nat\ n) = of\text{-}nat\ n$

⟨*proof*⟩

**lemma** *complex-Im-of-nat* [*simp*]: $Im\ (of\text{-}nat\ n) = 0$
⟨*proof*⟩

**lemma** *complex-Re-of-int* [*simp*]: $Re\ (of\text{-}int\ z) = of\text{-}int\ z$
⟨*proof*⟩

**lemma** *complex-Im-of-int* [*simp*]: $Im\ (of\text{-}int\ z) = 0$
⟨*proof*⟩

**lemma** *complex-Re-number-of* [*simp*]: $Re\ (number\text{-}of\ v) = number\text{-}of\ v$
⟨*proof*⟩

**lemma** *complex-Im-number-of* [*simp*]: $Im\ (number\text{-}of\ v) = 0$
⟨*proof*⟩

**lemma** *Complex-eq-number-of* [*simp*]:
$(Complex\ a\ b = number\text{-}of\ w) = (a = number\text{-}of\ w \wedge b = 0)$
⟨*proof*⟩

## 20.5 Scalar Multiplication

**instance** *complex* :: *scaleR*
*complex-scaleR-def*:
$scaleR\ r\ x \equiv Complex\ (r * Re\ x)\ (r * Im\ x)$ ⟨*proof*⟩

**lemma** *complex-scaleR* [*simp*]:
$scaleR\ r\ (Complex\ a\ b) = Complex\ (r * a)\ (r * b)$
⟨*proof*⟩

**lemma** *complex-Re-scaleR* [*simp*]: $Re\ (scaleR\ r\ x) = r * Re\ x$
⟨*proof*⟩

**lemma** *complex-Im-scaleR* [*simp*]: $Im\ (scaleR\ r\ x) = r * Im\ x$
⟨*proof*⟩

**instance** *complex* :: *real-field*
⟨*proof*⟩

## 20.6 Properties of Embedding from Reals

**abbreviation**
*complex-of-real* :: *real* ⇒ *complex* **where**
*complex-of-real* ≡ *of-real*

**lemma** *complex-of-real-def*: $complex\text{-}of\text{-}real\ r = Complex\ r\ 0$
⟨*proof*⟩

**lemma** *Re-complex-of-real* [*simp*]: $Re\ (complex\text{-}of\text{-}real\ z) = z$

⟨*proof*⟩

**lemma** *Im-complex-of-real* [*simp*]: $Im\ (complex\text{-}of\text{-}real\ z) = 0$
⟨*proof*⟩

**lemma** *Complex-add-complex-of-real* [*simp*]:
$Complex\ x\ y + complex\text{-}of\text{-}real\ r = Complex\ (x+r)\ y$
⟨*proof*⟩

**lemma** *complex-of-real-add-Complex* [*simp*]:
$complex\text{-}of\text{-}real\ r + Complex\ x\ y = Complex\ (r+x)\ y$
⟨*proof*⟩

**lemma** *Complex-mult-complex-of-real*:
$Complex\ x\ y * complex\text{-}of\text{-}real\ r = Complex\ (x*r)\ (y*r)$
⟨*proof*⟩

**lemma** *complex-of-real-mult-Complex*:
$complex\text{-}of\text{-}real\ r * Complex\ x\ y = Complex\ (r*x)\ (r*y)$
⟨*proof*⟩

## 20.7 Vector Norm

**instance** *complex* :: *norm*
*complex-norm-def*:
$norm\ z \equiv sqrt\ ((Re\ z)^2 + (Im\ z)^2)$ ⟨*proof*⟩

**abbreviation**
*cmod* :: *complex* ⇒ *real* **where**
$cmod \equiv norm$

**instance** *complex* :: *sgn*
*complex-sgn-def*: $sgn\ x == x\ /_R\ cmod\ x$ ⟨*proof*⟩

**lemmas** *cmod-def* = *complex-norm-def*

**lemma** *complex-norm* [*simp*]: $cmod\ (Complex\ x\ y) = sqrt\ (x^2 + y^2)$
⟨*proof*⟩

**instance** *complex* :: *real-normed-field*
⟨*proof*⟩

**lemma** *cmod-unit-one* [*simp*]: $cmod\ (Complex\ (cos\ a)\ (sin\ a)) = 1$
⟨*proof*⟩

**lemma** *cmod-complex-polar* [*simp*]:
$cmod\ (complex\text{-}of\text{-}real\ r * Complex\ (cos\ a)\ (sin\ a)) = abs\ r$
⟨*proof*⟩

**lemma** *complex-Re-le-cmod*: $Re\ x \leq cmod\ x$
⟨*proof*⟩

**lemma** *complex-mod-minus-le-complex-mod* [*simp*]: $-\ cmod\ x \leq cmod\ x$
⟨*proof*⟩

**lemma** *complex-mod-triangle-ineq2* [*simp*]: $cmod(b + a) - cmod\ b \leq cmod\ a$
⟨*proof*⟩

**lemmas** *real-sum-squared-expand* = *power2-sum* [**where** *'a=real*]

## 20.8 Completeness of the Complexes

**interpretation** *Re*: *bounded-linear* [*Re*]
⟨*proof*⟩

**interpretation** *Im*: *bounded-linear* [*Im*]
⟨*proof*⟩

**lemma** *LIMSEQ-Complex*:
$[\![X \ ----> a;\ Y \ ----> b]\!] \Longrightarrow (\lambda n.\ Complex\ (X\ n)\ (Y\ n)) \ ----> Complex\ a\ b$
⟨*proof*⟩

**instance** *complex* :: *banach*
⟨*proof*⟩

## 20.9 The Complex Number i

**definition**
*ii* :: *complex* (i) **where**
*i-def*: $ii \equiv Complex\ 0\ 1$

**lemma** *complex-Re-i* [*simp*]: $Re\ ii = 0$
⟨*proof*⟩

**lemma** *complex-Im-i* [*simp*]: $Im\ ii = 1$
⟨*proof*⟩

**lemma** *Complex-eq-i* [*simp*]: $(Complex\ x\ y = ii) = (x = 0 \land y = 1)$
⟨*proof*⟩

**lemma** *complex-i-not-zero* [*simp*]: $ii \neq 0$
⟨*proof*⟩

**lemma** *complex-i-not-one* [*simp*]: $ii \neq 1$
⟨*proof*⟩

**lemma** *complex-i-not-number-of* [*simp*]: $ii \neq number\text{-}of\ w$
⟨*proof*⟩

**lemma** *i-mult-Complex* [*simp*]: $ii * Complex\ a\ b = Complex\ (-\ b)\ a$
⟨*proof*⟩

**lemma** *Complex-mult-i* [*simp*]: $Complex\ a\ b * ii = Complex\ (-\ b)\ a$
⟨*proof*⟩

**lemma** *i-complex-of-real* [*simp*]: $ii * complex\text{-}of\text{-}real\ r = Complex\ 0\ r$
⟨*proof*⟩

**lemma** *complex-of-real-i* [*simp*]: $complex\text{-}of\text{-}real\ r * ii = Complex\ 0\ r$
⟨*proof*⟩

**lemma** *i-squared* [*simp*]: $ii * ii = -1$
⟨*proof*⟩

**lemma** *power2-i* [*simp*]: $ii^2 = -1$
⟨*proof*⟩

**lemma** *inverse-i* [*simp*]: $inverse\ ii = -\ ii$
⟨*proof*⟩

## 20.10 Complex Conjugation

**definition**
 $cnj :: complex \Rightarrow complex$ **where**
 $cnj\ z = Complex\ (Re\ z)\ (-\ Im\ z)$

**lemma** *complex-cnj* [*simp*]: $cnj\ (Complex\ a\ b) = Complex\ a\ (-\ b)$
⟨*proof*⟩

**lemma** *complex-Re-cnj* [*simp*]: $Re\ (cnj\ x) = Re\ x$
⟨*proof*⟩

**lemma** *complex-Im-cnj* [*simp*]: $Im\ (cnj\ x) = -\ Im\ x$
⟨*proof*⟩

**lemma** *complex-cnj-cancel-iff* [*simp*]: $(cnj\ x = cnj\ y) = (x = y)$
⟨*proof*⟩

**lemma** *complex-cnj-cnj* [*simp*]: $cnj\ (cnj\ z) = z$
⟨*proof*⟩

**lemma** *complex-cnj-zero* [*simp*]: $cnj\ 0 = 0$
⟨*proof*⟩

**lemma** *complex-cnj-zero-iff* [*iff*]: $(cnj\ z = 0) = (z = 0)$
⟨*proof*⟩

**lemma** *complex-cnj-add*: $cnj\ (x + y) = cnj\ x + cnj\ y$
⟨*proof*⟩

**lemma** *complex-cnj-diff*: $cnj\ (x - y) = cnj\ x - cnj\ y$
⟨*proof*⟩

**lemma** *complex-cnj-minus*: $cnj\ (- x) = - cnj\ x$
⟨*proof*⟩

**lemma** *complex-cnj-one* [*simp*]: $cnj\ 1 = 1$
⟨*proof*⟩

**lemma** *complex-cnj-mult*: $cnj\ (x * y) = cnj\ x * cnj\ y$
⟨*proof*⟩

**lemma** *complex-cnj-inverse*: $cnj\ (inverse\ x) = inverse\ (cnj\ x)$
⟨*proof*⟩

**lemma** *complex-cnj-divide*: $cnj\ (x\ /\ y) = cnj\ x\ /\ cnj\ y$
⟨*proof*⟩

**lemma** *complex-cnj-power*: $cnj\ (x\ \hat{}\ n) = cnj\ x\ \hat{}\ n$
⟨*proof*⟩

**lemma** *complex-cnj-of-nat* [*simp*]: $cnj\ (of\text{-}nat\ n) = of\text{-}nat\ n$
⟨*proof*⟩

**lemma** *complex-cnj-of-int* [*simp*]: $cnj\ (of\text{-}int\ z) = of\text{-}int\ z$
⟨*proof*⟩

**lemma** *complex-cnj-number-of* [*simp*]: $cnj\ (number\text{-}of\ w) = number\text{-}of\ w$
⟨*proof*⟩

**lemma** *complex-cnj-scaleR*: $cnj\ (scaleR\ r\ x) = scaleR\ r\ (cnj\ x)$
⟨*proof*⟩

**lemma** *complex-mod-cnj* [*simp*]: $cmod\ (cnj\ z) = cmod\ z$
⟨*proof*⟩

**lemma** *complex-cnj-complex-of-real* [*simp*]: $cnj\ (of\text{-}real\ x) = of\text{-}real\ x$
⟨*proof*⟩

**lemma** *complex-cnj-i* [*simp*]: $cnj\ ii = - ii$
⟨*proof*⟩

**lemma** *complex-add-cnj*: $z + cnj\ z = complex\text{-}of\text{-}real\ (2 * Re\ z)$
⟨*proof*⟩

**lemma** *complex-diff-cnj*: $z - cnj\ z = complex\text{-}of\text{-}real\ (2 * Im\ z) * ii$

⟨*proof*⟩

**lemma** *complex-mult-cnj*: $z * cnj\ z = complex\text{-}of\text{-}real\ ((Re\ z)^2 + (Im\ z)^2)$
⟨*proof*⟩

**lemma** *complex-mod-mult-cnj*: $cmod\ (z * cnj\ z) = (cmod\ z)^2$
⟨*proof*⟩

**interpretation** *cnj*: *bounded-linear* [*cnj*]
⟨*proof*⟩

## 20.11 The Functions *sgn* and *arg*

———— Argand ————-

**definition**
*arg* :: *complex* => *real* **where**
$arg\ z = (SOME\ a.\ Re(sgn\ z) = cos\ a\ \&\ Im(sgn\ z) = sin\ a\ \&\ -pi < a\ \&\ a \le pi)$

**lemma** *sgn-eq*: $sgn\ z = z\ /\ complex\text{-}of\text{-}real\ (cmod\ z)$
⟨*proof*⟩

**lemma** *i-mult-eq*: $ii * ii = complex\text{-}of\text{-}real\ (-1)$
⟨*proof*⟩

**lemma** *i-mult-eq2* [*simp*]: $ii * ii = -(1::complex)$
⟨*proof*⟩

**lemma** *complex-eq-cancel-iff2* [*simp*]:
$(Complex\ x\ y = complex\text{-}of\text{-}real\ xa) = (x = xa\ \&\ y = 0)$
⟨*proof*⟩

**lemma** *Re-sgn* [*simp*]: $Re(sgn\ z) = Re(z)/cmod\ z$
⟨*proof*⟩

**lemma** *Im-sgn* [*simp*]: $Im(sgn\ z) = Im(z)/cmod\ z$
⟨*proof*⟩

**lemma** *complex-inverse-complex-split*:
$inverse(complex\text{-}of\text{-}real\ x + ii * complex\text{-}of\text{-}real\ y) =$
$complex\text{-}of\text{-}real(x/(x\ \hat{}\ 2 + y\ \hat{}\ 2)) -$
$ii * complex\text{-}of\text{-}real(y/(x\ \hat{}\ 2 + y\ \hat{}\ 2))$
⟨*proof*⟩

**lemma** *cos-arg-i-mult-zero-pos*:
$0 < y ==> cos\ (arg(Complex\ 0\ y)) = 0$
⟨*proof*⟩

**lemma** *cos-arg-i-mult-zero-neg*:
$y < 0 ==> cos\ (arg(Complex\ 0\ y)) = 0$
⟨*proof*⟩

**lemma** *cos-arg-i-mult-zero* [*simp*]:
$y \neq 0 ==> cos\ (arg(Complex\ 0\ y)) = 0$
⟨*proof*⟩

## 20.12 Finally! Polar Form for Complex Numbers

**definition**

*cis* :: *real* => *complex* **where**
$cis\ a = Complex\ (cos\ a)\ (sin\ a)$

**definition**

*rcis* :: [*real*, *real*] => *complex* **where**
$rcis\ r\ a = complex\text{-}of\text{-}real\ r * cis\ a$

**definition**

*expi* :: *complex* => *complex* **where**
$expi\ z = complex\text{-}of\text{-}real(exp\ (Re\ z)) * cis\ (Im\ z)$

**lemma** *complex-split-polar*:
$\exists\, r\ a.\ z = complex\text{-}of\text{-}real\ r * (Complex\ (cos\ a)\ (sin\ a))$
⟨*proof*⟩

**lemma** *rcis-Ex*: $\exists\, r\ a.\ z = rcis\ r\ a$
⟨*proof*⟩

**lemma** *Re-rcis* [*simp*]: $Re(rcis\ r\ a) = r * cos\ a$
⟨*proof*⟩

**lemma** *Im-rcis* [*simp*]: $Im(rcis\ r\ a) = r * sin\ a$
⟨*proof*⟩

**lemma** *sin-cos-squared-add2-mult*: $(r * cos\ a)^2 + (r * sin\ a)^2 = r^2$
⟨*proof*⟩

**lemma** *complex-mod-rcis* [*simp*]: $cmod(rcis\ r\ a) = abs\ r$
⟨*proof*⟩

**lemma** *complex-Re-cnj* [*simp*]: *Re*(*cnj z*) = *Re z*
⟨*proof*⟩

**lemma** *complex-Im-cnj* [*simp*]: *Im*(*cnj z*) = − *Im z*
⟨*proof*⟩

**lemma** *complex-mod-sqrt-Re-mult-cnj*: *cmod z* = *sqrt* (*Re* (*z* ∗ *cnj z*))
⟨*proof*⟩

**lemma** *complex-In-mult-cnj-zero* [*simp*]: *Im* (*z* ∗ *cnj z*) = *0*
⟨*proof*⟩

**lemma** *cis-rcis-eq*: *cis a* = *rcis 1 a*
⟨*proof*⟩

**lemma** *rcis-mult*: *rcis r1 a* ∗ *rcis r2 b* = *rcis* (*r1*∗*r2*) (*a* + *b*)
⟨*proof*⟩

**lemma** *cis-mult*: *cis a* ∗ *cis b* = *cis* (*a* + *b*)
⟨*proof*⟩

**lemma** *cis-zero* [*simp*]: *cis 0* = *1*
⟨*proof*⟩

**lemma** *rcis-zero-mod* [*simp*]: *rcis 0 a* = *0*
⟨*proof*⟩

**lemma** *rcis-zero-arg* [*simp*]: *rcis r 0* = *complex-of-real r*
⟨*proof*⟩

**lemma** *complex-of-real-minus-one*:
  *complex-of-real* (−(*1*::*real*)) = −(*1*::*complex*)
⟨*proof*⟩

**lemma** *complex-i-mult-minus* [*simp*]: *ii* ∗ (*ii* ∗ *x*) = − *x*
⟨*proof*⟩

**lemma** *cis-real-of-nat-Suc-mult*:
  *cis* (*real* (*Suc n*) ∗ *a*) = *cis a* ∗ *cis* (*real n* ∗ *a*)
⟨*proof*⟩

**lemma** *DeMoivre*: (*cis a*) ˆ *n* = *cis* (*real n* ∗ *a*)
⟨*proof*⟩

**lemma** *DeMoivre2*: $(rcis\ r\ a)\ \hat{}\ n = rcis\ (r\ \hat{}\ n)\ (real\ n * a)$
⟨*proof*⟩

**lemma** *cis-inverse* [*simp*]: $inverse(cis\ a) = cis\ (-a)$
⟨*proof*⟩

**lemma** *rcis-inverse*: $inverse(rcis\ r\ a) = rcis\ (1/r)\ (-a)$
⟨*proof*⟩

**lemma** *cis-divide*: $cis\ a\ /\ cis\ b = cis\ (a - b)$
⟨*proof*⟩

**lemma** *rcis-divide*: $rcis\ r1\ a\ /\ rcis\ r2\ b = rcis\ (r1/r2)\ (a - b)$
⟨*proof*⟩

**lemma** *Re-cis* [*simp*]: $Re(cis\ a) = cos\ a$
⟨*proof*⟩

**lemma** *Im-cis* [*simp*]: $Im(cis\ a) = sin\ a$
⟨*proof*⟩

**lemma** *cos-n-Re-cis-pow-n*: $cos\ (real\ n * a) = Re(cis\ a\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *sin-n-Im-cis-pow-n*: $sin\ (real\ n * a) = Im(cis\ a\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *expi-add*: $expi(a + b) = expi(a) * expi(b)$
⟨*proof*⟩

**lemma** *expi-zero* [*simp*]: $expi\ (0::complex) = 1$
⟨*proof*⟩

**lemma** *complex-expi-Ex*: $\exists\, a\ r.\ z = complex\text{-}of\text{-}real\ r * expi\ a$
⟨*proof*⟩

**lemma** *expi-two-pi-i* [*simp*]: $expi((2::complex) * complex\text{-}of\text{-}real\ pi * ii) = 1$
⟨*proof*⟩

**end**

# 21 Zorn: Zorn's Lemma

**theory** *Zorn*
**imports** *Main*

**begin**

The lemma and section numbers refer to an unpublished article [**?**].

**definition**
$chain :: 'a\ set\ set => 'a\ set\ set\ set$ **where**
$chain\ S = \{F.\ F \subseteq S\ \&\ (\forall x \in F.\ \forall y \in F.\ x \subseteq y \mid y \subseteq x)\}$

**definition**
$super :: ['a\ set\ set,'a\ set\ set] => 'a\ set\ set\ set$ **where**
$super\ S\ c = \{d.\ d \in chain\ S\ \&\ c \subset d\}$

**definition**
$maxchain :: 'a\ set\ set => 'a\ set\ set\ set$ **where**
$maxchain\ S = \{c.\ c \in chain\ S\ \&\ super\ S\ c = \{\}\}$

**definition**
$succ :: ['a\ set\ set,'a\ set\ set] => 'a\ set\ set$ **where**
$succ\ S\ c =$
$(if\ c \notin chain\ S \mid c \in maxchain\ S$
$then\ c\ else\ SOME\ c'.\ c' \in super\ S\ c)$

**inductive-set**
$TFin :: 'a\ set\ set => 'a\ set\ set\ set$
**for** $S :: 'a\ set\ set$
**where**
$succI$: $x \in TFin\ S ==> succ\ S\ x \in TFin\ S$
| $Pow\text{-}UnionI$: $Y \in Pow(TFin\ S) ==> Union(Y) \in TFin\ S$
**monos** $Pow\text{-}mono$

## 21.1 Mathematical Preamble

**lemma** *Union-lemma0*:
$(\forall x \in C.\ x \subseteq A \mid B \subseteq x) ==> Union(C) \subseteq A \mid B \subseteq Union(C)$
⟨*proof*⟩

This is theorem *increasingD2* of ZF/Zorn.thy

**lemma** *Abrial-axiom1*: $x \subseteq succ\ S\ x$
⟨*proof*⟩

**lemmas** *TFin-UnionI* = *TFin.Pow-UnionI* [*OF PowI*]

**lemma** *TFin-induct*:
$[|\ n \in TFin\ S;$
$!!x.\ [|\ x \in TFin\ S;\ P(x)\ |] ==> P(succ\ S\ x);$
$!!Y.\ [|\ Y \subseteq TFin\ S;\ Ball\ Y\ P\ |] ==> P(Union\ Y)\ |]$
$==> P(n)$
⟨*proof*⟩

**lemma** *succ-trans*: $x \subseteq y ==> x \subseteq succ\ S\ y$

⟨*proof*⟩

Lemma 1 of section 3.1

**lemma** *TFin-linear-lemma1*:
$[| \ n \in \mathit{TFin}\ S; \ \ m \in \mathit{TFin}\ S;$
$\forall x \in \mathit{TFin}\ S.\ x \subseteq m \ {-}{-}{>} \ x = m \mid \mathit{succ}\ S\ x \subseteq m$
$|] \ {=}{=}{>} \ n \subseteq m \mid \mathit{succ}\ S\ m \subseteq n$
⟨*proof*⟩

Lemma 2 of section 3.2

**lemma** *TFin-linear-lemma2*:
$m \in \mathit{TFin}\ S \ {=}{=}{>} \ \forall n \in \mathit{TFin}\ S.\ n \subseteq m \ {-}{-}{>} \ n{=}m \mid \mathit{succ}\ S\ n \subseteq m$
⟨*proof*⟩

Re-ordering the premises of Lemma 2

**lemma** *TFin-subsetD*:
$[| \ n \subseteq m; \ \ m \in \mathit{TFin}\ S; \ \ n \in \mathit{TFin}\ S \ |] \ {=}{=}{>} \ n{=}m \mid \mathit{succ}\ S\ n \subseteq m$
⟨*proof*⟩

Consequences from section 3.3 – Property 3.2, the ordering is total

**lemma** *TFin-subset-linear*: $[| \ m \in \mathit{TFin}\ S; \ \ n \in \mathit{TFin}\ S|] \ {=}{=}{>} \ n \subseteq m \mid m \subseteq n$
⟨*proof*⟩

Lemma 3 of section 3.3

**lemma** *eq-succ-upper*: $[| \ n \in \mathit{TFin}\ S; \ \ m \in \mathit{TFin}\ S; \ \ m = \mathit{succ}\ S\ m \ |] \ {=}{=}{>} \ n \subseteq m$
⟨*proof*⟩

Property 3.3 of section 3.3

**lemma** *equal-succ-Union*: $m \in \mathit{TFin}\ S \ {=}{=}{>} \ (m = \mathit{succ}\ S\ m) = (m = \mathit{Union}(\mathit{TFin}\ S))$
⟨*proof*⟩

## 21.2 Hausdorff’s Theorem: Every Set Contains a Maximal Chain.

NB: We assume the partial ordering is ⊆, the subset relation!

**lemma** *empty-set-mem-chain*: $(\{\} :: {}'a\ \mathit{set}\ \mathit{set}) \in \mathit{chain}\ S$
⟨*proof*⟩

**lemma** *super-subset-chain*: $\mathit{super}\ S\ c \subseteq \mathit{chain}\ S$
⟨*proof*⟩

**lemma** *maxchain-subset-chain*: $\mathit{maxchain}\ S \subseteq \mathit{chain}\ S$
⟨*proof*⟩

**lemma** *mem-super-Ex*: $c \in \mathit{chain}\ S - \mathit{maxchain}\ S \ {=}{=}{>} \ ?\ d.\ d \in \mathit{super}\ S\ c$

⟨*proof*⟩

**lemma** *select-super*:
$c \in chain\ S\ -\ maxchain\ S\ ==>\ (\epsilon\, c'.\ c'\!: super\ S\ c): super\ S\ c$
⟨*proof*⟩

**lemma** *select-not-equals*:
$c \in chain\ S\ -\ maxchain\ S\ ==>\ (\epsilon\, c'.\ c'\!: super\ S\ c) \neq c$
⟨*proof*⟩

**lemma** *succI3*: $c \in chain\ S\ -\ maxchain\ S\ ==>\ succ\ S\ c = (\epsilon\, c'.\ c'\!: super\ S\ c)$
⟨*proof*⟩

**lemma** *succ-not-equals*: $c \in chain\ S\ -\ maxchain\ S\ ==>\ succ\ S\ c \neq c$
⟨*proof*⟩

**lemma** *TFin-chain-lemma4*: $c \in TFin\ S\ ==>\ (c :: 'a\ set\ set): chain\ S$
⟨*proof*⟩

**theorem** *Hausdorff*: $\exists\, c.\ (c :: 'a\ set\ set): maxchain\ S$
⟨*proof*⟩

## 21.3 Zorn's Lemma: If All Chains Have Upper Bounds Then There Is a Maximal Element

**lemma** *chain-extend*:
$[|\ c \in chain\ S;\ z \in S;$
$\forall\, x \in c.\ x \subseteq (z\!:: 'a\ set)\ |]\ ==>\ \{z\}\ Un\ c \in chain\ S$
⟨*proof*⟩

**lemma** *chain-Union-upper*: $[|\ c \in chain\ S;\ x \in c\ |]\ ==>\ x \subseteq Union(c)$
⟨*proof*⟩

**lemma** *chain-ball-Union-upper*: $c \in chain\ S\ ==>\ \forall\, x \in c.\ x \subseteq Union(c)$
⟨*proof*⟩

**lemma** *maxchain-Zorn*:
$[|\ c \in maxchain\ S;\ u \in S;\ Union(c) \subseteq u\ |]\ ==>\ Union(c) = u$
⟨*proof*⟩

**theorem** *Zorn-Lemma*:
$\forall\, c \in chain\ S.\ Union(c): S\ ==>\ \exists\, y \in S.\ \forall\, z \in S.\ y \subseteq z\ -->\ y = z$
⟨*proof*⟩

## 21.4 Alternative version of Zorn's Lemma

**lemma** *Zorn-Lemma2*:
$\forall\, c \in chain\ S.\ \exists\, y \in S.\ \forall\, x \in c.\ x \subseteq y$
$==>\ \exists\, y \in S.\ \forall\, x \in S.\ (y :: 'a\ set) \subseteq x\ -->\ y = x$

⟨*proof*⟩

Various other lemmas

**lemma** *chainD*: $[|\ c \in chain\ S;\ x \in c;\ y \in c\ |] ==> x \subseteq y \mid y \subseteq x$
⟨*proof*⟩

**lemma** *chainD2*: $!!(c :: 'a\ set\ set).\ c \in chain\ S ==> c \subseteq S$
⟨*proof*⟩

**end**

# 22 Filter: Filters and Ultrafilters

**theory** *Filter*
**imports** *Zorn Infinite-Set*
**begin**

## 22.1 Definitions and basic properties

### 22.1.1 Filters

**locale** *filter* =
**fixes** $F :: 'a\ set\ set$
**assumes** *UNIV* [*iff*]: $UNIV \in F$
**assumes** *empty* [*iff*]: $\{\} \notin F$
**assumes** *Int*: $[\![u \in F;\ v \in F]\!] \Longrightarrow u \cap v \in F$
**assumes** *subset*: $[\![u \in F;\ u \subseteq v]\!] \Longrightarrow v \in F$

**lemma** (**in** *filter*) *memD*: $A \in F \Longrightarrow -\ A \notin F$
⟨*proof*⟩

**lemma** (**in** *filter*) *not-memI*: $-\ A \in F \Longrightarrow A \notin F$
⟨*proof*⟩

**lemma** (**in** *filter*) *Int-iff*: $(x \cap y \in F) = (x \in F \wedge y \in F)$
⟨*proof*⟩

### 22.1.2 Ultrafilters

**locale** *ultrafilter* = *filter* +
**assumes** *ultra*: $A \in F \vee -\ A \in F$

**lemma** (**in** *ultrafilter*) *memI*: $-\ A \notin F \Longrightarrow A \in F$
⟨*proof*⟩

**lemma** (**in** *ultrafilter*) *not-memD*: $A \notin F \Longrightarrow -\ A \in F$
⟨*proof*⟩

**lemma** (**in** *ultrafilter*) *not-mem-iff*: $(A \notin F) = (- A \in F)$
⟨*proof*⟩

**lemma** (**in** *ultrafilter*) *Compl-iff*: $(- A \in F) = (A \notin F)$
⟨*proof*⟩

**lemma** (**in** *ultrafilter*) *Un-iff*: $(x \cup y \in F) = (x \in F \lor y \in F)$
⟨*proof*⟩

### 22.1.3 Free Ultrafilters

**locale** *freeultrafilter* = *ultrafilter* +
**assumes** *infinite*: $A \in F \Longrightarrow$ *infinite* $A$

**lemma** (**in** *freeultrafilter*) *finite*: *finite* $A \Longrightarrow A \notin F$
⟨*proof*⟩

**lemma** (**in** *freeultrafilter*) *singleton*: $\{x\} \notin F$
⟨*proof*⟩

**lemma** (**in** *freeultrafilter*) *insert-iff* [*simp*]: (*insert* $x$ $A \in F) = (A \in F)$
⟨*proof*⟩

**lemma** (**in** *freeultrafilter*) *filter*: *filter* $F$ ⟨*proof*⟩

**lemma** (**in** *freeultrafilter*) *ultrafilter*: *ultrafilter* $F$
⟨*proof*⟩

## 22.2 Collect properties

**lemma** (**in** *filter*) *Collect-ex*:
$(\{n.\ \exists x.\ P\ n\ x\} \in F) = (\exists X.\ \{n.\ P\ n\ (X\ n)\} \in F)$
⟨*proof*⟩

**lemma** (**in** *filter*) *Collect-conj*:
$(\{n.\ P\ n \land Q\ n\} \in F) = (\{n.\ P\ n\} \in F \land \{n.\ Q\ n\} \in F)$
⟨*proof*⟩

**lemma** (**in** *ultrafilter*) *Collect-not*:
$(\{n.\ \neg\ P\ n\} \in F) = (\{n.\ P\ n\} \notin F)$
⟨*proof*⟩

**lemma** (**in** *ultrafilter*) *Collect-disj*:
$(\{n.\ P\ n \lor Q\ n\} \in F) = (\{n.\ P\ n\} \in F \lor \{n.\ Q\ n\} \in F)$
⟨*proof*⟩

**lemma** (**in** *ultrafilter*) *Collect-all*:
$(\{n.\ \forall x.\ P\ n\ x\} \in F) = (\forall X.\ \{n.\ P\ n\ (X\ n)\} \in F)$
⟨*proof*⟩

## 22.3 Maximal filter = Ultrafilter

A filter F is an ultrafilter iff it is a maximal filter, i.e. whenever G is a filter and $F \subseteq G$ then $F = G$

Lemmas that shows existence of an extension to what was assumed to be a maximal filter. Will be used to derive contradiction in proof of property of ultrafilter.

**lemma** *extend-lemma1*: $UNIV \in F \Longrightarrow A \in \{X.\ \exists f{\in}F.\ A \cap f \subseteq X\}$
⟨*proof*⟩

**lemma** *extend-lemma2*: $F \subseteq \{X.\ \exists f{\in}F.\ A \cap f \subseteq X\}$
⟨*proof*⟩

**lemma** (**in** *filter*) *extend-filter*:
**assumes** *A*: $- A \notin F$
**shows** *filter* $\{X.\ \exists f{\in}F.\ A \cap f \subseteq X\}$ (**is** *filter* *?X*)
⟨*proof*⟩

**lemma** (**in** *filter*) *max-filter-ultrafilter*:
**assumes** *max*: $\bigwedge G.\ [\![filter\ G;\ F \subseteq G]\!] \Longrightarrow F = G$
**shows** *ultrafilter-axioms F*
⟨*proof*⟩

**lemma** (**in** *ultrafilter*) *max-filter*:
**assumes** *G*: *filter G* **and** *sub*: $F \subseteq G$ **shows** $F = G$
⟨*proof*⟩

## 22.4 Ultrafilter Theorem

A locale makes proof of ultrafilter Theorem more modular

**locale** (**open**) *UFT* =
  **fixes** *frechet* :: *'a set set*
  **and** *superfrechet* :: *'a set set set*

  **assumes** *infinite-UNIV*: *infinite* (*UNIV* :: *'a set*)

  **defines** *frechet-def*: $frechet \equiv \{A.\ finite\ (- A)\}$
  **and** *superfrechet-def*: $superfrechet \equiv \{G.\ filter\ G \wedge frechet \subseteq G\}$

**lemma** (**in** *UFT*) *superfrechetI*:
  $[\![filter\ G;\ frechet \subseteq G]\!] \Longrightarrow G \in superfrechet$
⟨*proof*⟩

**lemma** (**in** *UFT*) *superfrechetD1*:
  $G \in superfrechet \Longrightarrow filter\ G$
⟨*proof*⟩

**lemma** (**in** *UFT*) *superfrechetD2*:
$G \in$ *superfrechet* $\Longrightarrow$ *frechet* $\subseteq G$
⟨*proof*⟩

A few properties of free filters

**lemma** *filter-cofinite*:
**assumes** *inf*: *infinite* (*UNIV* :: *'a set*)
**shows** *filter* $\{A$:: *'a set*. *finite* $(- A)\}$ (**is** *filter ?F*)
⟨*proof*⟩

We prove: 1. Existence of maximal filter i.e. ultrafilter; 2. Freeness property i.e ultrafilter is free. Use a locale to prove various lemmas and then export main result: The ultrafilter Theorem

**lemma** (**in** *UFT*) *filter-frechet*: *filter frechet*
⟨*proof*⟩

**lemma** (**in** *UFT*) *frechet-in-superfrechet*: *frechet* $\in$ *superfrechet*
⟨*proof*⟩

**lemma** (**in** *UFT*) *lemma-mem-chain-filter*:
$[\![c \in$ *chain superfrechet*; $x \in c]\!] \Longrightarrow$ *filter* $x$
⟨*proof*⟩

### 22.4.1 Unions of chains of superfrechets

In this section we prove that superfrechet is closed with respect to unions of non-empty chains. We must show 1) Union of a chain is a filter, 2) Union of a chain contains frechet.

Number 2 is trivial, but 1 requires us to prove all the filter rules.

**lemma** (**in** *UFT*) *Union-chain-UNIV*:
$[\![c \in$ *chain superfrechet*; $c \neq \{\}]\!] \Longrightarrow$ *UNIV* $\in \bigcup c$
⟨*proof*⟩

**lemma** (**in** *UFT*) *Union-chain-empty*:
$c \in$ *chain superfrechet* $\Longrightarrow \{\} \notin \bigcup c$
⟨*proof*⟩

**lemma** (**in** *UFT*) *Union-chain-Int*:
$[\![c \in$ *chain superfrechet*; $u \in \bigcup c$; $v \in \bigcup c]\!] \Longrightarrow u \cap v \in \bigcup c$
⟨*proof*⟩

**lemma** (**in** *UFT*) *Union-chain-subset*:
$[\![c \in$ *chain superfrechet*; $u \in \bigcup c$; $u \subseteq v]\!] \Longrightarrow v \in \bigcup c$
⟨*proof*⟩

**lemma** (**in** *UFT*) *Union-chain-filter*:
**assumes** *chain*: $c \in$ *chain superfrechet* **and** *nonempty*: $c \neq \{\}$

**shows** *filter* $(\bigcup c)$
$\langle proof \rangle$

**lemma** (**in** *UFT*) *lemma-mem-chain-frechet-subset*:
$[\![c \in chain\ superfrechet;\ x \in c]\!] \Longrightarrow frechet \subseteq x$
$\langle proof \rangle$

**lemma** (**in** *UFT*) *Union-chain-superfrechet*:
$[\![c \neq \{\};\ c \in chain\ superfrechet]\!] \Longrightarrow \bigcup c \in superfrechet$
$\langle proof \rangle$

### 22.4.2 Existence of free ultrafilter

**lemma** (**in** *UFT*) *max-cofinite-filter-Ex*:
$\exists\, U{\in}superfrechet.\ \forall\, G{\in}superfrechet.\ U \subseteq G \longrightarrow U = G$
$\langle proof \rangle$

**lemma** (**in** *UFT*) *mem-superfrechet-all-infinite*:
$[\![U \in superfrechet;\ A \in U]\!] \Longrightarrow infinite\ A$
$\langle proof \rangle$

There exists a free ultrafilter on any infinite set

**lemma** (**in** *UFT*) *freeultrafilter-ex*:
$\exists\, U{::}'a\ set\ set.\ freeultrafilter\ U$
$\langle proof \rangle$

**lemmas** *freeultrafilter-Ex* = *UFT.freeultrafilter-ex*

**hide** (**open**) *const filter*

**end**

## 23 StarDef: Construction of Star Types Using Ultrafilters

**theory** *StarDef*
**imports** *Filter*
**uses** (*transfer.ML*)
**begin**

### 23.1 A Free Ultrafilter over the Naturals

**definition**
*FreeUltrafilterNat* :: *nat set set* ($\mathcal{U}$) **where**
$\mathcal{U} = (SOME\ U.\ freeultrafilter\ U)$

**lemma** *freeultrafilter-FreeUltrafilterNat*: *freeultrafilter* $\mathcal{U}$

⟨*proof*⟩

**interpretation** *FreeUltrafilterNat*: *freeultrafilter* [*FreeUltrafilterNat*]
⟨*proof*⟩

This rule takes the place of the old ultra tactic

**lemma** *ultra*:
$[\![\{n.\ P\ n\} \in \mathcal{U};\ \{n.\ P\ n \longrightarrow Q\ n\} \in \mathcal{U}]\!] \Longrightarrow \{n.\ Q\ n\} \in \mathcal{U}$
⟨*proof*⟩

## 23.2 Definition of *star* type constructor

**definition**
$starrel :: ((nat \Rightarrow 'a) \times (nat \Rightarrow 'a))\ set$ **where**
$starrel = \{(X,Y).\ \{n.\ X\ n = Y\ n\} \in \mathcal{U}\}$

**typedef** $'a\ star = (UNIV :: (nat \Rightarrow 'a)\ set)\ //\ starrel$
⟨*proof*⟩

**definition**
$star\text{-}n :: (nat \Rightarrow 'a) \Rightarrow 'a\ star$ **where**
$star\text{-}n\ X = Abs\text{-}star\ (starrel\ ``\ \{X\})$

**theorem** *star-cases* [*case-names star-n*, *cases type*: *star*]:
$(\bigwedge X.\ x = star\text{-}n\ X \Longrightarrow P) \Longrightarrow P$
⟨*proof*⟩

**lemma** *all-star-eq*: $(\forall x.\ P\ x) = (\forall X.\ P\ (star\text{-}n\ X))$
⟨*proof*⟩

**lemma** *ex-star-eq*: $(\exists x.\ P\ x) = (\exists X.\ P\ (star\text{-}n\ X))$
⟨*proof*⟩

Proving that *starrel* is an equivalence relation

**lemma** *starrel-iff* [*iff*]: $((X,Y) \in starrel) = (\{n.\ X\ n = Y\ n\} \in \mathcal{U})$
⟨*proof*⟩

**lemma** *equiv-starrel*: *equiv UNIV starrel*
⟨*proof*⟩

**lemmas** *equiv-starrel-iff* =
*eq-equiv-class-iff* [*OF equiv-starrel UNIV-I UNIV-I*]

**lemma** *starrel-in-star*: $starrel``\{x\} \in star$
⟨*proof*⟩

**lemma** *star-n-eq-iff*: $(star\text{-}n\ X = star\text{-}n\ Y) = (\{n.\ X\ n = Y\ n\} \in \mathcal{U})$
⟨*proof*⟩

## 23.3 Transfer principle

This introduction rule starts each transfer proof.

**lemma** *transfer-start*:
$P \equiv \{n.\ Q\} \in \mathcal{U} \Longrightarrow \mathit{Trueprop}\ P \equiv \mathit{Trueprop}\ Q$
$\langle \mathit{proof} \rangle$

Initialize transfer tactic.

$\langle \mathit{ML} \rangle$

Transfer introduction rules.

**lemma** *transfer-ex* [*transfer-intro*]:
$[\![\bigwedge X.\ p\ (\mathit{star\text{-}n}\ X) \equiv \{n.\ P\ n\ (X\ n)\} \in \mathcal{U}]\!]$
$\Longrightarrow \exists x{::}'a\ \mathit{star}.\ p\ x \equiv \{n.\ \exists x.\ P\ n\ x\} \in \mathcal{U}$
$\langle \mathit{proof} \rangle$

**lemma** *transfer-all* [*transfer-intro*]:
$[\![\bigwedge X.\ p\ (\mathit{star\text{-}n}\ X) \equiv \{n.\ P\ n\ (X\ n)\} \in \mathcal{U}]\!]$
$\Longrightarrow \forall x{::}'a\ \mathit{star}.\ p\ x \equiv \{n.\ \forall x.\ P\ n\ x\} \in \mathcal{U}$
$\langle \mathit{proof} \rangle$

**lemma** *transfer-not* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U}]\!] \Longrightarrow \neg\ p \equiv \{n.\ \neg\ P\ n\} \in \mathcal{U}$
$\langle \mathit{proof} \rangle$

**lemma** *transfer-conj* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ q \equiv \{n.\ Q\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow p \wedge q \equiv \{n.\ P\ n \wedge Q\ n\} \in \mathcal{U}$
$\langle \mathit{proof} \rangle$

**lemma** *transfer-disj* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ q \equiv \{n.\ Q\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow p \vee q \equiv \{n.\ P\ n \vee Q\ n\} \in \mathcal{U}$
$\langle \mathit{proof} \rangle$

**lemma** *transfer-imp* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ q \equiv \{n.\ Q\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow p \longrightarrow q \equiv \{n.\ P\ n \longrightarrow Q\ n\} \in \mathcal{U}$
$\langle \mathit{proof} \rangle$

**lemma** *transfer-iff* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ q \equiv \{n.\ Q\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow p = q \equiv \{n.\ P\ n = Q\ n\} \in \mathcal{U}$
$\langle \mathit{proof} \rangle$

**lemma** *transfer-if-bool* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ x \equiv \{n.\ X\ n\} \in \mathcal{U};\ y \equiv \{n.\ Y\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow (\mathit{if}\ p\ \mathit{then}\ x\ \mathit{else}\ y) \equiv \{n.\ \mathit{if}\ P\ n\ \mathit{then}\ X\ n\ \mathit{else}\ Y\ n\} \in \mathcal{U}$
$\langle \mathit{proof} \rangle$

**lemma** *transfer-eq* [*transfer-intro*]:
$[\![x \equiv \textit{star-n}\ X;\ y \equiv \textit{star-n}\ Y]\!] \Longrightarrow x = y \equiv \{n.\ X\ n = Y\ n\} \in \mathcal{U}$
⟨*proof*⟩

**lemma** *transfer-if* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ x \equiv \textit{star-n}\ X;\ y \equiv \textit{star-n}\ Y]\!]$
$\Longrightarrow (\textit{if}\ p\ \textit{then}\ x\ \textit{else}\ y) \equiv \textit{star-n}\ (\lambda n.\ \textit{if}\ P\ n\ \textit{then}\ X\ n\ \textit{else}\ Y\ n)$
⟨*proof*⟩

**lemma** *transfer-fun-eq* [*transfer-intro*]:
$[\![\bigwedge X.\ f\ (\textit{star-n}\ X) = g\ (\textit{star-n}\ X)$
$\equiv \{n.\ F\ n\ (X\ n) = G\ n\ (X\ n)\} \in \mathcal{U}]\!]$
$\Longrightarrow f = g \equiv \{n.\ F\ n = G\ n\} \in \mathcal{U}$
⟨*proof*⟩

**lemma** *transfer-star-n* [*transfer-intro*]: $\textit{star-n}\ X \equiv \textit{star-n}\ (\lambda n.\ X\ n)$
⟨*proof*⟩

**lemma** *transfer-bool* [*transfer-intro*]: $p \equiv \{n.\ p\} \in \mathcal{U}$
⟨*proof*⟩

## 23.4 Standard elements

**definition**
*star-of* :: $'a \Rightarrow 'a\ \textit{star}$ **where**
$\textit{star-of}\ x == \textit{star-n}\ (\lambda n.\ x)$

**definition**
*Standard* :: $'a\ \textit{star set}$ **where**
$\textit{Standard} = \textit{range star-of}$

Transfer tactic should remove occurrences of *star-of*

⟨*ML*⟩

**declare** *star-of-def* [*transfer-intro*]

**lemma** *star-of-inject*: $(\textit{star-of}\ x = \textit{star-of}\ y) = (x = y)$
⟨*proof*⟩

**lemma** *Standard-star-of* [*simp*]: $\textit{star-of}\ x \in \textit{Standard}$
⟨*proof*⟩

## 23.5 Internal functions

**definition**
*Ifun* :: $('a \Rightarrow 'b)\ \textit{star} \Rightarrow 'a\ \textit{star} \Rightarrow 'b\ \textit{star}$ (- $\star$ - [*300*,*301*] *300*) **where**
$\textit{Ifun}\ f \equiv \lambda x.\ \textit{Abs-star}$
$(\bigcup F{\in}\textit{Rep-star}\ f.\ \bigcup X{\in}\textit{Rep-star}\ x.\ \textit{starrel}``\{\lambda n.\ F\ n\ (X\ n)\})$

**lemma** *Ifun-congruent2*:
$congruent2\ starrel\ starrel\ (\lambda F\ X.\ starrel``\{\lambda n.\ F\ n\ (X\ n)\})$
⟨*proof*⟩

**lemma** *Ifun-star-n*: $star\text{-}n\ F \star star\text{-}n\ X = star\text{-}n\ (\lambda n.\ F\ n\ (X\ n))$
⟨*proof*⟩

Transfer tactic should remove occurrences of *Ifun*

⟨*ML*⟩

**lemma** *transfer-Ifun* [*transfer-intro*]:
$[\![f \equiv star\text{-}n\ F;\ x \equiv star\text{-}n\ X]\!] \Longrightarrow f \star x \equiv star\text{-}n\ (\lambda n.\ F\ n\ (X\ n))$
⟨*proof*⟩

**lemma** *Ifun-star-of* [*simp*]: $star\text{-}of\ f \star star\text{-}of\ x = star\text{-}of\ (f\ x)$
⟨*proof*⟩

**lemma** *Standard-Ifun* [*simp*]:
$[\![f \in Standard;\ x \in Standard]\!] \Longrightarrow f \star x \in Standard$
⟨*proof*⟩

Nonstandard extensions of functions

**definition**
$starfun :: ('a \Rightarrow 'b) \Rightarrow ('a\ star \Rightarrow 'b\ star)$ $(*f*\ \text{-}\ [80]\ 80)$ **where**
$starfun\ f == \lambda x.\ star\text{-}of\ f \star x$

**definition**
$starfun2 :: ('a \Rightarrow 'b \Rightarrow 'c) \Rightarrow ('a\ star \Rightarrow 'b\ star \Rightarrow 'c\ star)$
$(*f2*\ \text{-}\ [80]\ 80)$ **where**
$starfun2\ f == \lambda x\ y.\ star\text{-}of\ f \star x \star y$

**declare** *starfun-def* [*transfer-unfold*]
**declare** *starfun2-def* [*transfer-unfold*]

**lemma** *starfun-star-n*: $(\ *f*\ f)\ (star\text{-}n\ X) = star\text{-}n\ (\lambda n.\ f\ (X\ n))$
⟨*proof*⟩

**lemma** *starfun2-star-n*:
$(\ *f2*\ f)\ (star\text{-}n\ X)\ (star\text{-}n\ Y) = star\text{-}n\ (\lambda n.\ f\ (X\ n)\ (Y\ n))$
⟨*proof*⟩

**lemma** *starfun-star-of* [*simp*]: $(\ *f*\ f)\ (star\text{-}of\ x) = star\text{-}of\ (f\ x)$
⟨*proof*⟩

**lemma** *starfun2-star-of* [*simp*]: $(\ *f2*\ f)\ (star\text{-}of\ x) = *f*\ f\ x$
⟨*proof*⟩

**lemma** *Standard-starfun* [*simp*]: $x \in Standard \Longrightarrow starfun\ f\ x \in Standard$

⟨proof⟩

**lemma** *Standard-starfun2* [*simp*]:
  $[\![x \in \mathit{Standard};\ y \in \mathit{Standard}]\!] \Longrightarrow \mathit{starfun2}\ f\ x\ y \in \mathit{Standard}$
⟨proof⟩

**lemma** *Standard-starfun-iff*:
  **assumes** *inj*: $\bigwedge x\ y.\ f\ x = f\ y \Longrightarrow x = y$
  **shows** $(\mathit{starfun}\ f\ x \in \mathit{Standard}) = (x \in \mathit{Standard})$
⟨proof⟩

**lemma** *Standard-starfun2-iff*:
  **assumes** *inj*: $\bigwedge a\ b\ a'\ b'.\ f\ a\ b = f\ a'\ b' \Longrightarrow a = a' \wedge b = b'$
  **shows** $(\mathit{starfun2}\ f\ x\ y \in \mathit{Standard}) = (x \in \mathit{Standard} \wedge y \in \mathit{Standard})$
⟨proof⟩

## 23.6 Internal predicates

**definition**
  $\mathit{unstar} :: \mathit{bool}\ \mathit{star} \Rightarrow \mathit{bool}$ **where**
  $\mathit{unstar}\ b = (b = \mathit{star\text{-}of}\ \mathit{True})$

**lemma** *unstar-star-n*: $\mathit{unstar}\ (\mathit{star\text{-}n}\ P) = (\{n.\ P\ n\} \in \mathcal{U})$
⟨proof⟩

**lemma** *unstar-star-of* [*simp*]: $\mathit{unstar}\ (\mathit{star\text{-}of}\ p) = p$
⟨proof⟩

Transfer tactic should remove occurrences of *unstar*

⟨ML⟩

**lemma** *transfer-unstar* [*transfer-intro*]:
  $p \equiv \mathit{star\text{-}n}\ P \Longrightarrow \mathit{unstar}\ p \equiv \{n.\ P\ n\} \in \mathcal{U}$
⟨proof⟩

**definition**
  $\mathit{starP} :: ('a \Rightarrow \mathit{bool}) \Rightarrow 'a\ \mathit{star} \Rightarrow \mathit{bool}$ $(*p*\ \text{-}\ [80]\ 80)$ **where**
  $*p*\ P = (\lambda x.\ \mathit{unstar}\ (\mathit{star\text{-}of}\ P \star x))$

**definition**
  $\mathit{starP2} :: ('a \Rightarrow 'b \Rightarrow \mathit{bool}) \Rightarrow 'a\ \mathit{star} \Rightarrow 'b\ \mathit{star} \Rightarrow \mathit{bool}$ $(*p2*\ \text{-}\ [80]\ 80)$ **where**
  $*p2*\ P = (\lambda x\ y.\ \mathit{unstar}\ (\mathit{star\text{-}of}\ P \star x \star y))$

**declare** *starP-def* [*transfer-unfold*]
**declare** *starP2-def* [*transfer-unfold*]

**lemma** *starP-star-n*: $(\ *p*\ P)\ (\mathit{star\text{-}n}\ X) = (\{n.\ P\ (X\ n)\} \in \mathcal{U})$
⟨proof⟩

**lemma** *starP2-star-n*:
$( *p2* \; P) \; (star\text{-}n \; X) \; (star\text{-}n \; Y) = (\{n. \; P \; (X \; n) \; (Y \; n)\} \in \mathcal{U})$
⟨*proof*⟩

**lemma** *starP-star-of* [*simp*]: $( *p* \; P) \; (star\text{-}of \; x) = P \; x$
⟨*proof*⟩

**lemma** *starP2-star-of* [*simp*]: $( *p2* \; P) \; (star\text{-}of \; x) = *p* \; P \; x$
⟨*proof*⟩

## 23.7 Internal sets

**definition**
*Iset* :: $'a \; set \; star \Rightarrow {'a} \; star \; set$ **where**
$Iset \; A = \{x. \; ( *p2* \; op \in) \; x \; A\}$

**lemma** *Iset-star-n*:
$(star\text{-}n \; X \in Iset \; (star\text{-}n \; A)) = (\{n. \; X \; n \in A \; n\} \in \mathcal{U})$
⟨*proof*⟩

Transfer tactic should remove occurrences of *Iset*

⟨*ML*⟩

**lemma** *transfer-mem* [*transfer-intro*]:
$[\![x \equiv star\text{-}n \; X; \; a \equiv Iset \; (star\text{-}n \; A)]\!]$
$\Longrightarrow x \in a \equiv \{n. \; X \; n \in A \; n\} \in \mathcal{U}$
⟨*proof*⟩

**lemma** *transfer-Collect* [*transfer-intro*]:
$[\![\bigwedge X. \; p \; (star\text{-}n \; X) \equiv \{n. \; P \; n \; (X \; n)\} \in \mathcal{U}]\!]$
$\Longrightarrow Collect \; p \equiv Iset \; (star\text{-}n \; (\lambda n. \; Collect \; (P \; n)))$
⟨*proof*⟩

**lemma** *transfer-set-eq* [*transfer-intro*]:
$[\![a \equiv Iset \; (star\text{-}n \; A); \; b \equiv Iset \; (star\text{-}n \; B)]\!]$
$\Longrightarrow a = b \equiv \{n. \; A \; n = B \; n\} \in \mathcal{U}$
⟨*proof*⟩

**lemma** *transfer-ball* [*transfer-intro*]:
$[\![a \equiv Iset \; (star\text{-}n \; A); \; \bigwedge X. \; p \; (star\text{-}n \; X) \equiv \{n. \; P \; n \; (X \; n)\} \in \mathcal{U}]\!]$
$\Longrightarrow \forall x \in a. \; p \; x \equiv \{n. \; \forall x \in A \; n. \; P \; n \; x\} \in \mathcal{U}$
⟨*proof*⟩

**lemma** *transfer-bex* [*transfer-intro*]:
$[\![a \equiv Iset \; (star\text{-}n \; A); \; \bigwedge X. \; p \; (star\text{-}n \; X) \equiv \{n. \; P \; n \; (X \; n)\} \in \mathcal{U}]\!]$
$\Longrightarrow \exists x \in a. \; p \; x \equiv \{n. \; \exists x \in A \; n. \; P \; n \; x\} \in \mathcal{U}$
⟨*proof*⟩

**lemma** *transfer-Iset* [*transfer-intro*]:

$[\![a \equiv \textit{star-n } A]\!] \Longrightarrow \textit{Iset } a \equiv \textit{Iset } (\textit{star-n } (\lambda n.\ A\ n))$
⟨*proof*⟩

Nonstandard extensions of sets.

**definition**
*starset* :: $'a\ set \Rightarrow 'a\ star\ set$ (*∗s∗* - [80] 80) **where**
$\textit{starset } A = \textit{Iset } (\textit{star-of } A)$

**declare** *starset-def* [*transfer-unfold*]

**lemma** *starset-mem*: $(\textit{star-of } x \in *s* A) = (x \in A)$
⟨*proof*⟩

**lemma** *starset-UNIV*: $*s* (\textit{UNIV}::'a\ set) = (\textit{UNIV}::'a\ star\ set)$
⟨*proof*⟩

**lemma** *starset-empty*: $*s* \{\} = \{\}$
⟨*proof*⟩

**lemma** *starset-insert*: $*s* (\textit{insert } x\ A) = \textit{insert } (\textit{star-of } x)\ ( *s* A)$
⟨*proof*⟩

**lemma** *starset-Un*: $*s* (A \cup B) = *s* A \cup *s* B$
⟨*proof*⟩

**lemma** *starset-Int*: $*s* (A \cap B) = *s* A \cap *s* B$
⟨*proof*⟩

**lemma** *starset-Compl*: $*s* -A = -( *s* A)$
⟨*proof*⟩

**lemma** *starset-diff*: $*s* (A - B) = *s* A - *s* B$
⟨*proof*⟩

**lemma** *starset-image*: $*s* (f \text{ ` } A) = ( *f* f) \text{ ` } ( *s* A)$
⟨*proof*⟩

**lemma** *starset-vimage*: $*s* (f -\text{`} A) = ( *f* f) -\text{`} ( *s* A)$
⟨*proof*⟩

**lemma** *starset-subset*: $( *s* A \subseteq *s* B) = (A \subseteq B)$
⟨*proof*⟩

**lemma** *starset-eq*: $( *s* A = *s* B) = (A = B)$
⟨*proof*⟩

**lemmas** *starset-simps* [*simp*] =
*starset-mem* *starset-UNIV*
*starset-empty* *starset-insert*

*starset-Un* *starset-Int*
*starset-Compl* *starset-diff*
*starset-image* *starset-vimage*
*starset-subset* *starset-eq*

**end**

# 24 StarClasses: Class Instances

**theory** *StarClasses*
**imports** *StarDef*
**begin**

## 24.1 Syntactic classes

**instance** *star* :: (*zero*) *zero*
*star-zero-def*: $0 \equiv$ *star-of 0* ⟨*proof*⟩

**instance** *star* :: (*one*) *one*
*star-one-def*: $1 \equiv$ *star-of 1* ⟨*proof*⟩

**instance** *star* :: (*plus*) *plus*
*star-add-def*: $(op\ +) \equiv *f2*\ (op\ +)$ ⟨*proof*⟩

**instance** *star* :: (*times*) *times*
*star-mult-def*: $(op\ *) \equiv *f2*\ (op\ *)$ ⟨*proof*⟩

**instance** *star* :: (*minus*) *minus*
*star-minus-def*: $uminus \equiv *f*\ uminus$
*star-diff-def*: $(op\ -) \equiv *f2*\ (op\ -)$ ⟨*proof*⟩

**instance** *star* :: (*abs*) *abs*
*star-abs-def*: $abs \equiv *f*\ abs$ ⟨*proof*⟩

**instance** *star* :: (*sgn*) *sgn*
*star-sgn-def*: $sgn \equiv *f*\ sgn$ ⟨*proof*⟩

**instance** *star* :: (*inverse*) *inverse*
*star-divide-def*: $(op\ /) \equiv *f2*\ (op\ /)$
*star-inverse-def*: $inverse \equiv *f*\ inverse$ ⟨*proof*⟩

**instance** *star* :: (*number*) *number*
*star-number-def*: *number-of b* $\equiv$ *star-of* (*number-of b*) ⟨*proof*⟩

**instance** *star* :: (*Divides.div*) *Divides.div*
*star-div-def*: $(op\ div) \equiv *f2*\ (op\ div)$
*star-mod-def*: $(op\ mod) \equiv *f2*\ (op\ mod)$ ⟨*proof*⟩

**instance** *star* :: (*power*) *power*
*star-power-def*: $(op\ \hat{}\ ) \equiv \lambda x\ n.\ (\ *f*\ (\lambda x.\ x\ \hat{}\ n))\ x$ ⟨*proof*⟩

**instance** *star* :: (*ord*) *ord*
*star-le-def*: $(op \le) \equiv *p2*\ (op \le)$
*star-less-def*: $(op <) \equiv *p2*\ (op <)$ ⟨*proof*⟩

**lemmas** *star-class-defs* [*transfer-unfold*] =
*star-zero-def* *star-one-def* *star-number-def*
*star-add-def* *star-diff-def* *star-minus-def*
*star-mult-def* *star-divide-def* *star-inverse-def*
*star-le-def* *star-less-def* *star-abs-def* *star-sgn-def*
*star-div-def* *star-mod-def* *star-power-def*

Class operations preserve standard elements

**lemma** *Standard-zero*: $0 \in Standard$
⟨*proof*⟩

**lemma** *Standard-one*: $1 \in Standard$
⟨*proof*⟩

**lemma** *Standard-number-of*: $number\text{-}of\ b \in Standard$
⟨*proof*⟩

**lemma** *Standard-add*: $[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow x + y \in Standard$
⟨*proof*⟩

**lemma** *Standard-diff*: $[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow x - y \in Standard$
⟨*proof*⟩

**lemma** *Standard-minus*: $x \in Standard \Longrightarrow -\ x \in Standard$
⟨*proof*⟩

**lemma** *Standard-mult*: $[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow x * y \in Standard$
⟨*proof*⟩

**lemma** *Standard-divide*: $[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow x\ /\ y \in Standard$
⟨*proof*⟩

**lemma** *Standard-inverse*: $x \in Standard \Longrightarrow inverse\ x \in Standard$
⟨*proof*⟩

**lemma** *Standard-abs*: $x \in Standard \Longrightarrow abs\ x \in Standard$
⟨*proof*⟩

**lemma** *Standard-div*: $[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow x\ div\ y \in Standard$
⟨*proof*⟩

**lemma** *Standard-mod*: $[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow x\ mod\ y \in Standard$

⟨*proof*⟩

**lemma** *Standard-power*: $x \in Standard \Longrightarrow x \hat{}\ n \in Standard$
⟨*proof*⟩

**lemmas** *Standard-simps* [*simp*] =
*Standard-zero Standard-one Standard-number-of*
*Standard-add Standard-diff Standard-minus*
*Standard-mult Standard-divide Standard-inverse*
*Standard-abs Standard-div Standard-mod*
*Standard-power*

*star-of* preserves class operations

**lemma** *star-of-add*: $star\text{-}of\ (x + y) = star\text{-}of\ x + star\text{-}of\ y$
⟨*proof*⟩

**lemma** *star-of-diff*: $star\text{-}of\ (x - y) = star\text{-}of\ x - star\text{-}of\ y$
⟨*proof*⟩

**lemma** *star-of-minus*: $star\text{-}of\ (-x) = -\ star\text{-}of\ x$
⟨*proof*⟩

**lemma** *star-of-mult*: $star\text{-}of\ (x * y) = star\text{-}of\ x * star\text{-}of\ y$
⟨*proof*⟩

**lemma** *star-of-divide*: $star\text{-}of\ (x\ /\ y) = star\text{-}of\ x\ /\ star\text{-}of\ y$
⟨*proof*⟩

**lemma** *star-of-inverse*: $star\text{-}of\ (inverse\ x) = inverse\ (star\text{-}of\ x)$
⟨*proof*⟩

**lemma** *star-of-div*: $star\text{-}of\ (x\ div\ y) = star\text{-}of\ x\ div\ star\text{-}of\ y$
⟨*proof*⟩

**lemma** *star-of-mod*: $star\text{-}of\ (x\ mod\ y) = star\text{-}of\ x\ mod\ star\text{-}of\ y$
⟨*proof*⟩

**lemma** *star-of-power*: $star\text{-}of\ (x \hat{}\ n) = star\text{-}of\ x \hat{}\ n$
⟨*proof*⟩

**lemma** *star-of-abs*: $star\text{-}of\ (abs\ x) = abs\ (star\text{-}of\ x)$
⟨*proof*⟩

*star-of* preserves numerals

**lemma** *star-of-zero*: $star\text{-}of\ 0 = 0$
⟨*proof*⟩

**lemma** *star-of-one*: $star\text{-}of\ 1 = 1$
⟨*proof*⟩

**lemma** *star-of-number-of*: *star-of* (*number-of* $x$) = *number-of* $x$
⟨*proof*⟩

*star-of* preserves orderings

**lemma** *star-of-less*: (*star-of* $x$ < *star-of* $y$) = ($x$ < $y$)
⟨*proof*⟩

**lemma** *star-of-le*: (*star-of* $x$ ≤ *star-of* $y$) = ($x$ ≤ $y$)
⟨*proof*⟩

**lemma** *star-of-eq*: (*star-of* $x$ = *star-of* $y$) = ($x$ = $y$)
⟨*proof*⟩

As above, for 0

**lemmas** *star-of-0-less* = *star-of-less* [*of 0*, *simplified star-of-zero*]
**lemmas** *star-of-0-le* = *star-of-le* [*of 0*, *simplified star-of-zero*]
**lemmas** *star-of-0-eq* = *star-of-eq* [*of 0*, *simplified star-of-zero*]

**lemmas** *star-of-less-0* = *star-of-less* [*of - 0*, *simplified star-of-zero*]
**lemmas** *star-of-le-0* = *star-of-le* [*of - 0*, *simplified star-of-zero*]
**lemmas** *star-of-eq-0* = *star-of-eq* [*of - 0*, *simplified star-of-zero*]

As above, for 1

**lemmas** *star-of-1-less* = *star-of-less* [*of 1*, *simplified star-of-one*]
**lemmas** *star-of-1-le* = *star-of-le* [*of 1*, *simplified star-of-one*]
**lemmas** *star-of-1-eq* = *star-of-eq* [*of 1*, *simplified star-of-one*]

**lemmas** *star-of-less-1* = *star-of-less* [*of - 1*, *simplified star-of-one*]
**lemmas** *star-of-le-1* = *star-of-le* [*of - 1*, *simplified star-of-one*]
**lemmas** *star-of-eq-1* = *star-of-eq* [*of - 1*, *simplified star-of-one*]

As above, for numerals

**lemmas** *star-of-number-less* =
  *star-of-less* [*of number-of w*, *standard*, *simplified star-of-number-of*]
**lemmas** *star-of-number-le* =
  *star-of-le* [*of number-of w*, *standard*, *simplified star-of-number-of*]
**lemmas** *star-of-number-eq* =
  *star-of-eq* [*of number-of w*, *standard*, *simplified star-of-number-of*]

**lemmas** *star-of-less-number* =
  *star-of-less* [*of - number-of w*, *standard*, *simplified star-of-number-of*]
**lemmas** *star-of-le-number* =
  *star-of-le* [*of - number-of w*, *standard*, *simplified star-of-number-of*]
**lemmas** *star-of-eq-number* =
  *star-of-eq* [*of - number-of w*, *standard*, *simplified star-of-number-of*]

**lemmas** *star-of-simps* [*simp*] =

*star-of-add* *star-of-diff* *star-of-minus*
*star-of-mult* *star-of-divide* *star-of-inverse*
*star-of-div* *star-of-mod*
*star-of-power* *star-of-abs*
*star-of-zero* *star-of-one* *star-of-number-of*
*star-of-less* *star-of-le* *star-of-eq*
*star-of-0-less* *star-of-0-le* *star-of-0-eq*
*star-of-less-0* *star-of-le-0* *star-of-eq-0*
*star-of-1-less* *star-of-1-le* *star-of-1-eq*
*star-of-less-1* *star-of-le-1* *star-of-eq-1*
*star-of-number-less* *star-of-number-le* *star-of-number-eq*
*star-of-less-number* *star-of-le-number* *star-of-eq-number*

## 24.2 Ordering and lattice classes

**instance** *star* :: (*order*) *order*
⟨*proof*⟩

**instance** *star* :: (*lower-semilattice*) *lower-semilattice*
*star-inf-def* [*transfer-unfold*]: *inf* ≡ *f2* *inf*
⟨*proof*⟩

**instance** *star* :: (*upper-semilattice*) *upper-semilattice*
*star-sup-def* [*transfer-unfold*]: *sup* ≡ *f2* *sup*
⟨*proof*⟩

**instance** *star* :: (*lattice*) *lattice* ⟨*proof*⟩

**instance** *star* :: (*distrib-lattice*) *distrib-lattice*
⟨*proof*⟩

**lemma** *Standard-inf* [*simp*]:
$[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow inf\ x\ y \in Standard$
⟨*proof*⟩

**lemma** *Standard-sup* [*simp*]:
$[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow sup\ x\ y \in Standard$
⟨*proof*⟩

**lemma** *star-of-inf* [*simp*]: *star-of* (*inf x y*) = *inf* (*star-of x*) (*star-of y*)
⟨*proof*⟩

**lemma** *star-of-sup* [*simp*]: *star-of* (*sup x y*) = *sup* (*star-of x*) (*star-of y*)
⟨*proof*⟩

**instance** *star* :: (*linorder*) *linorder*
⟨*proof*⟩

**lemma** *star-max-def* [*transfer-unfold*]: *max* = *f2* *max*

⟨proof⟩

**lemma** *star-min-def* [*transfer-unfold*]: *min* = *∗f2∗ min*
⟨proof⟩

**lemma** *Standard-max* [*simp*]:
$[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow max\ x\ y \in Standard$
⟨proof⟩

**lemma** *Standard-min* [*simp*]:
$[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow min\ x\ y \in Standard$
⟨proof⟩

**lemma** *star-of-max* [*simp*]: *star-of* (*max x y*) = *max* (*star-of x*) (*star-of y*)
⟨proof⟩

**lemma** *star-of-min* [*simp*]: *star-of* (*min x y*) = *min* (*star-of x*) (*star-of y*)
⟨proof⟩

## 24.3 Ordered group classes

**instance** *star* :: (*semigroup-add*) *semigroup-add*
⟨proof⟩

**instance** *star* :: (*ab-semigroup-add*) *ab-semigroup-add*
⟨proof⟩

**instance** *star* :: (*semigroup-mult*) *semigroup-mult*
⟨proof⟩

**instance** *star* :: (*ab-semigroup-mult*) *ab-semigroup-mult*
⟨proof⟩

**instance** *star* :: (*comm-monoid-add*) *comm-monoid-add*
⟨proof⟩

**instance** *star* :: (*monoid-mult*) *monoid-mult*
⟨proof⟩

**instance** *star* :: (*comm-monoid-mult*) *comm-monoid-mult*
⟨proof⟩

**instance** *star* :: (*cancel-semigroup-add*) *cancel-semigroup-add*
⟨proof⟩

**instance** *star* :: (*cancel-ab-semigroup-add*) *cancel-ab-semigroup-add*
⟨proof⟩

**instance** *star* :: (*ab-group-add*) *ab-group-add*

⟨*proof*⟩

**instance** *star* :: (*pordered-ab-semigroup-add*) *pordered-ab-semigroup-add*
⟨*proof*⟩

**instance** *star* :: (*pordered-cancel-ab-semigroup-add*) *pordered-cancel-ab-semigroup-add*
⟨*proof*⟩

**instance** *star* :: (*pordered-ab-semigroup-add-imp-le*) *pordered-ab-semigroup-add-imp-le*
⟨*proof*⟩

**instance** *star* :: (*pordered-comm-monoid-add*) *pordered-comm-monoid-add* ⟨*proof*⟩
**instance** *star* :: (*pordered-ab-group-add*) *pordered-ab-group-add* ⟨*proof*⟩

**instance** *star* :: (*pordered-ab-group-add-abs*) *pordered-ab-group-add-abs*
⟨*proof*⟩

**instance** *star* :: (*ordered-cancel-ab-semigroup-add*) *ordered-cancel-ab-semigroup-add*
⟨*proof*⟩
**instance** *star* :: (*lordered-ab-group-add-meet*) *lordered-ab-group-add-meet* ⟨*proof*⟩
**instance** *star* :: (*lordered-ab-group-add-meet*) *lordered-ab-group-add-meet* ⟨*proof*⟩
**instance** *star* :: (*lordered-ab-group-add*) *lordered-ab-group-add* ⟨*proof*⟩

**instance** *star* :: (*lordered-ab-group-add-abs*) *lordered-ab-group-add-abs*
⟨*proof*⟩

## 24.4 Ring and field classes

**instance** *star* :: (*semiring*) *semiring*
⟨*proof*⟩

**instance** *star* :: (*semiring-0*) *semiring-0*
⟨*proof*⟩

**instance** *star* :: (*semiring-0-cancel*) *semiring-0-cancel* ⟨*proof*⟩

**instance** *star* :: (*comm-semiring*) *comm-semiring*
⟨*proof*⟩

**instance** *star* :: (*comm-semiring-0*) *comm-semiring-0* ⟨*proof*⟩
**instance** *star* :: (*comm-semiring-0-cancel*) *comm-semiring-0-cancel* ⟨*proof*⟩

**instance** *star* :: (*zero-neq-one*) *zero-neq-one*
⟨*proof*⟩

**instance** *star* :: (*semiring-1*) *semiring-1* ⟨*proof*⟩
**instance** *star* :: (*comm-semiring-1*) *comm-semiring-1* ⟨*proof*⟩

**instance** *star* :: (*no-zero-divisors*) *no-zero-divisors*

⟨*proof*⟩

**instance** *star* :: (*semiring-1-cancel*) *semiring-1-cancel* ⟨*proof*⟩
**instance** *star* :: (*comm-semiring-1-cancel*) *comm-semiring-1-cancel* ⟨*proof*⟩
**instance** *star* :: (*ring*) *ring* ⟨*proof*⟩
**instance** *star* :: (*comm-ring*) *comm-ring* ⟨*proof*⟩
**instance** *star* :: (*ring-1*) *ring-1* ⟨*proof*⟩
**instance** *star* :: (*comm-ring-1*) *comm-ring-1* ⟨*proof*⟩
**instance** *star* :: (*ring-no-zero-divisors*) *ring-no-zero-divisors* ⟨*proof*⟩
**instance** *star* :: (*ring-1-no-zero-divisors*) *ring-1-no-zero-divisors* ⟨*proof*⟩
**instance** *star* :: (*idom*) *idom* ⟨*proof*⟩

**instance** *star* :: (*division-ring*) *division-ring*
⟨*proof*⟩

**instance** *star* :: (*field*) *field*
⟨*proof*⟩

**instance** *star* :: (*division-by-zero*) *division-by-zero*
⟨*proof*⟩

**instance** *star* :: (*pordered-semiring*) *pordered-semiring*
⟨*proof*⟩

**instance** *star* :: (*pordered-cancel-semiring*) *pordered-cancel-semiring* ⟨*proof*⟩

**instance** *star* :: (*ordered-semiring-strict*) *ordered-semiring-strict*
⟨*proof*⟩

**instance** *star* :: (*pordered-comm-semiring*) *pordered-comm-semiring*
⟨*proof*⟩

**instance** *star* :: (*pordered-cancel-comm-semiring*) *pordered-cancel-comm-semiring*
⟨*proof*⟩

**instance** *star* :: (*ordered-comm-semiring-strict*) *ordered-comm-semiring-strict*
⟨*proof*⟩

**instance** *star* :: (*pordered-ring*) *pordered-ring* ⟨*proof*⟩
**instance** *star* :: (*pordered-ring-abs*) *pordered-ring-abs*
⟨*proof*⟩
**instance** *star* :: (*lordered-ring*) *lordered-ring* ⟨*proof*⟩

**instance** *star* :: (*abs-if*) *abs-if*
⟨*proof*⟩

**instance** *star* :: (*sgn-if*) *sgn-if*
⟨*proof*⟩

**instance** *star* :: (*ordered-ring-strict*) *ordered-ring-strict* ⟨*proof*⟩
**instance** *star* :: (*pordered-comm-ring*) *pordered-comm-ring* ⟨*proof*⟩

**instance** *star* :: (*ordered-semidom*) *ordered-semidom*
⟨*proof*⟩

**instance** *star* :: (*ordered-idom*) *ordered-idom* ⟨*proof*⟩
**instance** *star* :: (*ordered-field*) *ordered-field* ⟨*proof*⟩

## 24.5 Power classes

Proving the class axiom *power-Suc* for type *'a star* is a little tricky, because it quantifies over values of type *nat*. The transfer principle does not handle quantification over non-star types in general, but we can work around this by fixing an arbitrary *nat* value, and then applying the transfer principle.

**instance** *star* :: (*recpower*) *recpower*
⟨*proof*⟩

## 24.6 Number classes

**lemma** *star-of-nat-def* [*transfer-unfold*]: $\mathit{of\text{-}nat}\ n = \mathit{star\text{-}of}\ (\mathit{of\text{-}nat}\ n)$
⟨*proof*⟩

**lemma** *Standard-of-nat* [*simp*]: $\mathit{of\text{-}nat}\ n \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *star-of-of-nat* [*simp*]: $\mathit{star\text{-}of}\ (\mathit{of\text{-}nat}\ n) = \mathit{of\text{-}nat}\ n$
⟨*proof*⟩

**lemma** *star-of-int-def* [*transfer-unfold*]: $\mathit{of\text{-}int}\ z = \mathit{star\text{-}of}\ (\mathit{of\text{-}int}\ z)$
⟨*proof*⟩

**lemma** *Standard-of-int* [*simp*]: $\mathit{of\text{-}int}\ z \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *star-of-of-int* [*simp*]: $\mathit{star\text{-}of}\ (\mathit{of\text{-}int}\ z) = \mathit{of\text{-}int}\ z$
⟨*proof*⟩

**instance** *star* :: (*semiring-char-0*) *semiring-char-0*
⟨*proof*⟩

**instance** *star* :: (*ring-char-0*) *ring-char-0* ⟨*proof*⟩

**instance** *star* :: (*number-ring*) *number-ring*
⟨*proof*⟩

## 24.7 Finite class

**lemma** *starset-finite*: $\mathit{finite}\ A \Longrightarrow {*}s{*}\ A = \mathit{star\text{-}of}\ `\ A$

⟨*proof*⟩

**instance** *star* :: (*finite*) *finite*
⟨*proof*⟩

**end**

# 25 HyperNat: Hypernatural numbers

**theory** *HyperNat*
**imports** *StarClasses*
**begin**

**types** *hypnat* = *nat star*

**abbreviation**
  *hypnat-of-nat* :: *nat* => *nat star* **where**
  *hypnat-of-nat* == *star-of*

**definition**
  *hSuc* :: *hypnat* => *hypnat* **where**
  *hSuc-def* [*transfer-unfold*]: *hSuc* = ∗*f*∗ *Suc*

## 25.1 Properties Transferred from Naturals

**lemma** *hSuc-not-zero* [*iff*]: $\bigwedge m.\ hSuc\ m \neq 0$
⟨*proof*⟩

**lemma** *zero-not-hSuc* [*iff*]: $\bigwedge m.\ 0 \neq hSuc\ m$
⟨*proof*⟩

**lemma** *hSuc-hSuc-eq* [*iff*]: $\bigwedge m\ n.\ (hSuc\ m = hSuc\ n) = (m = n)$
⟨*proof*⟩

**lemma** *zero-less-hSuc* [*iff*]: $\bigwedge n.\ 0 < hSuc\ n$
⟨*proof*⟩

**lemma** *hypnat-minus-zero* [*simp*]: !!*z*. *z* − *z* = (*0*::*hypnat*)
⟨*proof*⟩

**lemma** *hypnat-diff-0-eq-0* [*simp*]: !!*n*. (*0*::*hypnat*) − *n* = *0*
⟨*proof*⟩

**lemma** *hypnat-add-is-0* [*iff*]: !!*m n*. (*m*+*n* = (*0*::*hypnat*)) = (*m*=*0* & *n*=*0*)
⟨*proof*⟩

**lemma** *hypnat-diff-diff-left*: !!*i j k*. (*i*::*hypnat*) − *j* − *k* = *i* − (*j*+*k*)
⟨*proof*⟩

**lemma** *hypnat-diff-commute*: $!!i\ j\ k.\ (i::hypnat) - j - k = i-k-j$
⟨*proof*⟩

**lemma** *hypnat-diff-add-inverse* [*simp*]: $!!m\ n.\ ((n::hypnat) + m) - n = m$
⟨*proof*⟩

**lemma** *hypnat-diff-add-inverse2* [*simp*]: $!!m\ n.\ ((m::hypnat) + n) - n = m$
⟨*proof*⟩

**lemma** *hypnat-diff-cancel* [*simp*]: $!!k\ m\ n.\ ((k::hypnat) + m) - (k+n) = m - n$
⟨*proof*⟩

**lemma** *hypnat-diff-cancel2* [*simp*]: $!!k\ m\ n.\ ((m::hypnat) + k) - (n+k) = m - n$
⟨*proof*⟩

**lemma** *hypnat-diff-add-0* [*simp*]: $!!m\ n.\ (n::hypnat) - (n+m) = (0::hypnat)$
⟨*proof*⟩

**lemma** *hypnat-diff-mult-distrib*: $!!k\ m\ n.\ ((m::hypnat) - n) * k = (m * k) - (n * k)$
⟨*proof*⟩

**lemma** *hypnat-diff-mult-distrib2*: $!!k\ m\ n.\ (k::hypnat) * (m - n) = (k * m) - (k * n)$
⟨*proof*⟩

**lemma** *hypnat-le-zero-cancel* [*iff*]: $!!n.\ (n \leq (0::hypnat)) = (n = 0)$
⟨*proof*⟩

**lemma** *hypnat-mult-is-0* [*simp*]: $!!m\ n.\ (m*n = (0::hypnat)) = (m=0 \mid n=0)$
⟨*proof*⟩

**lemma** *hypnat-diff-is-0-eq* [*simp*]: $!!m\ n.\ ((m::hypnat) - n = 0) = (m \leq n)$
⟨*proof*⟩

**lemma** *hypnat-not-less0* [*iff*]: $!!n.\ \sim n < (0::hypnat)$
⟨*proof*⟩

**lemma** *hypnat-less-one* [*iff*]:
$!!n.\ (n < (1::hypnat)) = (n=0)$
⟨*proof*⟩

**lemma** *hypnat-add-diff-inverse*: $!!m\ n.\ \sim m<n ==> n+(m-n) = (m::hypnat)$
⟨*proof*⟩

**lemma** *hypnat-le-add-diff-inverse* [*simp*]: $!!m\ n.\ n \leq m ==> n+(m-n) = (m::hypnat)$
⟨*proof*⟩

**lemma** *hypnat-le-add-diff-inverse2* [*simp*]: !!*m n*. *n*≤*m* ==> (*m*−*n*)+*n* = (*m*::*hypnat*)
⟨*proof*⟩

**declare** *hypnat-le-add-diff-inverse2* [*OF order-less-imp-le*]

**lemma** *hypnat-le0* [*iff*]: !!*n*. (*0*::*hypnat*) ≤ *n*
⟨*proof*⟩

**lemma** *hypnat-le-add1* [*simp*]: !!*x n*. (*x*::*hypnat*) ≤ *x* + *n*
⟨*proof*⟩

**lemma** *hypnat-add-self-le* [*simp*]: !!*x n*. (*x*::*hypnat*) ≤ *n* + *x*
⟨*proof*⟩

**lemma** *hypnat-add-one-self-less* [*simp*]: (*x*::*hypnat*) < *x* + (*1*::*hypnat*)
⟨*proof*⟩

**lemma** *hypnat-neq0-conv* [*iff*]: !!*n*. (*n* ≠ *0*) = (*0* < (*n*::*hypnat*))
⟨*proof*⟩

**lemma** *hypnat-gt-zero-iff*: ((*0*::*hypnat*) < *n*) = ((*1*::*hypnat*) ≤ *n*)
⟨*proof*⟩

**lemma** *hypnat-gt-zero-iff2*: (*0* < *n*) = (∃ *m*. *n* = *m* + (*1*::*hypnat*))
⟨*proof*⟩

**lemma** *hypnat-add-self-not-less*: ~ (*x* + *y* < (*x*::*hypnat*))
⟨*proof*⟩

**lemma** *hypnat-diff-split*:
*P*(*a* − *b*::*hypnat*) = ((*a*<*b* −−> *P 0*) & (*ALL d*. *a* = *b* + *d* −−> *P d*))
— elimination of − on *hypnat*
⟨*proof*⟩

## 25.2 Properties of the set of embedded natural numbers

**lemma** *of-nat-eq-star-of* [*simp*]: *of-nat* = *star-of*
⟨*proof*⟩

**lemma** *Nats-eq-Standard*: (*Nats* :: *nat star set*) = *Standard*
⟨*proof*⟩

**lemma** *hypnat-of-nat-mem-Nats* [*simp*]: *hypnat-of-nat n* ∈ *Nats*
⟨*proof*⟩

**lemma** *hypnat-of-nat-one* [*simp*]: *hypnat-of-nat* (*Suc 0*) = (*1*::*hypnat*)
⟨*proof*⟩

**lemma** *hypnat-of-nat-Suc* [*simp*]:

*hypnat-of-nat* (*Suc n*) = *hypnat-of-nat n* + (*1*::*hypnat*)
⟨*proof*⟩

**lemma** *of-nat-eq-add* [*rule-format*]:
∀ *d*::*hypnat*. *of-nat m* = *of-nat n* + *d* −−> *d* ∈ *range of-nat*
⟨*proof*⟩

**lemma** *Nats-diff* [*simp*]: [|*a* ∈ *Nats*; *b* ∈ *Nats*|] ==> (*a*−*b* :: *hypnat*) ∈ *Nats*
⟨*proof*⟩

## 25.3 Infinite Hypernatural Numbers – *HNatInfinite*

**definition**

*HNatInfinite* :: *hypnat set* **where**
*HNatInfinite* = {*n*. *n* ∉ *Nats*}

**lemma** *Nats-not-HNatInfinite-iff*: (*x* ∈ *Nats*) = (*x* ∉ *HNatInfinite*)
⟨*proof*⟩

**lemma** *HNatInfinite-not-Nats-iff*: (*x* ∈ *HNatInfinite*) = (*x* ∉ *Nats*)
⟨*proof*⟩

**lemma** *star-of-neq-HNatInfinite*: *N* ∈ *HNatInfinite* ⟹ *star-of n* ≠ *N*
⟨*proof*⟩

**lemma** *star-of-Suc-lessI*:
⋀*N*. ⟦*star-of n* < *N*; *star-of* (*Suc n*) ≠ *N*⟧ ⟹ *star-of* (*Suc n*) < *N*
⟨*proof*⟩

**lemma** *star-of-less-HNatInfinite*:
**assumes** *N*: *N* ∈ *HNatInfinite*
**shows** *star-of n* < *N*
⟨*proof*⟩

**lemma** *star-of-le-HNatInfinite*: *N* ∈ *HNatInfinite* ⟹ *star-of n* ≤ *N*
⟨*proof*⟩

### 25.3.1 Closure Rules

**lemma** *Nats-less-HNatInfinite*: ⟦*x* ∈ *Nats*; *y* ∈ *HNatInfinite*⟧ ⟹ *x* < *y*
⟨*proof*⟩

**lemma** *Nats-le-HNatInfinite*: ⟦*x* ∈ *Nats*; *y* ∈ *HNatInfinite*⟧ ⟹ *x* ≤ *y*
⟨*proof*⟩

**lemma** *zero-less-HNatInfinite*: *x* ∈ *HNatInfinite* ⟹ *0* < *x*
⟨*proof*⟩

**lemma** *one-less-HNatInfinite*: *x* ∈ *HNatInfinite* ⟹ *1* < *x*

⟨*proof*⟩

**lemma** *one-le-HNatInfinite*: $x \in HNatInfinite \Longrightarrow 1 \leq x$
⟨*proof*⟩

**lemma** *zero-not-mem-HNatInfinite* [*simp*]: $0 \notin HNatInfinite$
⟨*proof*⟩

**lemma** *Nats-downward-closed*:
$[\![x \in Nats; (y::hypnat) \leq x]\!] \Longrightarrow y \in Nats$
⟨*proof*⟩

**lemma** *HNatInfinite-upward-closed*:
$[\![x \in HNatInfinite; x \leq y]\!] \Longrightarrow y \in HNatInfinite$
⟨*proof*⟩

**lemma** *HNatInfinite-add*: $x \in HNatInfinite \Longrightarrow x + y \in HNatInfinite$
⟨*proof*⟩

**lemma** *HNatInfinite-add-one*: $x \in HNatInfinite \Longrightarrow x + 1 \in HNatInfinite$
⟨*proof*⟩

**lemma** *HNatInfinite-diff*:
$[\![x \in HNatInfinite; y \in Nats]\!] \Longrightarrow x - y \in HNatInfinite$
⟨*proof*⟩

**lemma** *HNatInfinite-is-Suc*: $x \in HNatInfinite ==> \exists y.\ x = y + (1::hypnat)$
⟨*proof*⟩

## 25.4 Existence of an infinite hypernatural number

**definition**

*whn* :: *hypnat* **where**
*hypnat-omega-def*: $whn = star\text{-}n\ (\%n::nat.\ n)$

**lemma** *hypnat-of-nat-neq-whn*: $hypnat\text{-}of\text{-}nat\ n \neq whn$
⟨*proof*⟩

**lemma** *whn-neq-hypnat-of-nat*: $whn \neq hypnat\text{-}of\text{-}nat\ n$
⟨*proof*⟩

**lemma** *whn-not-Nats* [*simp*]: $whn \notin Nats$
⟨*proof*⟩

**lemma** *HNatInfinite-whn* [*simp*]: $whn \in HNatInfinite$
⟨*proof*⟩

**lemma** *lemma-unbounded-set* [*simp*]: $\{n::nat.\ m < n\} \in FreeUltrafilterNat$

⟨*proof*⟩

**lemma** *Compl-Collect-le*: $- \{n::nat.\ N \le n\} = \{n.\ n < N\}$
⟨*proof*⟩

**lemma** *hypnat-of-nat-eq*:
  *hypnat-of-nat m = star-n* ($\%n::nat.\ m$)
⟨*proof*⟩

**lemma** *SHNat-eq*: $Nats = \{n.\ \exists N.\ n = hypnat\text{-}of\text{-}nat\ N\}$
⟨*proof*⟩

**lemma** *Nats-less-whn*: $n \in Nats \Longrightarrow n < whn$
⟨*proof*⟩

**lemma** *Nats-le-whn*: $n \in Nats \Longrightarrow n \le whn$
⟨*proof*⟩

**lemma** *hypnat-of-nat-less-whn* [*simp*]: *hypnat-of-nat* $n < whn$
⟨*proof*⟩

**lemma** *hypnat-of-nat-le-whn* [*simp*]: *hypnat-of-nat* $n \le whn$
⟨*proof*⟩

**lemma** *hypnat-zero-less-hypnat-omega* [*simp*]: $0 < whn$
⟨*proof*⟩

**lemma** *hypnat-one-less-hypnat-omega* [*simp*]: $1 < whn$
⟨*proof*⟩

### 25.4.1 Alternative characterization of the set of infinite hypernaturals

$HNatInfinite = \{N.\ \forall n \in \mathbb{N}.\ n < N\}$

**lemma** *HNatInfinite-FreeUltrafilterNat-lemma*:
  $\forall N::nat.\ \{n.\ f\ n \ne N\} \in FreeUltrafilterNat$
  $==> \{n.\ N < f\ n\} \in FreeUltrafilterNat$
⟨*proof*⟩

**lemma** *HNatInfinite-iff*: $HNatInfinite = \{N.\ \forall n \in Nats.\ n < N\}$
⟨*proof*⟩

### 25.4.2 Alternative Characterization of *HNatInfinite* using Free Ultrafilter

**lemma** *HNatInfinite-FreeUltrafilterNat*:
  *star-n* $X \in HNatInfinite ==> \forall u.\ \{n.\ u < X\ n\}:\ FreeUltrafilterNat$
⟨*proof*⟩

**lemma** *FreeUltrafilterNat-HNatInfinite*:
$\forall u.\ \{n.\ u < X\ n\}:\ \mathit{FreeUltrafilterNat} \Longrightarrow \mathit{star\text{-}n}\ X \in \mathit{HNatInfinite}$
⟨*proof*⟩

**lemma** *HNatInfinite-FreeUltrafilterNat-iff*:
$(\mathit{star\text{-}n}\ X \in \mathit{HNatInfinite}) = (\forall u.\ \{n.\ u < X\ n\}:\ \mathit{FreeUltrafilterNat})$
⟨*proof*⟩

## 25.5 Embedding of the Hypernaturals into other types

**definition**
*of-hypnat* :: *hypnat* ⇒ *'a::semiring-1-cancel star* **where**
*of-hypnat-def* [*transfer-unfold*]: *of-hypnat* = *∗f∗ of-nat*

**lemma** *of-hypnat-0* [*simp*]: *of-hypnat 0* = *0*
⟨*proof*⟩

**lemma** *of-hypnat-1* [*simp*]: *of-hypnat 1* = *1*
⟨*proof*⟩

**lemma** *of-hypnat-hSuc*: $\bigwedge m.$ *of-hypnat* (*hSuc m*) = *1* + *of-hypnat m*
⟨*proof*⟩

**lemma** *of-hypnat-add* [*simp*]:
$\bigwedge m\ n.$ *of-hypnat* (*m* + *n*) = *of-hypnat m* + *of-hypnat n*
⟨*proof*⟩

**lemma** *of-hypnat-mult* [*simp*]:
$\bigwedge m\ n.$ *of-hypnat* (*m* ∗ *n*) = *of-hypnat m* ∗ *of-hypnat n*
⟨*proof*⟩

**lemma** *of-hypnat-less-iff* [*simp*]:
$\bigwedge m\ n.$ (*of-hypnat m* < (*of-hypnat n::'a::ordered-semidom star*)) = (*m* < *n*)
⟨*proof*⟩

**lemma** *of-hypnat-0-less-iff* [*simp*]:
$\bigwedge n.$ (*0* < (*of-hypnat n::'a::ordered-semidom star*)) = (*0* < *n*)
⟨*proof*⟩

**lemma** *of-hypnat-less-0-iff* [*simp*]:
$\bigwedge m.$ ¬ (*of-hypnat m::'a::ordered-semidom star*) < *0*
⟨*proof*⟩

**lemma** *of-hypnat-le-iff* [*simp*]:
$\bigwedge m\ n.$ (*of-hypnat m* ≤ (*of-hypnat n::'a::ordered-semidom star*)) = (*m* ≤ *n*)
⟨*proof*⟩

**lemma** *of-hypnat-0-le-iff* [*simp*]:
$\bigwedge n.$ *0* ≤ (*of-hypnat n::'a::ordered-semidom star*)

⟨*proof*⟩

**lemma** *of-hypnat-le-0-iff* [*simp*]:
$\bigwedge m.\ ((\textit{of-hypnat}\ m{::}'a{::}\textit{ordered-semidom star}) \leq 0) = (m = 0)$
⟨*proof*⟩

**lemma** *of-hypnat-eq-iff* [*simp*]:
$\bigwedge m\ n.\ (\textit{of-hypnat}\ m = (\textit{of-hypnat}\ n{::}'a{::}\textit{ordered-semidom star})) = (m = n)$
⟨*proof*⟩

**lemma** *of-hypnat-eq-0-iff* [*simp*]:
$\bigwedge m.\ ((\textit{of-hypnat}\ m{::}'a{::}\textit{ordered-semidom star}) = 0) = (m = 0)$
⟨*proof*⟩

**lemma** *HNatInfinite-of-hypnat-gt-zero*:
$N \in \textit{HNatInfinite} \Longrightarrow (0{::}'a{::}\textit{ordered-semidom star}) < \textit{of-hypnat}\ N$
⟨*proof*⟩

**end**

# 26 HyperDef: Construction of Hyperreals Using Ultrafilters

**theory** *HyperDef*
**imports** *HyperNat ../Real/Real*
**uses** (*hypreal-arith.ML*)
**begin**

**types** *hypreal* = *real star*

**abbreviation**
*hypreal-of-real* :: *real* => *real star* **where**
*hypreal-of-real* == *star-of*

**abbreviation**
*hypreal-of-hypnat* :: *hypnat* ⇒ *hypreal* **where**
*hypreal-of-hypnat* ≡ *of-hypnat*

**definition**
*omega* :: *hypreal* **where**
— an infinite number = [<*1*,*2*,*3*,...>]
*omega* = *star-n* ($\lambda n.$ *real* (*Suc n*))

**definition**
*epsilon* :: *hypreal* **where**
— an infinitesimal number = [<*1*,*1*/*2*,*1*/*3*,...>]
*epsilon* = *star-n* ($\lambda n.$ *inverse* (*real* (*Suc n*)))

**notation** (*xsymbols*)
*omega* ($\omega$) **and**
*epsilon* ($\varepsilon$)

**notation** (*HTML* **output**)
*omega* ($\omega$) **and**
*epsilon* ($\varepsilon$)

## 26.1 Real vector class instances

**instance** *star* :: (*scaleR*) *scaleR* ⟨*proof*⟩

**defs** (**overloaded**)
*star-scaleR-def* [*transfer-unfold*]: *scaleR r* ≡ ∗*f*∗ (*scaleR r*)

**lemma** *Standard-scaleR* [*simp*]: *x* ∈ *Standard* ⟹ *scaleR r x* ∈ *Standard*
⟨*proof*⟩

**lemma** *star-of-scaleR* [*simp*]: *star-of* (*scaleR r x*) = *scaleR r* (*star-of x*)
⟨*proof*⟩

**instance** *star* :: (*real-vector*) *real-vector*
⟨*proof*⟩

**instance** *star* :: (*real-algebra*) *real-algebra*
⟨*proof*⟩

**instance** *star* :: (*real-algebra-1*) *real-algebra-1* ⟨*proof*⟩

**instance** *star* :: (*real-div-algebra*) *real-div-algebra* ⟨*proof*⟩

**instance** *star* :: (*real-field*) *real-field* ⟨*proof*⟩

**lemma** *star-of-real-def* [*transfer-unfold*]: *of-real r* = *star-of* (*of-real r*)
⟨*proof*⟩

**lemma** *Standard-of-real* [*simp*]: *of-real r* ∈ *Standard*
⟨*proof*⟩

**lemma** *star-of-of-real* [*simp*]: *star-of* (*of-real r*) = *of-real r*
⟨*proof*⟩

**lemma** *of-real-eq-star-of* [*simp*]: *of-real* = *star-of*
⟨*proof*⟩

**lemma** *Reals-eq-Standard*: (*Reals* :: *hypreal set*) = *Standard*
⟨*proof*⟩

## 26.2 Injection from *hypreal*

**definition**
*of-hypreal* :: *hypreal* $\Rightarrow$ *'a::real-algebra-1 star* **where**
*of-hypreal* = *f* *of-real*

**declare** *of-hypreal-def* [*transfer-unfold*]

**lemma** *Standard-of-hypreal* [*simp*]:
$r \in$ *Standard* $\Longrightarrow$ *of-hypreal* $r \in$ *Standard*
⟨*proof*⟩

**lemma** *of-hypreal-0* [*simp*]: *of-hypreal 0* = *0*
⟨*proof*⟩

**lemma** *of-hypreal-1* [*simp*]: *of-hypreal 1* = *1*
⟨*proof*⟩

**lemma** *of-hypreal-add* [*simp*]:
$\bigwedge x\ y.$ *of-hypreal* $(x + y)$ = *of-hypreal* $x$ + *of-hypreal* $y$
⟨*proof*⟩

**lemma** *of-hypreal-minus* [*simp*]: $\bigwedge x.$ *of-hypreal* $(-\ x) = -$ *of-hypreal* $x$
⟨*proof*⟩

**lemma** *of-hypreal-diff* [*simp*]:
$\bigwedge x\ y.$ *of-hypreal* $(x - y)$ = *of-hypreal* $x$ − *of-hypreal* $y$
⟨*proof*⟩

**lemma** *of-hypreal-mult* [*simp*]:
$\bigwedge x\ y.$ *of-hypreal* $(x * y)$ = *of-hypreal* $x$ * *of-hypreal* $y$
⟨*proof*⟩

**lemma** *of-hypreal-inverse* [*simp*]:
$\bigwedge x.$ *of-hypreal* (*inverse* $x$) =
*inverse* (*of-hypreal* $x$ :: *'a::{real-div-algebra,division-by-zero} star*)
⟨*proof*⟩

**lemma** *of-hypreal-divide* [*simp*]:
$\bigwedge x\ y.$ *of-hypreal* $(x\ /\ y)$ =
(*of-hypreal* $x$ / *of-hypreal* $y$ :: *'a::{real-field,division-by-zero} star*)
⟨*proof*⟩

**lemma** *of-hypreal-eq-iff* [*simp*]:
$\bigwedge x\ y.$ (*of-hypreal* $x$ = *of-hypreal* $y$) = $(x = y)$
⟨*proof*⟩

**lemma** *of-hypreal-eq-0-iff* [*simp*]:
$\bigwedge x.$ (*of-hypreal* $x$ = *0*) = $(x = 0)$
⟨*proof*⟩

## 26.3 Properties of *starrel*

**lemma** *lemma-starrel-refl* [*simp*]: $x \in starrel$ `` $\{x\}$
⟨*proof*⟩

**lemma** *starrel-in-hypreal* [*simp*]: *starrel*``$\{x\}$:*star*
⟨*proof*⟩

**declare** *Abs-star-inject* [*simp*] *Abs-star-inverse* [*simp*]
**declare** *equiv-starrel* [*THEN eq-equiv-class-iff*, *simp*]

## 26.4 *hypreal-of-real*: the Injection from *real* to *hypreal*

**lemma** *inj-star-of*: *inj star-of*
⟨*proof*⟩

**lemma** *mem-Rep-star-iff*: $(X \in \textit{Rep-star } x) = (x = \textit{star-n } X)$
⟨*proof*⟩

**lemma** *Rep-star-star-n-iff* [*simp*]:
$(X \in \textit{Rep-star } (\textit{star-n } Y)) = (\{n.\ Y\ n = X\ n\} \in \mathcal{U})$
⟨*proof*⟩

**lemma** *Rep-star-star-n*: $X \in \textit{Rep-star } (\textit{star-n } X)$
⟨*proof*⟩

## 26.5 Properties of *star-n*

**lemma** *star-n-add*:
$\textit{star-n } X + \textit{star-n } Y = \textit{star-n } (\%n.\ X\ n + Y\ n)$
⟨*proof*⟩

**lemma** *star-n-minus*:
$- \textit{star-n } X = \textit{star-n } (\%n.\ -(X\ n))$
⟨*proof*⟩

**lemma** *star-n-diff*:
$\textit{star-n } X - \textit{star-n } Y = \textit{star-n } (\%n.\ X\ n - Y\ n)$
⟨*proof*⟩

**lemma** *star-n-mult*:
$\textit{star-n } X * \textit{star-n } Y = \textit{star-n } (\%n.\ X\ n * Y\ n)$
⟨*proof*⟩

**lemma** *star-n-inverse*:
$\textit{inverse } (\textit{star-n } X) = \textit{star-n } (\%n.\ \textit{inverse}(X\ n))$
⟨*proof*⟩

**lemma** *star-n-le*:
$\textit{star-n } X \leq \textit{star-n } Y =$

$(\{n.\ X\ n \le Y\ n\} \in \mathit{FreeUltrafilterNat})$
$\langle proof \rangle$

**lemma** *star-n-less*:
$\mathit{star\text{-}n}\ X < \mathit{star\text{-}n}\ Y = (\{n.\ X\ n < Y\ n\} \in \mathit{FreeUltrafilterNat})$
$\langle proof \rangle$

**lemma** *star-n-zero-num*: $0 = \mathit{star\text{-}n}\ (\%n.\ 0)$
$\langle proof \rangle$

**lemma** *star-n-one-num*: $1 = \mathit{star\text{-}n}\ (\%n.\ 1)$
$\langle proof \rangle$

**lemma** *star-n-abs*:
$\mathit{abs}\ (\mathit{star\text{-}n}\ X) = \mathit{star\text{-}n}\ (\%n.\ \mathit{abs}\ (X\ n))$
$\langle proof \rangle$

## 26.6 Misc Others

**lemma** *hypreal-not-refl2*: $!!(x{::}\mathit{hypreal}).\ x < y ==> x \neq y$
$\langle proof \rangle$

**lemma** *hypreal-eq-minus-iff*: $((x{::}\mathit{hypreal}) = y) = (x + - y = 0)$
$\langle proof \rangle$

**lemma** *hypreal-mult-left-cancel*: $(c{::}\mathit{hypreal}) \neq 0 ==> (c*a=c*b) = (a=b)$
$\langle proof \rangle$

**lemma** *hypreal-mult-right-cancel*: $(c{::}\mathit{hypreal}) \neq 0 ==> (a*c=b*c) = (a=b)$
$\langle proof \rangle$

**lemma** *hypreal-omega-gt-zero* [*simp*]: $0 < \mathit{omega}$
$\langle proof \rangle$

## 26.7 Existence of Infinite Hyperreal Number

Existence of infinite number not corresponding to any real number. Use assumption that member $\mathcal{U}$ is not finite.

A few lemmas first

**lemma** *lemma-omega-empty-singleton-disj*: $\{n{::}\mathit{nat}.\ x = \mathit{real}\ n\} = \{\}\ |$
$(\exists\, y.\ \{n{::}\mathit{nat}.\ x = \mathit{real}\ n\} = \{y\})$
$\langle proof \rangle$

**lemma** *lemma-finite-omega-set*: $\mathit{finite}\ \{n{::}\mathit{nat}.\ x = \mathit{real}\ n\}$
$\langle proof \rangle$

**lemma** *not-ex-hypreal-of-real-eq-omega*:
$\sim (\exists\, x.\ \mathit{hypreal\text{-}of\text{-}real}\ x = \mathit{omega})$

⟨*proof*⟩

**lemma** *hypreal-of-real-not-eq-omega*: *hypreal-of-real* $x \neq$ *omega*
⟨*proof*⟩

Existence of infinitesimal number also not corresponding to any real number

**lemma** *lemma-epsilon-empty-singleton-disj*:
$\{n{::}nat.\ x = inverse(real(Suc\ n))\} = \{\} \mid$
$(\exists\, y.\ \{n{::}nat.\ x = inverse(real(Suc\ n))\} = \{y\})$
⟨*proof*⟩

**lemma** *lemma-finite-epsilon-set*: *finite* $\{n.\ x = inverse(real(Suc\ n))\}$
⟨*proof*⟩

**lemma** *not-ex-hypreal-of-real-eq-epsilon*: $\sim$ $(\exists\, x.$ *hypreal-of-real* $x =$ *epsilon*$)$
⟨*proof*⟩

**lemma** *hypreal-of-real-not-eq-epsilon*: *hypreal-of-real* $x \neq$ *epsilon*
⟨*proof*⟩

**lemma** *hypreal-epsilon-not-zero*: *epsilon* $\neq$ *0*
⟨*proof*⟩

**lemma** *hypreal-epsilon-inverse-omega*: *epsilon* $=$ *inverse*$($*omega*$)$
⟨*proof*⟩

**lemma** *hypreal-epsilon-gt-zero*: *0* $<$ *epsilon*
⟨*proof*⟩

## 26.8 Absolute Value Function for the Hyperreals

**lemma** *hrabs-add-less*:
$[\mid abs\ x < r;\ abs\ y < s \mid] ==> abs(x{+}y) < r + (s{::}hypreal)$
⟨*proof*⟩

**lemma** *hrabs-less-gt-zero*: $abs\ x < r ==> (0{::}hypreal) < r$
⟨*proof*⟩

**lemma** *hrabs-disj*: $abs\ x = (x{::}'a{::}abs\text{-}if) \mid abs\ x = -x$
⟨*proof*⟩

**lemma** *hrabs-add-lemma-disj*: $(y{::}hypreal) + -\ x + (y + -\ z) = abs\ (x + -\ z)$ $==> y = z \mid x = y$
⟨*proof*⟩

## 26.9 Embedding the Naturals into the Hyperreals

**abbreviation**
*hypreal-of-nat* :: *nat* => *hypreal* **where**

*hypreal-of-nat* == *of-nat*

**lemma** *SNat-eq*: *Nats* = {*n*. ∃ *N*. *n* = *hypreal-of-nat N*}
⟨*proof*⟩

**lemma** *hypreal-of-nat-eq*:
*hypreal-of-nat* (*n*::*nat*) = *hypreal-of-real* (*real n*)
⟨*proof*⟩

**lemma** *hypreal-of-nat*:
*hypreal-of-nat m* = *star-n* (%*n*. *real m*)
⟨*proof*⟩

⟨*ML*⟩

## 26.10 Exponentials on the Hyperreals

**lemma** *hpowr-0* [*simp*]: *r* ^ *0* = (*1*::*hypreal*)
⟨*proof*⟩

**lemma** *hpowr-Suc* [*simp*]: *r* ^ (*Suc n*) = (*r*::*hypreal*) * (*r* ^ *n*)
⟨*proof*⟩

**lemma** *hrealpow-two*: (*r*::*hypreal*) ^ *Suc* (*Suc 0*) = *r* * *r*
⟨*proof*⟩

**lemma** *hrealpow-two-le* [*simp*]: (*0*::*hypreal*) ≤ *r* ^ *Suc* (*Suc 0*)
⟨*proof*⟩

**lemma** *hrealpow-two-le-add-order* [*simp*]:
(*0*::*hypreal*) ≤ *u* ^ *Suc* (*Suc 0*) + *v* ^ *Suc* (*Suc 0*)
⟨*proof*⟩

**lemma** *hrealpow-two-le-add-order2* [*simp*]:
(*0*::*hypreal*) ≤ *u* ^ *Suc* (*Suc 0*) + *v* ^ *Suc* (*Suc 0*) + *w* ^ *Suc* (*Suc 0*)
⟨*proof*⟩

**lemma** *hypreal-add-nonneg-eq-0-iff*:
[| *0* ≤ *x*; *0* ≤ *y* |] ==> (*x*+*y* = *0*) = (*x* = *0* & *y* = (*0*::*hypreal*))
⟨*proof*⟩

FIXME: DELETE THESE

**lemma** *hypreal-three-squares-add-zero-iff*:
$(x*x + y*y + z*z = 0) = (x = 0 \& y = 0 \& z = (0::hypreal))$
⟨*proof*⟩

**lemma** *hrealpow-three-squares-add-zero-iff* [*simp*]:
$(x \hat{} Suc (Suc 0) + y \hat{} Suc (Suc 0) + z \hat{} Suc (Suc 0) = (0::hypreal)) =$
$(x = 0 \& y = 0 \& z = 0)$
⟨*proof*⟩

**lemma** *hrabs-hrealpow-two* [*simp*]:
$abs(x \hat{} Suc (Suc 0)) = (x::hypreal) \hat{} Suc (Suc 0)$
⟨*proof*⟩

**lemma** *two-hrealpow-ge-one* [*simp*]: $(1::hypreal) \leq 2 \hat{} n$
⟨*proof*⟩

**lemma** *two-hrealpow-gt* [*simp*]: $hypreal\text{-}of\text{-}nat\ n < 2 \hat{} n$
⟨*proof*⟩

**lemma** *hrealpow*:
$star\text{-}n\ X \hat{} m = star\text{-}n\ (\%n.\ (X\ n::real) \hat{} m)$
⟨*proof*⟩

**lemma** *hrealpow-sum-square-expand*:
$(x + (y::hypreal)) \hat{} Suc (Suc 0) =$
$x \hat{} Suc (Suc 0) + y \hat{} Suc (Suc 0) + (hypreal\text{-}of\text{-}nat (Suc (Suc 0)))*x*y$
⟨*proof*⟩

**lemma** *power-hypreal-of-real-number-of*:
$(number\text{-}of\ v :: hypreal) \hat{} n = hypreal\text{-}of\text{-}real ((number\text{-}of\ v) \hat{} n)$
⟨*proof*⟩
**declare** *power-hypreal-of-real-number-of* [*of - number-of w*, *standard*, *simp*]

## 26.11 Powers with Hypernatural Exponents

**definition**

*pow* :: [*'a::power star*, *nat star*] ⇒ *'a star* (**infixr** *pow 80*) **where**
*hyperpow-def* [*transfer-unfold*]:
$R\ pow\ N = (\ *f2*\ op\ \hat{}\ )\ R\ N$

**lemma** *Standard-hyperpow* [*simp*]:
$[\![r \in Standard; n \in Standard]\!] \Longrightarrow r\ pow\ n \in Standard$
⟨*proof*⟩

**lemma** *hyperpow*: $star\text{-}n\ X\ pow\ star\text{-}n\ Y = star\text{-}n\ (\%n.\ X\ n \hat{} Y\ n)$
⟨*proof*⟩

**lemma** *hyperpow-zero* [*simp*]:
$\bigwedge n.\ (0::'a::\{recpower,semiring\text{-}0\}\ star)\ pow\ (n + (1::hypnat)) = 0$
⟨*proof*⟩

**lemma** *hyperpow-not-zero*:
$\bigwedge r\ n.\ r \neq (0::'a::\{recpower,field\}\ star) ==> r\ pow\ n \neq 0$
⟨*proof*⟩

**lemma** *hyperpow-inverse*:
$\bigwedge r\ n.\ r \neq (0::'a::\{recpower,division\text{-}by\text{-}zero,field\}\ star)$
$\Longrightarrow inverse\ (r\ pow\ n) = (inverse\ r)\ pow\ n$
⟨*proof*⟩

**lemma** *hyperpow-hrabs*:
$\bigwedge r\ n.\ abs\ (r::'a::\{recpower,ordered\text{-}idom\}\ star)\ pow\ n = abs\ (r\ pow\ n)$
⟨*proof*⟩

**lemma** *hyperpow-add*:
$\bigwedge r\ n\ m.\ (r::'a::recpower\ star)\ pow\ (n + m) = (r\ pow\ n) * (r\ pow\ m)$
⟨*proof*⟩

**lemma** *hyperpow-one* [*simp*]:
$\bigwedge r.\ (r::'a::recpower\ star)\ pow\ (1::hypnat) = r$
⟨*proof*⟩

**lemma** *hyperpow-two*:
$\bigwedge r.\ (r::'a::recpower\ star)\ pow\ ((1::hypnat) + (1::hypnat)) = r * r$
⟨*proof*⟩

**lemma** *hyperpow-gt-zero*:
$\bigwedge r\ n.\ (0::'a::\{recpower,ordered\text{-}semidom\}\ star) < r \Longrightarrow 0 < r\ pow\ n$
⟨*proof*⟩

**lemma** *hyperpow-ge-zero*:
$\bigwedge r\ n.\ (0::'a::\{recpower,ordered\text{-}semidom\}\ star) \leq r \Longrightarrow 0 \leq r\ pow\ n$
⟨*proof*⟩

**lemma** *hyperpow-le*:
$\bigwedge x\ y\ n.\ [\![(0::'a::\{recpower,ordered\text{-}semidom\}\ star) < x;\ x \leq y]\!]$
$\Longrightarrow x\ pow\ n \leq y\ pow\ n$
⟨*proof*⟩

**lemma** *hyperpow-eq-one* [*simp*]:
$\bigwedge n.\ 1\ pow\ n = (1::'a::recpower\ star)$
⟨*proof*⟩

**lemma** *hrabs-hyperpow-minus-one* [*simp*]:
$\bigwedge n.\ abs(-1\ pow\ n) = (1::'a::\{number\text{-}ring,recpower,ordered\text{-}idom\}\ star)$
⟨*proof*⟩

**lemma** *hyperpow-mult*:
$\bigwedge r\ s\ n.\ (r * s::'a::\{comm\text{-}monoid\text{-}mult,recpower\}\ star)\ pow\ n$
$= (r\ pow\ n) * (s\ pow\ n)$
⟨*proof*⟩

**lemma** *hyperpow-two-le* [*simp*]:
$(0::'a::\{recpower,ordered\text{-}ring\text{-}strict\}\ star) \le r\ pow\ (1 + 1)$
⟨*proof*⟩

**lemma** *hrabs-hyperpow-two* [*simp*]:
$abs(x\ pow\ (1 + 1)) =$
$(x::'a::\{recpower,ordered\text{-}ring\text{-}strict\}\ star)\ pow\ (1 + 1)$
⟨*proof*⟩

**lemma** *hyperpow-two-hrabs* [*simp*]:
$abs(x::'a::\{recpower,ordered\text{-}idom\}\ star)\ pow\ (1 + 1) = x\ pow\ (1 + 1)$
⟨*proof*⟩

The precondition could be weakened to $(0::'a) \le x$

**lemma** *hypreal-mult-less-mono*:
$[|\ u<v;\ x<y;\ (0::hypreal) < v;\ 0 < x\ |] ==> u*x < v*\ y$
⟨*proof*⟩

**lemma** *hyperpow-two-gt-one*:
$\bigwedge r::'a::\{recpower,ordered\text{-}semidom\}\ star.\ 1 < r \Longrightarrow 1 < r\ pow\ (1 + 1)$
⟨*proof*⟩

**lemma** *hyperpow-two-ge-one*:
$\bigwedge r::'a::\{recpower,ordered\text{-}semidom\}\ star.\ 1 \le r \Longrightarrow 1 \le r\ pow\ (1 + 1)$
⟨*proof*⟩

**lemma** *two-hyperpow-ge-one* [*simp*]: $(1::hypreal) \le 2\ pow\ n$
⟨*proof*⟩

**lemma** *hyperpow-minus-one2* [*simp*]:
$!!n.\ -1\ pow\ ((1 + 1)*n) = (1::hypreal)$
⟨*proof*⟩

**lemma** *hyperpow-less-le*:
$!!r\ n\ N.\ [|(0::hypreal) \le r;\ r \le 1;\ n < N|] ==> r\ pow\ N \le r\ pow\ n$
⟨*proof*⟩

**lemma** *hyperpow-SHNat-le*:
$[|\ 0 \le r;\ r \le (1::hypreal);\ N \in HNatInfinite\ |]$
$==> ALL\ n:\ Nats.\ r\ pow\ N \le r\ pow\ n$
⟨*proof*⟩

**lemma** *hyperpow-realpow*:

$(hypreal\text{-}of\text{-}real\ r)\ pow\ (hypnat\text{-}of\text{-}nat\ n) = hypreal\text{-}of\text{-}real\ (r\ \hat{}\ n)$
⟨proof⟩

**lemma** *hyperpow-SReal* [*simp*]:
$(hypreal\text{-}of\text{-}real\ r)\ pow\ (hypnat\text{-}of\text{-}nat\ n) \in Reals$
⟨proof⟩

**lemma** *hyperpow-zero-HNatInfinite* [*simp*]:
$N \in HNatInfinite ==> (0::hypreal)\ pow\ N = 0$
⟨proof⟩

**lemma** *hyperpow-le-le*:
$[|\ (0::hypreal) \le r;\ r \le 1;\ n \le N\ |] ==> r\ pow\ N \le r\ pow\ n$
⟨proof⟩

**lemma** *hyperpow-Suc-le-self2*:
$[|\ (0::hypreal) \le r;\ r < 1\ |] ==> r\ pow\ (n + (1::hypnat)) \le r$
⟨proof⟩

**lemma** *hyperpow-hypnat-of-nat*: $\bigwedge x.\ x\ pow\ hypnat\text{-}of\text{-}nat\ n = x\ \hat{}\ n$
⟨proof⟩

**lemma** *of-hypreal-hyperpow*:
$\bigwedge x\ n.\ of\text{-}hypreal\ (x\ pow\ n) =$
$(of\text{-}hypreal\ x::'a::\{real\text{-}algebra\text{-}1,recpower\}\ star)\ pow\ n$
⟨proof⟩

**end**

# 27 NSA: Infinite Numbers, Infinitesimals, Infinitely Close Relation

**theory** *NSA*
**imports** *HyperDef ../Real/RComplete*
**begin**

**definition**
$hnorm :: 'a::norm\ star \Rightarrow real\ star$ **where**
$hnorm = *f*\ norm$

**definition**
$Infinitesimal :: ('a::real\text{-}normed\text{-}vector)\ star\ set$ **where**
$Infinitesimal = \{x.\ \forall r \in Reals.\ 0 < r --> hnorm\ x < r\}$

**definition**
$HFinite :: ('a::real\text{-}normed\text{-}vector)\ star\ set$ **where**
$HFinite = \{x.\ \exists r \in Reals.\ hnorm\ x < r\}$

**definition**
*HInfinite* :: ($'a$::*real-normed-vector*) *star set* **where**
*HInfinite* = $\{x.\ \forall\, r \in Reals.\ r < hnorm\ x\}$

**definition**
*approx* :: [$'a$::*real-normed-vector star*, $'a$ *star*] => *bool* (**infixl** @= *50*) **where**
— the 'infinitely close' relation
($x$ @= $y$) = (($x - y$) $\in$ *Infinitesimal*)

**definition**
*st* :: *hypreal* => *hypreal* **where**
— the standard part of a hyperreal
*st* = (%$x$. @$r$. $x \in$ *HFinite* & $r \in$ *Reals* & $r$ @= $x$)

**definition**
*monad* :: $'a$::*real-normed-vector star* => $'a$ *star set* **where**
*monad* $x$ = $\{y.\ x$ @= $y\}$

**definition**
*galaxy* :: $'a$::*real-normed-vector star* => $'a$ *star set* **where**
*galaxy* $x$ = $\{y.\ (x + -y) \in HFinite\}$

**notation** (*xsymbols*)
*approx* (**infixl** $\approx$ *50*)

**notation** (*HTML* **output**)
*approx* (**infixl** $\approx$ *50*)

**lemma** *SReal-def*: *Reals* == $\{x.\ \exists\, r.\ x = hypreal\text{-}of\text{-}real\ r\}$
⟨*proof*⟩

## 27.1 Nonstandard Extension of the Norm Function

**definition**
*scaleHR* :: *real star* $\Rightarrow$ $'a$ *star* $\Rightarrow$ $'a$::*real-normed-vector star* **where**
*scaleHR* = *starfun2 scaleR*

**declare** *hnorm-def* [*transfer-unfold*]
**declare** *scaleHR-def* [*transfer-unfold*]

**lemma** *Standard-hnorm* [*simp*]: $x \in Standard \Longrightarrow hnorm\ x \in Standard$
⟨*proof*⟩

**lemma** *star-of-norm* [*simp*]: *star-of* (*norm* $x$) = *hnorm* (*star-of* $x$)
⟨*proof*⟩

**lemma** *hnorm-ge-zero* [*simp*]:
$\bigwedge x$::$'a$::*real-normed-vector star*. $0 \le hnorm\ x$

⟨*proof*⟩

**lemma** *hnorm-eq-zero* [*simp*]:
$\bigwedge x::'a::real\text{-}normed\text{-}vector\ star.\ (hnorm\ x = 0) = (x = 0)$
⟨*proof*⟩

**lemma** *hnorm-triangle-ineq*:
$\bigwedge x\ y::'a::real\text{-}normed\text{-}vector\ star.\ hnorm\ (x + y) \leq hnorm\ x + hnorm\ y$
⟨*proof*⟩

**lemma** *hnorm-triangle-ineq3*:
$\bigwedge x\ y::'a::real\text{-}normed\text{-}vector\ star.\ |hnorm\ x - hnorm\ y| \leq hnorm\ (x - y)$
⟨*proof*⟩

**lemma** *hnorm-scaleR*:
$\bigwedge x::'a::real\text{-}normed\text{-}vector\ star.$
$hnorm\ (a *_R x) = |star\text{-}of\ a| * hnorm\ x$
⟨*proof*⟩

**lemma** *hnorm-scaleHR*:
$\bigwedge a\ (x::'a::real\text{-}normed\text{-}vector\ star).$
$hnorm\ (scaleHR\ a\ x) = |a| * hnorm\ x$
⟨*proof*⟩

**lemma** *hnorm-mult-ineq*:
$\bigwedge x\ y::'a::real\text{-}normed\text{-}algebra\ star.\ hnorm\ (x * y) \leq hnorm\ x * hnorm\ y$
⟨*proof*⟩

**lemma** *hnorm-mult*:
$\bigwedge x\ y::'a::real\text{-}normed\text{-}div\text{-}algebra\ star.\ hnorm\ (x * y) = hnorm\ x * hnorm\ y$
⟨*proof*⟩

**lemma** *hnorm-hyperpow*:
$\bigwedge(x::'a::\{real\text{-}normed\text{-}div\text{-}algebra,recpower\}\ star)\ n.$
$hnorm\ (x\ pow\ n) = hnorm\ x\ pow\ n$
⟨*proof*⟩

**lemma** *hnorm-one* [*simp*]:
$hnorm\ (1::'a::real\text{-}normed\text{-}div\text{-}algebra\ star) = 1$
⟨*proof*⟩

**lemma** *hnorm-zero* [*simp*]:
$hnorm\ (0::'a::real\text{-}normed\text{-}vector\ star) = 0$
⟨*proof*⟩

**lemma** *zero-less-hnorm-iff* [*simp*]:
$\bigwedge x::'a::real\text{-}normed\text{-}vector\ star.\ (0 < hnorm\ x) = (x \neq 0)$
⟨*proof*⟩

**lemma** *hnorm-minus-cancel* [*simp*]:
$\bigwedge x::'a::\mathit{real\text{-}normed\text{-}vector\ star}.\ \mathit{hnorm}\ (-\ x) = \mathit{hnorm}\ x$
⟨*proof*⟩

**lemma** *hnorm-minus-commute*:
$\bigwedge a\ b::'a::\mathit{real\text{-}normed\text{-}vector\ star}.\ \mathit{hnorm}\ (a - b) = \mathit{hnorm}\ (b - a)$
⟨*proof*⟩

**lemma** *hnorm-triangle-ineq2*:
$\bigwedge a\ b::'a::\mathit{real\text{-}normed\text{-}vector\ star}.\ \mathit{hnorm}\ a - \mathit{hnorm}\ b \leq \mathit{hnorm}\ (a - b)$
⟨*proof*⟩

**lemma** *hnorm-triangle-ineq4*:
$\bigwedge a\ b::'a::\mathit{real\text{-}normed\text{-}vector\ star}.\ \mathit{hnorm}\ (a - b) \leq \mathit{hnorm}\ a + \mathit{hnorm}\ b$
⟨*proof*⟩

**lemma** *abs-hnorm-cancel* [*simp*]:
$\bigwedge a::'a::\mathit{real\text{-}normed\text{-}vector\ star}.\ |\mathit{hnorm}\ a| = \mathit{hnorm}\ a$
⟨*proof*⟩

**lemma** *hnorm-of-hypreal* [*simp*]:
$\bigwedge r.\ \mathit{hnorm}\ (\mathit{of\text{-}hypreal}\ r::'a::\mathit{real\text{-}normed\text{-}algebra\text{-}1\ star}) = |r|$
⟨*proof*⟩

**lemma** *nonzero-hnorm-inverse*:
$\bigwedge a::'a::\mathit{real\text{-}normed\text{-}div\text{-}algebra\ star}.$
$a \neq 0 \Longrightarrow \mathit{hnorm}\ (\mathit{inverse}\ a) = \mathit{inverse}\ (\mathit{hnorm}\ a)$
⟨*proof*⟩

**lemma** *hnorm-inverse*:
$\bigwedge a::'a::\{\mathit{real\text{-}normed\text{-}div\text{-}algebra},\mathit{division\text{-}by\text{-}zero}\}\ \mathit{star}.$
$\mathit{hnorm}\ (\mathit{inverse}\ a) = \mathit{inverse}\ (\mathit{hnorm}\ a)$
⟨*proof*⟩

**lemma** *hnorm-divide*:
$\bigwedge a\ b::'a::\{\mathit{real\text{-}normed\text{-}field},\mathit{division\text{-}by\text{-}zero}\}\ \mathit{star}.$
$\mathit{hnorm}\ (a\ /\ b) = \mathit{hnorm}\ a\ /\ \mathit{hnorm}\ b$
⟨*proof*⟩

**lemma** *hypreal-hnorm-def* [*simp*]:
$\bigwedge r::\mathit{hypreal}.\ \mathit{hnorm}\ r \equiv |r|$
⟨*proof*⟩

**lemma** *hnorm-add-less*:
$\bigwedge(x::'a::\mathit{real\text{-}normed\text{-}vector\ star})\ y\ r\ s.$
$[\![\mathit{hnorm}\ x < r;\ \mathit{hnorm}\ y < s]\!] \Longrightarrow \mathit{hnorm}\ (x + y) < r + s$
⟨*proof*⟩

**lemma** *hnorm-mult-less*:

$\bigwedge(x::'a::real\text{-}normed\text{-}algebra\ star)\ y\ r\ s.$
$[\![hnorm\ x < r;\ hnorm\ y < s]\!] \Longrightarrow hnorm\ (x * y) < r * s$
⟨*proof*⟩

**lemma** *hnorm-scaleHR-less*:
$[\![|x| < r;\ hnorm\ y < s]\!] \Longrightarrow hnorm\ (scaleHR\ x\ y) < r * s$
⟨*proof*⟩

## 27.2 Closure Laws for the Standard Reals

**lemma** *Reals-minus-iff* [*simp*]: $(-x \in Reals) = (x \in Reals)$
⟨*proof*⟩

**lemma** *Reals-add-cancel*: $[\![x + y \in Reals;\ y \in Reals]\!] \Longrightarrow x \in Reals$
⟨*proof*⟩

**lemma** *SReal-hrabs*: $(x::hypreal) \in Reals$ ==> $abs\ x \in Reals$
⟨*proof*⟩

**lemma** *SReal-hypreal-of-real* [*simp*]: $hypreal\text{-}of\text{-}real\ x \in Reals$
⟨*proof*⟩

**lemma** *SReal-divide-number-of*: $r \in Reals$ ==> $r/(number\text{-}of\ w::hypreal) \in Reals$
⟨*proof*⟩

epsilon is not in Reals because it is an infinitesimal

**lemma** *SReal-epsilon-not-mem*: $epsilon \notin Reals$
⟨*proof*⟩

**lemma** *SReal-omega-not-mem*: $omega \notin Reals$
⟨*proof*⟩

**lemma** *SReal-UNIV-real*: $\{x.\ hypreal\text{-}of\text{-}real\ x \in Reals\} = (UNIV::real\ set)$
⟨*proof*⟩

**lemma** *SReal-iff*: $(x \in Reals) = (\exists\, y.\ x = hypreal\text{-}of\text{-}real\ y)$
⟨*proof*⟩

**lemma** *hypreal-of-real-image*: $hypreal\text{-}of\text{-}real$ ‘$(UNIV::real\ set) = Reals$
⟨*proof*⟩

**lemma** *inv-hypreal-of-real-image*: $inv\ hypreal\text{-}of\text{-}real$ ‘ $Reals = UNIV$
⟨*proof*⟩

**lemma** *SReal-hypreal-of-real-image*:
[| $\exists\, x.\ x$: $P$; $P \subseteq Reals$ |] ==> $\exists\, Q.\ P = hypreal\text{-}of\text{-}real$ ‘ $Q$
⟨*proof*⟩

**lemma** *SReal-dense*:

[| (*x*::*hypreal*) $\in$ *Reals*; *y* $\in$ *Reals*; *x*<*y* |] ==> $\exists$ *r* $\in$ *Reals*. *x*<*r* & *r*<*y*
⟨*proof*⟩

Completeness of Reals, but both lemmas are unused.

**lemma** *SReal-sup-lemma*:
*P* $\subseteq$ *Reals* ==> (($\exists$ *x* $\in$ *P*. *y* < *x*) =
($\exists$ *X*. *hypreal-of-real X* $\in$ *P* & *y* < *hypreal-of-real X*))
⟨*proof*⟩

**lemma** *SReal-sup-lemma2*:
[| *P* $\subseteq$ *Reals*; $\exists$ *x*. *x* $\in$ *P*; $\exists$ *y* $\in$ *Reals*. $\forall$ *x* $\in$ *P*. *x* < *y* |]
==> ($\exists$ *X*. *X* $\in$ {*w*. *hypreal-of-real w* $\in$ *P*}) &
($\exists$ *Y*. $\forall$ *X* $\in$ {*w*. *hypreal-of-real w* $\in$ *P*}. *X* < *Y*)
⟨*proof*⟩

## 27.3 Set of Finite Elements is a Subring of the Extended Reals

**lemma** *HFinite-add*: [|*x* $\in$ *HFinite*; *y* $\in$ *HFinite*|] ==> (*x*+*y*) $\in$ *HFinite*
⟨*proof*⟩

**lemma** *HFinite-mult*:
**fixes** *x y* :: *'a*::*real-normed-algebra star*
**shows** [|*x* $\in$ *HFinite*; *y* $\in$ *HFinite*|] ==> *x***y* $\in$ *HFinite*
⟨*proof*⟩

**lemma** *HFinite-scaleHR*:
[|*x* $\in$ *HFinite*; *y* $\in$ *HFinite*|] ==> *scaleHR x y* $\in$ *HFinite*
⟨*proof*⟩

**lemma** *HFinite-minus-iff*: ($-$*x* $\in$ *HFinite*) = (*x* $\in$ *HFinite*)
⟨*proof*⟩

**lemma** *HFinite-star-of* [*simp*]: *star-of x* $\in$ *HFinite*
⟨*proof*⟩

**lemma** *SReal-subset-HFinite*: (*Reals*::*hypreal set*) $\subseteq$ *HFinite*
⟨*proof*⟩

**lemma** *HFiniteD*: *x* $\in$ *HFinite* ==> $\exists$ *t* $\in$ *Reals*. *hnorm x* < *t*
⟨*proof*⟩

**lemma** *HFinite-hrabs-iff* [*iff*]: (*abs* (*x*::*hypreal*) $\in$ *HFinite*) = (*x* $\in$ *HFinite*)
⟨*proof*⟩

**lemma** *HFinite-hnorm-iff* [*iff*]:
(*hnorm* (*x*::*hypreal*) $\in$ *HFinite*) = (*x* $\in$ *HFinite*)
⟨*proof*⟩

**lemma** *HFinite-number-of* [*simp*]: *number-of* $w \in$ *HFinite*
⟨*proof*⟩

**lemma** *HFinite-0* [*simp*]: $0 \in$ *HFinite*
⟨*proof*⟩

**lemma** *HFinite-1* [*simp*]: $1 \in$ *HFinite*
⟨*proof*⟩

**lemma** *hrealpow-HFinite*:
**fixes** $x$ :: $'a$::{*real-normed-algebra*,*recpower*} *star*
**shows** $x \in$ *HFinite* ==> $x$ ^ $n \in$ *HFinite*
⟨*proof*⟩

**lemma** *HFinite-bounded*:
[|($x$::*hypreal*) $\in$ *HFinite*; $y \leq x$; $0 \leq y$ |] ==> $y \in$ *HFinite*
⟨*proof*⟩

## 27.4 Set of Infinitesimals is a Subring of the Hyperreals

**lemma** *InfinitesimalI*:
$(\bigwedge r.\ [\![r \in \mathbb{R};\ 0 < r]\!] \Longrightarrow hnorm\ x < r) \Longrightarrow x \in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *InfinitesimalD*:
$x \in$ *Infinitesimal* ==> $\forall\, r \in$ *Reals*. $0 < r$ --> *hnorm* $x < r$
⟨*proof*⟩

**lemma** *InfinitesimalI2*:
$(\bigwedge r.\ 0 < r \Longrightarrow hnorm\ x < star\text{-}of\ r) \Longrightarrow x \in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *InfinitesimalD2*:
$[\![x \in Infinitesimal;\ 0 < r]\!] \Longrightarrow hnorm\ x < star\text{-}of\ r$
⟨*proof*⟩

**lemma** *Infinitesimal-zero* [*iff*]: $0 \in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *hypreal-sum-of-halves*: $x/(2$::*hypreal*$) + x/(2$::*hypreal*$) = x$
⟨*proof*⟩

**lemma** *Infinitesimal-add*:
[| $x \in$ *Infinitesimal*; $y \in$ *Infinitesimal* |] ==> $(x+y) \in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *Infinitesimal-minus-iff* [*simp*]: $(-x$:*Infinitesimal*$) = (x$:*Infinitesimal*$)$

⟨*proof*⟩

**lemma** *Infinitesimal-hnorm-iff*:
(*hnorm* $x \in$ *Infinitesimal*) = ($x \in$ *Infinitesimal*)
⟨*proof*⟩

**lemma** *Infinitesimal-hrabs-iff* [*iff*]:
(*abs* ($x$::*hypreal*) $\in$ *Infinitesimal*) = ($x \in$ *Infinitesimal*)
⟨*proof*⟩

**lemma** *Infinitesimal-of-hypreal-iff* [*simp*]:
((*of-hypreal* $x$::'*a*::*real-normed-algebra-1 star*) $\in$ *Infinitesimal*) =
($x \in$ *Infinitesimal*)
⟨*proof*⟩

**lemma** *Infinitesimal-diff*:
[| $x \in$ *Infinitesimal*; $y \in$ *Infinitesimal* |] ==> $x-y \in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *Infinitesimal-mult*:
**fixes** $x$ $y$ :: '*a*::*real-normed-algebra star*
**shows** [|$x \in$ *Infinitesimal*; $y \in$ *Infinitesimal*|] ==> ($x * y$) $\in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *Infinitesimal-HFinite-mult*:
**fixes** $x$ $y$ :: '*a*::*real-normed-algebra star*
**shows** [| $x \in$ *Infinitesimal*; $y \in$ *HFinite* |] ==> ($x * y$) $\in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *Infinitesimal-HFinite-scaleHR*:
[| $x \in$ *Infinitesimal*; $y \in$ *HFinite* |] ==> *scaleHR* $x$ $y \in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *Infinitesimal-HFinite-mult2*:
**fixes** $x$ $y$ :: '*a*::*real-normed-algebra star*
**shows** [| $x \in$ *Infinitesimal*; $y \in$ *HFinite* |] ==> ($y * x$) $\in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *Infinitesimal-scaleR2*:
$x \in$ *Infinitesimal* ==> $a *_R x \in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *Compl-HFinite*: − *HFinite* = *HInfinite*
⟨*proof*⟩

**lemma** *HInfinite-inverse-Infinitesimal*:
**fixes** $x$ :: '*a*::*real-normed-div-algebra star*
**shows** $x \in$ *HInfinite* ==> *inverse* $x \in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *HInfiniteI*: $(\bigwedge r.\ r \in \mathbb{R} \Longrightarrow r < hnorm\ x) \Longrightarrow x \in HInfinite$
⟨*proof*⟩

**lemma** *HInfiniteD*: $[\![x \in HInfinite;\ r \in \mathbb{R}]\!] \Longrightarrow r < hnorm\ x$
⟨*proof*⟩

**lemma** *HInfinite-mult*:
  **fixes** $x\ y$ :: *'a::real-normed-div-algebra star*
  **shows** $[|x \in HInfinite;\ y \in HInfinite|]$ ==> $(x*y) \in HInfinite$
⟨*proof*⟩

**lemma** *hypreal-add-zero-less-le-mono*: $[|r < x;\ (0::hypreal) \leq y|]$ ==> $r < x+y$
⟨*proof*⟩

**lemma** *HInfinite-add-ge-zero*:
  $[|(x::hypreal) \in HInfinite;\ 0 \leq y;\ 0 \leq x|]$ ==> $(x + y)$: *HInfinite*
⟨*proof*⟩

**lemma** *HInfinite-add-ge-zero2*:
  $[|(x::hypreal) \in HInfinite;\ 0 \leq y;\ 0 \leq x|]$ ==> $(y + x)$: *HInfinite*
⟨*proof*⟩

**lemma** *HInfinite-add-gt-zero*:
  $[|(x::hypreal) \in HInfinite;\ 0 < y;\ 0 < x|]$ ==> $(x + y)$: *HInfinite*
⟨*proof*⟩

**lemma** *HInfinite-minus-iff*: $(-x \in HInfinite) = (x \in HInfinite)$
⟨*proof*⟩

**lemma** *HInfinite-add-le-zero*:
  $[|(x::hypreal) \in HInfinite;\ y \leq 0;\ x \leq 0|]$ ==> $(x + y)$: *HInfinite*
⟨*proof*⟩

**lemma** *HInfinite-add-lt-zero*:
  $[|(x::hypreal) \in HInfinite;\ y < 0;\ x < 0|]$ ==> $(x + y)$: *HInfinite*
⟨*proof*⟩

**lemma** *HFinite-sum-squares*:
  **fixes** $a\ b\ c$ :: *'a::real-normed-algebra star*
  **shows** $[|a$: *HFinite*; $b$: *HFinite*; $c$: *HFinite*$|]$
    ==> $a*a + b*b + c*c \in HFinite$
⟨*proof*⟩

**lemma** *not-Infinitesimal-not-zero*: $x \notin Infinitesimal$ ==> $x \neq 0$
⟨*proof*⟩

**lemma** *not-Infinitesimal-not-zero2*: $x \in HFinite - Infinitesimal$ ==> $x \neq 0$
⟨*proof*⟩

**lemma** *HFinite-diff-Infinitesimal-hrabs*:
$(x{::}hypreal) \in HFinite - Infinitesimal ==> abs\ x \in HFinite - Infinitesimal$
⟨*proof*⟩

**lemma** *hnorm-le-Infinitesimal*:
$[\![e \in Infinitesimal;\ hnorm\ x \le e]\!] \Longrightarrow x \in Infinitesimal$
⟨*proof*⟩

**lemma** *hnorm-less-Infinitesimal*:
$[\![e \in Infinitesimal;\ hnorm\ x < e]\!] \Longrightarrow x \in Infinitesimal$
⟨*proof*⟩

**lemma** *hrabs-le-Infinitesimal*:
$[|\ e \in Infinitesimal;\ abs\ (x{::}hypreal) \le e\ |] ==> x \in Infinitesimal$
⟨*proof*⟩

**lemma** *hrabs-less-Infinitesimal*:
$[|\ e \in Infinitesimal;\ abs\ (x{::}hypreal) < e\ |] ==> x \in Infinitesimal$
⟨*proof*⟩

**lemma** *Infinitesimal-interval*:
$[|\ e \in Infinitesimal;\ e' \in Infinitesimal;\ e' < x\ ;\ x < e\ |]$
$==> (x{::}hypreal) \in Infinitesimal$
⟨*proof*⟩

**lemma** *Infinitesimal-interval2*:
$[|\ e \in Infinitesimal;\ e' \in Infinitesimal;$
$e' \le x\ ;\ x \le e\ |] ==> (x{::}hypreal) \in Infinitesimal$
⟨*proof*⟩

**lemma** *lemma-Infinitesimal-hyperpow*:
$[|\ (x{::}hypreal) \in Infinitesimal;\ 0 < N\ |] ==> abs\ (x\ pow\ N) \le abs\ x$
⟨*proof*⟩

**lemma** *Infinitesimal-hyperpow*:
$[|\ (x{::}hypreal) \in Infinitesimal;\ 0 < N\ |] ==> x\ pow\ N \in Infinitesimal$
⟨*proof*⟩

**lemma** *hrealpow-hyperpow-Infinitesimal-iff*:
$(x\ \hat{}\ n \in Infinitesimal) = (x\ pow\ (hypnat\text{-}of\text{-}nat\ n) \in Infinitesimal)$
⟨*proof*⟩

**lemma** *Infinitesimal-hrealpow*:
$[|\ (x{::}hypreal) \in Infinitesimal;\ 0 < n\ |] ==> x\ \hat{}\ n \in Infinitesimal$
⟨*proof*⟩

**lemma** *not-Infinitesimal-mult*:

**fixes** $x\ y$ :: $'a$::*real-normed-div-algebra star*
**shows** [| $x \notin$ *Infinitesimal*; $y \notin$ *Infinitesimal*|] ==> $(x*y) \notin$*Infinitesimal*
⟨*proof*⟩

**lemma** *Infinitesimal-mult-disj*:
**fixes** $x\ y$ :: $'a$::*real-normed-div-algebra star*
**shows** $x*y \in$ *Infinitesimal* ==> $x \in$ *Infinitesimal* | $y \in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *HFinite-Infinitesimal-not-zero*: $x \in$ *HFinite*−*Infinitesimal* ==> $x \neq 0$
⟨*proof*⟩

**lemma** *HFinite-Infinitesimal-diff-mult*:
**fixes** $x\ y$ :: $'a$::*real-normed-div-algebra star*
**shows** [| $x \in$ *HFinite* − *Infinitesimal*;
$y \in$ *HFinite* − *Infinitesimal*
|] ==> $(x*y) \in$ *HFinite* − *Infinitesimal*
⟨*proof*⟩

**lemma** *Infinitesimal-subset-HFinite*:
*Infinitesimal* ⊆ *HFinite*
⟨*proof*⟩

**lemma** *Infinitesimal-star-of-mult*:
**fixes** $x$ :: $'a$::*real-normed-algebra star*
**shows** $x \in$ *Infinitesimal* ==> $x$ ∗ *star-of* $r \in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *Infinitesimal-star-of-mult2*:
**fixes** $x$ :: $'a$::*real-normed-algebra star*
**shows** $x \in$ *Infinitesimal* ==> *star-of* $r$ ∗ $x \in$ *Infinitesimal*
⟨*proof*⟩

## 27.5 The Infinitely Close Relation

**lemma** *mem-infmal-iff*: ($x \in$ *Infinitesimal*) = ($x$ @= *0*)
⟨*proof*⟩

**lemma** *approx-minus-iff*: ($x$ @= $y$) = ($x - y$ @= *0*)
⟨*proof*⟩

**lemma** *approx-minus-iff2*: ($x$ @= $y$) = ($-y + x$ @= *0*)
⟨*proof*⟩

**lemma** *approx-refl* [*iff*]: $x$ @= $x$
⟨*proof*⟩

**lemma** *hypreal-minus-distrib1*: $-(y + -(x$::$'a$::*ab-group-add*$))$ = $x + -y$
⟨*proof*⟩

**lemma** *approx-sym*: $x$ @= $y$ ==> $y$ @= $x$
⟨*proof*⟩

**lemma** *approx-trans*: [| $x$ @= $y$; $y$ @= $z$ |] ==> $x$ @= $z$
⟨*proof*⟩

**lemma** *approx-trans2*: [| $r$ @= $x$; $s$ @= $x$ |] ==> $r$ @= $s$
⟨*proof*⟩

**lemma** *approx-trans3*: [| $x$ @= $r$; $x$ @= $s$|] ==> $r$ @= $s$
⟨*proof*⟩

**lemma** *number-of-approx-reorient*: (*number-of* $w$ @= $x$) = ($x$ @= *number-of* $w$)
⟨*proof*⟩

**lemma** *zero-approx-reorient*: (*0* @= $x$) = ($x$ @= *0*)
⟨*proof*⟩

**lemma** *one-approx-reorient*: (*1* @= $x$) = ($x$ @= *1*)
⟨*proof*⟩

⟨*ML*⟩

**lemma** *Infinitesimal-approx-minus*: ($x-y \in$ *Infinitesimal*) = ($x$ @= $y$)
⟨*proof*⟩

**lemma** *approx-monad-iff*: ($x$ @= $y$) = (*monad*($x$)=*monad*($y$))
⟨*proof*⟩

**lemma** *Infinitesimal-approx*:
[| $x \in$ *Infinitesimal*; $y \in$ *Infinitesimal* |] ==> $x$ @= $y$
⟨*proof*⟩

**lemma** *approx-add*: [| $a$ @= $b$; $c$ @= $d$ |] ==> $a+c$ @= $b+d$
⟨*proof*⟩

**lemma** *approx-minus*: $a$ @= $b$ ==> $-a$ @= $-b$
⟨*proof*⟩

**lemma** *approx-minus2*: $-a$ @= $-b$ ==> $a$ @= $b$
⟨*proof*⟩

**lemma** *approx-minus-cancel* [*simp*]: ($-a$ @= $-b$) = ($a$ @= $b$)
⟨*proof*⟩

**lemma** *approx-add-minus*: [| $a$ @= $b$; $c$ @= $d$ |] ==> $a + -c$ @= $b + -d$
⟨*proof*⟩

**lemma** *approx-diff*: [| $a$ @= $b$; $c$ @= $d$ |] ==> $a - c$ @= $b - d$
⟨*proof*⟩

**lemma** *approx-mult1*:
**fixes** $a$ $b$ $c$ :: *'a::real-normed-algebra star*
**shows** [| $a$ @= $b$; $c$: *HFinite*|] ==> $a*c$ @= $b*c$
⟨*proof*⟩

**lemma** *approx-mult2*:
**fixes** $a$ $b$ $c$ :: *'a::real-normed-algebra star*
**shows** [|$a$ @= $b$; $c$: *HFinite*|] ==> $c*a$ @= $c*b$
⟨*proof*⟩

**lemma** *approx-mult-subst*:
**fixes** $u$ $v$ $x$ $y$ :: *'a::real-normed-algebra star*
**shows** [|$u$ @= $v*x$; $x$ @= $y$; $v \in$ *HFinite*|] ==> $u$ @= $v*y$
⟨*proof*⟩

**lemma** *approx-mult-subst2*:
**fixes** $u$ $v$ $x$ $y$ :: *'a::real-normed-algebra star*
**shows** [| $u$ @= $x*v$; $x$ @= $y$; $v \in$ *HFinite* |] ==> $u$ @= $y*v$
⟨*proof*⟩

**lemma** *approx-mult-subst-star-of*:
**fixes** $u$ $x$ $y$ :: *'a::real-normed-algebra star*
**shows** [| $u$ @= $x*$*star-of* $v$; $x$ @= $y$ |] ==> $u$ @= $y*$*star-of* $v$
⟨*proof*⟩

**lemma** *approx-eq-imp*: $a = b$ ==> $a$ @= $b$
⟨*proof*⟩

**lemma** *Infinitesimal-minus-approx*: $x \in$ *Infinitesimal* ==> $-x$ @= $x$
⟨*proof*⟩

**lemma** *bex-Infinitesimal-iff*: ($\exists\, y \in$ *Infinitesimal*. $x - z = y$) = ($x$ @= $z$)
⟨*proof*⟩

**lemma** *bex-Infinitesimal-iff2*: ($\exists\, y \in$ *Infinitesimal*. $x = z + y$) = ($x$ @= $z$)
⟨*proof*⟩

**lemma** *Infinitesimal-add-approx*: [| $y \in$ *Infinitesimal*; $x + y = z$ |] ==> $x$ @= $z$
⟨*proof*⟩

**lemma** *Infinitesimal-add-approx-self*: $y \in$ *Infinitesimal* ==> $x$ @= $x + y$
⟨*proof*⟩

**lemma** *Infinitesimal-add-approx-self2*: $y \in$ *Infinitesimal* ==> $x$ @= $y + x$
⟨*proof*⟩

**lemma** *Infinitesimal-add-minus-approx-self*: $y \in \textit{Infinitesimal} ==> x\ @= x + -y$
⟨*proof*⟩

**lemma** *Infinitesimal-add-cancel*: $[|\ y \in \textit{Infinitesimal};\ x+y\ @= z|] ==> x\ @= z$
⟨*proof*⟩

**lemma** *Infinitesimal-add-right-cancel*:
$[|\ y \in \textit{Infinitesimal};\ x\ @= z + y|] ==> x\ @= z$
⟨*proof*⟩

**lemma** *approx-add-left-cancel*: $d + b\ \ @= d + c ==> b\ @= c$
⟨*proof*⟩

**lemma** *approx-add-right-cancel*: $b + d\ @= c + d ==> b\ @= c$
⟨*proof*⟩

**lemma** *approx-add-mono1*: $b\ @= c ==> d + b\ @= d + c$
⟨*proof*⟩

**lemma** *approx-add-mono2*: $b\ @= c ==> b + a\ @= c + a$
⟨*proof*⟩

**lemma** *approx-add-left-iff* [*simp*]: $(a + b\ @= a + c) = (b\ @= c)$
⟨*proof*⟩

**lemma** *approx-add-right-iff* [*simp*]: $(b + a\ @= c + a) = (b\ @= c)$
⟨*proof*⟩

**lemma** *approx-HFinite*: $[|\ x \in \textit{HFinite};\ x\ @= y\ |] ==> y \in \textit{HFinite}$
⟨*proof*⟩

**lemma** *approx-star-of-HFinite*: $x\ @= \textit{star-of}\ D ==> x \in \textit{HFinite}$
⟨*proof*⟩

**lemma** *approx-mult-HFinite*:
**fixes** $a\ b\ c\ d$ :: $'a$::*real-normed-algebra star*
**shows** $[|a\ @= b;\ c\ @= d;\ b\text{: } \textit{HFinite};\ d\text{: } \textit{HFinite}|] ==> a{*}c\ @= b{*}d$
⟨*proof*⟩

**lemma** *scaleHR-left-diff-distrib*:
$\bigwedge a\ b\ x.\ \textit{scaleHR}\ (a - b)\ x = \textit{scaleHR}\ a\ x - \textit{scaleHR}\ b\ x$
⟨*proof*⟩

**lemma** *approx-scaleR1*:
$[|\ a\ @= \textit{star-of}\ b;\ c\text{: } \textit{HFinite}|] ==> \textit{scaleHR}\ a\ c\ @= b *_R c$
⟨*proof*⟩

**lemma** *approx-scaleR2*:

$a \ @= \ b \ ==> \ c *_R a \ @= \ c *_R b$
⟨proof⟩

**lemma** *approx-scaleR-HFinite*:
[|$a$ @= *star-of* $b$; $c$ @= $d$; $d$: *HFinite*|] ==> *scaleHR* $a$ $c$ @= $b *_R d$
⟨proof⟩

**lemma** *approx-mult-star-of*:
**fixes** $a$ $c$ :: $'a$::*real-normed-algebra star*
**shows** [|$a$ @= *star-of* $b$; $c$ @= *star-of* $d$ |]
==> $a*c$ @= *star-of* $b$**star-of* $d$
⟨proof⟩

**lemma** *approx-SReal-mult-cancel-zero*:
[| ($a$::*hypreal*) ∈ *Reals*; $a \neq 0$; $a*x$ @= $0$ |] ==> $x$ @= $0$
⟨proof⟩

**lemma** *approx-mult-SReal1*: [| ($a$::*hypreal*) ∈ *Reals*; $x$ @= $0$ |] ==> $x*a$ @= $0$
⟨proof⟩

**lemma** *approx-mult-SReal2*: [| ($a$::*hypreal*) ∈ *Reals*; $x$ @= $0$ |] ==> $a*x$ @= $0$
⟨proof⟩

**lemma** *approx-mult-SReal-zero-cancel-iff* [*simp*]:
[|($a$::*hypreal*) ∈ *Reals*; $a \neq 0$ |] ==> ($a*x$ @= $0$) = ($x$ @= $0$)
⟨proof⟩

**lemma** *approx-SReal-mult-cancel*:
[| ($a$::*hypreal*) ∈ *Reals*; $a \neq 0$; $a*$ $w$ @= $a*z$ |] ==> $w$ @= $z$
⟨proof⟩

**lemma** *approx-SReal-mult-cancel-iff1* [*simp*]:
[| ($a$::*hypreal*) ∈ *Reals*; $a \neq 0$|] ==> ($a*$ $w$ @= $a*z$) = ($w$ @= $z$)
⟨proof⟩

**lemma** *approx-le-bound*: [| ($z$::*hypreal*) $\leq f$; $f$ @= $g$; $g \leq z$ |] ==> $f$ @= $z$
⟨proof⟩

**lemma** *approx-hnorm*:
**fixes** $x$ $y$ :: $'a$::*real-normed-vector star*
**shows** $x \approx y \Longrightarrow$ *hnorm* $x \approx$ *hnorm* $y$
⟨proof⟩

## 27.6 Zero is the Only Infinitesimal that is also a Real

**lemma** *Infinitesimal-less-SReal*:
[| ($x$::*hypreal*) ∈ *Reals*; $y$ ∈ *Infinitesimal*; $0 < x$ |] ==> $y < x$
⟨proof⟩

**lemma** *Infinitesimal-less-SReal2*:
$(y::hypreal) \in Infinitesimal ==> \forall r \in Reals.\ 0 < r --> y < r$
$\langle proof \rangle$

**lemma** *SReal-not-Infinitesimal*:
$[|\ 0 < y;\ (y::hypreal) \in Reals|] ==> y \notin Infinitesimal$
$\langle proof \rangle$

**lemma** *SReal-minus-not-Infinitesimal*:
$[|\ y < 0;\ (y::hypreal) \in Reals\ |] ==> y \notin Infinitesimal$
$\langle proof \rangle$

**lemma** *SReal-Int-Infinitesimal-zero*: $Reals\ Int\ Infinitesimal = \{0::hypreal\}$
$\langle proof \rangle$

**lemma** *SReal-Infinitesimal-zero*:
$[|\ (x::hypreal) \in Reals;\ x \in Infinitesimal|] ==> x = 0$
$\langle proof \rangle$

**lemma** *SReal-HFinite-diff-Infinitesimal*:
$[|\ (x::hypreal) \in Reals;\ x \neq 0\ |] ==> x \in HFinite - Infinitesimal$
$\langle proof \rangle$

**lemma** *hypreal-of-real-HFinite-diff-Infinitesimal*:
$hypreal\text{-}of\text{-}real\ x \neq 0 ==> hypreal\text{-}of\text{-}real\ x \in HFinite - Infinitesimal$
$\langle proof \rangle$

**lemma** *star-of-Infinitesimal-iff-0* [*iff*]:
$(star\text{-}of\ x \in Infinitesimal) = (x = 0)$
$\langle proof \rangle$

**lemma** *star-of-HFinite-diff-Infinitesimal*:
$x \neq 0 ==> star\text{-}of\ x \in HFinite - Infinitesimal$
$\langle proof \rangle$

**lemma** *number-of-not-Infinitesimal* [*simp*]:
$number\text{-}of\ w \neq (0::hypreal) ==> (number\text{-}of\ w :: hypreal) \notin Infinitesimal$
$\langle proof \rangle$

**lemma** *one-not-Infinitesimal* [*simp*]:
$(1::'a::\{real\text{-}normed\text{-}vector,zero\text{-}neq\text{-}one\}\ star) \notin Infinitesimal$
$\langle proof \rangle$

**lemma** *approx-SReal-not-zero*:
$[|\ (y::hypreal) \in Reals;\ x\ @{=}\ y;\ y \neq 0\ |] ==> x \neq 0$
$\langle proof \rangle$

**lemma** *HFinite-diff-Infinitesimal-approx*:

[| $x$ @= $y$; $y \in$ *HFinite* $-$ *Infinitesimal* |]
==> $x \in$ *HFinite* $-$ *Infinitesimal*
⟨*proof*⟩

**lemma** *Infinitesimal-ratio*:
**fixes** *x y* :: *'a*::{*real-normed-div-algebra*,*field*} *star*
**shows** [| $y \neq 0$; $y \in$ *Infinitesimal*; $x/y \in$ *HFinite* |]
==> $x \in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *Infinitesimal-SReal-divide*:
[| (*x*::*hypreal*) $\in$ *Infinitesimal*; $y \in$ *Reals* |] ==> $x/y \in$ *Infinitesimal*
⟨*proof*⟩

## 27.7 Uniqueness: Two Infinitely Close Reals are Equal

**lemma** *star-of-approx-iff* [*simp*]: (*star-of x* @= *star-of y*) = ($x = y$)
⟨*proof*⟩

**lemma** *SReal-approx-iff*:
[|(*x*::*hypreal*) $\in$ *Reals*; $y \in$ *Reals*|] ==> ($x$ @= $y$) = ($x = y$)
⟨*proof*⟩

**lemma** *number-of-approx-iff* [*simp*]:
(*number-of v* @= (*number-of w* :: *'a*::{*number*,*real-normed-vector*} *star*)) =
(*number-of v* = (*number-of w* :: *'a*))
⟨*proof*⟩

**lemma** [*simp*]:
(*number-of w* @= (*0*::*'a*::{*number*,*real-normed-vector*} *star*)) =
(*number-of w* = (*0*::*'a*))
((*0*::*'a*::{*number*,*real-normed-vector*} *star*) @= *number-of w*) =
(*number-of w* = (*0*::*'a*))
(*number-of w* @= (*1*::*'b*::{*number*,*one*,*real-normed-vector*} *star*)) =
(*number-of w* = (*1*::*'b*))
((*1*::*'b*::{*number*,*one*,*real-normed-vector*} *star*) @= *number-of w*) =
(*number-of w* = (*1*::*'b*))
~ (*0* @= (*1*::*'c*::{*zero-neq-one*,*real-normed-vector*} *star*))
~ (*1* @= (*0*::*'c*::{*zero-neq-one*,*real-normed-vector*} *star*))
⟨*proof*⟩

**lemma** *star-of-approx-number-of-iff* [*simp*]:
(*star-of k* @= *number-of w*) = (*k* = *number-of w*)
⟨*proof*⟩

**lemma** *star-of-approx-zero-iff* [*simp*]: (*star-of k* @= *0*) = ($k = 0$)
⟨*proof*⟩

**lemma** *star-of-approx-one-iff* [*simp*]: (*star-of* $k$ @= *1*) = ($k$ = *1*)
⟨*proof*⟩

**lemma** *approx-unique-real*:
[| ($r$::*hypreal*) ∈ *Reals*; $s$ ∈ *Reals*; $r$ @= $x$; $s$ @= $x$|] ==> $r$ = $s$
⟨*proof*⟩

## 27.8 Existence of Unique Real Infinitely Close

### 27.8.1 Lifting of the Ub and Lub Properties

**lemma** *hypreal-of-real-isUb-iff*:
(*isUb* (*Reals*) (*hypreal-of-real* ‘ $Q$) (*hypreal-of-real* $Y$)) =
(*isUb* (*UNIV* :: *real set*) $Q$ $Y$)
⟨*proof*⟩

**lemma** *hypreal-of-real-isLub1*:
*isLub Reals* (*hypreal-of-real* ‘ $Q$) (*hypreal-of-real* $Y$)
==> *isLub* (*UNIV* :: *real set*) $Q$ $Y$
⟨*proof*⟩

**lemma** *hypreal-of-real-isLub2*:
*isLub* (*UNIV* :: *real set*) $Q$ $Y$
==> *isLub Reals* (*hypreal-of-real* ‘ $Q$) (*hypreal-of-real* $Y$)
⟨*proof*⟩

**lemma** *hypreal-of-real-isLub-iff*:
(*isLub Reals* (*hypreal-of-real* ‘ $Q$) (*hypreal-of-real* $Y$)) =
(*isLub* (*UNIV* :: *real set*) $Q$ $Y$)
⟨*proof*⟩

**lemma** *lemma-isUb-hypreal-of-real*:
*isUb Reals* $P$ $Y$ ==> ∃ *Yo*. *isUb Reals* $P$ (*hypreal-of-real Yo*)
⟨*proof*⟩

**lemma** *lemma-isLub-hypreal-of-real*:
*isLub Reals* $P$ $Y$ ==> ∃ *Yo*. *isLub Reals* $P$ (*hypreal-of-real Yo*)
⟨*proof*⟩

**lemma** *lemma-isLub-hypreal-of-real2*:
∃ *Yo*. *isLub Reals* $P$ (*hypreal-of-real Yo*) ==> ∃ $Y$. *isLub Reals* $P$ $Y$
⟨*proof*⟩

**lemma** *SReal-complete*:
[| $P$ ⊆ *Reals*; ∃ $x$. $x$ ∈ $P$; ∃ $Y$. *isUb Reals* $P$ $Y$ |]
==> ∃ $t$::*hypreal*. *isLub Reals* $P$ $t$
⟨*proof*⟩

**lemma** *hypreal-isLub-unique*:
$[|\ isLub\ R\ S\ x;\ isLub\ R\ S\ y\ |] ==> x = (y::hypreal)$
⟨*proof*⟩

**lemma** *lemma-st-part-ub*:
$(x::hypreal) \in HFinite ==> \exists\, u.\ isUb\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ u$
⟨*proof*⟩

**lemma** *lemma-st-part-nonempty*:
$(x::hypreal) \in HFinite ==> \exists\, y.\ y \in \{s.\ s \in Reals\ \&\ s < x\}$
⟨*proof*⟩

**lemma** *lemma-st-part-subset*: $\{s.\ s \in Reals\ \&\ s < x\} \subseteq Reals$
⟨*proof*⟩

**lemma** *lemma-st-part-lub*:
$(x::hypreal) \in HFinite ==> \exists\, t.\ isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t$
⟨*proof*⟩

**lemma** *lemma-hypreal-le-left-cancel*: $((t::hypreal) + r \le t) = (r \le 0)$
⟨*proof*⟩

**lemma** *lemma-st-part-le1*:
$[|\ (x::hypreal) \in HFinite;\ \ isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ \ 0 < r\ |] ==> x \le t + r$
⟨*proof*⟩

**lemma** *hypreal-setle-less-trans*:
$[|\ S *<= (x::hypreal);\ x < y\ |] ==> S *<= y$
⟨*proof*⟩

**lemma** *hypreal-gt-isUb*:
$[|\ isUb\ R\ S\ (x::hypreal);\ x < y;\ y \in R\ |] ==> isUb\ R\ S\ y$
⟨*proof*⟩

**lemma** *lemma-st-part-gt-ub*:
$[|\ (x::hypreal) \in HFinite;\ x < y;\ y \in Reals\ |]$
$==> isUb\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ y$
⟨*proof*⟩

**lemma** *lemma-minus-le-zero*: $t \le t + -r ==> r \le (0::hypreal)$
⟨*proof*⟩

**lemma** *lemma-st-part-le2*:
$[|\ (x::hypreal) \in HFinite;$
$isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ 0 < r\ |]$
$==> t + -r \le x$
⟨*proof*⟩

**lemma** *lemma-st-part1a*:
$[| (x::hypreal) \in HFinite;$
$isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ 0 < r\ |]$
$==> x + -t \le r$
⟨*proof*⟩

**lemma** *lemma-st-part2a*:
$[| (x::hypreal) \in HFinite;$
$isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ \ 0 < r\ |]$
$==> -(x + -t) \le r$
⟨*proof*⟩

**lemma** *lemma-SReal-ub*:
$(x::hypreal) \in Reals ==> isUb\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ x$
⟨*proof*⟩

**lemma** *lemma-SReal-lub*:
$(x::hypreal) \in Reals ==> isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ x$
⟨*proof*⟩

**lemma** *lemma-st-part-not-eq1*:
$[| (x::hypreal) \in HFinite;$
$isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ 0 < r\ |]$
$==> x + -t \ne r$
⟨*proof*⟩

**lemma** *lemma-st-part-not-eq2*:
$[| (x::hypreal) \in HFinite;$
$isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ 0 < r\ |]$
$==> -(x + -t) \ne r$
⟨*proof*⟩

**lemma** *lemma-st-part-major*:
$[| (x::hypreal) \in HFinite;$
$isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ 0 < r\ |]$
$==> abs\ (x - t) < r$
⟨*proof*⟩

**lemma** *lemma-st-part-major2*:
$[| (x::hypreal) \in HFinite;\ isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t\ |]$
$==> \forall r \in Reals.\ 0 < r --> abs\ (x - t) < r$
⟨*proof*⟩

Existence of real and Standard Part Theorem

**lemma** *lemma-st-part-Ex*:
  $(x::hypreal) \in HFinite$
   $==> \exists\, t \in Reals.\ \forall\, r \in Reals.\ 0 < r \;-->\; abs\ (x - t) < r$
⟨*proof*⟩

**lemma** *st-part-Ex*:
  $(x::hypreal) \in HFinite ==> \exists\, t \in Reals.\ x\ @{=}\ t$
⟨*proof*⟩

There is a unique real infinitely close

**lemma** *st-part-Ex1*: $x \in HFinite ==> EX!\ t::hypreal.\ t \in Reals\ \&\ x\ @{=}\ t$
⟨*proof*⟩

## 27.9 Finite, Infinite and Infinitesimal

**lemma** *HFinite-Int-HInfinite-empty* [*simp*]: *HFinite Int HInfinite* = {}
⟨*proof*⟩

**lemma** *HFinite-not-HInfinite*:
 **assumes** $x$: $x \in HFinite$ **shows** $x \notin HInfinite$
⟨*proof*⟩

**lemma** *not-HFinite-HInfinite*: $x \notin HFinite ==> x \in HInfinite$
⟨*proof*⟩

**lemma** *HInfinite-HFinite-disj*: $x \in HInfinite \mid x \in HFinite$
⟨*proof*⟩

**lemma** *HInfinite-HFinite-iff*: $(x \in HInfinite) = (x \notin HFinite)$
⟨*proof*⟩

**lemma** *HFinite-HInfinite-iff*: $(x \in HFinite) = (x \notin HInfinite)$
⟨*proof*⟩

**lemma** *HInfinite-diff-HFinite-Infinitesimal-disj*:
  $x \notin Infinitesimal ==> x \in HInfinite \mid x \in HFinite - Infinitesimal$
⟨*proof*⟩

**lemma** *HFinite-inverse*:
 **fixes** $x$ :: *'a::real-normed-div-algebra star*
 **shows** [| $x \in HFinite$; $x \notin Infinitesimal$ |] $==>$ *inverse* $x \in HFinite$
⟨*proof*⟩

**lemma** *HFinite-inverse2*:
 **fixes** $x$ :: *'a::real-normed-div-algebra star*
 **shows** $x \in HFinite - Infinitesimal ==>$ *inverse* $x \in HFinite$
⟨*proof*⟩

**lemma** *Infinitesimal-inverse-HFinite*:
**fixes** $x$ :: $'a$::*real-normed-div-algebra star*
**shows** $x \notin$ *Infinitesimal* ==> *inverse*$(x) \in$ *HFinite*
⟨*proof*⟩

**lemma** *HFinite-not-Infinitesimal-inverse*:
**fixes** $x$ :: $'a$::*real-normed-div-algebra star*
**shows** $x \in$ *HFinite* $-$ *Infinitesimal* ==> *inverse* $x \in$ *HFinite* $-$ *Infinitesimal*
⟨*proof*⟩

**lemma** *approx-inverse*:
**fixes** $x$ $y$ :: $'a$::*real-normed-div-algebra star*
**shows**
[| $x$ @= $y$; $y \in$ *HFinite* $-$ *Infinitesimal* |]
==> *inverse* $x$ @= *inverse* $y$
⟨*proof*⟩

**lemmas** *star-of-approx-inverse* = *star-of-HFinite-diff-Infinitesimal* [*THEN* [*2*] *approx-inverse*]
**lemmas** *hypreal-of-real-approx-inverse* = *hypreal-of-real-HFinite-diff-Infinitesimal*
[*THEN* [*2*] *approx-inverse*]

**lemma** *inverse-add-Infinitesimal-approx*:
**fixes** $x$ $h$ :: $'a$::*real-normed-div-algebra star*
**shows**
[| $x \in$ *HFinite* $-$ *Infinitesimal*;
$h \in$ *Infinitesimal* |] ==> *inverse*$(x + h)$ @= *inverse* $x$
⟨*proof*⟩

**lemma** *inverse-add-Infinitesimal-approx2*:
**fixes** $x$ $h$ :: $'a$::*real-normed-div-algebra star*
**shows**
[| $x \in$ *HFinite* $-$ *Infinitesimal*;
$h \in$ *Infinitesimal* |] ==> *inverse*$(h + x)$ @= *inverse* $x$
⟨*proof*⟩

**lemma** *inverse-add-Infinitesimal-approx-Infinitesimal*:
**fixes** $x$ $h$ :: $'a$::*real-normed-div-algebra star*
**shows**
[| $x \in$ *HFinite* $-$ *Infinitesimal*;
$h \in$ *Infinitesimal* |] ==> *inverse*$(x + h)$ $-$ *inverse* $x$ @= $h$
⟨*proof*⟩

**lemma** *Infinitesimal-square-iff*:
**fixes** $x$ :: $'a$::*real-normed-div-algebra star*
**shows** $(x \in$ *Infinitesimal*$)$ = $(x*x \in$ *Infinitesimal*$)$
⟨*proof*⟩
**declare** *Infinitesimal-square-iff* [*symmetric*, *simp*]

**lemma** *HFinite-square-iff* [*simp*]:
**fixes** $x$ :: $'a$::*real-normed-div-algebra star*
**shows** $(x*x \in HFinite) = (x \in HFinite)$
⟨*proof*⟩

**lemma** *HInfinite-square-iff* [*simp*]:
**fixes** $x$ :: $'a$::*real-normed-div-algebra star*
**shows** $(x*x \in HInfinite) = (x \in HInfinite)$
⟨*proof*⟩

**lemma** *approx-HFinite-mult-cancel*:
**fixes** $a\ w\ z$ :: $'a$::*real-normed-div-algebra star*
**shows** [| $a$: *HFinite−Infinitesimal*; $a*\ w$ @= $a*z$ |] ==> $w$ @= $z$
⟨*proof*⟩

**lemma** *approx-HFinite-mult-cancel-iff1*:
**fixes** $a\ w\ z$ :: $'a$::*real-normed-div-algebra star*
**shows** $a$: *HFinite−Infinitesimal* ==> ($a * w$ @= $a * z$) = ($w$ @= $z$)
⟨*proof*⟩

**lemma** *HInfinite-HFinite-add-cancel*:
[| $x + y \in HInfinite$; $y \in HFinite$ |] ==> $x \in HInfinite$
⟨*proof*⟩

**lemma** *HInfinite-HFinite-add*:
[| $x \in HInfinite$; $y \in HFinite$ |] ==> $x + y \in HInfinite$
⟨*proof*⟩

**lemma** *HInfinite-ge-HInfinite*:
[| $(x$::*hypreal*$) \in HInfinite$; $x \le y$; $0 \le x$ |] ==> $y \in HInfinite$
⟨*proof*⟩

**lemma** *Infinitesimal-inverse-HInfinite*:
**fixes** $x$ :: $'a$::*real-normed-div-algebra star*
**shows** [| $x \in Infinitesimal$; $x \neq 0$ |] ==> *inverse* $x \in HInfinite$
⟨*proof*⟩

**lemma** *HInfinite-HFinite-not-Infinitesimal-mult*:
**fixes** $x\ y$ :: $'a$::*real-normed-div-algebra star*
**shows** [| $x \in HInfinite$; $y \in HFinite - Infinitesimal$ |]
==> $x * y \in HInfinite$
⟨*proof*⟩

**lemma** *HInfinite-HFinite-not-Infinitesimal-mult2*:
**fixes** $x\ y$ :: $'a$::*real-normed-div-algebra star*
**shows** [| $x \in HInfinite$; $y \in HFinite - Infinitesimal$ |]
==> $y * x \in HInfinite$
⟨*proof*⟩

**lemma** *HInfinite-gt-SReal*:
 $[| (x::hypreal) \in HInfinite;\ 0 < x;\ y \in Reals\ |] ==> y < x$
⟨*proof*⟩

**lemma** *HInfinite-gt-zero-gt-one*:
 $[| (x::hypreal) \in HInfinite;\ 0 < x\ |] ==> 1 < x$
⟨*proof*⟩

**lemma** *not-HInfinite-one* [*simp*]: $1 \notin HInfinite$
⟨*proof*⟩

**lemma** *approx-hrabs-disj*: $abs\ (x::hypreal)\ @= x \mid abs\ x\ @= -x$
⟨*proof*⟩

## 27.10 Theorems about Monads

**lemma** *monad-hrabs-Un-subset*: $monad\ (abs\ x) \le monad(x::hypreal)\ Un\ monad(-x)$
⟨*proof*⟩

**lemma** *Infinitesimal-monad-eq*: $e \in Infinitesimal ==> monad\ (x+e) = monad\ x$
⟨*proof*⟩

**lemma** *mem-monad-iff*: $(u \in monad\ x) = (-u \in monad\ (-x))$
⟨*proof*⟩

**lemma** *Infinitesimal-monad-zero-iff*: $(x \in Infinitesimal) = (x \in monad\ 0)$
⟨*proof*⟩

**lemma** *monad-zero-minus-iff*: $(x \in monad\ 0) = (-x \in monad\ 0)$
⟨*proof*⟩

**lemma** *monad-zero-hrabs-iff*: $((x::hypreal) \in monad\ 0) = (abs\ x \in monad\ 0)$
⟨*proof*⟩

**lemma** *mem-monad-self* [*simp*]: $x \in monad\ x$
⟨*proof*⟩

## 27.11 Proof that $x \approx y$ implies $|x| \approx |y|$

**lemma** *approx-subset-monad*: $x\ @= y ==> \{x,y\} \le monad\ x$
⟨*proof*⟩

**lemma** *approx-subset-monad2*: $x\ @= y ==> \{x,y\} \le monad\ y$
⟨*proof*⟩

**lemma** *mem-monad-approx*: $u \in monad\ x ==> x\ @= u$
⟨*proof*⟩

**lemma** *approx-mem-monad*: x @= u ==> u ∈ monad x
⟨proof⟩

**lemma** *approx-mem-monad2*: x @= u ==> x ∈ monad u
⟨proof⟩

**lemma** *approx-mem-monad-zero*: [| x @= y;x ∈ monad 0 |] ==> y ∈ monad 0
⟨proof⟩

**lemma** *Infinitesimal-approx-hrabs*:
  [| x @= y; (x::hypreal) ∈ Infinitesimal |] ==> abs x @= abs y
⟨proof⟩

**lemma** *less-Infinitesimal-less*:
  [| 0 < x;  (x::hypreal) ∉Infinitesimal;  e :Infinitesimal |] ==> e < x
⟨proof⟩

**lemma** *Ball-mem-monad-gt-zero*:
  [| 0 < (x::hypreal);  x ∉ Infinitesimal; u ∈ monad x |] ==> 0 < u
⟨proof⟩

**lemma** *Ball-mem-monad-less-zero*:
  [| (x::hypreal) < 0; x ∉ Infinitesimal; u ∈ monad x |] ==> u < 0
⟨proof⟩

**lemma** *lemma-approx-gt-zero*:
  [|0 < (x::hypreal); x ∉ Infinitesimal; x @= y|] ==> 0 < y
⟨proof⟩

**lemma** *lemma-approx-less-zero*:
  [|(x::hypreal) < 0; x ∉ Infinitesimal; x @= y|] ==> y < 0
⟨proof⟩

**theorem** *approx-hrabs*: (x::hypreal) @= y ==> abs x @= abs y
⟨proof⟩

**lemma** *approx-hrabs-zero-cancel*: abs(x::hypreal) @= 0 ==> x @= 0
⟨proof⟩

**lemma** *approx-hrabs-add-Infinitesimal*:
  (e::hypreal) ∈ Infinitesimal ==> abs x @= abs(x+e)
⟨proof⟩

**lemma** *approx-hrabs-add-minus-Infinitesimal*:
  (e::hypreal) ∈ Infinitesimal ==> abs x @= abs(x + −e)
⟨proof⟩

**lemma** *hrabs-add-Infinitesimal-cancel*:
  [| (e::hypreal) ∈ Infinitesimal; e′ ∈ Infinitesimal;

$abs(x+e) = abs(y+e')|] ==> abs\ x\ @= abs\ y$
⟨*proof*⟩

**lemma** *hrabs-add-minus-Infinitesimal-cancel*:
[| $(e::hypreal) \in Infinitesimal$; $e' \in Infinitesimal$;
$abs(x + -e) = abs(y + -e')|] ==> abs\ x\ @= abs\ y$
⟨*proof*⟩

## 27.12 More *HFinite* and *Infinitesimal* Theorems

**lemma** *Infinitesimal-add-hypreal-of-real-less*:
[| $x < y$; $u \in Infinitesimal$ |]
==> *hypreal-of-real* $x + u <$ *hypreal-of-real* $y$
⟨*proof*⟩

**lemma** *Infinitesimal-add-hrabs-hypreal-of-real-less*:
[| $x \in Infinitesimal$; *abs*(*hypreal-of-real* $r$) $<$ *hypreal-of-real* $y$ |]
==> *abs* (*hypreal-of-real* $r + x$) $<$ *hypreal-of-real* $y$
⟨*proof*⟩

**lemma** *Infinitesimal-add-hrabs-hypreal-of-real-less2*:
[| $x \in Infinitesimal$; *abs*(*hypreal-of-real* $r$) $<$ *hypreal-of-real* $y$ |]
==> *abs* ($x$ + *hypreal-of-real* $r$) $<$ *hypreal-of-real* $y$
⟨*proof*⟩

**lemma** *hypreal-of-real-le-add-Infininitesimal-cancel*:
[| $u \in Infinitesimal$; $v \in Infinitesimal$;
*hypreal-of-real* $x + u \le$ *hypreal-of-real* $y + v$ |]
==> *hypreal-of-real* $x \le$ *hypreal-of-real* $y$
⟨*proof*⟩

**lemma** *hypreal-of-real-le-add-Infininitesimal-cancel2*:
[| $u \in Infinitesimal$; $v \in Infinitesimal$;
*hypreal-of-real* $x + u \le$ *hypreal-of-real* $y + v$ |]
==> $x \le y$
⟨*proof*⟩

**lemma** *hypreal-of-real-less-Infinitesimal-le-zero*:
[| *hypreal-of-real* $x < e$; $e \in Infinitesimal$ |] ==> *hypreal-of-real* $x \le 0$
⟨*proof*⟩

**lemma** *Infinitesimal-add-not-zero*:
[| $h \in Infinitesimal$; $x \neq 0$ |] ==> *star-of* $x + h \neq 0$
⟨*proof*⟩

**lemma** *Infinitesimal-square-cancel* [*simp*]:
$(x::hypreal)*x + y*y \in Infinitesimal ==> x*x \in Infinitesimal$
⟨*proof*⟩

**lemma** *HFinite-square-cancel* [*simp*]:
(*x*::*hypreal*)**x* + *y***y* ∈ *HFinite* ==> *x***x* ∈ *HFinite*
⟨*proof*⟩

**lemma** *Infinitesimal-square-cancel2* [*simp*]:
(*x*::*hypreal*)**x* + *y***y* ∈ *Infinitesimal* ==> *y***y* ∈ *Infinitesimal*
⟨*proof*⟩

**lemma** *HFinite-square-cancel2* [*simp*]:
(*x*::*hypreal*)**x* + *y***y* ∈ *HFinite* ==> *y***y* ∈ *HFinite*
⟨*proof*⟩

**lemma** *Infinitesimal-sum-square-cancel* [*simp*]:
(*x*::*hypreal*)**x* + *y***y* + *z***z* ∈ *Infinitesimal* ==> *x***x* ∈ *Infinitesimal*
⟨*proof*⟩

**lemma** *HFinite-sum-square-cancel* [*simp*]:
(*x*::*hypreal*)**x* + *y***y* + *z***z* ∈ *HFinite* ==> *x***x* ∈ *HFinite*
⟨*proof*⟩

**lemma** *Infinitesimal-sum-square-cancel2* [*simp*]:
(*y*::*hypreal*)**y* + *x***x* + *z***z* ∈ *Infinitesimal* ==> *x***x* ∈ *Infinitesimal*
⟨*proof*⟩

**lemma** *HFinite-sum-square-cancel2* [*simp*]:
(*y*::*hypreal*)**y* + *x***x* + *z***z* ∈ *HFinite* ==> *x***x* ∈ *HFinite*
⟨*proof*⟩

**lemma** *Infinitesimal-sum-square-cancel3* [*simp*]:
(*z*::*hypreal*)**z* + *y***y* + *x***x* ∈ *Infinitesimal* ==> *x***x* ∈ *Infinitesimal*
⟨*proof*⟩

**lemma** *HFinite-sum-square-cancel3* [*simp*]:
(*z*::*hypreal*)**z* + *y***y* + *x***x* ∈ *HFinite* ==> *x***x* ∈ *HFinite*
⟨*proof*⟩

**lemma** *monad-hrabs-less*:
[| *y* ∈ *monad x*; *0* < *hypreal-of-real e* |]
==> *abs* (*y* − *x*) < *hypreal-of-real e*
⟨*proof*⟩

**lemma** *mem-monad-SReal-HFinite*:
*x* ∈ *monad* (*hypreal-of-real a*) ==> *x* ∈ *HFinite*
⟨*proof*⟩

## 27.13 Theorems about Standard Part

**lemma** *st-approx-self*: *x* ∈ *HFinite* ==> *st x* @= *x*

⟨*proof*⟩

**lemma** *st-SReal*: $x \in \mathit{HFinite} ==> st\ x \in \mathit{Reals}$
⟨*proof*⟩

**lemma** *st-HFinite*: $x \in \mathit{HFinite} ==> st\ x \in \mathit{HFinite}$
⟨*proof*⟩

**lemma** *st-unique*: $[\![r \in \mathbb{R};\ r \approx x]\!] \Longrightarrow st\ x = r$
⟨*proof*⟩

**lemma** *st-SReal-eq*: $x \in \mathit{Reals} ==> st\ x = x$
⟨*proof*⟩

**lemma** *st-hypreal-of-real* [*simp*]: $st\ (\mathit{hypreal\text{-}of\text{-}real}\ x) = \mathit{hypreal\text{-}of\text{-}real}\ x$
⟨*proof*⟩

**lemma** *st-eq-approx*: $[|\ x \in \mathit{HFinite};\ y \in \mathit{HFinite};\ st\ x = st\ y\ |] ==> x\ @= y$
⟨*proof*⟩

**lemma** *approx-st-eq*:
  **assumes** $x \in \mathit{HFinite}$ **and** $y \in \mathit{HFinite}$ **and** $x\ @= y$
  **shows** $st\ x = st\ y$
⟨*proof*⟩

**lemma** *st-eq-approx-iff*:
  $[|\ x \in \mathit{HFinite};\ y \in \mathit{HFinite}|]$
    $==> (x\ @= y) = (st\ x = st\ y)$
⟨*proof*⟩

**lemma** *st-Infinitesimal-add-SReal*:
  $[|\ x \in \mathit{Reals};\ e \in \mathit{Infinitesimal}\ |] ==> st(x + e) = x$
⟨*proof*⟩

**lemma** *st-Infinitesimal-add-SReal2*:
  $[|\ x \in \mathit{Reals};\ e \in \mathit{Infinitesimal}\ |] ==> st(e + x) = x$
⟨*proof*⟩

**lemma** *HFinite-st-Infinitesimal-add*:
  $x \in \mathit{HFinite} ==> \exists\, e \in \mathit{Infinitesimal}.\ x = st(x) + e$
⟨*proof*⟩

**lemma** *st-add*: $[\![x \in \mathit{HFinite};\ y \in \mathit{HFinite}]\!] \Longrightarrow st\ (x + y) = st\ x + st\ y$
⟨*proof*⟩

**lemma** *st-number-of* [*simp*]: $st\ (\mathit{number\text{-}of}\ w) = \mathit{number\text{-}of}\ w$
⟨*proof*⟩

**lemma** [*simp*]: $st\ 0 = 0\ st\ 1 = 1$
⟨*proof*⟩

**lemma** *st-minus*: $x \in HFinite \Longrightarrow st\ (-\ x) = -\ st\ x$
⟨*proof*⟩

**lemma** *st-diff*: $[\![x \in HFinite;\ y \in HFinite]\!] \Longrightarrow st\ (x - y) = st\ x - st\ y$
⟨*proof*⟩

**lemma** *st-mult*: $[\![x \in HFinite;\ y \in HFinite]\!] \Longrightarrow st\ (x * y) = st\ x * st\ y$
⟨*proof*⟩

**lemma** *st-Infinitesimal*: $x \in Infinitesimal ==> st\ x = 0$
⟨*proof*⟩

**lemma** *st-not-Infinitesimal*: $st(x) \neq 0 ==> x \notin Infinitesimal$
⟨*proof*⟩

**lemma** *st-inverse*:
  $[|\ x \in HFinite;\ st\ x \neq 0\ |]$
  $==> st(inverse\ x) = inverse\ (st\ x)$
⟨*proof*⟩

**lemma** *st-divide* [*simp*]:
  $[|\ x \in HFinite;\ y \in HFinite;\ st\ y \neq 0\ |]$
  $==> st(x/y) = (st\ x)\ /\ (st\ y)$
⟨*proof*⟩

**lemma** *st-idempotent* [*simp*]: $x \in HFinite ==> st(st(x)) = st(x)$
⟨*proof*⟩

**lemma** *Infinitesimal-add-st-less*:
  $[|\ x \in HFinite;\ y \in HFinite;\ u \in Infinitesimal;\ st\ x < st\ y\ |]$
  $==> st\ x + u < st\ y$
⟨*proof*⟩

**lemma** *Infinitesimal-add-st-le-cancel*:
  $[|\ x \in HFinite;\ y \in HFinite;$
    $u \in Infinitesimal;\ st\ x \leq st\ y + u$
  $|] ==> st\ x \leq st\ y$
⟨*proof*⟩

**lemma** *st-le*: $[|\ x \in HFinite;\ y \in HFinite;\ x \leq y\ |] ==> st(x) \leq st(y)$
⟨*proof*⟩

**lemma** *st-zero-le*: $[|\ 0 \leq x;\ \ x \in HFinite\ |] ==> 0 \leq st\ x$
⟨*proof*⟩

**lemma** *st-zero-ge*: $[|\ x \leq 0;\ \ x \in HFinite\ |] ==> st\ x \leq 0$

⟨*proof*⟩

**lemma** *st-hrabs*: $x \in HFinite ==> abs(st\ x) = st(abs\ x)$
⟨*proof*⟩

## 27.14 Alternative Definitions using Free Ultrafilter

### 27.14.1 *HFinite*

**lemma** *HFinite-FreeUltrafilterNat*:
$star\text{-}n\ X \in HFinite$
$==> \exists u.\ \{n.\ norm\ (X\ n) < u\} \in FreeUltrafilterNat$
⟨*proof*⟩

**lemma** *FreeUltrafilterNat-HFinite*:
$\exists u.\ \{n.\ norm\ (X\ n) < u\} \in FreeUltrafilterNat$
$==>\ star\text{-}n\ X \in HFinite$
⟨*proof*⟩

**lemma** *HFinite-FreeUltrafilterNat-iff*:
$(star\text{-}n\ X \in HFinite) = (\exists u.\ \{n.\ norm\ (X\ n) < u\} \in FreeUltrafilterNat)$
⟨*proof*⟩

### 27.14.2 *HInfinite*

**lemma** *lemma-Compl-eq*: $- \{n.\ u < norm\ (xa\ n)\} = \{n.\ norm\ (xa\ n) \leq u\}$
⟨*proof*⟩

**lemma** *lemma-Compl-eq2*: $- \{n.\ norm\ (xa\ n) < u\} = \{n.\ u \leq norm\ (xa\ n)\}$
⟨*proof*⟩

**lemma** *lemma-Int-eq1*:
$\{n.\ norm\ (xa\ n) \leq u\}\ Int\ \{n.\ u \leq norm\ (xa\ n)\}$
$= \{n.\ norm(xa\ n) = u\}$
⟨*proof*⟩

**lemma** *lemma-FreeUltrafilterNat-one*:
$\{n.\ norm\ (xa\ n) = u\} \leq \{n.\ norm\ (xa\ n) < u + (1::real)\}$
⟨*proof*⟩

**lemma** *FreeUltrafilterNat-const-Finite*:
$\{n.\ norm\ (X\ n) = u\} \in FreeUltrafilterNat ==> star\text{-}n\ X \in HFinite$
⟨*proof*⟩

**lemma** *HInfinite-FreeUltrafilterNat*:
$star\text{-}n\ X \in HInfinite ==> \forall u.\ \{n.\ u < norm\ (X\ n)\} \in FreeUltrafilterNat$
⟨*proof*⟩

**lemma** *lemma-Int-HI*:

$\{n.\ norm\ (Xa\ n) < u\}\ Int\ \{n.\ X\ n = Xa\ n\} \subseteq \{n.\ norm\ (X\ n) < (u::real)\}$
⟨proof⟩

**lemma** *lemma-Int-HIa*: $\{n.\ u < norm\ (X\ n)\}\ Int\ \{n.\ norm\ (X\ n) < u\} = \{\}$
⟨proof⟩

**lemma** *FreeUltrafilterNat-HInfinite*:
$\forall u.\ \{n.\ u < norm\ (X\ n)\} \in FreeUltrafilterNat ==> star\text{-}n\ X \in HInfinite$
⟨proof⟩

**lemma** *HInfinite-FreeUltrafilterNat-iff*:
$(star\text{-}n\ X \in HInfinite) = (\forall u.\ \{n.\ u < norm\ (X\ n)\} \in FreeUltrafilterNat)$
⟨proof⟩

### 27.14.3 *Infinitesimal*

**lemma** *ball-SReal-eq*: $(\forall x::hypreal \in Reals.\ P\ x) = (\forall x::real.\ P\ (star\text{-}of\ x))$
⟨proof⟩

**lemma** *Infinitesimal-FreeUltrafilterNat*:
$star\text{-}n\ X \in Infinitesimal ==> \forall u{>}0.\ \{n.\ norm\ (X\ n) < u\} \in \mathcal{U}$
⟨proof⟩

**lemma** *FreeUltrafilterNat-Infinitesimal*:
$\forall u{>}0.\ \{n.\ norm\ (X\ n) < u\} \in \mathcal{U} ==> star\text{-}n\ X \in Infinitesimal$
⟨proof⟩

**lemma** *Infinitesimal-FreeUltrafilterNat-iff*:
$(star\text{-}n\ X \in Infinitesimal) = (\forall u{>}0.\ \{n.\ norm\ (X\ n) < u\} \in \mathcal{U})$
⟨proof⟩

**lemma** *lemma-Infinitesimal*:
$(\forall r.\ 0 < r --> x < r) = (\forall n.\ x < inverse(real\ (Suc\ n)))$
⟨proof⟩

**lemma** *lemma-Infinitesimal2*:
$(\forall r \in Reals.\ 0 < r --> x < r) =$
$(\forall n.\ x < inverse(hypreal\text{-}of\text{-}nat\ (Suc\ n)))$
⟨proof⟩

**lemma** *Infinitesimal-hypreal-of-nat-iff*:
$Infinitesimal = \{x.\ \forall n.\ hnorm\ x < inverse\ (hypreal\text{-}of\text{-}nat\ (Suc\ n))\}$
⟨proof⟩

## 27.15 Proof that $\omega$ is an infinite number

It will follow that epsilon is an infinitesimal number.

**lemma** *Suc-Un-eq*: $\{n.\ n < Suc\ m\} = \{n.\ n < m\}\ Un\ \{n.\ n = m\}$
⟨*proof*⟩

**lemma** *finite-nat-segment*: *finite* $\{n::nat.\ n < m\}$
⟨*proof*⟩

**lemma** *finite-real-of-nat-segment*: *finite* $\{n::nat.\ real\ n < real\ (m::nat)\}$
⟨*proof*⟩

**lemma** *finite-real-of-nat-less-real*: *finite* $\{n::nat.\ real\ n < u\}$
⟨*proof*⟩

**lemma** *lemma-real-le-Un-eq*:
$\{n.\ f\ n \le u\} = \{n.\ f\ n < u\}\ Un\ \{n.\ u = (f\ n :: real)\}$
⟨*proof*⟩

**lemma** *finite-real-of-nat-le-real*: *finite* $\{n::nat.\ real\ n \le u\}$
⟨*proof*⟩

**lemma** *finite-rabs-real-of-nat-le-real*: *finite* $\{n::nat.\ abs(real\ n) \le u\}$
⟨*proof*⟩

**lemma** *rabs-real-of-nat-le-real-FreeUltrafilterNat*:
$\{n.\ abs(real\ n) \le u\} \notin FreeUltrafilterNat$
⟨*proof*⟩

**lemma** *FreeUltrafilterNat-nat-gt-real*: $\{n.\ u < real\ n\} \in FreeUltrafilterNat$
⟨*proof*⟩

**lemma** *Compl-real-le-eq*: $-\ \{n::nat.\ real\ n \le u\} = \{n.\ u < real\ n\}$
⟨*proof*⟩

$\omega$ is a member of *HInfinite*

**lemma** *FreeUltrafilterNat-omega*: $\{n.\ u < real\ n\} \in FreeUltrafilterNat$
⟨*proof*⟩

**theorem** *HInfinite-omega* [*simp*]: *omega* $\in$ *HInfinite*
⟨*proof*⟩

**lemma** *Infinitesimal-epsilon* [*simp*]: *epsilon* $\in$ *Infinitesimal*
⟨*proof*⟩

**lemma** *HFinite-epsilon* [*simp*]: $epsilon \in HFinite$
⟨*proof*⟩

**lemma** *epsilon-approx-zero* [*simp*]: *epsilon* @= *0*
⟨*proof*⟩

**lemma** *real-of-nat-less-inverse-iff*:
$0 < u$ ==> $(u < inverse\ (real(Suc\ n))) = (real(Suc\ n) < inverse\ u)$
⟨*proof*⟩

**lemma** *finite-inverse-real-of-posnat-gt-real*:
$0 < u$ ==> $finite\ \{n.\ u < inverse(real(Suc\ n))\}$
⟨*proof*⟩

**lemma** *lemma-real-le-Un-eq2*:
$\{n.\ u \leq inverse(real(Suc\ n))\} =$
$\{n.\ u < inverse(real(Suc\ n))\}\ Un\ \{n.\ u = inverse(real(Suc\ n))\}$
⟨*proof*⟩

**lemma** *real-of-nat-inverse-eq-iff*:
$(u = inverse\ (real(Suc\ n))) = (real(Suc\ n) = inverse\ u)$
⟨*proof*⟩

**lemma** *lemma-finite-omega-set2*: $finite\ \{n::nat.\ u = inverse(real(Suc\ n))\}$
⟨*proof*⟩

**lemma** *finite-inverse-real-of-posnat-ge-real*:
$0 < u$ ==> $finite\ \{n.\ u \leq inverse(real(Suc\ n))\}$
⟨*proof*⟩

**lemma** *inverse-real-of-posnat-ge-real-FreeUltrafilterNat*:
$0 < u$ ==> $\{n.\ u \leq inverse(real(Suc\ n))\} \notin FreeUltrafilterNat$
⟨*proof*⟩

**lemma** *Compl-le-inverse-eq*:
$-\ \{n.\ u \leq inverse(real(Suc\ n))\} =$
$\{n.\ inverse(real(Suc\ n)) < u\}$
⟨*proof*⟩

**lemma** *FreeUltrafilterNat-inverse-real-of-posnat*:
$0 < u$ ==>
$\{n.\ inverse(real(Suc\ n)) < u\} \in FreeUltrafilterNat$
⟨*proof*⟩

Example of an hypersequence (i.e. an extended standard sequence) whose term with an hypernatural suffix is an infinitesimal i.e. the whn'nth term of the hypersequence is a member of Infinitesimal

**lemma** *SEQ-Infinitesimal*:
    $( *f* (\%n::nat.\ inverse(real(Suc\ n))))\ whn : Infinitesimal$
⟨*proof*⟩

Example where we get a hyperreal from a real sequence for which a particular property holds. The theorem is used in proofs about equivalence of nonstandard and standard neighbourhoods. Also used for equivalence of nonstandard ans standard definitions of pointwise limit.

**lemma** *real-seq-to-hypreal-Infinitesimal*:
    $\forall n.\ norm(X\ n\ -\ x) < inverse(real(Suc\ n))$
    $==> star\text{-}n\ X\ -\ star\text{-}of\ x \in Infinitesimal$
⟨*proof*⟩

**lemma** *real-seq-to-hypreal-approx*:
    $\forall n.\ norm(X\ n\ -\ x) < inverse(real(Suc\ n))$
    $==> star\text{-}n\ X\ @= star\text{-}of\ x$
⟨*proof*⟩

**lemma** *real-seq-to-hypreal-approx2*:
    $\forall n.\ norm(x\ -\ X\ n) < inverse(real(Suc\ n))$
        $==> star\text{-}n\ X\ @= star\text{-}of\ x$
⟨*proof*⟩

**lemma** *real-seq-to-hypreal-Infinitesimal2*:
    $\forall n.\ norm(X\ n\ -\ Y\ n) < inverse(real(Suc\ n))$
    $==> star\text{-}n\ X\ -\ star\text{-}n\ Y \in Infinitesimal$
⟨*proof*⟩

**end**

# 28 NSComplex: Nonstandard Complex Numbers

**theory** *NSComplex*
**imports** *Complex ../Hyperreal/NSA*
**begin**

**types** *hcomplex* = *complex star*

**abbreviation**
  *hcomplex-of-complex* :: *complex* => *complex star* **where**
  *hcomplex-of-complex* == *star-of*

**abbreviation**
  *hcmod* :: *complex star* => *real star* **where**
  *hcmod* == *hnorm*

**definition**
*hRe* :: *hcomplex* => *hypreal* **where**
*hRe* = *∗f∗ Re*

**definition**
*hIm* :: *hcomplex* => *hypreal* **where**
*hIm* = *∗f∗ Im*

**definition**
*iii* :: *hcomplex* **where**
*iii* = *star-of ii*

**definition**
*hcnj* :: *hcomplex* => *hcomplex* **where**
*hcnj* = *∗f∗ cnj*

**definition**
*hsgn* :: *hcomplex* => *hcomplex* **where**
*hsgn* = *∗f∗ sgn*

**definition**
*harg* :: *hcomplex* => *hypreal* **where**
*harg* = *∗f∗ arg*

**definition**

*hcis* :: *hypreal* => *hcomplex* **where**
*hcis* = *∗f∗ cis*

**abbreviation**
*hcomplex-of-hypreal* :: *hypreal* ⇒ *hcomplex* **where**
*hcomplex-of-hypreal* ≡ *of-hypreal*

**definition**

*hrcis* :: [*hypreal*, *hypreal*] => *hcomplex* **where** *hrcis* = $*f2*$ *rcis*

**definition**
*hexpi* :: *hcomplex* => *hcomplex* **where**
*hexpi* = $*f*$ *expi*

**definition**
*HComplex* :: [*hypreal*,*hypreal*] => *hcomplex* **where**
*HComplex* = $*f2*$ *Complex*

**lemmas** *hcomplex-defs* [*transfer-unfold*] =
*hRe-def hIm-def iii-def hcnj-def hsgn-def harg-def hcis-def*
*hrcis-def hexpi-def HComplex-def*

**lemma** *Standard-hRe* [*simp*]: $x \in \mathit{Standard} \Longrightarrow \mathit{hRe}\ x \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *Standard-hIm* [*simp*]: $x \in \mathit{Standard} \Longrightarrow \mathit{hIm}\ x \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *Standard-iii* [*simp*]: $\mathit{iii} \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *Standard-hcnj* [*simp*]: $x \in \mathit{Standard} \Longrightarrow \mathit{hcnj}\ x \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *Standard-hsgn* [*simp*]: $x \in \mathit{Standard} \Longrightarrow \mathit{hsgn}\ x \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *Standard-harg* [*simp*]: $x \in \mathit{Standard} \Longrightarrow \mathit{harg}\ x \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *Standard-hcis* [*simp*]: $r \in \mathit{Standard} \Longrightarrow \mathit{hcis}\ r \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *Standard-hexpi* [*simp*]: $x \in \mathit{Standard} \Longrightarrow \mathit{hexpi}\ x \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *Standard-hrcis* [*simp*]:
$[\![r \in \mathit{Standard};\ s \in \mathit{Standard}]\!] \Longrightarrow \mathit{hrcis}\ r\ s \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *Standard-HComplex* [*simp*]:
$[\![r \in \mathit{Standard};\ s \in \mathit{Standard}]\!] \Longrightarrow \mathit{HComplex}\ r\ s \in \mathit{Standard}$
⟨*proof*⟩

**lemma** *hcmod-def*: *hcmod* = $*f*$ *cmod*
⟨*proof*⟩

## 28.1 Properties of Nonstandard Real and Imaginary Parts

**lemma** *hcomplex-hRe-hIm-cancel-iff*:
$!!w\ z.\ (w{=}z) = (hRe(w) = hRe(z)\ \&\ hIm(w) = hIm(z))$
⟨*proof*⟩

**lemma** *hcomplex-equality* [*intro?*]:
$!!z\ w.\ hRe\ z = hRe\ w ==> hIm\ z = hIm\ w ==> z = w$
⟨*proof*⟩

**lemma** *hcomplex-hRe-zero* [*simp*]: $hRe\ 0 = 0$
⟨*proof*⟩

**lemma** *hcomplex-hIm-zero* [*simp*]: $hIm\ 0 = 0$
⟨*proof*⟩

**lemma** *hcomplex-hRe-one* [*simp*]: $hRe\ 1 = 1$
⟨*proof*⟩

**lemma** *hcomplex-hIm-one* [*simp*]: $hIm\ 1 = 0$
⟨*proof*⟩

## 28.2 Addition for Nonstandard Complex Numbers

**lemma** *hRe-add*: $!!x\ y.\ hRe(x + y) = hRe(x) + hRe(y)$
⟨*proof*⟩

**lemma** *hIm-add*: $!!x\ y.\ hIm(x + y) = hIm(x) + hIm(y)$
⟨*proof*⟩

## 28.3 More Minus Laws

**lemma** *hRe-minus*: $!!z.\ hRe(-z) = -\ hRe(z)$
⟨*proof*⟩

**lemma** *hIm-minus*: $!!z.\ hIm(-z) = -\ hIm(z)$
⟨*proof*⟩

**lemma** *hcomplex-add-minus-eq-minus*:
$x + y = (0::hcomplex) ==> x = -y$
⟨*proof*⟩

**lemma** *hcomplex-i-mult-eq* [*simp*]: $iii * iii = -\ 1$
⟨*proof*⟩

**lemma** *hcomplex-i-mult-left* [*simp*]: $!!z.\ iii * (iii * z) = -z$
⟨*proof*⟩

**lemma** *hcomplex-i-not-zero* [*simp*]: $iii \neq 0$
⟨*proof*⟩

## 28.4 More Multiplication Laws

**lemma** *hcomplex-mult-minus-one*: $- 1 * (z::hcomplex) = -z$
⟨*proof*⟩

**lemma** *hcomplex-mult-minus-one-right*: $(z::hcomplex) * - 1 = -z$
⟨*proof*⟩

**lemma** *hcomplex-mult-left-cancel*:
$(c::hcomplex) \neq (0::hcomplex) ==> (c*a=c*b) = (a=b)$
⟨*proof*⟩

**lemma** *hcomplex-mult-right-cancel*:
$(c::hcomplex) \neq (0::hcomplex) ==> (a*c=b*c) = (a=b)$
⟨*proof*⟩

## 28.5 Subraction and Division

**lemma** *hcomplex-diff-eq-eq* [*simp*]: $((x::hcomplex) - y = z) = (x = z + y)$

⟨*proof*⟩

## 28.6 Embedding Properties for *hcomplex-of-hypreal* Map

**lemma** *hRe-hcomplex-of-hypreal* [*simp*]: $!!z.\ hRe(hcomplex\text{-}of\text{-}hypreal\ z) = z$
⟨*proof*⟩

**lemma** *hIm-hcomplex-of-hypreal* [*simp*]: $!!z.\ hIm(hcomplex\text{-}of\text{-}hypreal\ z) = 0$
⟨*proof*⟩

**lemma** *hcomplex-of-hypreal-epsilon-not-zero* [*simp*]:
$hcomplex\text{-}of\text{-}hypreal\ epsilon \neq 0$
⟨*proof*⟩

## 28.7 HComplex theorems

**lemma** *hRe-HComplex* [*simp*]: $!!x\ y.\ hRe\ (HComplex\ x\ y) = x$
⟨*proof*⟩

**lemma** *hIm-HComplex* [*simp*]: $!!x\ y.\ hIm\ (HComplex\ x\ y) = y$
⟨*proof*⟩

**lemma** *hcomplex-surj* [*simp*]: $!!z.\ HComplex\ (hRe\ z)\ (hIm\ z) = z$
⟨*proof*⟩

**lemma** *hcomplex-induct* [*case-names rect*]:
$(\bigwedge x\ y.\ P\ (HComplex\ x\ y)) ==> P\ z$
⟨*proof*⟩

## 28.8 Modulus (Absolute Value) of Nonstandard Complex Number

**lemma** *hcomplex-of-hypreal-abs*:
*hcomplex-of-hypreal* (*abs x*) =
*hcomplex-of-hypreal*(*hcmod*(*hcomplex-of-hypreal x*))
⟨*proof*⟩

**lemma** *HComplex-inject* [*simp*]:
!!$x$ $y$ $x'$ $y'$. *HComplex* $x$ $y$ = *HComplex* $x'$ $y'$ = ($x$=$x'$ & $y$=$y'$)
⟨*proof*⟩

**lemma** *HComplex-add* [*simp*]:
!!$x1$ $y1$ $x2$ $y2$. *HComplex* $x1$ $y1$ + *HComplex* $x2$ $y2$ = *HComplex* ($x1$+$x2$) ($y1$+$y2$)
⟨*proof*⟩

**lemma** *HComplex-minus* [*simp*]: !!$x$ $y$. − *HComplex* $x$ $y$ = *HComplex* (−$x$) (−$y$)
⟨*proof*⟩

**lemma** *HComplex-diff* [*simp*]:
!!$x1$ $y1$ $x2$ $y2$. *HComplex* $x1$ $y1$ − *HComplex* $x2$ $y2$ = *HComplex* ($x1$−$x2$) ($y1$−$y2$)
⟨*proof*⟩

**lemma** *HComplex-mult* [*simp*]:
!!$x1$ $y1$ $x2$ $y2$. *HComplex* $x1$ $y1$ ∗ *HComplex* $x2$ $y2$ =
*HComplex* ($x1$∗$x2$ − $y1$∗$y2$) ($x1$∗$y2$ + $y1$∗$x2$)
⟨*proof*⟩

**lemma** *hcomplex-of-hypreal-eq*: !!$r$. *hcomplex-of-hypreal* $r$ = *HComplex* $r$ *0*
⟨*proof*⟩

**lemma** *HComplex-add-hcomplex-of-hypreal* [*simp*]:
!!$x$ $y$ $r$. *HComplex* $x$ $y$ + *hcomplex-of-hypreal* $r$ = *HComplex* ($x$+$r$) $y$
⟨*proof*⟩

**lemma** *hcomplex-of-hypreal-add-HComplex* [*simp*]:
!!$r$ $x$ $y$. *hcomplex-of-hypreal* $r$ + *HComplex* $x$ $y$ = *HComplex* ($r$+$x$) $y$
⟨*proof*⟩

**lemma** *HComplex-mult-hcomplex-of-hypreal*:
!!$x$ $y$ $r$. *HComplex* $x$ $y$ ∗ *hcomplex-of-hypreal* $r$ = *HComplex* ($x$∗$r$) ($y$∗$r$)
⟨*proof*⟩

**lemma** *hcomplex-of-hypreal-mult-HComplex*:
!!$r$ $x$ $y$. *hcomplex-of-hypreal* $r$ ∗ *HComplex* $x$ $y$ = *HComplex* ($r$∗$x$) ($r$∗$y$)
⟨*proof*⟩

**lemma** *i-hcomplex-of-hypreal* [*simp*]:

$!!r.\ iii * hcomplex\text{-}of\text{-}hypreal\ r = HComplex\ 0\ r$
⟨*proof*⟩

**lemma** *hcomplex-of-hypreal-i* [*simp*]:
$!!r.\ hcomplex\text{-}of\text{-}hypreal\ r * iii = HComplex\ 0\ r$
⟨*proof*⟩

## 28.9 Conjugation

**lemma** *hcomplex-hcnj-cancel-iff* [*iff*]: $!!x\ y.\ (hcnj\ x = hcnj\ y) = (x = y)$
⟨*proof*⟩

**lemma** *hcomplex-hcnj-hcnj* [*simp*]: $!!z.\ hcnj\ (hcnj\ z) = z$
⟨*proof*⟩

**lemma** *hcomplex-hcnj-hcomplex-of-hypreal* [*simp*]:
$!!x.\ hcnj\ (hcomplex\text{-}of\text{-}hypreal\ x) = hcomplex\text{-}of\text{-}hypreal\ x$
⟨*proof*⟩

**lemma** *hcomplex-hmod-hcnj* [*simp*]: $!!z.\ hcmod\ (hcnj\ z) = hcmod\ z$
⟨*proof*⟩

**lemma** *hcomplex-hcnj-minus*: $!!z.\ hcnj\ (-z) = -\ hcnj\ z$
⟨*proof*⟩

**lemma** *hcomplex-hcnj-inverse*: $!!z.\ hcnj(inverse\ z) = inverse(hcnj\ z)$
⟨*proof*⟩

**lemma** *hcomplex-hcnj-add*: $!!w\ z.\ hcnj(w + z) = hcnj(w) + hcnj(z)$
⟨*proof*⟩

**lemma** *hcomplex-hcnj-diff*: $!!w\ z.\ hcnj(w - z) = hcnj(w) - hcnj(z)$
⟨*proof*⟩

**lemma** *hcomplex-hcnj-mult*: $!!w\ z.\ hcnj(w * z) = hcnj(w) * hcnj(z)$
⟨*proof*⟩

**lemma** *hcomplex-hcnj-divide*: $!!w\ z.\ hcnj(w\ /\ z) = (hcnj\ w)/(hcnj\ z)$
⟨*proof*⟩

**lemma** *hcnj-one* [*simp*]: $hcnj\ 1 = 1$
⟨*proof*⟩

**lemma** *hcomplex-hcnj-zero* [*simp*]: $hcnj\ 0 = 0$
⟨*proof*⟩

**lemma** *hcomplex-hcnj-zero-iff* [*iff*]: $!!z.\ (hcnj\ z = 0) = (z = 0)$
⟨*proof*⟩

**lemma** *hcomplex-mult-hcnj*:
  $!!z.\ z * hcnj\ z = hcomplex\text{-}of\text{-}hypreal\ (hRe(z)\ \hat{}\ 2 + hIm(z)\ \hat{}\ 2)$
⟨*proof*⟩

## 28.10 More Theorems about the Function *hcmod*

**lemma** *hcmod-hcomplex-of-hypreal-of-nat* [*simp*]:
  $hcmod\ (hcomplex\text{-}of\text{-}hypreal(hypreal\text{-}of\text{-}nat\ n)) = hypreal\text{-}of\text{-}nat\ n$
⟨*proof*⟩

**lemma** *hcmod-hcomplex-of-hypreal-of-hypnat* [*simp*]:
  $hcmod\ (hcomplex\text{-}of\text{-}hypreal(hypreal\text{-}of\text{-}hypnat\ n)) = hypreal\text{-}of\text{-}hypnat\ n$
⟨*proof*⟩

**lemma** *hcmod-mult-hcnj*: $!!z.\ hcmod(z * hcnj(z)) = hcmod(z)\ \hat{}\ 2$
⟨*proof*⟩

**lemma** *hcmod-triangle-ineq2* [*simp*]:
 $!!a\ b.\ hcmod(b + a) - hcmod\ b \le hcmod\ a$
⟨*proof*⟩

**lemma** *hcmod-diff-ineq* [*simp*]: $!!a\ b.\ hcmod(a) - hcmod(b) \le hcmod(a + b)$
⟨*proof*⟩

## 28.11 Exponentiation

**lemma** *hcomplexpow-0* [*simp*]: $z\ \hat{}\ 0 \quad = (1::hcomplex)$
⟨*proof*⟩

**lemma** *hcomplexpow-Suc* [*simp*]: $z\ \hat{}\ (Suc\ n) = (z::hcomplex) * (z\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *hcomplexpow-i-squared* [*simp*]: $iii\ \hat{}\ 2 = -1$
⟨*proof*⟩

**lemma** *hcomplex-of-hypreal-pow*:
  $!!x.\ hcomplex\text{-}of\text{-}hypreal\ (x\ \hat{}\ n) = (hcomplex\text{-}of\text{-}hypreal\ x)\ \hat{}\ n$
⟨*proof*⟩

**lemma** *hcomplex-hcnj-pow*: $!!z.\ hcnj(z\ \hat{}\ n) = hcnj(z)\ \hat{}\ n$
⟨*proof*⟩

**lemma** *hcmod-hcomplexpow*: $!!x.\ hcmod(x\ \hat{}\ n) = hcmod(x)\ \hat{}\ n$
⟨*proof*⟩

**lemma** *hcpow-minus*:
  $!!x\ n.\ (-x::hcomplex)\ pow\ n =$
   $(if\ (\ *p*\ even)\ n\ then\ (x\ pow\ n)\ else\ -(x\ pow\ n))$
⟨*proof*⟩

**lemma** *hcpow-mult*:
$!!r\ s\ n.\ ((r{::}hcomplex) * s)\ pow\ n = (r\ pow\ n) * (s\ pow\ n)$
⟨*proof*⟩

**lemma** *hcpow-zero2* [*simp*]:
$\bigwedge n.\ 0\ pow\ (hSuc\ n) = (0{::}'a{::}\{recpower,semiring\text{-}0\}\ star)$
⟨*proof*⟩

**lemma** *hcpow-not-zero* [*simp*,*intro*]:
$!!r\ n.\ r \neq 0 ==> r\ pow\ n \neq (0{::}hcomplex)$
⟨*proof*⟩

**lemma** *hcpow-zero-zero*: $r\ pow\ n = (0{::}hcomplex) ==> r = 0$
⟨*proof*⟩

## 28.12 The Function *hsgn*

**lemma** *hsgn-zero* [*simp*]: $hsgn\ 0 = 0$
⟨*proof*⟩

**lemma** *hsgn-one* [*simp*]: $hsgn\ 1 = 1$
⟨*proof*⟩

**lemma** *hsgn-minus*: $!!z.\ hsgn\ (-z) = -\ hsgn(z)$
⟨*proof*⟩

**lemma** *hsgn-eq*: $!!z.\ hsgn\ z = z\ /\ hcomplex\text{-}of\text{-}hypreal\ (hcmod\ z)$
⟨*proof*⟩

**lemma** *hcmod-i*: $!!x\ y.\ hcmod\ (HComplex\ x\ y) = (\ *f*\ sqrt)\ (x\ \hat{}\ 2 + y\ \hat{}\ 2)$
⟨*proof*⟩

**lemma** *hcomplex-eq-cancel-iff1* [*simp*]:
$(hcomplex\text{-}of\text{-}hypreal\ xa = HComplex\ x\ y) = (xa = x\ \&\ y = 0)$
⟨*proof*⟩

**lemma** *hcomplex-eq-cancel-iff2* [*simp*]:
$(HComplex\ x\ y = hcomplex\text{-}of\text{-}hypreal\ xa) = (x = xa\ \&\ y = 0)$
⟨*proof*⟩

**lemma** *HComplex-eq-0* [*simp*]: $!!x\ y.\ (HComplex\ x\ y = 0) = (x = 0\ \&\ y = 0)$
⟨*proof*⟩

**lemma** *HComplex-eq-1* [*simp*]: $!!x\ y.\ (HComplex\ x\ y = 1) = (x = 1\ \&\ y = 0)$
⟨*proof*⟩

**lemma** *i-eq-HComplex-0-1*: $iii = HComplex\ 0\ 1$
⟨*proof*⟩

**lemma** *HComplex-eq-i* [*simp*]: $!!x\ y.\ (HComplex\ x\ y = iii) = (x = 0\ \&\ y = 1)$
⟨*proof*⟩

**lemma** *hRe-hsgn* [*simp*]: $!!z.\ hRe(hsgn\ z) = hRe(z)/hcmod\ z$
⟨*proof*⟩

**lemma** *hIm-hsgn* [*simp*]: $!!z.\ hIm(hsgn\ z) = hIm(z)/hcmod\ z$
⟨*proof*⟩

**lemma** *hcomplex-inverse-complex-split*:
$!!x\ y.\ inverse(hcomplex\text{-}of\text{-}hypreal\ x + iii * hcomplex\text{-}of\text{-}hypreal\ y) =$
$hcomplex\text{-}of\text{-}hypreal(x/(x\ \hat{}\ 2 + y\ \hat{}\ 2)) -$
$iii * hcomplex\text{-}of\text{-}hypreal(y/(x\ \hat{}\ 2 + y\ \hat{}\ 2))$
⟨*proof*⟩

**lemma** *HComplex-inverse*:
$!!x\ y.\ inverse\ (HComplex\ x\ y) =$
$HComplex\ (x/(x\ \hat{}\ 2 + y\ \hat{}\ 2))\ (-y/(x\ \hat{}\ 2 + y\ \hat{}\ 2))$
⟨*proof*⟩

**lemma** *hRe-mult-i-eq*[*simp*]:
$!!y.\ hRe\ (iii * hcomplex\text{-}of\text{-}hypreal\ y) = 0$
⟨*proof*⟩

**lemma** *hIm-mult-i-eq* [*simp*]:
$!!y.\ hIm\ (iii * hcomplex\text{-}of\text{-}hypreal\ y) = y$
⟨*proof*⟩

**lemma** *hcmod-mult-i* [*simp*]: $!!y.\ hcmod\ (iii * hcomplex\text{-}of\text{-}hypreal\ y) = abs\ y$
⟨*proof*⟩

**lemma** *hcmod-mult-i2* [*simp*]: $!!y.\ hcmod\ (hcomplex\text{-}of\text{-}hypreal\ y * iii) = abs\ y$
⟨*proof*⟩

**lemma** *cos-harg-i-mult-zero-pos*:
$!!y.\ 0 < y ==> (\ *f*\ cos)\ (harg(HComplex\ 0\ y)) = 0$
⟨*proof*⟩

**lemma** *cos-harg-i-mult-zero-neg*:
$!!y.\ y < 0 ==> (\ *f*\ cos)\ (harg(HComplex\ 0\ y)) = 0$
⟨*proof*⟩

**lemma** *cos-harg-i-mult-zero* [*simp*]:
$!!y.\ y \neq 0 ==> (\ *f*\ cos)\ (harg(HComplex\ 0\ y)) = 0$
⟨*proof*⟩

**lemma** *hcomplex-of-hypreal-zero-iff* [*simp*]:
$!!y.\ (\mathit{hcomplex\text{-}of\text{-}hypreal}\ y = 0) = (y = 0)$
⟨*proof*⟩

## 28.13 Polar Form for Nonstandard Complex Numbers

**lemma** *complex-split-polar2*:
$\forall\, n.\ \exists\, r\ a.\ (z\ n) = \ \mathit{complex\text{-}of\text{-}real}\ r * (\mathit{Complex}\ (\cos\ a)\ (\sin\ a))$
⟨*proof*⟩

**lemma** *hcomplex-split-polar*:
$!!z.\ \exists\, r\ a.\ z = \mathit{hcomplex\text{-}of\text{-}hypreal}\ r * (\mathit{HComplex}((\ {*f*}\ \cos)\ a)((\ {*f*}\ \sin)\ a))$
⟨*proof*⟩

**lemma** *hcis-eq*:
$!!a.\ \mathit{hcis}\ a =$
$(\mathit{hcomplex\text{-}of\text{-}hypreal}((\ {*f*}\ \cos)\ a) +$
$\mathit{iii} * \mathit{hcomplex\text{-}of\text{-}hypreal}((\ {*f*}\ \sin)\ a))$
⟨*proof*⟩

**lemma** *hrcis-Ex*: $!!z.\ \exists\, r\ a.\ z = \mathit{hrcis}\ r\ a$
⟨*proof*⟩

**lemma** *hRe-hcomplex-polar* [*simp*]:
$!!r\ a.\ \mathit{hRe}\ (\mathit{hcomplex\text{-}of\text{-}hypreal}\ r * \mathit{HComplex}\ ((\ {*f*}\ \cos)\ a)\ ((\ {*f*}\ \sin)\ a)) =$
$r * (\ {*f*}\ \cos)\ a$
⟨*proof*⟩

**lemma** *hRe-hrcis* [*simp*]: $!!r\ a.\ \mathit{hRe}(\mathit{hrcis}\ r\ a) = r * (\ {*f*}\ \cos)\ a$
⟨*proof*⟩

**lemma** *hIm-hcomplex-polar* [*simp*]:
$!!r\ a.\ \mathit{hIm}\ (\mathit{hcomplex\text{-}of\text{-}hypreal}\ r * \mathit{HComplex}\ ((\ {*f*}\ \cos)\ a)\ ((\ {*f*}\ \sin)\ a)) =$
$r * (\ {*f*}\ \sin)\ a$
⟨*proof*⟩

**lemma** *hIm-hrcis* [*simp*]: $!!r\ a.\ \mathit{hIm}(\mathit{hrcis}\ r\ a) = r * (\ {*f*}\ \sin)\ a$
⟨*proof*⟩

**lemma** *hcmod-unit-one* [*simp*]:
$!!a.\ \mathit{hcmod}\ (\mathit{HComplex}\ ((\ {*f*}\ \cos)\ a)\ ((\ {*f*}\ \sin)\ a)) = 1$
⟨*proof*⟩

**lemma** *hcmod-complex-polar* [*simp*]:
$!!r\ a.\ \mathit{hcmod}\ (\mathit{hcomplex\text{-}of\text{-}hypreal}\ r * \mathit{HComplex}\ ((\ {*f*}\ \cos)\ a)\ ((\ {*f*}\ \sin)\ a)) =$
$\mathit{abs}\ r$
⟨*proof*⟩

**lemma** *hcmod-hrcis* [*simp*]: !!$r$ $a$. *hcmod*(*hrcis* $r$ $a$) = *abs* $r$
⟨*proof*⟩

**lemma** *hcis-hrcis-eq*: !!$a$. *hcis* $a$ = *hrcis* *1* $a$
⟨*proof*⟩
**declare** *hcis-hrcis-eq* [*symmetric*, *simp*]

**lemma** *hrcis-mult*:
!!$a$ $b$ *r1* *r2*. *hrcis* *r1* $a$ ∗ *hrcis* *r2* $b$ = *hrcis* (*r1*∗*r2*) ($a$ + $b$)
⟨*proof*⟩

**lemma** *hcis-mult*: !!$a$ $b$. *hcis* $a$ ∗ *hcis* $b$ = *hcis* ($a$ + $b$)
⟨*proof*⟩

**lemma** *hcis-zero* [*simp*]: *hcis* *0* = *1*
⟨*proof*⟩

**lemma** *hrcis-zero-mod* [*simp*]: !!$a$. *hrcis* *0* $a$ = *0*
⟨*proof*⟩

**lemma** *hrcis-zero-arg* [*simp*]: !!$r$. *hrcis* $r$ *0* = *hcomplex-of-hypreal* $r$
⟨*proof*⟩

**lemma** *hcomplex-i-mult-minus* [*simp*]: !!$x$. *iii* ∗ (*iii* ∗ $x$) = − $x$
⟨*proof*⟩

**lemma** *hcomplex-i-mult-minus2* [*simp*]: *iii* ∗ *iii* ∗ $x$ = − $x$
⟨*proof*⟩

**lemma** *hcis-hypreal-of-nat-Suc-mult*:
!!$a$. *hcis* (*hypreal-of-nat* (*Suc* $n$) ∗ $a$) =
*hcis* $a$ ∗ *hcis* (*hypreal-of-nat* $n$ ∗ $a$)
⟨*proof*⟩

**lemma** *NSDeMoivre*: !!$a$. (*hcis* $a$) ˆ $n$ = *hcis* (*hypreal-of-nat* $n$ ∗ $a$)
⟨*proof*⟩

**lemma** *hcis-hypreal-of-hypnat-Suc-mult*:
!! $a$ $n$. *hcis* (*hypreal-of-hypnat* ($n$ + *1*) ∗ $a$) =
*hcis* $a$ ∗ *hcis* (*hypreal-of-hypnat* $n$ ∗ $a$)
⟨*proof*⟩

**lemma** *NSDeMoivre-ext*:
!!$a$ $n$. (*hcis* $a$) *pow* $n$ = *hcis* (*hypreal-of-hypnat* $n$ ∗ $a$)
⟨*proof*⟩

**lemma** *NSDeMoivre2*:
$!!a\ r.\ (hrcis\ r\ a)\ \hat{}\ n = hrcis\ (r\ \hat{}\ n)\ (hypreal\text{-}of\text{-}nat\ n * a)$
⟨*proof*⟩

**lemma** *DeMoivre2-ext*:
$!!\ a\ r\ n.\ (hrcis\ r\ a)\ pow\ n = hrcis\ (r\ pow\ n)\ (hypreal\text{-}of\text{-}hypnat\ n * a)$
⟨*proof*⟩

**lemma** *hcis-inverse* [*simp*]: $!!a.\ inverse(hcis\ a) = hcis\ (-a)$
⟨*proof*⟩

**lemma** *hrcis-inverse*: $!!a\ r.\ inverse(hrcis\ r\ a) = hrcis\ (inverse\ r)\ (-a)$
⟨*proof*⟩

**lemma** *hRe-hcis* [*simp*]: $!!a.\ hRe(hcis\ a) = (\ *f*\ cos)\ a$
⟨*proof*⟩

**lemma** *hIm-hcis* [*simp*]: $!!a.\ hIm(hcis\ a) = (\ *f*\ sin)\ a$
⟨*proof*⟩

**lemma** *cos-n-hRe-hcis-pow-n*: $(\ *f*\ cos)\ (hypreal\text{-}of\text{-}nat\ n * a) = hRe(hcis\ a\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *sin-n-hIm-hcis-pow-n*: $(\ *f*\ sin)\ (hypreal\text{-}of\text{-}nat\ n * a) = hIm(hcis\ a\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *cos-n-hRe-hcis-hcpow-n*: $(\ *f*\ cos)\ (hypreal\text{-}of\text{-}hypnat\ n * a) = hRe(hcis\ a\ pow\ n)$
⟨*proof*⟩

**lemma** *sin-n-hIm-hcis-hcpow-n*: $(\ *f*\ sin)\ (hypreal\text{-}of\text{-}hypnat\ n * a) = hIm(hcis\ a\ pow\ n)$
⟨*proof*⟩

**lemma** *hexpi-add*: $!!a\ b.\ hexpi(a + b) = hexpi(a) * hexpi(b)$
⟨*proof*⟩

## 28.14 *hcomplex-of-complex*: the Injection from type *complex* to to *hcomplex*

**lemma** *inj-hcomplex-of-complex*: $inj(hcomplex\text{-}of\text{-}complex)$

⟨*proof*⟩

**lemma** *hcomplex-of-complex-i*: $iii = hcomplex\text{-}of\text{-}complex\ ii$
⟨*proof*⟩

**lemma** *hRe-hcomplex-of-complex*:

*hRe* (*hcomplex-of-complex* $z$) = *hypreal-of-real* (*Re* $z$)
⟨*proof*⟩

**lemma** *hIm-hcomplex-of-complex*:
*hIm* (*hcomplex-of-complex* $z$) = *hypreal-of-real* (*Im* $z$)
⟨*proof*⟩

**lemma** *hcmod-hcomplex-of-complex*:
*hcmod* (*hcomplex-of-complex* $x$) = *hypreal-of-real* (*cmod* $x$)
⟨*proof*⟩

## 28.15 Numerals and Arithmetic

**lemma** *hcomplex-number-of-def*: (*number-of* $w$ :: *hcomplex*) == *of-int* $w$
⟨*proof*⟩

**lemma** *hcomplex-of-hypreal-eq-hcomplex-of-complex*:
*hcomplex-of-hypreal* (*hypreal-of-real* $x$) =
*hcomplex-of-complex* (*complex-of-real* $x$)
⟨*proof*⟩

**lemma** *hcomplex-hypreal-number-of*:
*hcomplex-of-complex* (*number-of* $w$) = *hcomplex-of-hypreal*(*number-of* $w$)
⟨*proof*⟩

**lemma** *hcomplex-number-of-hcnj* [*simp*]:
*hcnj* (*number-of* $v$ :: *hcomplex*) = *number-of* $v$
⟨*proof*⟩

**lemma** *hcomplex-number-of-hcmod* [*simp*]:
*hcmod*(*number-of* $v$ :: *hcomplex*) = *abs* (*number-of* $v$ :: *hypreal*)
⟨*proof*⟩

**lemma** *hcomplex-number-of-hRe* [*simp*]:
*hRe*(*number-of* $v$ :: *hcomplex*) = *number-of* $v$
⟨*proof*⟩

**lemma** *hcomplex-number-of-hIm* [*simp*]:
*hIm*(*number-of* $v$ :: *hcomplex*) = *0*
⟨*proof*⟩

**end**

# 29 Star: Star-Transforms in Non-Standard Analysis

**theory** *Star*
**imports** *NSA*
**begin**

**definition**

*starset-n* :: (*nat* => *'a set*) => *'a star set* (**sn** - [*80*] *80*) **where**
**sn** *As* = *Iset* (*star-n As*)

**definition**
*InternalSets* :: *'a star set set* **where**
*InternalSets* = $\{X.\ \exists As.\ X = *sn*\ As\}$

**definition**

*is-starext* :: [*'a star* => *'a star*, *'a* => *'a*] => *bool* **where**
*is-starext F f* = ($\forall x\ y.\ \exists X \in$ *Rep-star*(*x*). $\exists Y \in$ *Rep-star*(*y*).
((*y* = (*F x*)) = ({*n*. *Y n* = *f*(*X n*)} : *FreeUltrafilterNat*)))

**definition**

*starfun-n* :: (*nat* => (*'a* => *'b*)) => *'a star* => *'b star* (**fn** - [*80*] *80*) **where**
**fn** *F* = *Ifun* (*star-n F*)

**definition**
*InternalFuns* :: (*'a star* => *'b star*) *set* **where**
*InternalFuns* = $\{X.\ \exists F.\ X = *fn*\ F\}$

**lemma** *no-choice*: $\forall x.\ \exists y.\ Q\ x\ y$ ==> $\exists$(*f* :: *'a* => *nat*). $\forall x.\ Q\ x\ (f\ x)$
⟨*proof*⟩

## 29.1 Properties of the Star-transform Applied to Sets of Reals

**lemma** *STAR-star-of-image-subset*: *star-of* ‘ *A* <= **s** *A*
⟨*proof*⟩

**lemma** *STAR-hypreal-of-real-Int*: **s** *X Int Reals* = *hypreal-of-real* ‘ *X*
⟨*proof*⟩

**lemma** *STAR-star-of-Int*: $*s*$ $X$ *Int Standard* $=$ *star-of* ‘ $X$
⟨*proof*⟩

**lemma** *lemma-not-hyprealA*: $x \notin$ *hypreal-of-real* ‘ $A$ $==>$ $\forall\, y \in A.\ x \neq$ *hypreal-of-real* $y$
⟨*proof*⟩

**lemma** *lemma-not-starA*: $x \notin$ *star-of* ‘ $A$ $==>$ $\forall\, y \in A.\ x \neq$ *star-of* $y$
⟨*proof*⟩

**lemma** *lemma-Compl-eq*: $-\ \{n.\ X\ n = xa\} = \{n.\ X\ n \neq xa\}$
⟨*proof*⟩

**lemma** *STAR-real-seq-to-hypreal*:
$\forall\, n.\ (X\ n) \notin M$ $==>$ *star-n* $X \notin *s*\ M$
⟨*proof*⟩

**lemma** *STAR-singleton*: $*s*\ \{x\} = \{$*star-of* $x\}$
⟨*proof*⟩

**lemma** *STAR-not-mem*: $x \notin F$ $==>$ *star-of* $x \notin *s*\ F$
⟨*proof*⟩

**lemma** *STAR-subset-closed*: [| $x : *s*\ A$; $A <= B$ |] $==>$ $x : *s*\ B$
⟨*proof*⟩

Nonstandard extension of a set (defined using a constant sequence) as a special case of an internal set

**lemma** *starset-n-starset*: $\forall\, n.\ (As\ n = A)$ $==>$ $*sn*\ As = *s*\ A$
⟨*proof*⟩

**lemma** *starfun-n-starfun*: $\forall\, n.\ (F\ n = f)$ $==>$ $*fn*\ F = *f*\ f$
⟨*proof*⟩

**lemma** *hrabs-is-starext-rabs*: *is-starext abs abs*

⟨proof⟩

Nonstandard extension of functions

**lemma** *starfun*:
$( *f* \ f) \ (star\text{-}n \ X) = star\text{-}n \ (\%n. \ f \ (X \ n))$
⟨proof⟩

**lemma** *starfun-if-eq*:
$!!w. \ w \neq star\text{-}of \ x$
$==> ( *f* \ (\lambda z. \ if \ z = x \ then \ a \ else \ g \ z)) \ w = ( *f* \ g) \ w$
⟨proof⟩

**lemma** *starfun-mult*: $!!x. \ ( *f* \ f) \ x * ( *f* \ g) \ x = ( *f* \ (\%x. \ f \ x * g \ x)) \ x$
⟨proof⟩
**declare** *starfun-mult* [*symmetric*, *simp*]

**lemma** *starfun-add*: $!!x. \ ( *f* \ f) \ x + ( *f* \ g) \ x = ( *f* \ (\%x. \ f \ x + g \ x)) \ x$
⟨proof⟩
**declare** *starfun-add* [*symmetric*, *simp*]

**lemma** *starfun-minus*: $!!x. \ - ( *f* \ f) \ x = ( *f* \ (\%x. \ - f \ x)) \ x$
⟨proof⟩
**declare** *starfun-minus* [*symmetric*, *simp*]

**lemma** *starfun-add-minus*: $!!x. \ ( *f* \ f) \ x + -( *f* \ g) \ x = ( *f* \ (\%x. \ f \ x + -g \ x)) \ x$
⟨proof⟩
**declare** *starfun-add-minus* [*symmetric*, *simp*]

**lemma** *starfun-diff*: $!!x. \ ( *f* \ f) \ x \ - ( *f* \ g) \ x = ( *f* \ (\%x. \ f \ x - g \ x)) \ x$
⟨proof⟩
**declare** *starfun-diff* [*symmetric*, *simp*]

**lemma** *starfun-o2*: $(\%x. \ ( *f* \ f) \ (( *f* \ g) \ x)) = *f* \ (\%x. \ f \ (g \ x))$
⟨proof⟩

**lemma** *starfun-o*: $( *f* \ f) \ o \ ( *f* \ g) = ( *f* \ (f \ o \ g))$
⟨proof⟩

NS extension of constant function

**lemma** *starfun-const-fun* [*simp*]: $!!x. \ ( *f* \ (\%x. \ k)) \ x = star\text{-}of \ k$
⟨proof⟩

the NS extension of the identity function

**lemma** *starfun-Id* [*simp*]: $!!x.\ (\ *f*\ (\%x.\ x))\ x = x$
⟨*proof*⟩

**lemma** *starfun-Idfun-approx*:
$x\ @{=}\ \textit{star-of}\ a \ ==>\ (\ *f*\ (\%x.\ x))\ x\ @{=}\ \textit{star-of}\ a$
⟨*proof*⟩

The Star-function is a (nonstandard) extension of the function

**lemma** *is-starext-starfun*: *is-starext* $(\ *f*\ f)\ f$
⟨*proof*⟩

Any nonstandard extension is in fact the Star-function

**lemma** *is-starfun-starext*: *is-starext* $F\ f\ ==>\ F = *f*\ f$
⟨*proof*⟩

**lemma** *is-starext-starfun-iff*: $(\textit{is-starext}\ F\ f) = (F = *f*\ f)$
⟨*proof*⟩

extented function has same solution as its standard version for real arguments. i.e they are the same for all real arguments

**lemma** *starfun-eq*: $(\ *f*\ f)\ (\textit{star-of}\ a) = \textit{star-of}\ (f\ a)$
⟨*proof*⟩

**lemma** *starfun-approx*: $(\ *f*\ f)\ (\textit{star-of}\ a)\ @{=}\ \textit{star-of}\ (f\ a)$
⟨*proof*⟩

**lemma** *starfun-lambda-cancel*:
$!!x'.\ (\ *f*\ (\%h.\ f\ (x + h)))\ x' = (\ *f*\ f)\ (\textit{star-of}\ x + x')$
⟨*proof*⟩

**lemma** *starfun-lambda-cancel2*:
$(\ *f*\ (\%h.\ f(g(x + h))))\ x' = (\ *f*\ (f\ o\ g))\ (\textit{star-of}\ x + x')$
⟨*proof*⟩

**lemma** *starfun-mult-HFinite-approx*:
**fixes** $l\ m :: {}'a{::}\textit{real-normed-algebra star}$
**shows** $[|\ (\ *f*\ f)\ x\ @{=}\ l;\ (\ *f*\ g)\ x\ @{=}\ m;$
$l{:}\ \textit{HFinite};\ m{:}\ \textit{HFinite}$
$|]\ ==>\ (\ *f*\ (\%x.\ f\ x * g\ x))\ x\ @{=}\ l * m$
⟨*proof*⟩

**lemma** *starfun-add-approx*: $[|\ (\ *f*\ f)\ x\ @{=}\ l;\ (\ *f*\ g)\ x\ @{=}\ m$
$|]\ ==>\ (\ *f*\ (\%x.\ f\ x + g\ x))\ x\ @{=}\ l + m$
⟨*proof*⟩

Examples: hrabs is nonstandard extension of rabs inverse is nonstandard extension of inverse

**lemma** *starfun-rabs-hrabs*: $*f*\ abs = abs$
⟨*proof*⟩

**lemma** *starfun-inverse-inverse* [*simp*]: $(\ *f*\ inverse)\ x = inverse(x)$
⟨*proof*⟩

**lemma** *starfun-inverse*: $!!x.\ inverse\ ((\ *f*\ f)\ x) = (\ *f*\ (\%x.\ inverse\ (f\ x)))\ x$
⟨*proof*⟩
**declare** *starfun-inverse* [*symmetric*, *simp*]

**lemma** *starfun-divide*: $!!x.\ (\ *f*\ f)\ x\ /\ (\ *f*\ g)\ x = (\ *f*\ (\%x.\ f\ x\ /\ g\ x))\ x$
⟨*proof*⟩
**declare** *starfun-divide* [*symmetric*, *simp*]

**lemma** *starfun-inverse2*: $!!x.\ inverse\ ((\ *f*\ f)\ x) = (\ *f*\ (\%x.\ inverse\ (f\ x)))\ x$
⟨*proof*⟩

General lemma/theorem needed for proofs in elementary topology of the reals

**lemma** *starfun-mem-starset*:
$!!x.\ (\ *f*\ f)\ x : *s*\ A ==> x : *s*\ \{x.\ f\ x\ \in A\}$
⟨*proof*⟩

Alternative definition for hrabs with rabs function applied entrywise to equivalence class representative. This is easily proved using starfun and ns extension thm

**lemma** *hypreal-hrabs*:
$abs\ (star\text{-}n\ X) = star\text{-}n\ (\%n.\ abs\ (X\ n))$
⟨*proof*⟩

nonstandard extension of set through nonstandard extension of rabs function i.e hrabs. A more general result should be where we replace rabs by some arbitrary function f and hrabs by its NS extenson. See second NS set extension below.

**lemma** *STAR-rabs-add-minus*:
$*s*\ \{x.\ abs\ (x + -\ y) < r\} =$
$\{x.\ abs(x + -star\text{-}of\ y) < star\text{-}of\ r\}$
⟨*proof*⟩

**lemma** *STAR-starfun-rabs-add-minus*:
$*s*\ \{x.\ abs\ (f\ x + -\ y) < r\} =$
$\{x.\ abs((\ *f*\ f)\ x + -star\text{-}of\ y) < star\text{-}of\ r\}$
⟨*proof*⟩

Another characterization of Infinitesimal and one of @= relation. In this theory since *hypreal-hrabs* proved here. Maybe move both theorems??

**lemma** *Infinitesimal-FreeUltrafilterNat-iff2*:

$(star\text{-}n\ X \in Infinitesimal) =$
$(\forall\, m.\ \{n.\ norm(X\ n) < inverse(real(Suc\ m))\}$
$\in\ FreeUltrafilterNat)$
⟨*proof*⟩

**lemma** *HNatInfinite-inverse-Infinitesimal* [*simp*]:
$n \in HNatInfinite ==> inverse\ (hypreal\text{-}of\text{-}hypnat\ n) \in Infinitesimal$
⟨*proof*⟩

**lemma** *approx-FreeUltrafilterNat-iff*: $star\text{-}n\ X\ @{=}\ star\text{-}n\ Y =$
$(\forall\, r{>}0.\ \{n.\ norm\ (X\ n\ -\ Y\ n) < r\} : FreeUltrafilterNat)$
⟨*proof*⟩

**lemma** *approx-FreeUltrafilterNat-iff2*: $star\text{-}n\ X\ @{=}\ star\text{-}n\ Y =$
$(\forall\, m.\ \{n.\ norm\ (X\ n\ -\ Y\ n) <$
$inverse(real(Suc\ m))\} : FreeUltrafilterNat)$
⟨*proof*⟩

**lemma** *inj-starfun*: *inj starfun*
⟨*proof*⟩

**end**

# 30 NatStar: Star-transforms for the Hypernaturals

**theory** *NatStar*
**imports** *Star*
**begin**

**lemma** *star-n-eq-starfun-whn*: $star\text{-}n\ X = (\ {*f*}\ X)\ whn$
⟨*proof*⟩

**lemma** *starset-n-Un*: $*sn*\ (\%n.\ (A\ n)\ Un\ (B\ n)) = *sn*\ A\ Un\ *sn*\ B$
⟨*proof*⟩

**lemma** *InternalSets-Un*:
$[|\ X \in InternalSets;\ Y \in InternalSets\ |]$
$==> (X\ Un\ Y) \in InternalSets$
⟨*proof*⟩

**lemma** *starset-n-Int*:
$*sn*\ (\%n.\ (A\ n)\ Int\ (B\ n)) = *sn*\ A\ Int\ *sn*\ B$
⟨*proof*⟩

**lemma** *InternalSets-Int*:
$[|\ X \in InternalSets;\ Y \in InternalSets\ |]$

$==> (X\ Int\ Y) \in InternalSets$
⟨*proof*⟩

**lemma** *starset-n-Compl*: $*sn* ((\%n. - A\ n)) = -( *sn* A)$
⟨*proof*⟩

**lemma** *InternalSets-Compl*: $X \in InternalSets ==> -X \in InternalSets$
⟨*proof*⟩

**lemma** *starset-n-diff*: $*sn* (\%n. (A\ n) - (B\ n)) = *sn* A - *sn* B$
⟨*proof*⟩

**lemma** *InternalSets-diff*:
$[| X \in InternalSets; Y \in InternalSets |]$
$==> (X - Y) \in InternalSets$
⟨*proof*⟩

**lemma** *NatStar-SHNat-subset*: $Nats \le *s* (UNIV :: nat\ set)$
⟨*proof*⟩

**lemma** *NatStar-hypreal-of-real-Int*:
$*s* X\ Int\ Nats = hypnat\text{-}of\text{-}nat\ `\ X$
⟨*proof*⟩

**lemma** *starset-starset-n-eq*: $*s* X = *sn* (\%n. X)$
⟨*proof*⟩

**lemma** *InternalSets-starset-n* [*simp*]: $( *s* X) \in InternalSets$
⟨*proof*⟩

**lemma** *InternalSets-UNIV-diff*:
$X \in InternalSets ==> UNIV - X \in InternalSets$
⟨*proof*⟩

## 30.1 Nonstandard Extensions of Functions

Example of transfer of a property from reals to hyperreals — used for limit comparison of sequences

**lemma** *starfun-le-mono*:
$\forall n. N \le n --> f\ n \le g\ n$
$==> \forall n. hypnat\text{-}of\text{-}nat\ N \le n --> ( *f* f)\ n \le ( *f* g)\ n$
⟨*proof*⟩

**lemma** *starfun-less-mono*:
$\forall n. N \le n --> f\ n < g\ n$
$==> \forall n. hypnat\text{-}of\text{-}nat\ N \le n --> ( *f* f)\ n < ( *f* g)\ n$
⟨*proof*⟩

Nonstandard extension when we increment the argument by one

**lemma** *starfun-shift-one*:
$!!N.\ (\ *f*\ (\%n.\ f\ (Suc\ n)))\ N = (\ *f*\ f)\ (N + (1::hypnat))$
⟨*proof*⟩

Nonstandard extension with absolute value

**lemma** *starfun-abs*: $!!N.\ (\ *f*\ (\%n.\ abs\ (f\ n)))\ N = abs((\ *f*\ f)\ N)$
⟨*proof*⟩

The hyperpow function as a nonstandard extension of realpow

**lemma** *starfun-pow*: $!!N.\ (\ *f*\ (\%n.\ r\ \hat{}\ n))\ N = (hypreal\text{-}of\text{-}real\ r)\ pow\ N$
⟨*proof*⟩

**lemma** *starfun-pow2*:
$!!N.\ (\ *f*\ (\%n.\ (X\ n)\ \hat{}\ m))\ N = (\ *f*\ X)\ N\ pow\ hypnat\text{-}of\text{-}nat\ m$
⟨*proof*⟩

**lemma** *starfun-pow3*: $!!R.\ (\ *f*\ (\%r.\ r\ \hat{}\ n))\ R = (R)\ pow\ hypnat\text{-}of\text{-}nat\ n$
⟨*proof*⟩

The *hypreal-of-hypnat* function as a nonstandard extension of *real-of-nat*

**lemma** *starfunNat-real-of-nat*: $(\ *f*\ real) = hypreal\text{-}of\text{-}hypnat$
⟨*proof*⟩

**lemma** *starfun-inverse-real-of-nat-eq*:
$N \in HNatInfinite$
$==> (\ *f*\ (\%x::nat.\ inverse(real\ x)))\ N = inverse(hypreal\text{-}of\text{-}hypnat\ N)$
⟨*proof*⟩

Internal functions - some redundancy with *f* now

**lemma** *starfun-n*: $(\ *fn*\ f)\ (star\text{-}n\ X) = star\text{-}n\ (\%n.\ f\ n\ (X\ n))$
⟨*proof*⟩

Multiplication: $(\ *fn)\ x\ (\ *gn) = *(fn\ x\ gn)$

**lemma** *starfun-n-mult*:
$(\ *fn*\ f)\ z * (\ *fn*\ g)\ z = (\ *fn*\ (\%\ i\ x.\ f\ i\ x * g\ i\ x))\ z$
⟨*proof*⟩

Addition: $(\ *fn) + (\ *gn) = *(fn + gn)$

**lemma** *starfun-n-add*:
$(\ *fn*\ f)\ z + (\ *fn*\ g)\ z = (\ *fn*\ (\%i\ x.\ f\ i\ x + g\ i\ x))\ z$
⟨*proof*⟩

Subtraction: $(\ *fn) - (\ *gn) = *(fn + -\ gn)$

**lemma** *starfun-n-add-minus*:
$(\ *fn*\ f)\ z + -(\ *fn*\ g)\ z = (\ *fn*\ (\%i\ x.\ f\ i\ x + -g\ i\ x))\ z$
⟨*proof*⟩

Composition: $( *fn)\ o\ ( *gn) = *(fn\ o\ gn)$

**lemma** *starfun-n-const-fun* [*simp*]:
$( *fn* (\%i\ x.\ k))\ z = star\text{-}of\ k$
⟨*proof*⟩

**lemma** *starfun-n-minus*: $-\ ( *fn*\ f)\ x = ( *fn* (\%i\ x.\ -\ (f\ i)\ x))\ x$
⟨*proof*⟩

**lemma** *starfun-n-eq* [*simp*]:
$( *fn*\ f)\ (star\text{-}of\ n) = star\text{-}n\ (\%i.\ f\ i\ n)$
⟨*proof*⟩

**lemma** *starfun-eq-iff*: $(( *f*\ f) = ( *f*\ g)) = (f = g)$
⟨*proof*⟩

**lemma** *starfunNat-inverse-real-of-nat-Infinitesimal* [*simp*]:
$N \in HNatInfinite ==> ( *f* (\%x.\ inverse\ (real\ x)))\ N \in Infinitesimal$
⟨*proof*⟩

## 30.2 Nonstandard Characterization of Induction

**lemma** *hypnat-induct-obj*:
$!!n.\ (( *p*\ P)\ (0::hypnat)\ \&$
$(\forall n.\ ( *p*\ P)(n) --> ( *p*\ P)(n + 1)))$
$--> ( *p*\ P)(n)$
⟨*proof*⟩

**lemma** *hypnat-induct*:
$!!n.\ [|\ ( *p*\ P)\ (0::hypnat);$
$!!n.\ ( *p*\ P)(n) ==> ( *p*\ P)(n + 1)|]$
$==> ( *p*\ P)(n)$
⟨*proof*⟩

**lemma** *starP2-eq-iff*: $( *p2* (op\ =)) = (op\ =)$
⟨*proof*⟩

**lemma** *starP2-eq-iff2*: $( *p2* (\%x\ y.\ x = y))\ X\ Y = (X = Y)$
⟨*proof*⟩

**lemma** *nonempty-nat-set-Least-mem*:
$c \in (S :: nat\ set) ==> (LEAST\ n.\ n \in S) \in S$
⟨*proof*⟩

**lemma** *nonempty-set-star-has-least*:
$!!S::nat\ set\ star.\ Iset\ S \neq \{\} ==> \exists n \in Iset\ S.\ \forall m \in Iset\ S.\ n \leq m$
⟨*proof*⟩

**lemma** *nonempty-InternalNatSet-has-least*:
$[|\ (S::hypnat\ set) \in InternalSets;\ S \neq \{\}\ |] ==> \exists n \in S.\ \forall m \in S.\ n \leq m$

⟨*proof*⟩

Goldblatt page 129 Thm 11.3.2

**lemma** *internal-induct-lemma*:
!!*X*::*nat set star*. [| (*0*::*hypnat*) ∈ *Iset X*; ∀ *n*. *n* ∈ *Iset X* −−> *n* + *1* ∈ *Iset X* |]
==> *Iset X* = (*UNIV*:: *hypnat set*)
⟨*proof*⟩

**lemma** *internal-induct*:
[| *X* ∈ *InternalSets*; (*0*::*hypnat*) ∈ *X*; ∀ *n*. *n* ∈ *X* −−> *n* + *1* ∈ *X* |]
==> *X* = (*UNIV*:: *hypnat set*)
⟨*proof*⟩

**end**

# 31 HSEQ: Sequences and Convergence (Nonstandard)

**theory** *HSEQ*
**imports** *SEQ NatStar*
**begin**

**definition**
*NSLIMSEQ* :: [*nat* => ′*a*::*real-normed-vector*, ′*a*] => *bool*
(((-)/ −−−−*NS*> (-)) [*60*, *60*] *60*) **where**
— Nonstandard definition of convergence of sequence
*X* −−−−*NS*> *L* = (∀ *N* ∈ *HNatInfinite*. ( *∗f∗ X*) *N* ≈ *star-of L*)

**definition**
*nslim* :: (*nat* => ′*a*::*real-normed-vector*) => ′*a* **where**
— Nonstandard definition of limit using choice operator
*nslim X* = (*THE L*. *X* −−−−*NS*> *L*)

**definition**
*NSconvergent* :: (*nat* => ′*a*::*real-normed-vector*) => *bool* **where**
— Nonstandard definition of convergence
*NSconvergent X* = (∃ *L*. *X* −−−−*NS*> *L*)

**definition**
*NSBseq* :: (*nat* => ′*a*::*real-normed-vector*) => *bool* **where**
— Nonstandard definition for bounded sequence
*NSBseq X* = (∀ *N* ∈ *HNatInfinite*. ( *∗f∗ X*) *N* : *HFinite*)

**definition**
*NSCauchy* :: (*nat* => ′*a*::*real-normed-vector*) => *bool* **where**

— Nonstandard definition
$NSCauchy\ X = (\forall M \in HNatInfinite.\ \forall N \in HNatInfinite.\ (\ {*}f{*}\ X)\ M \approx (\ {*}f{*}\ X)\ N)$

## 31.1 Limits of Sequences

**lemma** *NSLIMSEQ-iff*:
$(X\ {----}NS{>}\ L) = (\forall N \in HNatInfinite.\ (\ {*}f{*}\ X)\ N \approx star\text{-}of\ L)$
⟨*proof*⟩

**lemma** *NSLIMSEQ-I*:
$(\bigwedge N.\ N \in HNatInfinite \Longrightarrow starfun\ X\ N \approx star\text{-}of\ L) \Longrightarrow X\ {----}NS{>}\ L$
⟨*proof*⟩

**lemma** *NSLIMSEQ-D*:
$[\![X\ {----}NS{>}\ L;\ N \in HNatInfinite]\!] \Longrightarrow starfun\ X\ N \approx star\text{-}of\ L$
⟨*proof*⟩

**lemma** *NSLIMSEQ-const*: $(\%n.\ k)\ {----}NS{>}\ k$
⟨*proof*⟩

**lemma** *NSLIMSEQ-add*:
$[|\ X\ {----}NS{>}\ a;\ Y\ {----}NS{>}\ b\ |] ==> (\%n.\ X\ n + Y\ n)\ {----}NS{>}\ a + b$
⟨*proof*⟩

**lemma** *NSLIMSEQ-add-const*: $f\ {----}NS{>}\ a ==> (\%n.(f\ n + b))\ {----}NS{>}\ a + b$
⟨*proof*⟩

**lemma** *NSLIMSEQ-mult*:
**fixes** $a\ b :: 'a::real\text{-}normed\text{-}algebra$
**shows** $[|\ X\ {----}NS{>}\ a;\ Y\ {----}NS{>}\ b\ |] ==> (\%n.\ X\ n * Y\ n)\ {----}NS{>}\ a * b$
⟨*proof*⟩

**lemma** *NSLIMSEQ-minus*: $X\ {----}NS{>}\ a ==> (\%n.\ -(X\ n))\ {----}NS{>}\ -a$
⟨*proof*⟩

**lemma** *NSLIMSEQ-minus-cancel*: $(\%n.\ -(X\ n))\ {----}NS{>}\ -a ==> X\ {----}NS{>}\ a$
⟨*proof*⟩

**lemma** *NSLIMSEQ-add-minus*:
$[|\ X\ {----}NS{>}\ a;\ Y\ {----}NS{>}\ b\ |] ==> (\%n.\ X\ n + -Y\ n)\ {----}NS{>}\ a + -b$
⟨*proof*⟩

**lemma** *NSLIMSEQ-diff*:
[| $X$ ----NS> $a$; $Y$ ----NS> $b$ |] ==> (%$n$. $X$ $n$ − $Y$ $n$) ----NS> $a$ − $b$
⟨*proof*⟩

**lemma** *NSLIMSEQ-diff-const*: $f$ ----NS> $a$ ==> (%$n$.($f$ $n$ − $b$)) ----NS> $a$ − $b$
⟨*proof*⟩

**lemma** *NSLIMSEQ-inverse*:
**fixes** $a$ :: $'a$::*real-normed-div-algebra*
**shows** [| $X$ ----NS> $a$; $a$ ~= *0* |] ==> (%$n$. *inverse*($X$ $n$)) ----NS> *inverse*($a$)
⟨*proof*⟩

**lemma** *NSLIMSEQ-mult-inverse*:
**fixes** $a$ $b$ :: $'a$::*real-normed-field*
**shows**
[| $X$ ----NS> $a$; $Y$ ----NS> $b$; $b$ ~= *0* |] ==> (%$n$. $X$ $n$ / $Y$ $n$) ----NS> $a/b$
⟨*proof*⟩

**lemma** *starfun-hnorm*: $\bigwedge x$. *hnorm* (( *f* $f$) $x$) = ( *f* ($\lambda x$. *norm* ($f$ $x$))) $x$
⟨*proof*⟩

**lemma** *NSLIMSEQ-norm*: $X$ ----NS> $a$ $\Longrightarrow$ ($\lambda n$. *norm* ($X$ $n$)) ----NS> *norm* $a$
⟨*proof*⟩

Uniqueness of limit

**lemma** *NSLIMSEQ-unique*: [| $X$ ----NS> $a$; $X$ ----NS> $b$ |] ==> $a$ = $b$
⟨*proof*⟩

**lemma** *NSLIMSEQ-pow* [*rule-format*]:
**fixes** $a$ :: $'a$::{*real-normed-algebra*,*recpower*}
**shows** ($X$ ----NS> $a$) --> ((%$n$. ($X$ $n$) ^ $m$) ----NS> $a$ ^ $m$)
⟨*proof*⟩

We can now try and derive a few properties of sequences, starting with the limit comparison property for sequences.

**lemma** *NSLIMSEQ-le*:
[| $f$ ----NS> $l$; $g$ ----NS> $m$;
$\exists N. \forall n \geq N. f(n) \leq g(n)$
|] ==> $l \leq$ ($m$::*real*)
⟨*proof*⟩

**lemma** *NSLIMSEQ-le-const*: [| $X$ ----NS> ($r$::*real*); $\forall n$. $a \leq X$ $n$ |] ==> $a \leq r$
⟨*proof*⟩

**lemma** *NSLIMSEQ-le-const2*: [| $X$ ----NS> ($r$::*real*); $\forall\, n.\ X\ n \le a$ |] ==> $r \le a$
⟨*proof*⟩

Shift a convergent series by 1: By the equivalence between Cauchiness and convergence and because the successor of an infinite hypernatural is also infinite.

**lemma** *NSLIMSEQ-Suc*: $f$ ----NS> $l$ ==> (%$n$. $f$(*Suc* $n$)) ----NS> $l$
⟨*proof*⟩

**lemma** *NSLIMSEQ-imp-Suc*: (%$n$. $f$(*Suc* $n$)) ----NS> $l$ ==> $f$ ----NS> $l$
⟨*proof*⟩

**lemma** *NSLIMSEQ-Suc-iff*: ((%$n$. $f$(*Suc* $n$)) ----NS> $l$) = ($f$ ----NS> $l$)
⟨*proof*⟩

### 31.1.1 Equivalence of *LIMSEQ* and *NSLIMSEQ*

**lemma** *LIMSEQ-NSLIMSEQ*:
**assumes** $X$: $X$ ----> $L$ **shows** $X$ ----NS> $L$
⟨*proof*⟩

**lemma** *NSLIMSEQ-LIMSEQ*:
**assumes** $X$: $X$ ----NS> $L$ **shows** $X$ ----> $L$
⟨*proof*⟩

**theorem** *LIMSEQ-NSLIMSEQ-iff*: ($f$ ----> $L$) = ($f$ ----NS> $L$)
⟨*proof*⟩

**lemma** *NSLIMSEQ-finite-set*:
!!($f$::*nat*=>*nat*). $\forall\, n.\ n \le f\ n$ ==> *finite* $\{n.\ f\ n \le u\}$
⟨*proof*⟩

### 31.1.2 Derived theorems about *NSLIMSEQ*

We prove the NS version from the standard one, since the NS proof seems more complicated than the standard one above!

**lemma** *NSLIMSEQ-norm-zero*: (($\lambda n$. *norm* ($X\ n$)) ----NS> *0*) = ($X$ ----NS> *0*)
⟨*proof*⟩

**lemma** *NSLIMSEQ-rabs-zero*: ((%$n$. $|f\ n|$) ----NS> *0*) = ($f$ ----NS> (*0*::*real*))
⟨*proof*⟩

Generalization to other limits

**lemma** *NSLIMSEQ-imp-rabs*: $f$ ----NS> ($l$::*real*) ==> (%$n$. $|f\,n|$) ----NS> $|l|$
⟨*proof*⟩

**lemma** *NSLIMSEQ-inverse-zero*:
$\forall\, y$::*real*. $\exists\, N$. $\forall\, n \geq N$. $y < f(n)$
==> (%$n$. *inverse*($f\,n$)) ----NS> *0*
⟨*proof*⟩

**lemma** *NSLIMSEQ-inverse-real-of-nat*: (%$n$. *inverse*(*real*(*Suc n*))) ----NS> *0*
⟨*proof*⟩

**lemma** *NSLIMSEQ-inverse-real-of-nat-add*:
(%$n$. $r$ + *inverse*(*real*(*Suc n*))) ----NS> $r$
⟨*proof*⟩

**lemma** *NSLIMSEQ-inverse-real-of-nat-add-minus*:
(%$n$. $r$ + −*inverse*(*real*(*Suc n*))) ----NS> $r$
⟨*proof*⟩

**lemma** *NSLIMSEQ-inverse-real-of-nat-add-minus-mult*:
(%$n$. $r$*( *1* + −*inverse*(*real*(*Suc n*)))) ----NS> $r$
⟨*proof*⟩

## 31.2 Convergence

**lemma** *nslimI*: $X$ ----NS> $L$ ==> *nslim* $X$ = $L$
⟨*proof*⟩

**lemma** *lim-nslim-iff*: *lim* $X$ = *nslim* $X$
⟨*proof*⟩

**lemma** *NSconvergentD*: *NSconvergent* $X$ ==> $\exists\, L$. ($X$ ----NS> $L$)
⟨*proof*⟩

**lemma** *NSconvergentI*: ($X$ ----NS> $L$) ==> *NSconvergent* $X$
⟨*proof*⟩

**lemma** *convergent-NSconvergent-iff*: *convergent* $X$ = *NSconvergent* $X$
⟨*proof*⟩

**lemma** *NSconvergent-NSLIMSEQ-iff*: *NSconvergent* $X$ = ($X$ ----NS> *nslim* $X$)
⟨*proof*⟩

## 31.3 Bounded Monotonic Sequences

**lemma** *NSBseqD*: [| *NSBseq* $X$; $N$: *HNatInfinite* |] ==> ( *f* $X$) $N$ : *HFinite*
⟨*proof*⟩

**lemma** *Standard-subset-HFinite*: *Standard* $\subseteq$ *HFinite*
⟨*proof*⟩

**lemma** *NSBseqD2*: *NSBseq X* $\Longrightarrow$ ( $*f*$ *X*) *N* $\in$ *HFinite*
⟨*proof*⟩

**lemma** *NSBseqI*: $\forall$ *N* $\in$ *HNatInfinite*. ( $*f*$ *X*) *N* : *HFinite* ==> *NSBseq X*
⟨*proof*⟩

The standard definition implies the nonstandard definition

**lemma** *Bseq-NSBseq*: *Bseq X* ==> *NSBseq X*
⟨*proof*⟩

The nonstandard definition implies the standard definition

**lemma** *SReal-less-omega*: $r \in \mathbb{R} \Longrightarrow r < \omega$
⟨*proof*⟩

**lemma** *NSBseq-Bseq*: *NSBseq X* $\Longrightarrow$ *Bseq X*
⟨*proof*⟩

Equivalence of nonstandard and standard definitions for a bounded sequence

**lemma** *Bseq-NSBseq-iff*: (*Bseq X*) = (*NSBseq X*)
⟨*proof*⟩

A convergent sequence is bounded: Boundedness as a necessary condition for convergence. The nonstandard version has no existential, as usual

**lemma** *NSconvergent-NSBseq*: *NSconvergent X* ==> *NSBseq X*
⟨*proof*⟩

Standard Version: easily now proved using equivalence of NS and standard definitions

**lemma** *convergent-Bseq*: *convergent X* ==> *Bseq X*
⟨*proof*⟩

### 31.3.1 Upper Bounds and Lubs of Bounded Sequences

**lemma** *NSBseq-isUb*: *NSBseq X* ==> $\exists$ *U*::*real*. *isUb UNIV* $\{x. \exists n. X\ n = x\}$ *U*
⟨*proof*⟩

**lemma** *NSBseq-isLub*: *NSBseq X* ==> $\exists$ *U*::*real*. *isLub UNIV* $\{x. \exists n. X\ n = x\}$ *U*
⟨*proof*⟩

### 31.3.2 A Bounded and Monotonic Sequence Converges

The best of both worlds: Easier to prove this result as a standard theorem and then use equivalence to "transfer" it into the equivalent nonstandard form if needed!

**lemma** *Bmonoseq-NSLIMSEQ*: $\forall\, n \geq m.\ X\ n = X\ m$ ==> $\exists\, L.\ (X$ ----NS> $L)$
⟨*proof*⟩

**lemma** *NSBseq-mono-NSconvergent*:
[| *NSBseq* $X$; $\forall\, m.\ \forall\, n \geq m.\ X\ m \leq X\ n$ |] ==> *NSconvergent* ($X$::*nat*=>*real*)
⟨*proof*⟩

## 31.4 Cauchy Sequences

**lemma** *NSCauchyI*:
$(\bigwedge M\ N.\ [\![M \in \mathit{HNatInfinite};\ N \in \mathit{HNatInfinite}]\!] \Longrightarrow \mathit{starfun}\ X\ M \approx \mathit{starfun}\ X\ N)$
$\Longrightarrow \mathit{NSCauchy}\ X$
⟨*proof*⟩

**lemma** *NSCauchyD*:
$[\![\mathit{NSCauchy}\ X;\ M \in \mathit{HNatInfinite};\ N \in \mathit{HNatInfinite}]\!]$
$\Longrightarrow \mathit{starfun}\ X\ M \approx \mathit{starfun}\ X\ N$
⟨*proof*⟩

### 31.4.1 Equivalence Between NS and Standard

**lemma** *Cauchy-NSCauchy*:
**assumes** $X$: *Cauchy* $X$ **shows** *NSCauchy* $X$
⟨*proof*⟩

**lemma** *NSCauchy-Cauchy*:
**assumes** $X$: *NSCauchy* $X$ **shows** *Cauchy* $X$
⟨*proof*⟩

**theorem** *NSCauchy-Cauchy-iff*: *NSCauchy* $X$ = *Cauchy* $X$
⟨*proof*⟩

### 31.4.2 Cauchy Sequences are Bounded

A Cauchy sequence is bounded – nonstandard version

**lemma** *NSCauchy-NSBseq*: *NSCauchy* $X$ ==> *NSBseq* $X$
⟨*proof*⟩

### 31.4.3 Cauchy Sequences are Convergent

Equivalence of Cauchy criterion and convergence: We will prove this using our NS formulation which provides a much easier proof than using the standard definition. We do not need to use properties of subsequences such as boundedness, monotonicity etc... Compare with Harrison's corresponding proof in HOL which is much longer and more complicated. Of course, we do not have problems which he encountered with guessing the right instantiations for his 'espsilon-delta' proof(s) in this case since the NS formulations do not involve existential quantifiers.

**lemma** *NSconvergent-NSCauchy*: *NSconvergent X* $\Longrightarrow$ *NSCauchy X*
⟨*proof*⟩

**lemma** *real-NSCauchy-NSconvergent*:
  **fixes** *X* :: *nat* ⇒ *real*
  **shows** *NSCauchy X* $\Longrightarrow$ *NSconvergent X*
⟨*proof*⟩

**lemma** *NSCauchy-NSconvergent*:
  **fixes** *X* :: *nat* ⇒ *'a::banach*
  **shows** *NSCauchy X* $\Longrightarrow$ *NSconvergent X*
⟨*proof*⟩

**lemma** *NSCauchy-NSconvergent-iff*:
  **fixes** *X* :: *nat* ⇒ *'a::banach*
  **shows** *NSCauchy X* = *NSconvergent X*
⟨*proof*⟩

## 31.5 Power Sequences

The sequence $x \ \hat{} \ n$ tends to 0 if $(0::'a) \leq x$ and $x < (1::'a)$. Proof will use (NS) Cauchy equivalence for convergence and also fact that bounded and monotonic sequence converges.

We now use NS criterion to bring proof of theorem through

**lemma** *NSLIMSEQ-realpow-zero*:
  [| *0* ≤ (*x::real*); *x* < *1* |] ==> (%*n*. *x* ^ *n*) −−−−*NS*> *0*
⟨*proof*⟩

**lemma** *NSLIMSEQ-rabs-realpow-zero*: |*c*| < (*1::real*) ==> (%*n*. |*c*| ^ *n*) −−−−*NS*> *0*
⟨*proof*⟩

**lemma** *NSLIMSEQ-rabs-realpow-zero2*: |*c*| < (*1::real*) ==> (%*n*. *c* ^ *n*) −−−−*NS*> *0*
⟨*proof*⟩

**end**

# 32 HSeries: Finite Summation and Infinite Series for Hyperreals

**theory** *HSeries*
**imports** *Series HSEQ*
**begin**

**definition**
  *sumhr* :: (*hypnat* ∗ *hypnat* ∗ (*nat=>real*)) => *hypreal* **where**
  *sumhr* =
    (%(*M*,*N*,*f*). *starfun2* (%*m n*. *setsum f* {*m*..<*n*}) *M N*)

**definition**
  *NSsums* :: [*nat=>real*,*real*] => *bool*   (**infixr** *NSsums 80*) **where**
  *f NSsums s* = (%*n*. *setsum f* {*0*..<*n*}) −−−−*NS*> *s*

**definition**
  *NSsummable* :: (*nat=>real*) => *bool* **where**
  *NSsummable f* = (∃ *s*. *f NSsums s*)

**definition**
  *NSsuminf* :: (*nat=>real*) => *real* **where**
  *NSsuminf f* = (*THE s*. *f NSsums s*)

**lemma** *sumhr-app*: *sumhr*(*M*,*N*,*f*) = ( ∗*f2*∗ (λ*m n*. *setsum f* {*m*..<*n*})) *M N*
⟨*proof*⟩

Base case in definition of *sumr*

**lemma** *sumhr-zero* [*simp*]: !!*m*. *sumhr* (*m*,*0*,*f*) = *0*
⟨*proof*⟩

Recursive case in definition of *sumr*

**lemma** *sumhr-if*:
  !!*m n*. *sumhr*(*m*,*n*+*1*,*f*) =
    (*if n* + *1* ≤ *m then 0 else sumhr*(*m*,*n*,*f*) + ( ∗*f*∗ *f*) *n*)
⟨*proof*⟩

**lemma** *sumhr-Suc-zero* [*simp*]: !!*n*. *sumhr* (*n* + *1*, *n*, *f*) = *0*
⟨*proof*⟩

**lemma** *sumhr-eq-bounds* [*simp*]: !!*n*. *sumhr* (*n*,*n*,*f*) = *0*
⟨*proof*⟩

**lemma** *sumhr-Suc* [*simp*]: !!*m*. *sumhr* (*m*,*m* + *1*,*f*) = ( ∗*f*∗ *f*) *m*
⟨*proof*⟩

**lemma** *sumhr-add-lbound-zero* [*simp*]: !!*k m*. *sumhr*(*m*+*k*,*k*,*f*) = *0*
⟨*proof*⟩

**lemma** *sumhr-add*:
$!!m\ n.\ sumhr\ (m,n,f) + sumhr(m,n,g) = sumhr(m,n,\%i.\ f\ i + g\ i)$
⟨*proof*⟩

**lemma** *sumhr-mult*:
$!!m\ n.\ hypreal\text{-}of\text{-}real\ r * sumhr(m,n,f) = sumhr(m,n,\%n.\ r * f\ n)$
⟨*proof*⟩

**lemma** *sumhr-split-add*:
$!!n\ p.\ n < p ==> sumhr(0,n,f) + sumhr(n,p,f) = sumhr(0,p,f)$
⟨*proof*⟩

**lemma** *sumhr-split-diff*: $n<p ==> sumhr(0,p,f) - sumhr(0,n,f) = sumhr(n,p,f)$
⟨*proof*⟩

**lemma** *sumhr-hrabs*: $!!m\ n.\ abs(sumhr(m,n,f)) \le sumhr(m,n,\%i.\ abs(f\ i))$
⟨*proof*⟩

other general version also needed

**lemma** *sumhr-fun-hypnat-eq*:
$(\forall r.\ m \le r\ \&\ r < n --> f\ r = g\ r) -->$
$sumhr(hypnat\text{-}of\text{-}nat\ m,\ hypnat\text{-}of\text{-}nat\ n,\ f) =$
$sumhr(hypnat\text{-}of\text{-}nat\ m,\ hypnat\text{-}of\text{-}nat\ n,\ g)$
⟨*proof*⟩

**lemma** *sumhr-const*:
$!!n.\ sumhr(0,\ n,\ \%i.\ r) = hypreal\text{-}of\text{-}hypnat\ n * hypreal\text{-}of\text{-}real\ r$
⟨*proof*⟩

**lemma** *sumhr-less-bounds-zero* [*simp*]: $!!m\ n.\ n < m ==> sumhr(m,n,f) = 0$
⟨*proof*⟩

**lemma** *sumhr-minus*: $!!m\ n.\ sumhr(m,\ n,\ \%i.\ -\ f\ i) = -\ sumhr(m,\ n,\ f)$
⟨*proof*⟩

**lemma** *sumhr-shift-bounds*:
$!!m\ n.\ sumhr(m+hypnat\text{-}of\text{-}nat\ k,n+hypnat\text{-}of\text{-}nat\ k,f) =$
$sumhr(m,n,\%i.\ f(i + k))$
⟨*proof*⟩

## 32.1 Nonstandard Sums

Infinite sums are obtained by summing to some infinite hypernatural (such as *whn*)

**lemma** *sumhr-hypreal-of-hypnat-omega*:
$sumhr(0,whn,\%i.\ 1) = hypreal\text{-}of\text{-}hypnat\ whn$
⟨*proof*⟩

**lemma** *sumhr-hypreal-omega-minus-one*: $sumhr(0,\ whn,\ \%i.\ 1) = omega - 1$
⟨*proof*⟩

**lemma** *sumhr-minus-one-realpow-zero* [*simp*]:
$!!N.\ sumhr(0,\ N + N,\ \%i.\ (-1)\ \hat{}\ (i+1)) = 0$
⟨*proof*⟩

**lemma** *sumhr-interval-const*:
$(\forall n.\ m \le Suc\ n \dashrightarrow f\ n = r)\ \&\ m \le na$
$\Longrightarrow sumhr(hypnat\text{-}of\text{-}nat\ m, hypnat\text{-}of\text{-}nat\ na, f) =$
$(hypreal\text{-}of\text{-}nat\ (na - m) * hypreal\text{-}of\text{-}real\ r)$
⟨*proof*⟩

**lemma** *starfunNat-sumr*: $!!N.\ (\ *f*\ (\%n.\ setsum\ f\ \{0..<n\}))\ N = sumhr(0,N,f)$
⟨*proof*⟩

**lemma** *sumhr-hrabs-approx* [*simp*]: $sumhr(0,\ M,\ f)$ @= $sumhr(0,\ N,\ f)$
$\Longrightarrow abs\ (sumhr(M,\ N,\ f))$ @= $0$
⟨*proof*⟩

**lemma** *sums-NSsums-iff*: $(f\ sums\ l) = (f\ NSsums\ l)$
⟨*proof*⟩

**lemma** *summable-NSsummable-iff*: $(summable\ f) = (NSsummable\ f)$
⟨*proof*⟩

**lemma** *suminf-NSsuminf-iff*: $(suminf\ f) = (NSsuminf\ f)$
⟨*proof*⟩

**lemma** *NSsums-NSsummable*: $f\ NSsums\ l \Longrightarrow NSsummable\ f$
⟨*proof*⟩

**lemma** *NSsummable-NSsums*: $NSsummable\ f \Longrightarrow f\ NSsums\ (NSsuminf\ f)$
⟨*proof*⟩

**lemma** *NSsums-unique*: $f\ NSsums\ s \Longrightarrow (s = NSsuminf\ f)$
⟨*proof*⟩

**lemma** *NSseries-zero*:
$\forall m.\ n \le Suc\ m \dashrightarrow f(m) = 0 \Longrightarrow f\ NSsums\ (setsum\ f\ \{0..<n\})$
⟨*proof*⟩

**lemma** *NSsummable-NSCauchy*:
$NSsummable\ f =$
$(\forall M \in HNatInfinite.\ \forall N \in HNatInfinite.\ abs\ (sumhr(M,N,f))$ @= $0)$
⟨*proof*⟩

Terms of a convergent series tend to zero

**lemma** *NSsummable-NSLIMSEQ-zero*: *NSsummable f ==> f ----NS> 0*
⟨*proof*⟩

Nonstandard comparison test

**lemma** *NSsummable-comparison-test*:
[| $\exists N. \forall n.\ N \le n$ --> *abs*(*f n*) $\le$ *g n*; *NSsummable g* |] ==> *NSsummable f*
⟨*proof*⟩

**lemma** *NSsummable-rabs-comparison-test*:
[| $\exists N. \forall n.\ N \le n$ --> *abs*(*f n*) $\le$ *g n*; *NSsummable g* |]
==> *NSsummable* (%*k*. *abs* (*f k*))
⟨*proof*⟩

**end**

# 33 HLim: Limits and Continuity (Nonstandard)

**theory** *HLim*
**imports** *Star Lim*
**begin**

Nonstandard Definitions

**definition**
*NSLIM* :: ['*a*::*real-normed-vector* => '*b*::*real-normed-vector*, '*a*, '*b*] => *bool*
(((-)/ -- (-)/ --NS> (-)) [*60*, *0*, *60*] *60*) **where**
*f* -- *a* --NS> *L* =
($\forall x$. ($x \neq$ *star-of a* & *x* @= *star-of a* --> ( **f** *f*) *x* @= *star-of L*))

**definition**
*isNSCont* :: ['*a*::*real-normed-vector* => '*b*::*real-normed-vector*, '*a*] => *bool* **where**
— NS definition dispenses with limit notions
*isNSCont f a* = ($\forall y$. *y* @= *star-of a* -->
( **f** *f*) *y* @= *star-of* (*f a*))

**definition**
*isNSUCont* :: ['*a*::*real-normed-vector* => '*b*::*real-normed-vector*] => *bool* **where**
*isNSUCont f* = ($\forall x\ y$. *x* @= *y* --> ( **f** *f*) *x* @= ( **f** *f*) *y*)

## 33.1 Limits of Functions

**lemma** *NSLIM-I*:
($\bigwedge x$. ⟦$x \neq$ *star-of a*; $x \approx$ *star-of a*⟧ $\Longrightarrow$ *starfun f x* $\approx$ *star-of L*)
$\Longrightarrow$ *f* -- *a* --NS> *L*
⟨*proof*⟩

**lemma** *NSLIM-D*:

$[\![f\ \text{--}\ a\ \text{--}NS\text{>}\ L;\ x \neq \textit{star-of}\ a;\ x \approx \textit{star-of}\ a]\!]$
$\Longrightarrow \textit{starfun}\ f\ x \approx \textit{star-of}\ L$
⟨proof⟩

Proving properties of limits using nonstandard definition. The properties hold for standard limits as well!

**lemma** *NSLIM-mult*:
**fixes** *l m* :: *'a::real-normed-algebra*
**shows** [| *f* −− *x* −−*NS*> *l*; *g* −− *x* −−*NS*> *m* |]
==> (%*x*. *f*(*x*) ∗ *g*(*x*)) −− *x* −−*NS*> (*l* ∗ *m*)
⟨proof⟩

**lemma** *starfun-scaleR* [*simp*]:
$\textit{starfun}\ (\lambda x.\ f\ x *_R g\ x) = (\lambda x.\ \textit{scaleHR}\ (\textit{starfun}\ f\ x)\ (\textit{starfun}\ g\ x))$
⟨proof⟩

**lemma** *NSLIM-scaleR*:
[| *f* −− *x* −−*NS*> *l*; *g* −− *x* −−*NS*> *m* |]
==> (%*x*. *f*(*x*) $*_R$ *g*(*x*)) −− *x* −−*NS*> (*l* $*_R$ *m*)
⟨proof⟩

**lemma** *NSLIM-add*:
[| *f* −− *x* −−*NS*> *l*; *g* −− *x* −−*NS*> *m* |]
==> (%*x*. *f*(*x*) + *g*(*x*)) −− *x* −−*NS*> (*l* + *m*)
⟨proof⟩

**lemma** *NSLIM-const* [*simp*]: (%*x*. *k*) −− *x* −−*NS*> *k*
⟨proof⟩

**lemma** *NSLIM-minus*: *f* −− *a* −−*NS*> *L* ==> (%*x*. −*f*(*x*)) −− *a* −−*NS*> −*L*
⟨proof⟩

**lemma** *NSLIM-diff*:
$[\![f$ −− *x* −−*NS*> *l*; *g* −− *x* −−*NS*> $m]\!] \Longrightarrow (\lambda x.\ f\ x - g\ x)$ −− *x* −−*NS*> (*l* − *m*)
⟨proof⟩

**lemma** *NSLIM-add-minus*: [| *f* −− *x* −−*NS*> *l*; *g* −− *x* −−*NS*> *m* |] ==> (%*x*. *f*(*x*) + −*g*(*x*)) −− *x* −−*NS*> (*l* + −*m*)
⟨proof⟩

**lemma** *NSLIM-inverse*:
**fixes** *L* :: *'a::real-normed-div-algebra*
**shows** [| *f* −− *a* −−*NS*> *L*; *L* ≠ *0* |]
==> (%*x*. *inverse*(*f*(*x*))) −− *a* −−*NS*> (*inverse L*)
⟨proof⟩

**lemma** *NSLIM-zero*:
**assumes** *f*: *f* −− *a* −−*NS*> *l* **shows** (%*x*. *f*(*x*) − *l*) −− *a* −−*NS*> *0*

⟨*proof*⟩

**lemma** *NSLIM-zero-cancel*: (%$x$. $f(x) - l$) −− $x$ −−$NS$> $0$ ==> $f$ −− $x$ −−$NS$> $l$
⟨*proof*⟩

**lemma** *NSLIM-const-not-eq*:
  **fixes** $a$ :: *'a::real-normed-algebra-1*
  **shows** $k \neq L \Longrightarrow \neg$ ($\lambda x$. $k$) −− $a$ −−$NS$> $L$
⟨*proof*⟩

**lemma** *NSLIM-not-zero*:
  **fixes** $a$ :: *'a::real-normed-algebra-1*
  **shows** $k \neq 0 \Longrightarrow \neg$ ($\lambda x$. $k$) −− $a$ −−$NS$> $0$
⟨*proof*⟩

**lemma** *NSLIM-const-eq*:
  **fixes** $a$ :: *'a::real-normed-algebra-1*
  **shows** ($\lambda x$. $k$) −− $a$ −−$NS$> $L \Longrightarrow k = L$
⟨*proof*⟩

**lemma** *NSLIM-unique*:
  **fixes** $a$ :: *'a::real-normed-algebra-1*
  **shows** ⟦$f$ −− $a$ −−$NS$> $L$; $f$ −− $a$ −−$NS$> $M$⟧ $\Longrightarrow L = M$
⟨*proof*⟩

**lemma** *NSLIM-mult-zero*:
  **fixes** $f\ g$ :: *'a::real-normed-vector* ⇒ *'b::real-normed-algebra*
  **shows** [| $f$ −− $x$ −−$NS$> $0$; $g$ −− $x$ −−$NS$> $0$ |] ==> (%$x$. $f(x)*g(x)$) −− $x$ −−$NS$> $0$
⟨*proof*⟩

**lemma** *NSLIM-self*: (%$x$. $x$) −− $a$ −−$NS$> $a$
⟨*proof*⟩

### 33.1.1 Equivalence of *LIM* and *NSLIM*

**lemma** *LIM-NSLIM*:
  **assumes** $f$: $f$ −− $a$ −−> $L$ **shows** $f$ −− $a$ −−$NS$> $L$
⟨*proof*⟩

**lemma** *NSLIM-LIM*:
  **assumes** $f$: $f$ −− $a$ −−$NS$> $L$ **shows** $f$ −− $a$ −−> $L$
⟨*proof*⟩

**theorem** *LIM-NSLIM-iff*: ($f$ −− $x$ −−> $L$) = ($f$ −− $x$ −−$NS$> $L$)
⟨*proof*⟩

## 33.2 Continuity

**lemma** *isNSContD*:
$[\![isNSCont\ f\ a;\ y \approx star\text{-}of\ a]\!] \Longrightarrow (\ *f*\ f)\ y \approx star\text{-}of\ (f\ a)$
⟨*proof*⟩

**lemma** *isNSCont-NSLIM*: $isNSCont\ f\ a ==> f\ --\ a\ --NS> (f\ a)$
⟨*proof*⟩

**lemma** *NSLIM-isNSCont*: $f\ --\ a\ --NS> (f\ a) ==> isNSCont\ f\ a$
⟨*proof*⟩

NS continuity can be defined using NS Limit in similar fashion to standard def of continuity

**lemma** *isNSCont-NSLIM-iff*: $(isNSCont\ f\ a) = (f\ --\ a\ --NS> (f\ a))$
⟨*proof*⟩

Hence, NS continuity can be given in terms of standard limit

**lemma** *isNSCont-LIM-iff*: $(isNSCont\ f\ a) = (f\ --\ a\ --> (f\ a))$
⟨*proof*⟩

Moreover, it's trivial now that NS continuity is equivalent to standard continuity

**lemma** *isNSCont-isCont-iff*: $(isNSCont\ f\ a) = (isCont\ f\ a)$
⟨*proof*⟩

Standard continuity ==¿ NS continuity

**lemma** *isCont-isNSCont*: $isCont\ f\ a ==> isNSCont\ f\ a$
⟨*proof*⟩

NS continuity ==¿ Standard continuity

**lemma** *isNSCont-isCont*: $isNSCont\ f\ a ==> isCont\ f\ a$
⟨*proof*⟩

Alternative definition of continuity

**lemma** *NSLIM-h-iff*: $(f\ --\ a\ --NS> L) = ((\%h.\ f(a + h))\ --\ 0\ --NS> L)$
⟨*proof*⟩

**lemma** *NSLIM-isCont-iff*: $(f\ --\ a\ --NS> f\ a) = ((\%h.\ f(a + h))\ --\ 0\ --NS> f\ a)$
⟨*proof*⟩

**lemma** *isNSCont-minus*: $isNSCont\ f\ a ==> isNSCont\ (\%x.\ - f\ x)\ a$
⟨*proof*⟩

**lemma** *isNSCont-inverse*:
**fixes** $f :: 'a{::}real\text{-}normed\text{-}vector \Rightarrow 'b{::}real\text{-}normed\text{-}div\text{-}algebra$
**shows** $[|\ isNSCont\ f\ x;\ f\ x \neq 0\ |] ==> isNSCont\ (\%x.\ inverse\ (f\ x))\ x$

⟨*proof*⟩

**lemma** *isNSCont-const* [*simp*]: *isNSCont* (%*x*. *k*) *a*
⟨*proof*⟩

**lemma** *isNSCont-abs* [*simp*]: *isNSCont abs* (*a*::*real*)
⟨*proof*⟩

## 33.3 Uniform Continuity

**lemma** *isNSUContD*: [| *isNSUCont f*; *x* ≈ *y*|] ==> ( **f** *f*) *x* ≈ ( **f** *f*) *y*
⟨*proof*⟩

**lemma** *isUCont-isNSUCont*:
  **fixes** *f* :: ′*a*::*real-normed-vector* ⇒ ′*b*::*real-normed-vector*
  **assumes** *f*: *isUCont f* **shows** *isNSUCont f*
⟨*proof*⟩

**lemma** *isNSUCont-isUCont*:
  **fixes** *f* :: ′*a*::*real-normed-vector* ⇒ ′*b*::*real-normed-vector*
  **assumes** *f*: *isNSUCont f* **shows** *isUCont f*
⟨*proof*⟩

**end**

# 34 HDeriv: Differentiation (Nonstandard)

**theory** *HDeriv*
**imports** *Deriv HLim*
**begin**

Nonstandard Definitions

**definition**
  *nsderiv* :: [′*a*::*real-normed-field* ⇒ ′*a*, ′*a*, ′*a*] ⇒ *bool*
        ((*NSDERIV* (-)/ (-)/ :> (-)) [*1000*, *1000*, *60*] *60*) **where**
  *NSDERIV f x* :> *D* = (∀ *h* ∈ *Infinitesimal* − {*0*}.
      (( **f** *f*)(*star-of x* + *h*)
       − *star-of* (*f x*))/*h* @= *star-of D*)

**definition**
  *NSdifferentiable* :: [′*a*::*real-normed-field* ⇒ ′*a*, ′*a*] ⇒ *bool*
    (**infixl** *NSdifferentiable 60*) **where**
  *f NSdifferentiable x* = (∃ *D*. *NSDERIV f x* :> *D*)

**definition**
  *increment* :: [*real*=>*real*,*real*,*hypreal*] => *hypreal* **where**
  *increment f x h* = (@*inc*. *f NSdifferentiable x* &
          *inc* = ( **f** *f*)(*hypreal-of-real x* + *h*) − *hypreal-of-real* (*f x*))

## 34.1 Derivatives

**lemma** *DERIV-NS-iff*:
$(DERIV\ f\ x :> D) = ((\%h.\ (f(x + h) - f(x))/h) -- \ 0 \ --NS> D)$
⟨*proof*⟩

**lemma** *NS-DERIV-D*: $DERIV\ f\ x :> D ==> (\%h.\ (f(x + h) - f(x))/h) -- \ 0$ $--NS> D$
⟨*proof*⟩

**lemma** *hnorm-of-hypreal*:
$\bigwedge r.\ hnorm\ ((\ *f*\ of\text{-}real)\ r::'a::real\text{-}normed\text{-}div\text{-}algebra\ star) = |r|$
⟨*proof*⟩

**lemma** *Infinitesimal-of-hypreal*:
$x \in Infinitesimal \Longrightarrow$
$((\ *f*\ of\text{-}real)\ x::'a::real\text{-}normed\text{-}div\text{-}algebra\ star) \in Infinitesimal$
⟨*proof*⟩

**lemma** *of-hypreal-eq-0-iff*:
$\bigwedge x.\ ((\ *f*\ of\text{-}real)\ x = (0::'a::real\text{-}algebra\text{-}1\ star)) = (x = 0)$
⟨*proof*⟩

**lemma** *NSDeriv-unique*:
$[|\ NSDERIV\ f\ x :> D;\ NSDERIV\ f\ x :> E\ |] ==> D = E$
⟨*proof*⟩

First NSDERIV in terms of NSLIM

first equivalence

**lemma** *NSDERIV-NSLIM-iff*:
$(NSDERIV\ f\ x :> D) = ((\%h.\ (f(x + h) - f(x))/h) -- \ 0 \ --NS> D)$
⟨*proof*⟩

second equivalence

**lemma** *NSDERIV-NSLIM-iff2*:
$(NSDERIV\ f\ x :> D) = ((\%z.\ (f(z) - f(x))\ /\ (z-x)) -- \ x \ --NS> D)$
⟨*proof*⟩

**lemma** *NSDERIV-iff2*:
$(NSDERIV\ f\ x :> D) =$
$(\forall w.$
$w \neq star\text{-}of\ x\ \&\ w \approx star\text{-}of\ x\ -->$
$(\ *f*\ (\%z.\ (f\ z - f\ x)\ /\ (z-x)))\ w \approx star\text{-}of\ D)$
⟨*proof*⟩

**lemma** *hypreal-not-eq-minus-iff*:

$(x \neq a) = (x - a \neq (0::'a::ab\text{-}group\text{-}add))$
⟨proof⟩

**lemma** *NSDERIVD5*:
$(NSDERIV\ f\ x :> D) ==>$
$(\forall u.\ u \approx hypreal\text{-}of\text{-}real\ x \ -->$
$(\ *f*\ (\%z.\ f\ z - f\ x))\ u \approx hypreal\text{-}of\text{-}real\ D * (u - hypreal\text{-}of\text{-}real\ x))$
⟨proof⟩

**lemma** *NSDERIVD4*:
$(NSDERIV\ f\ x :> D) ==>$
$(\forall h \in Infinitesimal.$
$((\ *f*\ f)(hypreal\text{-}of\text{-}real\ x + h) -$
$hypreal\text{-}of\text{-}real\ (f\ x)) \approx (hypreal\text{-}of\text{-}real\ D) * h)$
⟨proof⟩

**lemma** *NSDERIVD3*:
$(NSDERIV\ f\ x :> D) ==>$
$(\forall h \in Infinitesimal - \{0\}.$
$((\ *f*\ f)(hypreal\text{-}of\text{-}real\ x + h) -$
$hypreal\text{-}of\text{-}real\ (f\ x)) \approx (hypreal\text{-}of\text{-}real\ D) * h)$
⟨proof⟩

Differentiability implies continuity nice and simple "algebraic" proof

**lemma** *NSDERIV-isNSCont*: $NSDERIV\ f\ x :> D ==> isNSCont\ f\ x$
⟨proof⟩

Differentiation rules for combinations of functions follow from clear, straightforard, algebraic manipulations

Constant function

**lemma** *NSDERIV-const* [*simp*]: $(NSDERIV\ (\%x.\ k)\ x :> 0)$
⟨proof⟩

Sum of functions- proved easily

**lemma** *NSDERIV-add*: $[|\ NSDERIV\ f\ x :> Da;\ \ NSDERIV\ g\ x :> Db\ |]$
$==> NSDERIV\ (\%x.\ f\ x + g\ x)\ x :> Da + Db$
⟨proof⟩

Product of functions - Proof is trivial but tedious and long due to rearrangement of terms

**lemma** *lemma-nsderiv1*:
**fixes** $a\ b\ c\ d :: 'a::comm\text{-}ring\ star$
**shows** $(a*b) - (c*d) = (b*(a - c)) + (c*(b - d))$
⟨proof⟩

**lemma** *lemma-nsderiv2*:
**fixes** $x\ y\ z :: 'a::real\text{-}normed\text{-}field\ star$

**shows** [| $(x - y) / z = \mathit{star\text{-}of}\ D + yb$; $z \neq 0$;
$z \in \mathit{Infinitesimal}$; $yb \in \mathit{Infinitesimal}$ |]
$==> x - y \approx 0$
⟨*proof*⟩

**lemma** *NSDERIV-mult*: [| *NSDERIV f x* :> *Da*; *NSDERIV g x* :> *Db* |]
$==>$ *NSDERIV* $(\%x.\ f\ x * g\ x)\ x$ :> $(Da * g(x)) + (Db * f(x))$
⟨*proof*⟩

Multiplying by a constant

**lemma** *NSDERIV-cmult*: *NSDERIV f x* :> *D*
$==>$ *NSDERIV* $(\%x.\ c * f\ x)\ x$ :> $c{*}D$
⟨*proof*⟩

Negation of function

**lemma** *NSDERIV-minus*: *NSDERIV f x* :> *D* $==>$ *NSDERIV* $(\%x.\ -(f\ x))\ x$ :> $-D$
⟨*proof*⟩

Subtraction

**lemma** *NSDERIV-add-minus*: [| *NSDERIV f x* :> *Da*; *NSDERIV g x* :> *Db* |]
$==>$ *NSDERIV* $(\%x.\ f\ x + -g\ x)\ x$ :> $Da + -Db$
⟨*proof*⟩

**lemma** *NSDERIV-diff*:
[| *NSDERIV f x* :> *Da*; *NSDERIV g x* :> *Db* |]
$==>$ *NSDERIV* $(\%x.\ f\ x - g\ x)\ x$ :> $Da{-}Db$
⟨*proof*⟩

Similarly to the above, the chain rule admits an entirely straightforward derivation. Compare this with Harrison's HOL proof of the chain rule, which proved to be trickier and required an alternative characterisation of differentiability- the so-called Carathedory derivative. Our main problem is manipulation of terms.

**lemma** *NSDERIV-zero*:
[| *NSDERIV g x* :> *D*;
$( *f* g)\ (\mathit{star\text{-}of}\ x + xa) = \mathit{star\text{-}of}\ (g\ x)$;
$xa \in \mathit{Infinitesimal}$;
$xa \neq 0$
|] $==> D = 0$
⟨*proof*⟩

**lemma** *NSDERIV-approx*:
[| *NSDERIV f x* :> *D*; $h \in \mathit{Infinitesimal}$; $h \neq 0$ |]
$==> ( *f* f)\ (\mathit{star\text{-}of}\ x + h) - \mathit{star\text{-}of}\ (f\ x) \approx 0$
⟨*proof*⟩

**lemma** *NSDERIVD1*: $[\mid$ *NSDERIV* $f\ (g\ x)$ :> $Da$;
$(\ *f*\ g)\ (star\text{-}of(x) + xa) \neq star\text{-}of\ (g\ x)$;
$(\ *f*\ g)\ (star\text{-}of(x) + xa) \approx star\text{-}of\ (g\ x)$
$\mid]$ ==> $((\ *f*\ f)\ ((\ *f*\ g)\ (star\text{-}of(x) + xa))$
$-\ star\text{-}of\ (f\ (g\ x)))$
$/\ ((\ *f*\ g)\ (star\text{-}of(x) + xa) - star\text{-}of\ (g\ x))$
$\approx star\text{-}of(Da)$
⟨*proof*⟩

**lemma** *NSDERIVD2*: $[\mid$ *NSDERIV* $g\ x$ :> $Db$; $xa \in$ *Infinitesimal*; $xa \neq 0 \mid]$
==> $((\ *f*\ g)\ (star\text{-}of(x) + xa) - star\text{-}of(g\ x))\ /\ xa$
$\approx star\text{-}of(Db)$
⟨*proof*⟩

**lemma** *lemma-chain*: $(z::'a::real\text{-}normed\text{-}field\ star) \neq 0$ ==> $x*y = (x*inverse(z))*(z*y)$
⟨*proof*⟩

This proof uses both definitions of differentiability.

**lemma** *NSDERIV-chain*: $[\mid$ *NSDERIV* $f\ (g\ x)$ :> $Da$; *NSDERIV* $g\ x$ :> $Db \mid]$
==> *NSDERIV* $(f\ o\ g)\ x$ :> $Da * Db$
⟨*proof*⟩

Differentiation of natural number powers

**lemma** *NSDERIV-Id* [*simp*]: *NSDERIV* $(\%x.\ x)\ x$ :> *1*
⟨*proof*⟩

**lemma** *NSDERIV-cmult-Id* [*simp*]: *NSDERIV* $(op * c)\ x$ :> $c$
⟨*proof*⟩

**lemma** *NSDERIV-inverse*:
**fixes** $x :: 'a::\{real\text{-}normed\text{-}field,recpower\}$
**shows** $x \neq 0$ ==> *NSDERIV* $(\%x.\ inverse(x))\ x$ :> $(-\ (inverse\ x\ \hat{}\ Suc\ (Suc\ 0)))$
⟨*proof*⟩

### 34.1.1 Equivalence of NS and Standard definitions

**lemma** *divideR-eq-divide*: $x\ /_R\ y = x\ /\ y$
⟨*proof*⟩

Now equivalence between NSDERIV and DERIV

**lemma** *NSDERIV-DERIV-iff*: (*NSDERIV* $f\ x$ :> $D$) = (*DERIV* $f\ x$ :> $D$)
⟨*proof*⟩

**lemma** *NSDERIV-pow*: *NSDERIV* (%*x*. *x* ^ *n*) *x* :> *real n* ∗ (*x* ^ (*n* − *Suc 0*))
⟨*proof*⟩

Derivative of inverse

**lemma** *NSDERIV-inverse-fun*:
**fixes** *x* :: *'a*::{*real-normed-field*,*recpower*}
**shows** [| *NSDERIV f x* :> *d*; *f*(*x*) ≠ *0* |]
==> *NSDERIV* (%*x*. *inverse*(*f x*)) *x* :> (− (*d* ∗ *inverse*(*f*(*x*) ^ *Suc* (*Suc 0*))))
⟨*proof*⟩

Derivative of quotient

**lemma** *NSDERIV-quotient*:
**fixes** *x* :: *'a*::{*real-normed-field*,*recpower*}
**shows** [| *NSDERIV f x* :> *d*; *NSDERIV g x* :> *e*; *g*(*x*) ≠ *0* |]
==> *NSDERIV* (%*y*. *f*(*y*) / (*g y*)) *x* :> (*d*∗*g*(*x*)
− (*e*∗*f*(*x*))) / (*g*(*x*) ^ *Suc* (*Suc 0*))
⟨*proof*⟩

**lemma** *CARAT-NSDERIV*: *NSDERIV f x* :> *l* ==>
∃ *g*. (∀ *z*. *f z* − *f x* = *g z* ∗ (*z*−*x*)) & *isNSCont g x* & *g x* = *l*
⟨*proof*⟩

**lemma** *hypreal-eq-minus-iff3*: (*x* = *y* + *z*) = (*x* + −*z* = (*y*::*hypreal*))
⟨*proof*⟩

**lemma** *CARAT-DERIVD*:
**assumes** *all*: ∀ *z*. *f z* − *f x* = *g z* ∗ (*z*−*x*)
**and** *nsc*: *isNSCont g x*
**shows** *NSDERIV f x* :> *g x*
⟨*proof*⟩

### 34.1.2 Differentiability predicate

**lemma** *NSdifferentiableD*: *f NSdifferentiable x* ==> ∃ *D*. *NSDERIV f x* :> *D*
⟨*proof*⟩

**lemma** *NSdifferentiableI*: *NSDERIV f x* :> *D* ==> *f NSdifferentiable x*
⟨*proof*⟩

## 34.2 (NS) Increment

**lemma** *incrementI*:
*f NSdifferentiable x* ==>
*increment f x h* = ( ∗*f*∗ *f*) (*hypreal-of-real*(*x*) + *h*) −
*hypreal-of-real* (*f x*)
⟨*proof*⟩

**lemma** *incrementI2*: *NSDERIV f x* :> *D* ==>
*increment f x h* = ( **f* f*) (*hypreal-of-real*(*x*) + *h*) −
*hypreal-of-real* (*f x*)
⟨*proof*⟩

**lemma** *increment-thm*: [| *NSDERIV f x* :> *D*; *h* ∈ *Infinitesimal*; *h* ≠ *0* |]
==> ∃ *e* ∈ *Infinitesimal*. *increment f x h* = *hypreal-of-real*(*D*)**h* + *e***h*
⟨*proof*⟩

**lemma** *increment-thm2*:
[| *NSDERIV f x* :> *D*; *h* ≈ *0*; *h* ≠ *0* |]
==> ∃ *e* ∈ *Infinitesimal*. *increment f x h* =
*hypreal-of-real*(*D*)**h* + *e***h*
⟨*proof*⟩

**lemma** *increment-approx-zero*: [| *NSDERIV f x* :> *D*; *h* ≈ *0*; *h* ≠ *0* |]
==> *increment f x h* ≈ *0*
⟨*proof*⟩

**end**

# 35 HTranscendental: Nonstandard Extensions of Transcendental Functions

**theory** *HTranscendental*
**imports** *Transcendental HSeries HDeriv*
**begin**

**definition**
*exphr* :: *real* => *hypreal* **where**
— define exponential function using standard part
*exphr x* = *st*(*sumhr* (*0*, *whn*, %*n*. *inverse*(*real* (*fact n*)) ∗ (*x* ˆ *n*)))

**definition**
*sinhr* :: *real* => *hypreal* **where**
*sinhr x* = *st*(*sumhr* (*0*, *whn*, %*n*. (*if even*(*n*) *then 0 else*
((−*1*) ˆ ((*n* − *1*) *div 2*))/(*real* (*fact n*))) ∗ (*x* ˆ *n*)))

**definition**
*coshr* :: *real* => *hypreal* **where**
*coshr x* = *st*(*sumhr* (*0*, *whn*, %*n*. (*if even*(*n*) *then*
((−*1*) ˆ (*n div 2*))/(*real* (*fact n*)) *else 0*) ∗ *x* ˆ *n*))

## 35.1 Nonstandard Extension of Square Root Function

**lemma** *STAR-sqrt-zero* [*simp*]: ( *f* sqrt) 0 = 0
⟨*proof*⟩

**lemma** *STAR-sqrt-one* [*simp*]: ( *f* sqrt) 1 = 1
⟨*proof*⟩

**lemma** *hypreal-sqrt-pow2-iff*: (( *f* sqrt)(x) ^ 2 = x) = (0 ≤ x)
⟨*proof*⟩

**lemma** *hypreal-sqrt-gt-zero-pow2*: !!x. 0 < x ==> ( *f* sqrt) (x) ^ 2 = x
⟨*proof*⟩

**lemma** *hypreal-sqrt-pow2-gt-zero*: 0 < x ==> 0 < ( *f* sqrt) (x) ^ 2
⟨*proof*⟩

**lemma** *hypreal-sqrt-not-zero*: 0 < x ==> ( *f* sqrt) (x) ≠ 0
⟨*proof*⟩

**lemma** *hypreal-inverse-sqrt-pow2*:
  0 < x ==> inverse (( *f* sqrt)(x)) ^ 2 = inverse x
⟨*proof*⟩

**lemma** *hypreal-sqrt-mult-distrib*:
  !!x y. [|0 < x; 0 <y |] ==>
   ( *f* sqrt)(x*y) = ( *f* sqrt)(x) * ( *f* sqrt)(y)
⟨*proof*⟩

**lemma** *hypreal-sqrt-mult-distrib2*:
  [|0≤x; 0≤y |] ==>
  ( *f* sqrt)(x*y) = ( *f* sqrt)(x) * ( *f* sqrt)(y)
⟨*proof*⟩

**lemma** *hypreal-sqrt-approx-zero* [*simp*]:
  0 < x ==> (( *f* sqrt)(x) @= 0) = (x @= 0)
⟨*proof*⟩

**lemma** *hypreal-sqrt-approx-zero2* [*simp*]:
  0 ≤ x ==> (( *f* sqrt)(x) @= 0) = (x @= 0)
⟨*proof*⟩

**lemma** *hypreal-sqrt-sum-squares* [*simp*]:
  (( *f* sqrt)(x*x + y*y + z*z) @= 0) = (x*x + y*y + z*z @= 0)
⟨*proof*⟩

**lemma** *hypreal-sqrt-sum-squares2* [*simp*]:
  (( *f* sqrt)(x*x + y*y) @= 0) = (x*x + y*y @= 0)
⟨*proof*⟩

**lemma** *hypreal-sqrt-gt-zero*: $!!x.\ 0 < x ==> 0 < ( *f* \ sqrt)(x)$
⟨*proof*⟩

**lemma** *hypreal-sqrt-ge-zero*: $0 \leq x ==> 0 \leq ( *f* \ sqrt)(x)$
⟨*proof*⟩

**lemma** *hypreal-sqrt-hrabs* [*simp*]: $!!x.\ ( *f* \ sqrt)(x \ \hat{} \ 2) = abs(x)$
⟨*proof*⟩

**lemma** *hypreal-sqrt-hrabs2* [*simp*]: $!!x.\ ( *f* \ sqrt)(x*x) = abs(x)$
⟨*proof*⟩

**lemma** *hypreal-sqrt-hyperpow-hrabs* [*simp*]:
$!!x.\ ( *f* \ sqrt)(x \ pow \ (hypnat\text{-}of\text{-}nat \ 2)) = abs(x)$
⟨*proof*⟩

**lemma** *star-sqrt-HFinite*: $[\![x \in HFinite;\ 0 \leq x]\!] \Longrightarrow ( *f* \ sqrt) \ x \in HFinite$
⟨*proof*⟩

**lemma** *st-hypreal-sqrt*:
$[|\ x \in HFinite;\ 0 \leq x\ |] ==> st(( *f* \ sqrt) \ x) = ( *f* \ sqrt)(st \ x)$
⟨*proof*⟩

**lemma** *hypreal-sqrt-sum-squares-ge1* [*simp*]: $!!x \ y.\ x \leq ( *f* \ sqrt)(x \ \hat{} \ 2 + y \ \hat{} \ 2)$
⟨*proof*⟩

**lemma** *HFinite-hypreal-sqrt*:
$[|\ 0 \leq x;\ x \in HFinite\ |] ==> ( *f* \ sqrt) \ x \in HFinite$
⟨*proof*⟩

**lemma** *HFinite-hypreal-sqrt-imp-HFinite*:
$[|\ 0 \leq x;\ ( *f* \ sqrt) \ x \in HFinite\ |] ==> x \in HFinite$
⟨*proof*⟩

**lemma** *HFinite-hypreal-sqrt-iff* [*simp*]:
$0 \leq x ==> (( *f* \ sqrt) \ x \in HFinite) = (x \in HFinite)$
⟨*proof*⟩

**lemma** *HFinite-sqrt-sum-squares* [*simp*]:
$(( *f* \ sqrt)(x*x + y*y) \in HFinite) = (x*x + y*y \in HFinite)$
⟨*proof*⟩

**lemma** *Infinitesimal-hypreal-sqrt*:
$[|\ 0 \leq x;\ x \in Infinitesimal\ |] ==> ( *f* \ sqrt) \ x \in Infinitesimal$
⟨*proof*⟩

**lemma** *Infinitesimal-hypreal-sqrt-imp-Infinitesimal*:
$[|\ 0 \leq x;\ ( *f* \ sqrt) \ x \in Infinitesimal\ |] ==> x \in Infinitesimal$
⟨*proof*⟩

**lemma** *Infinitesimal-hypreal-sqrt-iff* [*simp*]:
  $0 \le x$ ==> (( *f* *sqrt*) $x \in$ *Infinitesimal*) = ($x \in$ *Infinitesimal*)
⟨*proof*⟩

**lemma** *Infinitesimal-sqrt-sum-squares* [*simp*]:
  (( *f* *sqrt*)($x*x + y*y$) $\in$ *Infinitesimal*) = ($x*x + y*y \in$ *Infinitesimal*)
⟨*proof*⟩

**lemma** *HInfinite-hypreal-sqrt*:
  [| $0 \le x$; $x \in$ *HInfinite* |] ==> ( *f* *sqrt*) $x \in$ *HInfinite*
⟨*proof*⟩

**lemma** *HInfinite-hypreal-sqrt-imp-HInfinite*:
  [| $0 \le x$; ( *f* *sqrt*) $x \in$ *HInfinite* |] ==> $x \in$ *HInfinite*
⟨*proof*⟩

**lemma** *HInfinite-hypreal-sqrt-iff* [*simp*]:
  $0 \le x$ ==> (( *f* *sqrt*) $x \in$ *HInfinite*) = ($x \in$ *HInfinite*)
⟨*proof*⟩

**lemma** *HInfinite-sqrt-sum-squares* [*simp*]:
  (( *f* *sqrt*)($x*x + y*y$) $\in$ *HInfinite*) = ($x*x + y*y \in$ *HInfinite*)
⟨*proof*⟩

**lemma** *HFinite-exp* [*simp*]:
  *sumhr* (*0*, *whn*, %*n*. *inverse* (*real* (*fact n*)) * $x$ ^ $n$) $\in$ *HFinite*
⟨*proof*⟩

**lemma** *exphr-zero* [*simp*]: *exphr 0 = 1*
⟨*proof*⟩

**lemma** *coshr-zero* [*simp*]: *coshr 0 = 1*
⟨*proof*⟩

**lemma** *STAR-exp-zero-approx-one* [*simp*]: ( *f* *exp*) (*0*::*hypreal*) @= *1*
⟨*proof*⟩

**lemma** *STAR-exp-Infinitesimal*: $x \in$ *Infinitesimal* ==> ( *f* *exp*) ($x$::*hypreal*) @= *1*
⟨*proof*⟩

**lemma** *STAR-exp-epsilon* [*simp*]: ( *f* *exp*) *epsilon* @= *1*
⟨*proof*⟩

**lemma** *STAR-exp-add*: !!$x$ $y$. ( *f* *exp*)($x + y$) = ( *f* *exp*) $x$ * ( *f* *exp*) $y$
⟨*proof*⟩

**lemma** *exphr-hypreal-of-real-exp-eq*: *exphr* $x$ = *hypreal-of-real* (*exp* $x$)

⟨*proof*⟩

**lemma** *starfun-exp-ge-add-one-self* [*simp*]: !!*x*::*hypreal*. *0* ≤ *x* ==> (*1* + *x*) ≤ ( ∗*f*∗ *exp*) *x*
⟨*proof*⟩

**lemma** *starfun-exp-HInfinite*:
[| *x* ∈ *HInfinite*; *0* ≤ *x* |] ==> ( ∗*f*∗ *exp*) (*x*::*hypreal*) ∈ *HInfinite*
⟨*proof*⟩

**lemma** *starfun-exp-minus*: !!*x*. ( ∗*f*∗ *exp*) (−*x*) = *inverse*(( ∗*f*∗ *exp*) *x*)
⟨*proof*⟩

**lemma** *starfun-exp-Infinitesimal*:
[| *x* ∈ *HInfinite*; *x* ≤ *0* |] ==> ( ∗*f*∗ *exp*) (*x*::*hypreal*) ∈ *Infinitesimal*
⟨*proof*⟩

**lemma** *starfun-exp-gt-one* [*simp*]: !!*x*::*hypreal*. *0* < *x* ==> *1* < ( ∗*f*∗ *exp*) *x*
⟨*proof*⟩

**lemma** *starfun-ln-exp* [*simp*]: !!*x*. ( ∗*f*∗ *ln*) (( ∗*f*∗ *exp*) *x*) = *x*
⟨*proof*⟩

**lemma** *starfun-exp-ln-iff* [*simp*]: !!*x*. (( ∗*f*∗ *exp*)(( ∗*f*∗ *ln*) *x*) = *x*) = (*0* < *x*)
⟨*proof*⟩

**lemma** *starfun-exp-ln-eq*: !!*u x*. ( ∗*f*∗ *exp*) *u* = *x* ==> ( ∗*f*∗ *ln*) *x* = *u*
⟨*proof*⟩

**lemma** *starfun-ln-less-self* [*simp*]: !!*x*. *0* < *x* ==> ( ∗*f*∗ *ln*) *x* < *x*
⟨*proof*⟩

**lemma** *starfun-ln-ge-zero* [*simp*]: !!*x*. *1* ≤ *x* ==> *0* ≤ ( ∗*f*∗ *ln*) *x*
⟨*proof*⟩

**lemma** *starfun-ln-gt-zero* [*simp*]: !!*x* .*1* < *x* ==> *0* < ( ∗*f*∗ *ln*) *x*
⟨*proof*⟩

**lemma** *starfun-ln-not-eq-zero* [*simp*]: !!*x*. [| *0* < *x*; *x* ≠ *1* |] ==> ( ∗*f*∗ *ln*) *x* ≠ *0*
⟨*proof*⟩

**lemma** *starfun-ln-HFinite*: [| *x* ∈ *HFinite*; *1* ≤ *x* |] ==> ( ∗*f*∗ *ln*) *x* ∈ *HFinite*
⟨*proof*⟩

**lemma** *starfun-ln-inverse*: $!!x.\ 0 < x ==> (\ *f*\ ln)\ (inverse\ x) = -(\ *f*\ ln)\ x$
⟨*proof*⟩

**lemma** *starfun-abs-exp-cancel*: $\bigwedge x.\ |(\ *f*\ exp)\ (x::hypreal)| = (\ *f*\ exp)\ x$
⟨*proof*⟩

**lemma** *starfun-exp-less-mono*: $\bigwedge x\ y::hypreal.\ x < y \Longrightarrow (\ *f*\ exp)\ x < (\ *f*\ exp)\ y$
⟨*proof*⟩

**lemma** *starfun-exp-HFinite*: $x \in HFinite ==> (\ *f*\ exp)\ (x::hypreal) \in HFinite$
⟨*proof*⟩

**lemma** *starfun-exp-add-HFinite-Infinitesimal-approx*:
$[|x \in Infinitesimal;\ z \in HFinite\ |] ==> (\ *f*\ exp)\ (z + x::hypreal)\ @= (\ *f*\ exp)\ z$
⟨*proof*⟩

**lemma** *starfun-ln-HInfinite*:
$[|\ x \in HInfinite;\ 0 < x\ |] ==> (\ *f*\ ln)\ x \in HInfinite$
⟨*proof*⟩

**lemma** *starfun-exp-HInfinite-Infinitesimal-disj*:
$x \in HInfinite ==> (\ *f*\ exp)\ x \in HInfinite\ |\ (\ *f*\ exp)\ (x::hypreal) \in Infinitesimal$
⟨*proof*⟩

**lemma** *starfun-ln-HFinite-not-Infinitesimal*:
$[|\ x \in HFinite - Infinitesimal;\ 0 < x\ |] ==> (\ *f*\ ln)\ x \in HFinite$
⟨*proof*⟩

**lemma** *starfun-ln-Infinitesimal-HInfinite*:
$[|\ x \in Infinitesimal;\ 0 < x\ |] ==> (\ *f*\ ln)\ x \in HInfinite$
⟨*proof*⟩

**lemma** *starfun-ln-less-zero*: $!!x.\ [|\ 0 < x;\ x < 1\ |] ==> (\ *f*\ ln)\ x < 0$
⟨*proof*⟩

**lemma** *starfun-ln-Infinitesimal-less-zero*:
$[|\ x \in Infinitesimal;\ 0 < x\ |] ==> (\ *f*\ ln)\ x < 0$
⟨*proof*⟩

**lemma** *starfun-ln-HInfinite-gt-zero*:
$[|\ x \in HInfinite;\ 0 < x\ |] ==> 0 < (\ *f*\ ln)\ x$
⟨*proof*⟩

**lemma** *HFinite-sin* [*simp*]:
  *sumhr* (*0*, *whn*, %*n*. (*if even*(*n*) *then 0 else*
    (−*1* ˆ ((*n* − *1*) *div 2*))/(*real* (*fact n*))) ∗ *x* ˆ *n*)
    ∈ *HFinite*
⟨*proof*⟩

**lemma** *STAR-sin-zero* [*simp*]: ( ∗*f*∗ *sin*) *0* = *0*
⟨*proof*⟩

**lemma** *STAR-sin-Infinitesimal* [*simp*]: *x* ∈ *Infinitesimal* ==> ( ∗*f*∗ *sin*) *x* @= *x*
⟨*proof*⟩

**lemma** *HFinite-cos* [*simp*]:
  *sumhr* (*0*, *whn*, %*n*. (*if even*(*n*) *then*
    (−*1* ˆ (*n div 2*))/(*real* (*fact n*)) *else*
    *0*) ∗ *x* ˆ *n*) ∈ *HFinite*
⟨*proof*⟩

**lemma** *STAR-cos-zero* [*simp*]: ( ∗*f*∗ *cos*) *0* = *1*
⟨*proof*⟩

**lemma** *STAR-cos-Infinitesimal* [*simp*]: *x* ∈ *Infinitesimal* ==> ( ∗*f*∗ *cos*) *x* @= *1*
⟨*proof*⟩

**lemma** *STAR-tan-zero* [*simp*]: ( ∗*f*∗ *tan*) *0* = *0*
⟨*proof*⟩

**lemma** *STAR-tan-Infinitesimal*: *x* ∈ *Infinitesimal* ==> ( ∗*f*∗ *tan*) *x* @= *x*
⟨*proof*⟩

**lemma** *STAR-sin-cos-Infinitesimal-mult*:
  *x* ∈ *Infinitesimal* ==> ( ∗*f*∗ *sin*) *x* ∗ ( ∗*f*∗ *cos*) *x* @= *x*
⟨*proof*⟩

**lemma** *HFinite-pi*: *hypreal-of-real pi* ∈ *HFinite*
⟨*proof*⟩

**lemma** *lemma-split-hypreal-of-real*:
  *N* ∈ *HNatInfinite*
  ==> *hypreal-of-real a* =
    *hypreal-of-hypnat N* ∗ (*inverse*(*hypreal-of-hypnat N*) ∗ *hypreal-of-real a*)
⟨*proof*⟩

**lemma** *STAR-sin-Infinitesimal-divide*:
  [|*x* ∈ *Infinitesimal*; *x* ≠ *0* |] ==> ( ∗*f*∗ *sin*) *x*/*x* @= *1*

⟨*proof*⟩

**lemma** *lemma-sin-pi*:
 *n* ∈ *HNatInfinite*
 ==> ( *f* *sin*) (*inverse* (*hypreal-of-hypnat n*))/(*inverse* (*hypreal-of-hypnat n*)) @= *1*
⟨*proof*⟩

**lemma** *STAR-sin-inverse-HNatInfinite*:
 *n* ∈ *HNatInfinite*
 ==> ( *f* *sin*) (*inverse* (*hypreal-of-hypnat n*)) * *hypreal-of-hypnat n* @= *1*
⟨*proof*⟩

**lemma** *Infinitesimal-pi-divide-HNatInfinite*:
 *N* ∈ *HNatInfinite*
 ==> *hypreal-of-real pi*/(*hypreal-of-hypnat N*) ∈ *Infinitesimal*
⟨*proof*⟩

**lemma** *pi-divide-HNatInfinite-not-zero* [*simp*]:
 *N* ∈ *HNatInfinite* ==> *hypreal-of-real pi*/(*hypreal-of-hypnat N*) ≠ *0*
⟨*proof*⟩

**lemma** *STAR-sin-pi-divide-HNatInfinite-approx-pi*:
 *n* ∈ *HNatInfinite*
 ==> ( *f* *sin*) (*hypreal-of-real pi*/(*hypreal-of-hypnat n*)) * *hypreal-of-hypnat n*
 @= *hypreal-of-real pi*
⟨*proof*⟩

**lemma** *STAR-sin-pi-divide-HNatInfinite-approx-pi2*:
 *n* ∈ *HNatInfinite*
 ==> *hypreal-of-hypnat n* *
 ( *f* *sin*) (*hypreal-of-real pi*/(*hypreal-of-hypnat n*))
 @= *hypreal-of-real pi*
⟨*proof*⟩

**lemma** *starfunNat-pi-divide-n-Infinitesimal*:
 *N* ∈ *HNatInfinite* ==> ( *f* (%*x*. *pi* / *real x*)) *N* ∈ *Infinitesimal*
⟨*proof*⟩

**lemma** *STAR-sin-pi-divide-n-approx*:
 *N* ∈ *HNatInfinite* ==>
 ( *f* *sin*) (( *f* (%*x*. *pi* / *real x*)) *N*) @=
 *hypreal-of-real pi*/(*hypreal-of-hypnat N*)
⟨*proof*⟩

**lemma** *NSLIMSEQ-sin-pi*: (%n. real n * sin (pi / real n)) ––––NS> pi
⟨*proof*⟩

**lemma** *NSLIMSEQ-cos-one*: (%n. cos (pi / real n))––––NS> 1
⟨*proof*⟩

**lemma** *NSLIMSEQ-sin-cos-pi*:
(%n. real n * sin (pi / real n) * cos (pi / real n)) ––––NS> pi
⟨*proof*⟩

A familiar approximation to *cos x* when *x* is small

**lemma** *STAR-cos-Infinitesimal-approx*:
x ∈ Infinitesimal ==> ( *f* cos) x @= 1 – x ˆ 2
⟨*proof*⟩

**lemma** *STAR-cos-Infinitesimal-approx2*:
x ∈ Infinitesimal ==> ( *f* cos) x @= 1 – (x ˆ 2)/2
⟨*proof*⟩

**end**

# 36 NSCA: Non-Standard Complex Analysis

**theory** *NSCA*
**imports** *NSComplex ../Hyperreal/HTranscendental*
**begin**

**abbreviation**

*SComplex* :: *hcomplex set* **where**
*SComplex* ≡ *Standard*

**definition**
*stc* :: *hcomplex* => *hcomplex* **where**
— standard part map
*stc x* = (*SOME r. x* ∈ *HFinite* & *r*:*SComplex* & *r* @= *x*)

## 36.1 Closure Laws for SComplex, the Standard Complex Numbers

**lemma** *SComplex-minus-iff* [*simp*]: (–x ∈ SComplex) = (x ∈ SComplex)
⟨*proof*⟩

**lemma** *SComplex-add-cancel*:
[| x + y ∈ SComplex; y ∈ SComplex |] ==> x ∈ SComplex
⟨*proof*⟩

**lemma** *SReal-hcmod-hcomplex-of-complex* [*simp*]:
  *hcmod* (*hcomplex-of-complex r*) $\in$ *Reals*
⟨*proof*⟩

**lemma** *SReal-hcmod-number-of* [*simp*]: *hcmod* (*number-of w* ::*hcomplex*) $\in$ *Reals*
⟨*proof*⟩

**lemma** *SReal-hcmod-SComplex*: *x* $\in$ *SComplex* ==> *hcmod x* $\in$ *Reals*
⟨*proof*⟩

**lemma** *SComplex-divide-number-of*:
  *r* $\in$ *SComplex* ==> *r*/(*number-of w*::*hcomplex*) $\in$ *SComplex*
⟨*proof*⟩

**lemma** *SComplex-UNIV-complex*:
  {*x*. *hcomplex-of-complex x* $\in$ *SComplex*} = (*UNIV*::*complex set*)
⟨*proof*⟩

**lemma** *SComplex-iff*: (*x* $\in$ *SComplex*) = ($\exists$ *y*. *x* = *hcomplex-of-complex y*)
⟨*proof*⟩

**lemma** *hcomplex-of-complex-image*:
  *hcomplex-of-complex* '(*UNIV*::*complex set*) = *SComplex*
⟨*proof*⟩

**lemma** *inv-hcomplex-of-complex-image*: *inv hcomplex-of-complex* '*SComplex* = *UNIV*
⟨*proof*⟩

**lemma** *SComplex-hcomplex-of-complex-image*:
  [| $\exists$ *x*. *x*: *P*; *P* $\leq$ *SComplex* |] ==> $\exists$ *Q*. *P* = *hcomplex-of-complex* ' *Q*
⟨*proof*⟩

**lemma** *SComplex-SReal-dense*:
  [| *x* $\in$ *SComplex*; *y* $\in$ *SComplex*; *hcmod x* < *hcmod y*
   |] ==> $\exists$ *r* $\in$ *Reals*. *hcmod x*< *r* & *r* < *hcmod y*
⟨*proof*⟩

**lemma** *SComplex-hcmod-SReal*:
  *z* $\in$ *SComplex* ==> *hcmod z* $\in$ *Reals*
⟨*proof*⟩

## 36.2 The Finite Elements form a Subring

**lemma** *HFinite-hcmod-hcomplex-of-complex* [*simp*]:
  *hcmod* (*hcomplex-of-complex r*) $\in$ *HFinite*
⟨*proof*⟩

**lemma** *HFinite-hcmod-iff*: (*x* $\in$ *HFinite*) = (*hcmod x* $\in$ *HFinite*)
⟨*proof*⟩

**lemma** *HFinite-bounded-hcmod*:
$[|x \in HFinite;\ y \leq hcmod\ x;\ 0 \leq y\ |] ==> y: HFinite$
⟨*proof*⟩

## 36.3 The Complex Infinitesimals form a Subring

**lemma** *hcomplex-sum-of-halves*: $x/(2::hcomplex) + x/(2::hcomplex) = x$
⟨*proof*⟩

**lemma** *Infinitesimal-hcmod-iff*:
$(z \in Infinitesimal) = (hcmod\ z \in Infinitesimal)$
⟨*proof*⟩

**lemma** *HInfinite-hcmod-iff*: $(z \in HInfinite) = (hcmod\ z \in HInfinite)$
⟨*proof*⟩

**lemma** *HFinite-diff-Infinitesimal-hcmod*:
$x \in HFinite - Infinitesimal ==> hcmod\ x \in HFinite - Infinitesimal$
⟨*proof*⟩

**lemma** *hcmod-less-Infinitesimal*:
$[|\ e \in Infinitesimal;\ hcmod\ x < hcmod\ e\ |] ==> x \in Infinitesimal$
⟨*proof*⟩

**lemma** *hcmod-le-Infinitesimal*:
$[|\ e \in Infinitesimal;\ hcmod\ x \leq hcmod\ e\ |] ==> x \in Infinitesimal$
⟨*proof*⟩

**lemma** *Infinitesimal-interval-hcmod*:
$[|\ e \in Infinitesimal;$
$e' \in Infinitesimal;$
$hcmod\ e' < hcmod\ x\ ;\ hcmod\ x < hcmod\ e$
$|] ==> x \in Infinitesimal$
⟨*proof*⟩

**lemma** *Infinitesimal-interval2-hcmod*:
$[|\ e \in Infinitesimal;$
$e' \in Infinitesimal;$
$hcmod\ e' \leq hcmod\ x\ ;\ hcmod\ x \leq hcmod\ e$
$|] ==> x \in Infinitesimal$
⟨*proof*⟩

## 36.4 The "Infinitely Close" Relation

**lemma** *approx-SComplex-mult-cancel-zero*:
$[|\ a \in SComplex;\ a \neq 0;\ a*x\ @{=}\ 0\ |] ==> x\ @{=}\ 0$
⟨*proof*⟩

**lemma** *approx-mult-SComplex1*: $[|\ a \in SComplex;\ x\ @{=}\ 0\ |] ==> x*a\ @{=}\ 0$

⟨*proof*⟩

**lemma** *approx-mult-SComplex2*: $[| \ a \in SComplex; \ x \ @= \ 0 \ |] ==> a*x \ @= \ 0$
⟨*proof*⟩

**lemma** *approx-mult-SComplex-zero-cancel-iff* [*simp*]:
$[|a \in SComplex; \ a \neq 0 \ |] ==> (a*x \ @= \ 0) = (x \ @= \ 0)$
⟨*proof*⟩

**lemma** *approx-SComplex-mult-cancel*:
$[| \ a \in SComplex; \ a \neq 0; \ a* \ w \ @= \ a*z \ |] ==> w \ @= \ z$
⟨*proof*⟩

**lemma** *approx-SComplex-mult-cancel-iff1* [*simp*]:
$[| \ a \in SComplex; \ a \neq 0|] ==> (a* \ w \ @= \ a*z) = (w \ @= \ z)$
⟨*proof*⟩

**lemma** *approx-hcmod-approx-zero*: $(x \ @= \ y) = (hcmod \ (y - x) \ @= \ 0)$
⟨*proof*⟩

**lemma** *approx-approx-zero-iff*: $(x \ @= \ 0) = (hcmod \ x \ @= \ 0)$
⟨*proof*⟩

**lemma** *approx-minus-zero-cancel-iff* [*simp*]: $(-x \ @= \ 0) = (x \ @= \ 0)$
⟨*proof*⟩

**lemma** *Infinitesimal-hcmod-add-diff*:
$u \ @= \ 0 ==> hcmod(x + u) - hcmod \ x \in Infinitesimal$
⟨*proof*⟩

**lemma** *approx-hcmod-add-hcmod*: $u \ @= \ 0 ==> hcmod(x + u) \ @= \ hcmod \ x$
⟨*proof*⟩

## 36.5 Zero is the Only Infinitesimal Complex Number

**lemma** *Infinitesimal-less-SComplex*:
$[| \ x \in SComplex; \ y \in Infinitesimal; \ 0 < hcmod \ x \ |] ==> hcmod \ y < hcmod \ x$
⟨*proof*⟩

**lemma** *SComplex-Int-Infinitesimal-zero*: $SComplex \ Int \ Infinitesimal = \{0\}$
⟨*proof*⟩

**lemma** *SComplex-Infinitesimal-zero*:
$[| \ x \in SComplex; \ x \in Infinitesimal|] ==> x = 0$
⟨*proof*⟩

**lemma** *SComplex-HFinite-diff-Infinitesimal*:

$[| \ x \in SComplex; \ x \neq 0 \ |] ==> x \in HFinite - Infinitesimal$
⟨proof⟩

**lemma** *hcomplex-of-complex-HFinite-diff-Infinitesimal*:
$hcomplex\text{-}of\text{-}complex \ x \neq 0$
$==> hcomplex\text{-}of\text{-}complex \ x \in HFinite - Infinitesimal$
⟨proof⟩

**lemma** *number-of-not-Infinitesimal* [*simp*]:
$number\text{-}of \ w \neq (0::hcomplex) ==> (number\text{-}of \ w::hcomplex) \notin Infinitesimal$
⟨proof⟩

**lemma** *approx-SComplex-not-zero*:
$[| \ y \in SComplex; \ x \ @= \ y; \ y \neq 0 \ |] ==> x \neq 0$
⟨proof⟩

**lemma** *SComplex-approx-iff*:
$[|x \in SComplex; \ y \in SComplex|] ==> (x \ @= \ y) = (x = y)$
⟨proof⟩

**lemma** *number-of-Infinitesimal-iff* [*simp*]:
$((number\text{-}of \ w :: hcomplex) \in Infinitesimal) =$
$(number\text{-}of \ w = (0::hcomplex))$
⟨proof⟩

**lemma** *approx-unique-complex*:
$[| \ r \in SComplex; \ s \in SComplex; \ r \ @= \ x; \ s \ @= \ x|] ==> r = s$
⟨proof⟩

## 36.6 Properties of *hRe*, *hIm* and *HComplex*

**lemma** *abs-Re-le-cmod*: $|Re \ x| \leq cmod \ x$
⟨proof⟩

**lemma** *abs-Im-le-cmod*: $|Im \ x| \leq cmod \ x$
⟨proof⟩

**lemma** *abs-hRe-le-hcmod*: $\bigwedge x. \ |hRe \ x| \leq hcmod \ x$
⟨proof⟩

**lemma** *abs-hIm-le-hcmod*: $\bigwedge x. \ |hIm \ x| \leq hcmod \ x$
⟨proof⟩

**lemma** *Infinitesimal-hRe*: $x \in Infinitesimal \Longrightarrow hRe \ x \in Infinitesimal$
⟨proof⟩

**lemma** *Infinitesimal-hIm*: $x \in Infinitesimal \Longrightarrow hIm \ x \in Infinitesimal$
⟨proof⟩

**lemma** *real-sqrt-lessI*: $[\![0 < u;\ x < u^2]\!] \Longrightarrow sqrt\ x < u$

$\langle proof \rangle$

**lemma** *hypreal-sqrt-lessI*:
$\bigwedge x\ u.\ [\![0 < u;\ x < u^2]\!] \Longrightarrow (\ {*}f{*}\ sqrt)\ x < u$
$\langle proof \rangle$

**lemma** *hypreal-sqrt-ge-zero*: $\bigwedge x.\ 0 \leq x \Longrightarrow 0 \leq (\ {*}f{*}\ sqrt)\ x$
$\langle proof \rangle$

**lemma** *Infinitesimal-sqrt*:
$[\![x \in Infinitesimal;\ 0 \leq x]\!] \Longrightarrow (\ {*}f{*}\ sqrt)\ x \in Infinitesimal$
$\langle proof \rangle$

**lemma** *Infinitesimal-HComplex*:
$[\![x \in Infinitesimal;\ y \in Infinitesimal]\!] \Longrightarrow HComplex\ x\ y \in Infinitesimal$
$\langle proof \rangle$

**lemma** *hcomplex-Infinitesimal-iff*:
$(x \in Infinitesimal) = (hRe\ x \in Infinitesimal \wedge hIm\ x \in Infinitesimal)$
$\langle proof \rangle$

**lemma** *hRe-diff* [*simp*]: $\bigwedge x\ y.\ hRe\ (x - y) = hRe\ x - hRe\ y$
$\langle proof \rangle$

**lemma** *hIm-diff* [*simp*]: $\bigwedge x\ y.\ hIm\ (x - y) = hIm\ x - hIm\ y$
$\langle proof \rangle$

**lemma** *approx-hRe*: $x \approx y \Longrightarrow hRe\ x \approx hRe\ y$
$\langle proof \rangle$

**lemma** *approx-hIm*: $x \approx y \Longrightarrow hIm\ x \approx hIm\ y$
$\langle proof \rangle$

**lemma** *approx-HComplex*:
$[\![a \approx b;\ c \approx d]\!] \Longrightarrow HComplex\ a\ c \approx HComplex\ b\ d$
$\langle proof \rangle$

**lemma** *hcomplex-approx-iff*:
$(x \approx y) = (hRe\ x \approx hRe\ y \wedge hIm\ x \approx hIm\ y)$
$\langle proof \rangle$

**lemma** *HFinite-hRe*: $x \in HFinite \Longrightarrow hRe\ x \in HFinite$
$\langle proof \rangle$

**lemma** *HFinite-hIm*: $x \in HFinite \Longrightarrow hIm\ x \in HFinite$
$\langle proof \rangle$

**lemma** *HFinite-HComplex*:
$[\![x \in HFinite;\ y \in HFinite]\!] \Longrightarrow HComplex\ x\ y \in HFinite$
⟨*proof*⟩

**lemma** *hcomplex-HFinite-iff*:
$(x \in HFinite) = (hRe\ x \in HFinite \wedge hIm\ x \in HFinite)$
⟨*proof*⟩

**lemma** *hcomplex-HInfinite-iff*:
$(x \in HInfinite) = (hRe\ x \in HInfinite \vee hIm\ x \in HInfinite)$
⟨*proof*⟩

**lemma** *hcomplex-of-hypreal-approx-iff* [*simp*]:
(*hcomplex-of-hypreal x* @= *hcomplex-of-hypreal z*) = (*x* @= *z*)
⟨*proof*⟩

**lemma** *Standard-HComplex*:
$[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow HComplex\ x\ y \in Standard$
⟨*proof*⟩

**lemma** *stc-part-Ex*: *x*:*HFinite* ==> $\exists\, t \in SComplex.$ *x* @= *t*
⟨*proof*⟩

**lemma** *stc-part-Ex1*: *x*:*HFinite* ==> *EX*! *t*. $t \in SComplex$ & *x* @= *t*
⟨*proof*⟩

**lemmas** *hcomplex-of-complex-approx-inverse* =
*hcomplex-of-complex-HFinite-diff-Infinitesimal* [*THEN* [*2*] *approx-inverse*]

## 36.7 Theorems About Monads

**lemma** *monad-zero-hcmod-iff*: $(x \in monad\ 0) = (hcmod\ x{:}monad\ 0)$
⟨*proof*⟩

## 36.8 Theorems About Standard Part

**lemma** *stc-approx-self*: $x \in HFinite$ ==> *stc x* @= *x*
⟨*proof*⟩

**lemma** *stc-SComplex*: $x \in HFinite$ ==> $stc\ x \in SComplex$
⟨*proof*⟩

**lemma** *stc-HFinite*: $x \in HFinite$ ==> $stc\ x \in HFinite$
⟨*proof*⟩

**lemma** *stc-unique*: $[\![y \in SComplex;\ y \approx x]\!] \Longrightarrow stc\ x = y$
⟨*proof*⟩

**lemma** *stc-SComplex-eq* [*simp*]: $x \in SComplex$ ==> $stc\ x = x$

⟨*proof*⟩

**lemma** *stc-hcomplex-of-complex*:
*stc (hcomplex-of-complex x) = hcomplex-of-complex x*
⟨*proof*⟩

**lemma** *stc-eq-approx*:
*[| x ∈ HFinite; y ∈ HFinite; stc x = stc y |] ==> x @= y*
⟨*proof*⟩

**lemma** *approx-stc-eq*:
*[| x ∈ HFinite; y ∈ HFinite; x @= y |] ==> stc x = stc y*
⟨*proof*⟩

**lemma** *stc-eq-approx-iff*:
*[| x ∈ HFinite; y ∈ HFinite|] ==> (x @= y) = (stc x = stc y)*
⟨*proof*⟩

**lemma** *stc-Infinitesimal-add-SComplex*:
*[| x ∈ SComplex; e ∈ Infinitesimal |] ==> stc(x + e) = x*
⟨*proof*⟩

**lemma** *stc-Infinitesimal-add-SComplex2*:
*[| x ∈ SComplex; e ∈ Infinitesimal |] ==> stc(e + x) = x*
⟨*proof*⟩

**lemma** *HFinite-stc-Infinitesimal-add*:
*x ∈ HFinite ==> ∃ e ∈ Infinitesimal. x = stc(x) + e*
⟨*proof*⟩

**lemma** *stc-add*:
*[| x ∈ HFinite; y ∈ HFinite |] ==> stc (x + y) = stc(x) + stc(y)*
⟨*proof*⟩

**lemma** *stc-number-of* [*simp*]: *stc (number-of w) = number-of w*
⟨*proof*⟩

**lemma** *stc-zero* [*simp*]: *stc 0 = 0*
⟨*proof*⟩

**lemma** *stc-one* [*simp*]: *stc 1 = 1*
⟨*proof*⟩

**lemma** *stc-minus*: *y ∈ HFinite ==> stc(−y) = −stc(y)*
⟨*proof*⟩

**lemma** *stc-diff*:
*[| x ∈ HFinite; y ∈ HFinite |] ==> stc (x−y) = stc(x) − stc(y)*
⟨*proof*⟩

**lemma** *stc-mult*:
  $[| \ x \in HFinite; \ y \in HFinite \ |]$
    $==> stc \ (x * y) = stc(x) * stc(y)$
⟨*proof*⟩

**lemma** *stc-Infinitesimal*: $x \in Infinitesimal ==> stc \ x = 0$
⟨*proof*⟩

**lemma** *stc-not-Infinitesimal*: $stc(x) \neq 0 ==> x \notin Infinitesimal$
⟨*proof*⟩

**lemma** *stc-inverse*:
  $[| \ x \in HFinite; \ stc \ x \neq 0 \ |]$
  $==> stc(inverse \ x) = inverse \ (stc \ x)$
⟨*proof*⟩

**lemma** *stc-divide* [*simp*]:
  $[| \ x \in HFinite; \ y \in HFinite; \ stc \ y \neq 0 \ |]$
  $==> stc(x/y) = (stc \ x) \ / \ (stc \ y)$
⟨*proof*⟩

**lemma** *stc-idempotent* [*simp*]: $x \in HFinite ==> stc(stc(x)) = stc(x)$
⟨*proof*⟩

**lemma** *HFinite-HFinite-hcomplex-of-hypreal*:
  $z \in HFinite ==> hcomplex\text{-}of\text{-}hypreal \ z \in HFinite$
⟨*proof*⟩

**lemma** *SComplex-SReal-hcomplex-of-hypreal*:
  $x \in Reals ==> \ hcomplex\text{-}of\text{-}hypreal \ x \in SComplex$
⟨*proof*⟩

**lemma** *stc-hcomplex-of-hypreal*:
 $z \in HFinite ==> stc(hcomplex\text{-}of\text{-}hypreal \ z) = hcomplex\text{-}of\text{-}hypreal \ (st \ z)$
⟨*proof*⟩

**lemma** *Infinitesimal-hcnj-iff* [*simp*]:
  $(hcnj \ z \in Infinitesimal) = (z \in Infinitesimal)$
⟨*proof*⟩

**lemma** *Infinitesimal-hcomplex-of-hypreal-epsilon* [*simp*]:
  $hcomplex\text{-}of\text{-}hypreal \ epsilon \in Infinitesimal$
⟨*proof*⟩

**end**

# 37 CStar: Star-transforms in NSA, Extending Sets of Complex Numbers and Complex Functions

**theory** *CStar*
**imports** *NSCA*
**begin**

## 37.1 Properties of the *-Transform Applied to Sets of Reals

**lemma** *STARC-hcomplex-of-complex-Int*:
$*s*$ *X Int SComplex* = *hcomplex-of-complex* ‘ *X*
⟨*proof*⟩

**lemma** *lemma-not-hcomplexA*:
$x \notin$ *hcomplex-of-complex* ‘ *A* ==> $\forall\, y \in A.\ x \neq$ *hcomplex-of-complex y*
⟨*proof*⟩

## 37.2 Theorems about Nonstandard Extensions of Functions

**lemma** *starfunC-hcpow*: !!*Z*. ( $*f*$ (%*z*. *z* ˆ *n*)) *Z* = *Z pow hypnat-of-nat n*
⟨*proof*⟩

**lemma** *starfunCR-cmod*: $*f*$ *cmod* = *hcmod*
⟨*proof*⟩

## 37.3 Internal Functions - Some Redundancy With *f* Now

**lemma** *starfun-Re*: ( $*f*$ ($\lambda x$. *Re* (*f x*))) = ($\lambda x$. *hRe* (( $*f*$ *f*) *x*))
⟨*proof*⟩

**lemma** *starfun-Im*: ( $*f*$ ($\lambda x$. *Im* (*f x*))) = ($\lambda x$. *hIm* (( $*f*$ *f*) *x*))
⟨*proof*⟩

**lemma** *starfunC-eq-Re-Im-iff*:
(( $*f*$ *f*) *x* = *z*) = ((( $*f*$ (%*x*. *Re*(*f x*))) *x* = *hRe* (*z*)) &
(( $*f*$ (%*x*. *Im*(*f x*))) *x* = *hIm* (*z*)))
⟨*proof*⟩

**lemma** *starfunC-approx-Re-Im-iff*:
(( $*f*$ *f*) *x* @= *z*) = ((( $*f*$ (%*x*. *Re*(*f x*))) *x* @= *hRe* (*z*)) &
(( $*f*$ (%*x*. *Im*(*f x*))) *x* @= *hIm* (*z*)))
⟨*proof*⟩

**end**

# 38 CLim: Limits, Continuity and Differentiation for Complex Functions

**theory** *CLim*
**imports** *CStar*
**begin**

**declare** *hypreal-epsilon-not-zero* [*simp*]

**lemma** *lemma-complex-mult-inverse-squared* [*simp*]:
$x \neq (0::complex) \Longrightarrow (x * inverse(x)\ \hat{}\ 2) = inverse\ x$
⟨*proof*⟩

Changing the quantified variable. Install earlier?

**lemma** *all-shift*: $(\forall x::'a::comm\text{-}ring\text{-}1.\ P\ x) = (\forall x.\ P\ (x-a))$
⟨*proof*⟩

**lemma** *complex-add-minus-iff* [*simp*]: $(x + -\ a = (0::complex)) = (x=a)$
⟨*proof*⟩

**lemma** *complex-add-eq-0-iff* [*iff*]: $(x+y = (0::complex)) = (y = -x)$
⟨*proof*⟩

## 38.1 Limit of Complex to Complex Function

**lemma** *NSLIM-Re*: $f$ −− $a$ −−NS> $L$ ==> (%x. Re(f x)) −− $a$ −−NS> Re(L)
⟨*proof*⟩

**lemma** *NSLIM-Im*: $f$ −− $a$ −−NS> $L$ ==> (%x. Im(f x)) −− $a$ −−NS> Im(L)
⟨*proof*⟩

**lemma** *LIM-Re*: $f$ −− $a$ −−> $L$ ==> (%x. Re(f x)) −− $a$ −−> Re(L)
⟨*proof*⟩

**lemma** *LIM-Im*: $f$ −− $a$ −−> $L$ ==> (%x. Im(f x)) −− $a$ −−> Im(L)
⟨*proof*⟩

**lemma** *LIM-cnj*: $f$ −− $a$ −−> $L$ ==> (%x. cnj (f x)) −− $a$ −−> cnj L
⟨*proof*⟩

**lemma** *LIM-cnj-iff*: ((%x. cnj (f x)) −− $a$ −−> cnj L) = (f −− $a$ −−> L)
⟨*proof*⟩

**lemma** *starfun-norm*: $( *f* (\lambda x.\ norm\ (f\ x))) = (\lambda x.\ hnorm\ (( *f* f)\ x))$
⟨*proof*⟩

**lemma** *star-of-Re* [*simp*]: *star-of* (*Re x*) = *hRe* (*star-of x*)
⟨*proof*⟩

**lemma** *star-of-Im* [*simp*]: *star-of* (*Im x*) = *hIm* (*star-of x*)
⟨*proof*⟩

**lemma** *NSCLIM-NSCRLIM-iff*:
(*f* −− *x* −−*NS*> *L*) = ((%*y*. *cmod*(*f y* − *L*)) −− *x* −−*NS*> *0*)
⟨*proof*⟩

**lemma** *CLIM-CRLIM-iff*: (*f* −− *x* −−> *L*) = ((%*y*. *cmod*(*f y* − *L*)) −− *x* −−> *0*)
⟨*proof*⟩

**lemma** *NSCLIM-NSCRLIM-iff2*:
(*f* −− *x* −−*NS*> *L*) = ((%*y*. *cmod*(*f y* − *L*)) −− *x* −−*NS*> *0*)
⟨*proof*⟩

**lemma** *NSLIM-NSCRLIM-Re-Im-iff*:
(*f* −− *a* −−*NS*> *L*) = ((%*x*. *Re*(*f x*)) −− *a* −−*NS*> *Re*(*L*) &
(%*x*. *Im*(*f x*)) −− *a* −−*NS*> *Im*(*L*))
⟨*proof*⟩

**lemma** *LIM-CRLIM-Re-Im-iff*:
(*f* −− *a* −−> *L*) = ((%*x*. *Re*(*f x*)) −− *a* −−> *Re*(*L*) &
(%*x*. *Im*(*f x*)) −− *a* −−> *Im*(*L*))
⟨*proof*⟩

## 38.2 Continuity

**lemma** *NSLIM-isContc-iff*:
(*f* −− *a* −−*NS*> *f a*) = ((%*h*. *f*(*a* + *h*)) −− *0* −−*NS*> *f a*)
⟨*proof*⟩

## 38.3 Functions from Complex to Reals

**lemma** *isNSContCR-cmod* [*simp*]: *isNSCont cmod* (*a*)
⟨*proof*⟩

**lemma** *isContCR-cmod* [*simp*]: *isCont cmod* (*a*)
⟨*proof*⟩

**lemma** *isCont-Re*: *isCont f a* ==> *isCont* (%*x*. *Re* (*f x*)) *a*
⟨*proof*⟩

**lemma** *isCont-Im*: *isCont f a* ==> *isCont* (%*x*. *Im* (*f x*)) *a*
⟨*proof*⟩

## 38.4 Differentiation of Natural Number Powers

**lemma** *CDERIV-pow* [*simp*]:
$DERIV\ (\%x.\ x\ \hat{}\ n)\ x :> (complex\text{-}of\text{-}real\ (real\ n)) * (x\ \hat{}\ (n - Suc\ 0))$
⟨*proof*⟩

Nonstandard version

**lemma** *NSCDERIV-pow*:
$NSDERIV\ (\%x.\ x\ \hat{}\ n)\ x :> complex\text{-}of\text{-}real\ (real\ n) * (x\ \hat{}\ (n - 1))$
⟨*proof*⟩

Can't relax the premise $x \neq (0{::}'a)$: it isn't continuous at zero

**lemma** *NSCDERIV-inverse*:
$(x{::}complex) \neq 0 ==> NSDERIV\ (\%x.\ inverse(x))\ x :> (-\ (inverse\ x\ \hat{}\ 2))$
⟨*proof*⟩

**lemma** *CDERIV-inverse*:
$(x{::}complex) \neq 0 ==> DERIV\ (\%x.\ inverse(x))\ x :> (-(inverse\ x\ \hat{}\ 2))$
⟨*proof*⟩

## 38.5 Derivative of Reciprocals (Function *inverse*)

**lemma** *CDERIV-inverse-fun*:
$[|\ DERIV\ f\ x :> d;\ f(x) \neq (0{::}complex)\ |]$
$==> DERIV\ (\%x.\ inverse(f\ x))\ x :> (-\ (d * inverse(f(x)\ \hat{}\ 2)))$
⟨*proof*⟩

**lemma** *NSCDERIV-inverse-fun*:
$[|\ NSDERIV\ f\ x :> d;\ f(x) \neq (0{::}complex)\ |]$
$==> NSDERIV\ (\%x.\ inverse(f\ x))\ x :> (-\ (d * inverse(f(x)\ \hat{}\ 2)))$
⟨*proof*⟩

## 38.6 Derivative of Quotient

**lemma** *CDERIV-quotient*:
$[|\ DERIV\ f\ x :> d;\ DERIV\ g\ x :> e;\ g(x) \neq (0{::}complex)\ |]$
$==> DERIV\ (\%y.\ f(y)\ /\ (g\ y))\ x :> (d{*}g(x) - (e{*}f(x)))\ /\ (g(x)\ \hat{}\ 2)$
⟨*proof*⟩

**lemma** *NSCDERIV-quotient*:
$[|\ NSDERIV\ f\ x :> d;\ NSDERIV\ g\ x :> e;\ g(x) \neq (0{::}complex)\ |]$
$==> NSDERIV\ (\%y.\ f(y)\ /\ (g\ y))\ x :> (d{*}g(x) - (e{*}f(x)))\ /\ (g(x)\ \hat{}\ 2)$
⟨*proof*⟩

## 38.7 Caratheodory Formulation of Derivative at a Point: Standard Proof

**lemma** *CARAT-CDERIVD*:
$(\forall z.\ f\ z - f\ x = g\ z * (z - x))\ \&\ isNSCont\ g\ x\ \&\ g\ x = l$
$==> NSDERIV\ f\ x :> l$

⟨*proof*⟩

**end**

# 39 Ln: Properties of ln

**theory** *Ln*
**imports** *Transcendental*
**begin**

**lemma** *exp-first-two-terms*: *exp x = 1 + x + suminf (%n. inverse(real (fact (n+2))) * (x ^ (n+2)))*
⟨*proof*⟩

**lemma** *exp-tail-after-first-two-terms-summable*:
*summable (%n. inverse(real (fact (n+2))) * (x ^ (n+2)))*
⟨*proof*⟩

**lemma** *aux1*: **assumes** *a*: *0 <= x* **and** *b*: *x <= 1*
**shows** *inverse (real (fact (n + 2))) * x ^ (n + 2) <= (x^2/2) * ((1/2)^n)*
⟨*proof*⟩

**lemma** *aux2*: *(%n. (x::real) ^ 2 / 2 * (1 / 2) ^ n) sums x^2*
⟨*proof*⟩

**lemma** *exp-bound*: *0 <= (x::real) ==> x <= 1 ==> exp x <= 1 + x + x^2*
⟨*proof*⟩

**lemma** *aux4*: *0 <= (x::real) ==> x <= 1 ==> exp (x − x^2) <= 1 + x*
⟨*proof*⟩

**lemma** *ln-one-plus-pos-lower-bound*: *0 <= x ==> x <= 1 ==>*
*x − x^2 <= ln (1 + x)*
⟨*proof*⟩

**lemma** *ln-one-minus-pos-upper-bound*: *0 <= x ==> x < 1 ==> ln (1 − x) <= − x*
⟨*proof*⟩

**lemma** *aux5*: *x < 1 ==> ln(1 − x) = − ln(1 + x / (1 − x))*
⟨*proof*⟩

**lemma** *ln-one-minus-pos-lower-bound*: *0 <= x ==> x <= (1 / 2) ==>*
*− x − 2 * x^2 <= ln (1 − x)*
⟨*proof*⟩

**lemma** *exp-ge-add-one-self* [*simp*]: *1 + (x::real) <= exp x*
⟨*proof*⟩

**lemma** *ln-add-one-self-le-self2*: *−1 < x ==> ln(1 + x) <= x*
⟨*proof*⟩

**lemma** *abs-ln-one-plus-x-minus-x-bound-nonneg*:
*0 <= x ==> x <= 1 ==> abs(ln (1 + x) − x) <= x^2*
⟨*proof*⟩

**lemma** *abs-ln-one-plus-x-minus-x-bound-nonpos*:
*−(1 / 2) <= x ==> x <= 0 ==> abs(ln (1 + x) − x) <= 2 * x^2*
⟨*proof*⟩

**lemma** *abs-ln-one-plus-x-minus-x-bound*:
*abs x <= 1 / 2 ==> abs(ln (1 + x) − x) <= 2 * x^2*
⟨*proof*⟩

**lemma** *DERIV-ln*: *0 < x ==> DERIV ln x :> 1 / x*
⟨*proof*⟩

**lemma** *ln-x-over-x-mono*: *exp 1 <= x ==> x <= y ==> (ln y / y) <= (ln x / x)*
⟨*proof*⟩

**end**

# 40 Poly: Univariate Real Polynomials

**theory** *Poly*
**imports** *Deriv*
**begin**

Application of polynomial as a real function.

**consts** *poly* :: *real list => real => real*
**primrec**
*poly-Nil*: *poly [] x = 0*
*poly-Cons*: *poly (h#t) x = h + x * poly t x*

## 40.1 Arithmetic Operations on Polynomials

addition

**consts** *padd* :: [*real list*, *real list*] => *real list* (**infixl** +++ *65*)
**primrec**
*padd-Nil*: [] +++ *l2* = *l2*
*padd-Cons*: (*h#t*) +++ *l2* = (*if l2* = [] *then h#t*
*else* (*h* + *hd l2*)#(*t* +++ *tl l2*))

Multiplication by a constant

**consts** *cmult* :: [*real*, *real list*] => *real list* (**infixl** %∗ *70*)
**primrec**
*cmult-Nil*: *c* %∗ [] = []
*cmult-Cons*: *c* %∗ (*h*#*t*) = (*c* ∗ *h*)#(*c* %∗ *t*)

Multiplication by a polynomial

**consts** *pmult* :: [*real list*, *real list*] => *real list* (**infixl** ∗∗∗ *70*)
**primrec**
*pmult-Nil*: [] ∗∗∗ *l2* = []
*pmult-Cons*: (*h*#*t*) ∗∗∗ *l2* = (*if t* = [] *then h* %∗ *l2*
*else* (*h* %∗ *l2*) +++ ((*0*) # (*t* ∗∗∗ *l2*)))

Repeated multiplication by a polynomial

**consts** *mulexp* :: [*nat*, *real list*, *real list*] => *real list*
**primrec**
*mulexp-zero*: *mulexp 0 p q* = *q*
*mulexp-Suc*: *mulexp* (*Suc n*) *p q* = *p* ∗∗∗ *mulexp n p q*

Exponential

**consts** *pexp* :: [*real list*, *nat*] => *real list* (**infixl** %ˆ *80*)
**primrec**
*pexp-0*: *p* %ˆ *0* = [*1*]
*pexp-Suc*: *p* %ˆ (*Suc n*) = *p* ∗∗∗ (*p* %ˆ *n*)

Quotient related value of dividing a polynomial by x + a

**consts** *pquot* :: [*real list*, *real*] => *real list*
**primrec**
*pquot-Nil*: *pquot* [] *a*= []
*pquot-Cons*: *pquot* (*h*#*t*) *a* = (*if t* = [] *then* [*h*]
*else* (*inverse*(*a*) ∗ (*h* − *hd*( *pquot t a*)))#(*pquot t a*))

Differentiation of polynomials (needs an auxiliary function).

**consts** *pderiv-aux* :: *nat* => *real list* => *real list*
**primrec**
*pderiv-aux-Nil*: *pderiv-aux n* [] = []
*pderiv-aux-Cons*: *pderiv-aux n* (*h*#*t*) =
(*real n* ∗ *h*)#(*pderiv-aux* (*Suc n*) *t*)

normalization of polynomials (remove extra 0 coeff)

**consts** *pnormalize* :: *real list* => *real list*
**primrec**
*pnormalize-Nil*: *pnormalize* [] = []
*pnormalize-Cons*: *pnormalize* (*h*#*p*) = (*if* ( (*pnormalize p*) = [])
*then* (*if* (*h* = *0*) *then* [] *else* [*h*])
*else* (*h*#(*pnormalize p*)))

**definition** *pnormal p* = ((*pnormalize p* = *p*) ∧ *p* ≠ [])
**definition** *nonconstant p* = (*pnormal p* ∧ (∀ *x*. *p* ≠ [*x*]))

Other definitions

**definition**
*poly-minus* :: *real list* => *real list* (−− - [*80*] *80*) **where**
−− *p* = (− *1*) %∗ *p*

**definition**
*pderiv* :: *real list* => *real list* **where**
*pderiv p* = (*if p* = [] *then* [] *else pderiv-aux 1* (*tl p*))

**definition**
*divides* :: [*real list*,*real list*] => *bool* (**infixl** *divides 70*) **where**
*p1 divides p2* = (∃ *q*. *poly p2* = *poly*(*p1* ∗∗∗ *q*))

**definition**
*order* :: *real* => *real list* => *nat* **where**
— order of a polynomial
*order a p* = (*SOME n*. ([−*a*, *1*] %ˆ *n*) *divides p* &
~ (([−*a*, *1*] %ˆ (*Suc n*)) *divides p*))

**definition**
*degree* :: *real list* => *nat* **where**
— degree of a polynomial
*degree p* = *length* (*pnormalize p*) − *1*

**definition**
*rsquarefree* :: *real list* => *bool* **where**
— squarefree polynomials — NB with respect to real roots only.
*rsquarefree p* = (*poly p* ≠ *poly* [] &
(∀ *a*. (*order a p* = *0*) | (*order a p* = *1*)))

**lemma** *padd-Nil2*: *p* +++ [] = *p*
⟨*proof*⟩
**declare** *padd-Nil2* [*simp*]

**lemma** *padd-Cons-Cons*: (*h1* # *p1*) +++ (*h2* # *p2*) = (*h1* + *h2*) # (*p1* +++ *p2*)
⟨*proof*⟩

**lemma** *pminus-Nil*: −− [] = []
⟨*proof*⟩
**declare** *pminus-Nil* [*simp*]

**lemma** *pmult-singleton*: [*h1*] ∗∗∗ *p1* = *h1* %∗ *p1*
⟨*proof*⟩

**lemma** *poly-ident-mult*: *1* %∗ *t* = *t*
⟨*proof*⟩

**declare** *poly-ident-mult* [*simp*]

**lemma** *poly-simple-add-Cons*: $[a]$ +++ $((0)\#t) = (a\#t)$
⟨*proof*⟩
**declare** *poly-simple-add-Cons* [*simp*]

Handy general properties

**lemma** *padd-commut*: $b$ +++ $a = a$ +++ $b$
⟨*proof*⟩

**lemma** *padd-assoc* [*rule-format*]: $\forall\, b\ c.\ (a$ +++ $b)$ +++ $c = a$ +++ $(b$ +++ $c)$
⟨*proof*⟩

**lemma** *poly-cmult-distr* [*rule-format*]:
  $\forall\, q.\ a$ %* ( $p$ +++ $q) = (a$ %* $p$ +++ $a$ %* $q)$
⟨*proof*⟩

**lemma** *pmult-by-x*: $[0,\ 1]$ *** $t = ((0)\#t)$
⟨*proof*⟩
**declare** *pmult-by-x* [*simp*]

properties of evaluation of polynomials.

**lemma** *poly-add*: *poly* $(p1$ +++ $p2)\ x =$ *poly* $p1\ x +$ *poly* $p2\ x$
⟨*proof*⟩

**lemma** *poly-cmult*: *poly* $(c$ %* $p)\ x = c *$ *poly* $p\ x$
⟨*proof*⟩

**lemma** *poly-minus*: *poly* $(--\ p)\ x = -$ (*poly* $p\ x)$
⟨*proof*⟩

**lemma** *poly-mult*: *poly* $(p1$ *** $p2)\ x =$ *poly* $p1\ x *$ *poly* $p2\ x$
⟨*proof*⟩

**lemma** *poly-exp*: *poly* $(p$ %^ $n)\ x =$ (*poly* $p\ x)$ ^ $n$
⟨*proof*⟩

More Polynomial Evaluation Lemmas

**lemma** *poly-add-rzero*: *poly* $(a$ +++ $[])\ x =$ *poly* $a\ x$
⟨*proof*⟩
**declare** *poly-add-rzero* [*simp*]

**lemma** *poly-mult-assoc*: *poly* $((a$ *** $b)$ *** $c)\ x =$ *poly* $(a$ *** $(b$ *** $c))\ x$
⟨*proof*⟩

**lemma** *poly-mult-Nil2*: *poly* $(p$ *** $[])\ x = 0$
⟨*proof*⟩
**declare** *poly-mult-Nil2* [*simp*]

**lemma** *poly-exp-add*: *poly* $(p\ \%\hat{}\ (n + d))\ x = poly(\ p\ \%\hat{}\ n *** p\ \%\hat{}\ d)\ x$
⟨*proof*⟩

The derivative

**lemma** *pderiv-Nil*: *pderiv* [] = []

⟨*proof*⟩
**declare** *pderiv-Nil* [*simp*]

**lemma** *pderiv-singleton*: *pderiv* [*c*] = []
⟨*proof*⟩
**declare** *pderiv-singleton* [*simp*]

**lemma** *pderiv-Cons*: *pderiv* (*h*#*t*) = *pderiv-aux 1 t*
⟨*proof*⟩

**lemma** *DERIV-cmult2*: *DERIV f x* :> *D* ==> *DERIV* (%*x*. (*f x*) ∗ *c* :: *real*) *x* :> *D* ∗ *c*
⟨*proof*⟩

**lemma** *DERIV-pow2*: *DERIV* (%*x*. *x* ˆ *Suc n*) *x* :> *real* (*Suc n*) ∗ (*x* ˆ *n*)
⟨*proof*⟩
**declare** *DERIV-pow2* [*simp*] *DERIV-pow* [*simp*]

**lemma** *lemma-DERIV-poly1*: ∀ *n*. *DERIV* (%*x*. (*x* ˆ (*Suc n*) ∗ *poly p x*)) *x* :>
    *x* ˆ *n* ∗ *poly* (*pderiv-aux* (*Suc n*) *p*) *x*
⟨*proof*⟩

**lemma** *lemma-DERIV-poly*: *DERIV* (%*x*. (*x* ˆ (*Suc n*) ∗ *poly p x*)) *x* :>
    *x* ˆ *n* ∗ *poly* (*pderiv-aux* (*Suc n*) *p*) *x*
⟨*proof*⟩

**lemma** *DERIV-add-const*: *DERIV f x* :> *D* ==> *DERIV* (%*x*. *a* + *f x* :: *real*) *x* :> *D*
⟨*proof*⟩

**lemma** *poly-DERIV*: *DERIV* (%*x*. *poly p x*) *x* :> *poly* (*pderiv p*) *x*
⟨*proof*⟩
**declare** *poly-DERIV* [*simp*]

Consequences of the derivative theorem above

**lemma** *poly-differentiable*: (%*x*. *poly p x*) *differentiable x*

⟨*proof*⟩
**declare** *poly-differentiable* [*simp*]

**lemma** *poly-isCont*: *isCont* (%*x*. *poly p x*) *x*
⟨*proof*⟩
**declare** *poly-isCont* [*simp*]

**lemma** *poly-IVT-pos*: $[| \ a < b; \ poly \ p \ a < 0; \ 0 < poly \ p \ b \ |]$
$==> \exists x. \ a < x \ \& \ x < b \ \& \ (poly \ p \ x = 0)$
⟨*proof*⟩

**lemma** *poly-IVT-neg*: $[| \ a < b; \ 0 < poly \ p \ a; \ poly \ p \ b < 0 \ |]$
$==> \exists x. \ a < x \ \& \ x < b \ \& \ (poly \ p \ x = 0)$
⟨*proof*⟩

**lemma** *poly-MVT*: $a < b ==>$
$\exists x. \ a < x \ \& \ x < b \ \& \ (poly \ p \ b - poly \ p \ a = (b - a) * poly \ (pderiv \ p) \ x)$
⟨*proof*⟩

Lemmas for Derivatives

**lemma** *lemma-poly-pderiv-aux-add*: $\forall p2 \ n. \ poly \ (pderiv\text{-}aux \ n \ (p1 \ {+}{+}{+} \ p2)) \ x =$
$poly \ (pderiv\text{-}aux \ n \ p1 \ {+}{+}{+} \ pderiv\text{-}aux \ n \ p2) \ x$
⟨*proof*⟩

**lemma** *poly-pderiv-aux-add*: $poly \ (pderiv\text{-}aux \ n \ (p1 \ {+}{+}{+} \ p2)) \ x =$
$poly \ (pderiv\text{-}aux \ n \ p1 \ {+}{+}{+} \ pderiv\text{-}aux \ n \ p2) \ x$
⟨*proof*⟩

**lemma** *lemma-poly-pderiv-aux-cmult*: $\forall n. \ poly \ (pderiv\text{-}aux \ n \ (c \ \%{*} \ p)) \ x = poly \ (c \ \%{*} \ pderiv\text{-}aux \ n \ p) \ x$
⟨*proof*⟩

**lemma** *poly-pderiv-aux-cmult*: $poly \ (pderiv\text{-}aux \ n \ (c \ \%{*} \ p)) \ x = poly \ (c \ \%{*} \ pderiv\text{-}aux \ n \ p) \ x$
⟨*proof*⟩

**lemma** *poly-pderiv-aux-minus*:
$poly \ (pderiv\text{-}aux \ n \ (-- \ p)) \ x = poly \ (-- \ pderiv\text{-}aux \ n \ p) \ x$
⟨*proof*⟩

**lemma** *lemma-poly-pderiv-aux-mult1*: $\forall n. \ poly \ (pderiv\text{-}aux \ (Suc \ n) \ p) \ x = poly \ ((pderiv\text{-}aux \ n \ p) \ {+}{+}{+} \ p) \ x$
⟨*proof*⟩

**lemma** *lemma-poly-pderiv-aux-mult*: $poly \ (pderiv\text{-}aux \ (Suc \ n) \ p) \ x = poly \ ((pderiv\text{-}aux \ n \ p) \ {+}{+}{+} \ p) \ x$
⟨*proof*⟩

**lemma** *lemma-poly-pderiv-add*: $\forall q. \ poly \ (pderiv \ (p \ {+}{+}{+} \ q)) \ x = poly \ (pderiv \ p \ {+}{+}{+} \ pderiv \ q) \ x$
⟨*proof*⟩

**lemma** *poly-pderiv-add*: $poly \ (pderiv \ (p \ {+}{+}{+} \ q)) \ x = poly \ (pderiv \ p \ {+}{+}{+} \ pderiv \ q) \ x$
⟨*proof*⟩

**lemma** *poly-pderiv-cmult*: $poly\ (pderiv\ (c\ \%{*}\ p))\ x = poly\ (c\ \%{*}\ (pderiv\ p))\ x$
⟨*proof*⟩

**lemma** *poly-pderiv-minus*: $poly\ (pderiv\ (--p))\ x = poly\ (--(pderiv\ p))\ x$
⟨*proof*⟩

**lemma** *lemma-poly-mult-pderiv*:
$poly\ (pderiv\ (h\#t))\ x = poly\ ((0\ \#\ (pderiv\ t))\ {+}{+}{+}\ t)\ x$
⟨*proof*⟩

**lemma** *poly-pderiv-mult*: $\forall\, q.\ poly\ (pderiv\ (p\ {*}{*}{*}\ q))\ x =$
$poly\ (p\ {*}{*}{*}\ (pderiv\ q)\ {+}{+}{+}\ q\ {*}{*}{*}\ (pderiv\ p))\ x$
⟨*proof*⟩

**lemma** *poly-pderiv-exp*: $poly\ (pderiv\ (p\ \%\hat{}\ (Suc\ n)))\ x =$
$poly\ ((real\ (Suc\ n))\ \%{*}\ (p\ \%\hat{}\ n)\ {*}{*}{*}\ pderiv\ p)\ x$
⟨*proof*⟩

**lemma** *poly-pderiv-exp-prime*: $poly\ (pderiv\ ([-a,\ 1]\ \%\hat{}\ (Suc\ n)))\ x =$
$poly\ (real\ (Suc\ n)\ \%{*}\ ([-a,\ 1]\ \%\hat{}\ n))\ x$
⟨*proof*⟩

## 40.2 Key Property: if $f\ a = (0{::}'a)$ then $x - a$ divides $p\ x$

**lemma** *lemma-poly-linear-rem*: $\forall\, h.\ \exists\, q\ r.\ h\#t = [r]\ {+}{+}{+}\ [-a,\ 1]\ {*}{*}{*}\ q$
⟨*proof*⟩

**lemma** *poly-linear-rem*: $\exists\, q\ r.\ h\#t = [r]\ {+}{+}{+}\ [-a,\ 1]\ {*}{*}{*}\ q$
⟨*proof*⟩

**lemma** *poly-linear-divides*: $(poly\ p\ a = 0) = ((p = [])\ |\ (\exists\, q.\ p = [-a,\ 1]\ {*}{*}{*}\ q))$
⟨*proof*⟩

**lemma** *lemma-poly-length-mult*: $\forall\, h\ k\ a.\ length\ (k\ \%{*}\ p\ {+}{+}{+}\ \ (h\ \#\ (a\ \%{*}\ p))) = Suc\ (length\ p)$
⟨*proof*⟩
**declare** *lemma-poly-length-mult* [*simp*]

**lemma** *lemma-poly-length-mult2*: $\forall\, h\ k.\ length\ (k\ \%{*}\ p\ {+}{+}{+}\ \ (h\ \#\ p)) = Suc\ (length\ p)$
⟨*proof*⟩
**declare** *lemma-poly-length-mult2* [*simp*]

**lemma** *poly-length-mult*: $length([-a,1]\ {*}{*}{*}\ q) = Suc\ (length\ q)$
⟨*proof*⟩
**declare** *poly-length-mult* [*simp*]

## 40.3 Polynomial length

**lemma** *poly-cmult-length*: $length\ (a\ \%{*}\ p) = length\ p$
⟨*proof*⟩
**declare** *poly-cmult-length* [*simp*]

**lemma** *poly-add-length* [*rule-format*]:
$\forall\, p2.\ length\ (p1\ {+}{+}{+}\ p2) =$
$(if\ (length\ p1 < length\ p2)\ then\ length\ p2\ else\ length\ p1)$
⟨*proof*⟩

**lemma** *poly-root-mult-length*: $length([a,b]\ {*}{*}{*}\ p) = Suc\ (length\ p)$
⟨*proof*⟩
**declare** *poly-root-mult-length* [*simp*]

**lemma** *poly-mult-not-eq-poly-Nil*: $(poly\ (p\ {*}{*}{*}\ q)\ x \neq poly\ []\ x) =$
$(poly\ p\ x \neq poly\ []\ x\ \&\ poly\ q\ x \neq poly\ []\ x)$
⟨*proof*⟩
**declare** *poly-mult-not-eq-poly-Nil* [*simp*]

**lemma** *poly-mult-eq-zero-disj*: $(poly\ (p\ {*}{*}{*}\ q)\ x = 0) = (poly\ p\ x = 0\ |\ poly\ q\ x = 0)$
⟨*proof*⟩

Normalisation Properties

**lemma** *poly-normalized-nil*: $(pnormalize\ p = []) \dashrightarrow (poly\ p\ x = 0)$
⟨*proof*⟩

A nontrivial polynomial of degree n has no more than n roots

**lemma** *poly-roots-index-lemma* [*rule-format*]:
$\forall\, p\ x.\ poly\ p\ x \neq poly\ []\ x\ \&\ length\ p = n$
$\dashrightarrow (\exists\, i.\ \forall\, x.\ (poly\ p\ x = (0{::}real)) \dashrightarrow (\exists\, m.\ (m \leq n\ \&\ x = i\ m)))$
⟨*proof*⟩
**lemmas** *poly-roots-index-lemma2* = *conjI* [*THEN poly-roots-index-lemma*, *standard*]

**lemma** *poly-roots-index-length*: $poly\ p\ x \neq poly\ []\ x \Longrightarrow$
$\exists\, i.\ \forall\, x.\ (poly\ p\ x = 0) \dashrightarrow (\exists\, n.\ n \leq length\ p\ \&\ x = i\ n)$
⟨*proof*⟩

**lemma** *poly-roots-finite-lemma*: $poly\ p\ x \neq poly\ []\ x \Longrightarrow$
$\exists\, N\ i.\ \forall\, x.\ (poly\ p\ x = 0) \dashrightarrow (\exists\, n.\ (n{::}nat) < N\ \&\ x = i\ n)$
⟨*proof*⟩

**lemma** *real-finite-lemma* [*rule-format* (*no-asm*)]:
$\forall\, P.\ (\forall\, x.\ P\ x \dashrightarrow (\exists\, n.\ n < N\ \&\ x = (j{::}nat{=}{>}real)\ n))$
$\dashrightarrow (\exists\, a.\ \forall\, x.\ P\ x \dashrightarrow x < a)$
⟨*proof*⟩

**lemma** *poly-roots-finite*: $(poly\ p \neq poly\ []) =$
$(\exists N\ j.\ \forall x.\ poly\ p\ x = 0 \dashrightarrow (\exists n.\ (n::nat) < N\ \&\ x = j\ n))$
⟨*proof*⟩

Entirety and Cancellation for polynomials

**lemma** *poly-entire-lemma*: $[|\ poly\ p \neq poly\ []\ ;\ poly\ q \neq poly\ []\ |]$
$==>\ poly\ (p\ {*}{*}{*}\ q) \neq poly\ []$
⟨*proof*⟩

**lemma** *poly-entire*: $(poly\ (p\ {*}{*}{*}\ q) = poly\ []) = ((poly\ p = poly\ [])\ |\ (poly\ q = poly\ []))$
⟨*proof*⟩

**lemma** *poly-entire-neg*: $(poly\ (p\ {*}{*}{*}\ q) \neq poly\ []) = ((poly\ p \neq poly\ [])\ \&\ (poly\ q \neq poly\ []))$
⟨*proof*⟩

**lemma** *fun-eq*: $(f = g) = (\forall x.\ f\ x = g\ x)$
⟨*proof*⟩

**lemma** *poly-add-minus-zero-iff*: $(poly\ (p\ {+}{+}{+}\ {-}{-}\ q) = poly\ []) = (poly\ p = poly\ q)$
⟨*proof*⟩

**lemma** *poly-add-minus-mult-eq*: $poly\ (p\ {*}{*}{*}\ q\ {+}{+}{+}\ {-}{-}(p\ {*}{*}{*}\ r)) = poly\ (p\ {*}{*}{*}\ (q\ {+}{+}{+}\ {-}{-}\ r))$
⟨*proof*⟩

**lemma** *poly-mult-left-cancel*: $(poly\ (p\ {*}{*}{*}\ q) = poly\ (p\ {*}{*}{*}\ r)) = (poly\ p = poly\ []\ |\ poly\ q = poly\ r)$
⟨*proof*⟩

**lemma** *real-mult-zero-disj-iff*: $(x * y = 0) = (x = (0::real)\ |\ y = 0)$
⟨*proof*⟩

**lemma** *poly-exp-eq-zero*:
$(poly\ (p\ \%\hat{}\ n) = poly\ []) = (poly\ p = poly\ []\ \&\ n \neq 0)$
⟨*proof*⟩
**declare** *poly-exp-eq-zero* [*simp*]

**lemma** *poly-prime-eq-zero*: $poly\ [a,1] \neq poly\ []$
⟨*proof*⟩
**declare** *poly-prime-eq-zero* [*simp*]

**lemma** *poly-exp-prime-eq-zero*: $(poly\ ([a,\ 1]\ \%\hat{}\ n) \neq poly\ [])$
⟨*proof*⟩
**declare** *poly-exp-prime-eq-zero* [*simp*]

A more constructive notion of polynomials being trivial

**lemma** *poly-zero-lemma*: *poly* (*h* # *t*) = *poly* [] ==> *h* = *0* & *poly t* = *poly* []
⟨*proof*⟩

**lemma** *poly-zero*: (*poly p* = *poly* []) = *list-all* (%*c*. *c* = *0*) *p*
⟨*proof*⟩

**declare** *real-mult-zero-disj-iff* [*simp*]

**lemma** *pderiv-aux-iszero* [*rule-format*, *simp*]:
∀ *n*. *list-all* (%*c*. *c* = *0*) (*pderiv-aux* (*Suc n*) *p*) = *list-all* (%*c*. *c* = *0*) *p*
⟨*proof*⟩

**lemma** *pderiv-aux-iszero-num*: (*number-of n* :: *nat*) ≠ *0*
==> (*list-all* (%*c*. *c* = *0*) (*pderiv-aux* (*number-of n*) *p*) =
*list-all* (%*c*. *c* = *0*) *p*)
⟨*proof*⟩

**lemma** *pderiv-iszero* [*rule-format*]:
*poly* (*pderiv p*) = *poly* [] --> (∃ *h*. *poly p* = *poly* [*h*])
⟨*proof*⟩

**lemma** *pderiv-zero-obj*: *poly p* = *poly* [] --> (*poly* (*pderiv p*) = *poly* [])
⟨*proof*⟩

**lemma** *pderiv-zero*: *poly p* = *poly* [] ==> (*poly* (*pderiv p*) = *poly* [])
⟨*proof*⟩
**declare** *pderiv-zero* [*simp*]

**lemma** *poly-pderiv-welldef*: *poly p* = *poly q* ==> (*poly* (*pderiv p*) = *poly* (*pderiv q*))
⟨*proof*⟩

Basics of divisibility.

**lemma** *poly-primes*: ([*a*, *1*] *divides* (*p* *** *q*)) = ([*a*, *1*] *divides p* | [*a*, *1*] *divides q*)
⟨*proof*⟩

**lemma** *poly-divides-refl*: *p divides p*
⟨*proof*⟩
**declare** *poly-divides-refl* [*simp*]

**lemma** *poly-divides-trans*: [| *p divides q*; *q divides r* |] ==> *p divides r*
⟨*proof*⟩

**lemma** *poly-divides-exp*: *m* ≤ *n* ==> (*p* %^ *m*) *divides* (*p* %^ *n*)
⟨*proof*⟩

**lemma** *poly-exp-divides*: [| (*p* %^ *n*) *divides q*; *m*≤*n* |] ==> (*p* %^ *m*) *divides q*

⟨*proof*⟩

**lemma** *poly-divides-add*:
[| *p divides q*; *p divides r* |] ==> *p divides* (*q* +++ *r*)
⟨*proof*⟩

**lemma** *poly-divides-diff*:
[| *p divides q*; *p divides* (*q* +++ *r*) |] ==> *p divides r*
⟨*proof*⟩

**lemma** *poly-divides-diff2*: [| *p divides r*; *p divides* (*q* +++ *r*) |] ==> *p divides q*
⟨*proof*⟩

**lemma** *poly-divides-zero*: *poly p* = *poly* [] ==> *q divides p*
⟨*proof*⟩

**lemma** *poly-divides-zero2*: *q divides* []
⟨*proof*⟩
**declare** *poly-divides-zero2* [*simp*]

At last, we can consider the order of a root.

**lemma** *poly-order-exists-lemma* [*rule-format*]:
$\forall$ *p*. *length p* = *d* −−> *poly p* $\neq$ *poly* []
−−> ($\exists$ *n q*. *p* = *mulexp n* [−*a*, *1*] *q* & *poly q a* $\neq$ *0*)
⟨*proof*⟩

**lemma** *poly-order-exists*:
[| *length p* = *d*; *poly p* $\neq$ *poly* [] |]
==> $\exists$ *n*. ([−*a*, *1*] %^ *n*) *divides p* &
~(([−*a*, *1*] %^ (*Suc n*)) *divides p*)
⟨*proof*⟩

**lemma** *poly-one-divides*: [*1*] *divides p*
⟨*proof*⟩
**declare** *poly-one-divides* [*simp*]

**lemma** *poly-order*: *poly p* $\neq$ *poly* []
==> *EX*! *n*. ([−*a*, *1*] %^ *n*) *divides p* &
~(([−*a*, *1*] %^ (*Suc n*)) *divides p*)
⟨*proof*⟩

Order

**lemma** *some1-equalityD*: [| *n* = (@*n*. *P n*); *EX*! *n*. *P n* |] ==> *P n*
⟨*proof*⟩

**lemma** *order*:
(([−*a*, *1*] %^ *n*) *divides p* &
~(([−*a*, *1*] %^ (*Suc n*)) *divides p*)) =

    ((*n* = *order a p*) & ~(*poly p* = *poly* []))
⟨*proof*⟩

**lemma** *order2*: [| *poly p* ≠ *poly* [] |]
    ==> ([−*a*, *1*] %^ (*order a p*)) *divides p* &
        ~(([−*a*, *1*] %^ (*Suc*(*order a p*))) *divides p*)
⟨*proof*⟩

**lemma** *order-unique*: [| *poly p* ≠ *poly* []; ([−*a*, *1*] %^ *n*) *divides p*;
    ~(([−*a*, *1*] %^ (*Suc n*)) *divides p*)
    |] ==> (*n* = *order a p*)
⟨*proof*⟩

**lemma** *order-unique-lemma*: (*poly p* ≠ *poly* [] & ([−*a*, *1*] %^ *n*) *divides p* &
    ~(([−*a*, *1*] %^ (*Suc n*)) *divides p*))
    ==> (*n* = *order a p*)
⟨*proof*⟩

**lemma** *order-poly*: *poly p* = *poly q* ==> *order a p* = *order a q*
⟨*proof*⟩

**lemma** *pexp-one*: *p* %^ (*Suc 0*) = *p*
⟨*proof*⟩
**declare** *pexp-one* [*simp*]

**lemma** *lemma-order-root* [*rule-format*]:
    ∀ *p a*. *n* > *0* & [− *a*, *1*] %^ *n divides p* & ~ [− *a*, *1*] %^ (*Suc n*) *divides p*
        −−> *poly p a* = *0*
⟨*proof*⟩

**lemma** *order-root*: (*poly p a* = *0*) = ((*poly p* = *poly* []) | *order a p* ≠ *0*)
⟨*proof*⟩

**lemma** *order-divides*: (([−*a*, *1*] %^ *n*) *divides p*) = ((*poly p* = *poly* []) | *n* ≤ *order a p*)
⟨*proof*⟩

**lemma** *order-decomp*:
    *poly p* ≠ *poly* []
    ==> ∃ *q*. (*poly p* = *poly* (([−*a*, *1*] %^ (*order a p*)) *** *q*)) &
        ~([−*a*, *1*] *divides q*)
⟨*proof*⟩

Important composition properties of orders.

**lemma** *order-mult*: *poly* (*p* *** *q*) ≠ *poly* []
    ==> *order a* (*p* *** *q*) = *order a p* + *order a q*
⟨*proof*⟩

**lemma** *lemma-order-pderiv* [*rule-format*]:
  $\forall\, p\ q\ a.\ n > 0\ \&$
    *poly* (*pderiv p*) $\neq$ *poly* [] &
    *poly p* = *poly* ([$-$ *a*, *1*] %^ *n* *** *q*) & ~ [$-$ *a*, *1*] *divides q*
    --> *n* = *Suc* (*order a* (*pderiv p*))
⟨*proof*⟩

**lemma** *order-pderiv*: [| *poly* (*pderiv p*) $\neq$ *poly* []; *order a p* $\neq$ *0* |]
  ==> (*order a p* = *Suc* (*order a* (*pderiv p*)))
⟨*proof*⟩

Now justify the standard squarefree decomposition, i.e. f / gcd(f,f'). *) (* 'a la Harrison

**lemma** *poly-squarefree-decomp-order*: [| *poly* (*pderiv p*) $\neq$ *poly* [];
    *poly p* = *poly* (*q* *** *d*);
    *poly* (*pderiv p*) = *poly* (*e* *** *d*);
    *poly d* = *poly* (*r* *** *p* +++ *s* *** *pderiv p*)
  |] ==> *order a q* = (*if order a p* = *0 then 0 else 1*)
⟨*proof*⟩

**lemma** *poly-squarefree-decomp-order2*: [| *poly* (*pderiv p*) $\neq$ *poly* [];
    *poly p* = *poly* (*q* *** *d*);
    *poly* (*pderiv p*) = *poly* (*e* *** *d*);
    *poly d* = *poly* (*r* *** *p* +++ *s* *** *pderiv p*)
  |] ==> $\forall$ *a*. *order a q* = (*if order a p* = *0 then 0 else 1*)
⟨*proof*⟩

**lemma** *order-root2*: *poly p* $\neq$ *poly* [] ==> (*poly p a* = *0*) = (*order a p* $\neq$ *0*)
⟨*proof*⟩

**lemma** *order-pderiv2*: [| *poly* (*pderiv p*) $\neq$ *poly* []; *order a p* $\neq$ *0* |]
  ==> (*order a* (*pderiv p*) = *n*) = (*order a p* = *Suc n*)
⟨*proof*⟩

**lemma** *rsquarefree-roots*:
 *rsquarefree p* = ($\forall$ *a*. ~(*poly p a* = *0* & *poly* (*pderiv p*) *a* = *0*))
⟨*proof*⟩

**lemma** *pmult-one*: [*1*] *** *p* = *p*
⟨*proof*⟩
**declare** *pmult-one* [*simp*]

**lemma** *poly-Nil-zero*: *poly* [] = *poly* [*0*]
⟨*proof*⟩

**lemma** *rsquarefree-decomp*:
  [| *rsquarefree p*; *poly p a* = *0* |]
  ==> $\exists$ *q*. (*poly p* = *poly* ([$-$*a*, *1*] *** *q*)) & *poly q a* $\neq$ *0*

⟨*proof*⟩

**lemma** *poly-squarefree-decomp*: $[|$ *poly* (*pderiv p*) $\neq$ *poly* [];
*poly p* = *poly* (*q* $***$ *d*);
*poly* (*pderiv p*) = *poly* (*e* $***$ *d*);
*poly d* = *poly* (*r* $***$ *p* +++ *s* $***$ *pderiv p*)
$|]$ ==> *rsquarefree q* & ($\forall$ *a*. (*poly q a* = *0*) = (*poly p a* = *0*))
⟨*proof*⟩

Normalization of a polynomial.

**lemma** *poly-normalize*: *poly* (*pnormalize p*) = *poly p*
⟨*proof*⟩
**declare** *poly-normalize* [*simp*]

The degree of a polynomial.

**lemma** *lemma-degree-zero*:
*list-all* (%*c*. *c* = *0*) *p* $\longleftrightarrow$ *pnormalize p* = []
⟨*proof*⟩

**lemma** *degree-zero*: (*poly p* = *poly* []) $\Longrightarrow$ (*degree p* = *0*)
⟨*proof*⟩

**lemma** *pnormalize-sing*: (*pnormalize* [*x*] = [*x*]) $\longleftrightarrow$ *x* $\neq$ *0* ⟨*proof*⟩
**lemma** *pnormalize-pair*: *y* $\neq$ *0* $\longleftrightarrow$ (*pnormalize* [*x*, *y*] = [*x*, *y*]) ⟨*proof*⟩
**lemma** *pnormal-cons*: *pnormal p* $\Longrightarrow$ *pnormal* (*c*#*p*)
⟨*proof*⟩
**lemma** *pnormal-tail*: *p*$\neq$[] $\Longrightarrow$ *pnormal* (*c*#*p*) $\Longrightarrow$ *pnormal p*
⟨*proof*⟩
**lemma** *pnormal-last-nonzero*: *pnormal p* ==> *last p* $\neq$ *0*
⟨*proof*⟩
**lemma** *pnormal-length*: *pnormal p* $\Longrightarrow$ *0* < *length p*
⟨*proof*⟩
**lemma** *pnormal-last-length*: $[\![$*0* < *length p* ; *last p* $\neq$ *0*$]\!]$ $\Longrightarrow$ *pnormal p*
⟨*proof*⟩
**lemma** *pnormal-id*: *pnormal p* $\longleftrightarrow$ (*0* < *length p* $\wedge$ *last p* $\neq$ *0*)
⟨*proof*⟩

Tidier versions of finiteness of roots.

**lemma** *poly-roots-finite-set*: *poly p* $\neq$ *poly* [] ==> *finite* {*x*. *poly p x* = *0*}
⟨*proof*⟩

bound for polynomial.

**lemma** *poly-mono*: *abs*(*x*) $\leq$ *k* ==> *abs*(*poly p x*) $\leq$ *poly* (*map abs p*) *k*
⟨*proof*⟩

**lemma** *poly-Sing*: *poly* [*c*] *x* = *c* ⟨*proof*⟩
**end**

# 41 MacLaurin: MacLaurin Series

**theory** *MacLaurin*
**imports** *Transcendental*
**begin**

## 41.1 Maclaurin's Theorem with Lagrange Form of Remainder

This is a very long, messy proof even now that it's been broken down into lemmas.

**lemma** *Maclaurin-lemma*:
$0 < h ==>$
$\exists B.\ f\ h = (\sum m{=}0..{<}n.\ (j\ m\ /\ real\ (fact\ m)) * (h\hat{\ }m)) +$
$(B * ((h\hat{\ }n)\ /\ real(fact\ n)))$
⟨*proof*⟩

**lemma** *eq-diff-eq'*: $(x = y - z) = (y = x + (z{::}real))$
⟨*proof*⟩

A crude tactic to differentiate by proof.

**lemmas** *deriv-rulesI* =
*DERIV-ident DERIV-const DERIV-cos DERIV-cmult*
*DERIV-sin DERIV-exp DERIV-inverse DERIV-pow*
*DERIV-add DERIV-diff DERIV-mult DERIV-minus*
*DERIV-inverse-fun DERIV-quotient DERIV-fun-pow*
*DERIV-fun-exp DERIV-fun-sin DERIV-fun-cos*
*DERIV-ident DERIV-const DERIV-cos*

⟨*ML*⟩

**lemma** *Maclaurin-lemma2*:
$[|\ \forall m\ t.\ m < n \wedge 0{\le}t \wedge t{\le}h \longrightarrow DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t;$
$n = Suc\ k;$
$difg =$
$(\lambda m\ t.\ diff\ m\ t\ -$
$((\sum p = 0..{<}n - m.\ diff\ (m + p)\ 0\ /\ real\ (fact\ p) * t\ \hat{}\ p) +$
$B * (t\ \hat{}\ (n - m)\ /\ real\ (fact\ (n - m)))))|] ==>$
$\forall m\ t.\ m < n\ \&\ 0 \le t\ \&\ t \le h\ -->$
$DERIV\ (difg\ m)\ t :> difg\ (Suc\ m)\ t$
⟨*proof*⟩

**lemma** *Maclaurin-lemma3*:
**fixes** $difg :: nat => real => real$ **shows**
$[|\forall k\ t.\ k < Suc\ m \wedge 0{\le}t\ \&\ t{\le}h \longrightarrow DERIV\ (difg\ k)\ t :> difg\ (Suc\ k)\ t;$
$\forall k{<}Suc\ m.\ difg\ k\ 0 = 0;\ DERIV\ (difg\ n)\ t :> 0;\ \ n < m;\ 0 < t;$
$t < h|]$

$==> \exists ta.\ 0 < ta \ \&\ ta < t \ \&\ DERIV\ (difg\ (Suc\ n))\ ta :> 0$
$\langle proof \rangle$

**lemma** *Maclaurin*:
$[|\ 0 < h;\ n > 0;\ diff\ 0 = f;$
$\quad \forall m\ t.\ m < n \ \&\ 0 \leq t \ \&\ t \leq h \ --> DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t\ |]$
$==> \exists t.\ 0 < t \ \&$
$\quad t < h \ \&$
$\quad f\ h =$
$\quad setsum\ (\%m.\ (diff\ m\ 0\ /\ real\ (fact\ m)) * h\ \hat{}\ m)\ \{0..<n\} +$
$\quad (diff\ n\ t\ /\ real\ (fact\ n)) * h\ \hat{}\ n$
$\langle proof \rangle$

**lemma** *Maclaurin-objl*:
$0 < h \ \&\ n>0 \ \&\ diff\ 0 = f \ \&$
$(\forall m\ t.\ m < n \ \&\ 0 \leq t \ \&\ t \leq h \ --> DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t)$
$--> (\exists t.\ 0 < t \ \&\ t < h \ \&$
$\quad f\ h = (\sum m=0..<n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$\quad\quad diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n)$
$\langle proof \rangle$

**lemma** *Maclaurin2*:
$[|\ 0 < h;\ diff\ 0 = f;$
$\quad \forall m\ t.$
$\quad\quad m < n \ \&\ 0 \leq t \ \&\ t \leq h \ --> DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t\ |]$
$==> \exists t.\ 0 < t \ \&$
$\quad t \leq h \ \&$
$\quad f\ h =$
$\quad (\sum m=0..<n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$\quad diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n$
$\langle proof \rangle$

**lemma** *Maclaurin2-objl*:
$0 < h \ \&\ diff\ 0 = f \ \&$
$\quad (\forall m\ t.$
$\quad\quad m < n \ \&\ 0 \leq t \ \&\ t \leq h \ --> DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t)$
$--> (\exists t.\ 0 < t \ \&$
$\quad t \leq h \ \&$
$\quad f\ h =$
$\quad (\sum m=0..<n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$\quad diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n)$
$\langle proof \rangle$

**lemma** *Maclaurin-minus*:
$[|\ h < 0;\ n > 0;\ diff\ 0 = f;$
$\quad \forall m\ t.\ m < n \ \&\ h \leq t \ \&\ t \leq 0 \ --> DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t\ |]$
$==> \exists t.\ h < t \ \&$
$\quad t < 0 \ \&$

$f\ h =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n$
⟨proof⟩

**lemma** *Maclaurin-minus-objl*:
$(h < 0\ \&\ n > 0\ \&\ diff\ 0 = f\ \&$
$(\forall m\ t.$
$m < n\ \&\ h \leq t\ \&\ t \leq 0\ {-}{-}{>}\ DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t))$
${-}{-}{>}\ (\exists t.\ h < t\ \&$
$t < 0\ \&$
$f\ h =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n)$
⟨proof⟩

## 41.2 More Convenient "Bidirectional" Version.

**lemma** *Maclaurin-bi-le-lemma* [*rule-format*]:
$n{>}0 \longrightarrow$
$diff\ 0\ 0 =$
$(\sum m = 0..{<}n.\ diff\ m\ 0 * 0\ \hat{}\ m\ /\ real\ (fact\ m)) +$
$diff\ n\ 0 * 0\ \hat{}\ n\ /\ real\ (fact\ n)$
⟨proof⟩

**lemma** *Maclaurin-bi-le*:
$[|\ diff\ 0 = f;$
$\forall m\ t.\ m < n\ \&\ abs\ t \leq abs\ x\ {-}{-}{>}\ DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t\ |]$
${=}{=}{>}\ \exists t.\ abs\ t \leq abs\ x\ \&$
$f\ x =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * x\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * x\ \hat{}\ n$
⟨proof⟩

**lemma** *Maclaurin-all-lt*:
$[|\ diff\ 0 = f;$
$\forall m\ x.\ DERIV\ (diff\ m)\ x :> diff(Suc\ m)\ x;$
$x\ {\tilde{}}{=}\ 0;\ n > 0$
$|]\ {=}{=}{>}\ \exists t.\ 0 < abs\ t\ \&\ abs\ t < abs\ x\ \&$
$f\ x = (\sum m{=}0..{<}n.\ (diff\ m\ 0\ /\ real\ (fact\ m)) * x\ \hat{}\ m) +$
$(diff\ n\ t\ /\ real\ (fact\ n)) * x\ \hat{}\ n$
⟨proof⟩

**lemma** *Maclaurin-all-lt-objl*:
$diff\ 0 = f\ \&$
$(\forall m\ x.\ DERIV\ (diff\ m)\ x :> diff(Suc\ m)\ x)\ \&$
$x\ {\tilde{}}{=}\ 0\ \&\ n > 0$
${-}{-}{>}\ (\exists t.\ 0 < abs\ t\ \&\ abs\ t < abs\ x\ \&$
$f\ x = (\sum m{=}0..{<}n.\ (diff\ m\ 0\ /\ real\ (fact\ m)) * x\ \hat{}\ m) +$

$(diff\ n\ t\ /\ real\ (fact\ n))*x\ \hat{}\ n)$
⟨proof⟩

**lemma** *Maclaurin-zero* [*rule-format*]:
$x = (0::real)$
$==> n \neq 0 \ -->$
$(\sum m=0..<n.\ (diff\ m\ (0::real)\ /\ real\ (fact\ m))*x\ \hat{}\ m) =$
$diff\ 0\ 0$
⟨proof⟩

**lemma** *Maclaurin-all-le*: $[|\ diff\ 0 = f;$
$\forall m\ x.\ DERIV\ (diff\ m)\ x :> diff\ (Suc\ m)\ x$
$|] ==> \exists t.\ abs\ t \leq abs\ x\ \&$
$f\ x = (\sum m=0..<n.\ (diff\ m\ 0\ /\ real\ (fact\ m))*x\ \hat{}\ m) +$
$(diff\ n\ t\ /\ real\ (fact\ n))*x\ \hat{}\ n$
⟨proof⟩

**lemma** *Maclaurin-all-le-objl*: $diff\ 0 = f\ \&$
$(\forall m\ x.\ DERIV\ (diff\ m)\ x :> diff\ (Suc\ m)\ x)$
$--> (\exists t.\ abs\ t \leq abs\ x\ \&$
$f\ x = (\sum m=0..<n.\ (diff\ m\ 0\ /\ real\ (fact\ m))*x\ \hat{}\ m) +$
$(diff\ n\ t\ /\ real\ (fact\ n))*x\ \hat{}\ n)$
⟨proof⟩

## 41.3 Version for Exponential Function

**lemma** *Maclaurin-exp-lt*: $[|\ x\ \sim= 0;\ n > 0\ |]$
$==> (\exists t.\ 0 < abs\ t\ \&$
$abs\ t < abs\ x\ \&$
$exp\ x = (\sum m=0..<n.\ (x\ \hat{}\ m)\ /\ real\ (fact\ m)) +$
$(exp\ t\ /\ real\ (fact\ n))*x\ \hat{}\ n)$
⟨proof⟩

**lemma** *Maclaurin-exp-le*:
$\exists t.\ abs\ t \leq abs\ x\ \&$
$exp\ x = (\sum m=0..<n.\ (x\ \hat{}\ m)\ /\ real\ (fact\ m)) +$
$(exp\ t\ /\ real\ (fact\ n))*x\ \hat{}\ n$
⟨proof⟩

## 41.4 Version for Sine Function

**lemma** *MVT2*:
$[|\ a < b;\ \forall x.\ a \leq x\ \&\ x \leq b\ --> DERIV\ f\ x :> f'(x)\ |]$
$==> \exists z::real.\ a < z\ \&\ z < b\ \&\ (f\ b - f\ a = (b - a)*f'(z))$
⟨proof⟩

**lemma** *mod-exhaust-less-4*:
$m\ mod\ 4 = 0\ |\ m\ mod\ 4 = 1\ |\ m\ mod\ 4 = 2\ |\ m\ mod\ 4 = (3::nat)$
⟨proof⟩

**lemma** *Suc-Suc-mult-two-diff-two* [*rule-format*, *simp*]:
  $n \neq 0$ −−> *Suc* (*Suc* (*2* ∗ *n* − *2*)) = *2*∗*n*
⟨*proof*⟩

**lemma** *lemma-Suc-Suc-4n-diff-2* [*rule-format*, *simp*]:
  $n \neq 0$ −−> *Suc* (*Suc* (*4*∗*n* − *2*)) = *4*∗*n*
⟨*proof*⟩

**lemma** *Suc-mult-two-diff-one* [*rule-format*, *simp*]:
  $n \neq 0$ −−> *Suc* (*2* ∗ *n* − *1*) = *2*∗*n*
⟨*proof*⟩

It is unclear why so many variant results are needed.

**lemma** *Maclaurin-sin-expansion2*:
    ∃ *t*. *abs t* ≤ *abs x* &
      *sin x* =
      (∑ *m*=*0*..<*n*. (*if even m then 0*
                    *else* (−*1* ˆ ((*m* − *Suc 0*) *div 2*)) / *real* (*fact m*)) ∗
                    *x* ˆ *m*)
     + ((*sin*(*t* + *1*/*2* ∗ *real* (*n*) ∗*pi*) / *real* (*fact n*)) ∗ *x* ˆ *n*)
⟨*proof*⟩

**lemma** *Maclaurin-sin-expansion*:
    ∃ *t*. *sin x* =
      (∑ *m*=*0*..<*n*. (*if even m then 0*
                    *else* (−*1* ˆ ((*m* − *Suc 0*) *div 2*)) / *real* (*fact m*)) ∗
                    *x* ˆ *m*)
     + ((*sin*(*t* + *1*/*2* ∗ *real* (*n*) ∗*pi*) / *real* (*fact n*)) ∗ *x* ˆ *n*)
⟨*proof*⟩

**lemma** *Maclaurin-sin-expansion3*:
    [| *n* > *0*; *0* < *x* |] ==>
      ∃ *t*. *0* < *t* & *t* < *x* &
      *sin x* =
      (∑ *m*=*0*..<*n*. (*if even m then 0*
                    *else* (−*1* ˆ ((*m* − *Suc 0*) *div 2*)) / *real* (*fact m*)) ∗
                    *x* ˆ *m*)
     + ((*sin*(*t* + *1*/*2* ∗ *real*(*n*) ∗*pi*) / *real* (*fact n*)) ∗ *x* ˆ *n*)
⟨*proof*⟩

**lemma** *Maclaurin-sin-expansion4*:
    *0* < *x* ==>
      ∃ *t*. *0* < *t* & *t* ≤ *x* &
      *sin x* =
      (∑ *m*=*0*..<*n*. (*if even m then 0*
                    *else* (−*1* ˆ ((*m* − *Suc 0*) *div 2*)) / *real* (*fact m*)) ∗
                    *x* ˆ *m*)

$+ ((sin(t + 1/2 * real\ (n) * pi) / real\ (fact\ n)) * x \hat{}\ n)$
⟨*proof*⟩

## 41.5 Maclaurin Expansion for Cosine Function

**lemma** *sumr-cos-zero-one* [*simp*]:
$(\sum m{=}0..{<}(Suc\ n).$
$(if\ even\ m\ then\ -1\ \hat{}\ (m\ div\ 2)/(real\ \ (fact\ m))\ else\ 0) * 0\ \hat{}\ m) = 1$
⟨*proof*⟩

**lemma** *Maclaurin-cos-expansion*:
$\exists t.\ abs\ t \leq abs\ x\ \&$
$cos\ x =$
$(\sum m{=}0..{<}n.\ (if\ even\ m$
$then\ -1\ \hat{}\ (m\ div\ 2)/(real\ (fact\ m))$
$else\ 0) *$
$x\ \hat{}\ m)$
$+ ((cos(t + 1/2 * real\ (n) * pi) / real\ (fact\ n)) * x\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *Maclaurin-cos-expansion2*:
$[|\ 0 < x;\ n > 0\ |] ==>$
$\exists t.\ 0 < t\ \&\ t < x\ \&$
$cos\ x =$
$(\sum m{=}0..{<}n.\ (if\ even\ m$
$then\ -1\ \hat{}\ (m\ div\ 2)/(real\ (fact\ m))$
$else\ 0) *$
$x\ \hat{}\ m)$
$+ ((cos(t + 1/2 * real\ (n) * pi) / real\ (fact\ n)) * x\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *Maclaurin-minus-cos-expansion*:
$[|\ x < 0;\ n > 0\ |] ==>$
$\exists t.\ x < t\ \&\ t < 0\ \&$
$cos\ x =$
$(\sum m{=}0..{<}n.\ (if\ even\ m$
$then\ -1\ \hat{}\ (m\ div\ 2)/(real\ (fact\ m))$
$else\ 0) *$
$x\ \hat{}\ m)$
$+ ((cos(t + 1/2 * real\ (n) * pi) / real\ (fact\ n)) * x\ \hat{}\ n)$
⟨*proof*⟩

**lemma** *sin-bound-lemma*:
$[|x = y;\ abs\ u \leq (v{::}real)\ |] ==> |(x + u) - y| \leq v$
⟨*proof*⟩

**lemma** *Maclaurin-sin-bound*:
 $abs(sin\ x - (\sum m{=}0..{<}n.\ (if\ even\ m\ then\ 0\ else\ (-1\ \hat{}\ ((m - Suc\ 0)\ div\ 2))\ /\ real\ (fact\ m))\ *$
 $x\ \hat{}\ m)) \le inverse(real\ (fact\ n)) * |x|\ \hat{}\ n$
⟨*proof*⟩

**end**

# 42 Taylor: Taylor series

**theory** *Taylor*
**imports** *MacLaurin*
**begin**

We use MacLaurin and the translation of the expansion point $c$ to $0$ to prove Taylor's theorem.

**lemma** *taylor-up*:
 **assumes** *INIT*: $n{>}0\ diff\ 0 = f$
 **and** *DERIV*: $(\forall\ m\ t.\ m < n\ \&\ a \le t\ \&\ t \le b \longrightarrow DERIV\ (diff\ m)\ t :> (diff\ (Suc\ m)\ t))$
 **and** *INTERV*: $a \le c\ \ c < b$
 **shows** $\exists\ t.\ c < t\ \&\ t < b\ \&$
  $f\ b = setsum\ (\%m.\ (diff\ m\ c\ /\ real\ (fact\ m)) * (b - c)\hat{}m)\ \{0..{<}n\} +$
   $(diff\ n\ t\ /\ real\ (fact\ n)) * (b - c)\hat{}n$
⟨*proof*⟩

**lemma** *taylor-down*:
 **assumes** *INIT*: $n{>}0\ diff\ 0 = f$
 **and** *DERIV*: $(\forall\ m\ t.\ m < n\ \&\ a \le t\ \&\ t \le b \longrightarrow DERIV\ (diff\ m)\ t :> (diff\ (Suc\ m)\ t))$
 **and** *INTERV*: $a < c\ \ c \le b$
 **shows** $\exists\ t.\ a < t\ \&\ t < c\ \&$
  $f\ a = setsum\ (\%\ m.\ (diff\ m\ c\ /\ real\ (fact\ m)) * (a - c)\hat{}m)\ \{0..{<}n\} +$
   $(diff\ n\ t\ /\ real\ (fact\ n)) * (a - c)\hat{}n$
⟨*proof*⟩

**lemma** *taylor*:
 **assumes** *INIT*: $n{>}0\ diff\ 0 = f$
 **and** *DERIV*: $(\forall\ m\ t.\ m < n\ \&\ a \le t\ \&\ t \le b \longrightarrow DERIV\ (diff\ m)\ t :> (diff\ (Suc\ m)\ t))$
 **and** *INTERV*: $a \le c\ \ c \le b\ \ a \le x\ \ x \le b\ \ x \ne c$
 **shows** $\exists\ t.\ (if\ x{<}c\ then\ (x < t\ \&\ t < c)\ else\ (c < t\ \&\ t < x))\ \&$
  $f\ x = setsum\ (\%\ m.\ (diff\ m\ c\ /\ real\ (fact\ m)) * (x - c)\hat{}m)\ \{0..{<}n\} +$
   $(diff\ n\ t\ /\ real\ (fact\ n)) * (x - c)\hat{}n$
⟨*proof*⟩

**end**

# 43 Integration: Theory of Integration

**theory** *Integration*
**imports** *MacLaurin*
**begin**

We follow John Harrison in formalizing the Gauge integral.

**definition**
— Partitions and tagged partitions etc.

*partition* :: [(*real*∗*real*),*nat* => *real*] => *bool* **where**
*partition* = (%(*a*,*b*) *D*. *D 0* = *a* &
(∃ *N*. (∀ *n* < *N*. *D*(*n*) < *D*(*Suc n*)) &
(∀ *n* ≥ *N*. *D*(*n*) = *b*)))

**definition**
*psize* :: (*nat* => *real*) => *nat* **where**
*psize D* = (*SOME N*. (∀ *n* < *N*. *D*(*n*) < *D*(*Suc n*)) &
(∀ *n* ≥ *N*. *D*(*n*) = *D*(*N*)))

**definition**
*tpart* :: [(*real*∗*real*),((*nat* => *real*)∗(*nat* =>*real*))] => *bool* **where**
*tpart* = (%(*a*,*b*) (*D*,*p*). *partition*(*a*,*b*) *D* &
(∀ *n*. *D*(*n*) ≤ *p*(*n*) & *p*(*n*) ≤ *D*(*Suc n*)))

— Gauges and gauge-fine divisions

**definition**
*gauge* :: [*real* => *bool*, *real* => *real*] => *bool* **where**
*gauge E g* = (∀ *x*. *E x* −−> *0* < *g*(*x*))

**definition**
*fine* :: [*real* => *real*, ((*nat* => *real*)∗(*nat* => *real*))] => *bool* **where**
*fine* = (%*g* (*D*,*p*). ∀ *n*. *n* < (*psize D*) −−> *D*(*Suc n*) − *D*(*n*) < *g*(*p n*))

— Riemann sum

**definition**
*rsum* :: [((*nat*=>*real*)∗(*nat*=>*real*)),*real*=>*real*] => *real* **where**
*rsum* = (%(*D*,*p*) *f*. ∑ *n*=*0*..<*psize*(*D*). *f*(*p n*) ∗ (*D*(*Suc n*) − *D*(*n*)))

— Gauge integrability (definite)

**definition**
*Integral* :: [(*real*∗*real*),*real*=>*real*,*real*] => *bool* **where**
*Integral* = (%(*a*,*b*) *f k*. ∀ *e* > *0*.
(∃ *g*. *gauge*(%*x*. *a* ≤ *x* & *x* ≤ *b*) *g* &

$$(\forall D\ p.\ tpart(a,b)\ (D,p)\ \&\ fine(g)(D,p)\ -->$$
$$|rsum(D,p)\ f\ -\ k|\ <\ e)))$$

**lemma** *partition-zero* [*simp*]: $a = b ==> psize\ (\%n.\ if\ n = 0\ then\ a\ else\ b) = 0$
⟨*proof*⟩

**lemma** *partition-one* [*simp*]: $a < b ==> psize\ (\%n.\ if\ n = 0\ then\ a\ else\ b) = 1$
⟨*proof*⟩

**lemma** *partition-single* [*simp*]:
$a \le b ==> partition(a,b)(\%n.\ if\ n = 0\ then\ a\ else\ b)$
⟨*proof*⟩

**lemma** *partition-lhs*: $partition(a,b)\ D ==> (D(0) = a)$
⟨*proof*⟩

**lemma** *partition*:
$(partition(a,b)\ D) =$
$((D\ 0 = a)\ \&$
$(\forall n < psize\ D.\ D\ n < D(Suc\ n))\ \&$
$(\forall n \ge psize\ D.\ D\ n = b))$
⟨*proof*⟩

**lemma** *partition-rhs*: $partition(a,b)\ D ==> (D(psize\ D) = b)$
⟨*proof*⟩

**lemma** *partition-rhs2*: $[|partition(a,b)\ D;\ psize\ D \le n\ |] ==> (D\ n = b)$
⟨*proof*⟩

**lemma** *lemma-partition-lt-gen* [*rule-format*]:
$partition(a,b)\ D\ \&\ m + Suc\ d \le n\ \&\ n \le (psize\ D) --> D(m) < D(m + Suc\ d)$
⟨*proof*⟩

**lemma** *less-eq-add-Suc*: $m < n ==> \exists d.\ n = m + Suc\ d$
⟨*proof*⟩

**lemma** *partition-lt-gen*:
$[|partition(a,b)\ D;\ m < n;\ n \le (psize\ D)|] ==> D(m) < D(n)$
⟨*proof*⟩

**lemma** *partition-lt*: $partition(a,b)\ D ==> n < (psize\ D) ==> D(0) < D(Suc\ n)$
⟨*proof*⟩

**lemma** *partition-le*: $partition(a,b)\ D ==> a \le b$
⟨*proof*⟩

**lemma** *partition-gt*: $[|partition(a,b)\ D;\ n < (psize\ D)|] ==> D(n) < D(psize\ D)$

⟨*proof*⟩

**lemma** *partition-eq*: $partition(a,b)\ D ==> ((a = b) = (psize\ D = 0))$
⟨*proof*⟩

**lemma** *partition-lb*: $partition(a,b)\ D ==> a \le D(r)$
⟨*proof*⟩

**lemma** *partition-lb-lt*: $[|\ partition(a,b)\ D;\ psize\ D\ \tilde{}= 0\ |] ==> a < D(Suc\ n)$
⟨*proof*⟩

**lemma** *partition-ub*: $partition(a,b)\ D ==> D(r) \le b$
⟨*proof*⟩

**lemma** *partition-ub-lt*: $[|\ partition(a,b)\ D;\ n < psize\ D\ |] ==> D(n) < b$
⟨*proof*⟩

**lemma** *lemma-partition-append1*:
$[|\ partition\ (a,\ b)\ D1;\ partition\ (b,\ c)\ D2\ |]$
$==> (\forall\, n < psize\ D1 + psize\ D2.$
$(if\ n < psize\ D1\ then\ D1\ n\ else\ D2\ (n - psize\ D1))$
$< (if\ Suc\ n < psize\ D1\ then\ D1\ (Suc\ n)$
$else\ D2\ (Suc\ n - psize\ D1))) \ \&$
$(\forall\, n \ge psize\ D1 + psize\ D2.$
$(if\ n < psize\ D1\ then\ D1\ n\ else\ D2\ (n - psize\ D1)) =$
$(if\ psize\ D1 + psize\ D2 < psize\ D1\ then\ D1\ (psize\ D1 + psize\ D2)$
$else\ D2\ (psize\ D1 + psize\ D2 - psize\ D1)))$
⟨*proof*⟩

**lemma** *lemma-psize1*:
$[|\ partition\ (a,\ b)\ D1;\ partition\ (b,\ c)\ D2;\ N < psize\ D1\ |]$
$==> D1(N) < D2\ (psize\ D2)$
⟨*proof*⟩

**lemma** *lemma-partition-append2*:
$[|\ partition\ (a,\ b)\ D1;\ partition\ (b,\ c)\ D2\ |]$
$==> psize\ (\%n.\ if\ n < psize\ D1\ then\ D1\ n\ else\ D2\ (n - psize\ D1)) =$
$psize\ D1 + psize\ D2$
⟨*proof*⟩

**lemma** *tpart-eq-lhs-rhs*: $[|psize\ D = 0;\ tpart(a,b)\ (D,p)|] ==> a = b$
⟨*proof*⟩

**lemma** *tpart-partition*: $tpart(a,b)\ (D,p) ==> partition(a,b)\ D$
⟨*proof*⟩

**lemma** *partition-append*:
$[|\ tpart(a,b)\ (D1,p1);\ fine(g)\ (D1,p1);$
$tpart(b,c)\ (D2,p2);\ fine(g)\ (D2,p2)\ |]$

$==> \exists D\ p.\ tpart(a,c)\ (D,p)\ \&\ fine(g)\ (D,p)$
⟨proof⟩

We can always find a division that is fine wrt any gauge

**lemma** *partition-exists*:
$[|\ a \le b;\ gauge(\%x.\ a \le x\ \&\ x \le b)\ g\ |]$
$==> \exists D\ p.\ tpart(a,b)\ (D,p)\ \&\ fine\ g\ (D,p)$
⟨proof⟩

Lemmas about combining gauges

**lemma** *gauge-min*:
$[|\ gauge(E)\ g1;\ gauge(E)\ g2\ |]$
$==> gauge(E)\ (\%x.\ if\ g1(x) < g2(x)\ then\ g1(x)\ else\ g2(x))$
⟨proof⟩

**lemma** *fine-min*:
$fine\ (\%x.\ if\ g1(x) < g2(x)\ then\ g1(x)\ else\ g2(x))\ (D,p)$
$==> fine(g1)\ (D,p)\ \&\ fine(g2)\ (D,p)$
⟨proof⟩

The integral is unique if it exists

**lemma** *Integral-unique*:
$[|\ a \le b;\ Integral(a,b)\ f\ k1;\ Integral(a,b)\ f\ k2\ |] ==> k1 = k2$
⟨proof⟩

**lemma** *Integral-zero* [*simp*]: $Integral(a,a)\ f\ 0$
⟨proof⟩

**lemma** *sumr-partition-eq-diff-bounds* [*simp*]:
$(\sum n{=}0..{<}m.\ D\ (Suc\ n) - D\ n{::}real) = D(m) - D\ 0$
⟨proof⟩

**lemma** *Integral-eq-diff-bounds*: $a \le b ==> Integral(a,b)\ (\%x.\ 1)\ (b - a)$
⟨proof⟩

**lemma** *Integral-mult-const*: $a \le b ==> Integral(a,b)\ (\%x.\ c)\ \ (c*(b - a))$
⟨proof⟩

**lemma** *Integral-mult*:
$[|\ a \le b;\ Integral(a,b)\ f\ k\ |] ==> Integral(a,b)\ (\%x.\ c * f\ x)\ (c * k)$
⟨proof⟩

Fundamental theorem of calculus (Part I)

"Straddle Lemma" : Swartz and Thompson: AMM 95(7) 1988

**lemma** *choiceP*: $\forall x.\ P(x) --> (\exists y.\ Q\ x\ y) ==> \exists f.\ (\forall x.\ P(x) --> Q\ x\ (f\ x))$
⟨proof⟩

**lemma** *strad1*:
$[\![\forall\, xa{::}real.\ xa \neq x \wedge |xa - x| < s \longrightarrow$
$|(f\, xa - f\, x)\ /\ (xa - x) - f'\, x| * 2 < e;$
$0 < e;\ a \leq x;\ x \leq b;\ 0 < s]\!]$
$\Longrightarrow \forall\, z.\ |z - x| < s \ \text{-->}|f\, z - f\, x - f'\, x * (z - x)| * 2 \leq e * |z - x|$
$\langle proof \rangle$

**lemma** *lemma-straddle*:
$[|\ \forall\, x.\ a \leq x\ \&\ x \leq b \ \text{-->}\ DERIV\, f\, x :> f'(x);\ 0 < e\ |]$
$\text{==>}\ \exists\, g.\ gauge(\%x.\ a \leq x\ \&\ x \leq b)\ g\ \&$
$(\forall\, x\, u\, v.\ a \leq u\ \&\ u \leq x\ \&\ x \leq v\ \&\ v \leq b\ \&\ (v - u) < g(x)$
$\text{-->}\ |(f(v) - f(u)) - (f'(x) * (v - u))| \leq e * (v - u))$
$\langle proof \rangle$

**lemma** *FTC1*: $[|a \leq b;\ \forall\, x.\ a \leq x\ \&\ x \leq b \ \text{-->}\ DERIV\, f\, x :> f'(x)\ |]$
$\text{==>}\ Integral(a,b)\ f'\ (f(b) - f(a))$
$\langle proof \rangle$

**lemma** *Integral-subst*: $[|\ Integral(a,b)\ f\, k1;\ k2{=}k1\ |]\ \text{==>}\ Integral(a,b)\ f\, k2$
$\langle proof \rangle$

**lemma** *Integral-add*:
$[|\ a \leq b;\ b \leq c;\ Integral(a,b)\ f'\ k1;\ Integral(b,c)\ f'\ k2;$
$\forall\, x.\ a \leq x\ \&\ x \leq c \ \text{-->}\ DERIV\, f\, x :> f'\, x\ |]$
$\text{==>}\ Integral(a,c)\ f'\ (k1 + k2)$
$\langle proof \rangle$

**lemma** *partition-psize-Least*:
$partition(a,b)\ D\ \text{==>}\ psize\ D = (LEAST\ n.\ D(n) = b)$
$\langle proof \rangle$

**lemma** *lemma-partition-bounded*: $partition\ (a,\ c)\ D\ \text{==>}\ \sim (\exists\, n.\ c < D(n))$
$\langle proof \rangle$

**lemma** *lemma-partition-eq*:
$partition\ (a,\ c)\ D\ \text{==>}\ D = (\%n.\ if\ D\ n < c\ then\ D\ n\ else\ c)$
$\langle proof \rangle$

**lemma** *lemma-partition-eq2*:
$partition\ (a,\ c)\ D\ \text{==>}\ D = (\%n.\ if\ D\ n \leq c\ then\ D\ n\ else\ c)$
$\langle proof \rangle$

**lemma** *partition-lt-Suc*:
$[|\ partition(a,b)\ D;\ n < psize\ D\ |]\ \text{==>}\ D\ n < D\ (Suc\ n)$

⟨*proof*⟩

**lemma** *tpart-tag-eq*: *tpart*(*a*,*c*) (*D*,*p*) ==> *p* = (%*n*. *if D n* < *c then p n else c*)
⟨*proof*⟩

## 43.1 Lemmas for Additivity Theorem of Gauge Integral

**lemma** *lemma-additivity1*:
[| *a* ≤ *D n*; *D n* < *b*; *partition*(*a*,*b*) *D* |] ==> *n* < *psize D*
⟨*proof*⟩

**lemma** *lemma-additivity2*: [| *a* ≤ *D n*; *partition*(*a*,*D n*) *D* |] ==> *psize D* ≤ *n*
⟨*proof*⟩

**lemma** *partition-eq-bound*:
[| *partition*(*a*,*b*) *D*; *psize D* < *m* |] ==> *D*(*m*) = *D*(*psize D*)
⟨*proof*⟩

**lemma** *partition-ub2*: [| *partition*(*a*,*b*) *D*; *psize D* < *m* |] ==> *D*(*r*) ≤ *D*(*m*)
⟨*proof*⟩

**lemma** *tag-point-eq-partition-point*:
[| *tpart*(*a*,*b*) (*D*,*p*); *psize D* ≤ *m* |] ==> *p*(*m*) = *D*(*m*)
⟨*proof*⟩

**lemma** *partition-lt-cancel*: [| *partition*(*a*,*b*) *D*; *D m* < *D n* |] ==> *m* < *n*
⟨*proof*⟩

**lemma** *lemma-additivity4-psize-eq*:
[| *a* ≤ *D n*; *D n* < *b*; *partition* (*a*, *b*) *D* |]
==> *psize* (%*x*. *if D x* < *D n then D*(*x*) *else D n*) = *n*
⟨*proof*⟩

**lemma** *lemma-psize-left-less-psize*:
*partition* (*a*, *b*) *D*
==> *psize* (%*x*. *if D x* < *D n then D*(*x*) *else D n*) ≤ *psize D*
⟨*proof*⟩

**lemma** *lemma-psize-left-less-psize2*:
[| *partition*(*a*,*b*) *D*; *na* < *psize* (%*x*. *if D x* < *D n then D*(*x*) *else D n*) |]
==> *na* < *psize D*
⟨*proof*⟩

**lemma** *lemma-additivity3*:
[| *partition*(*a*,*b*) *D*; *D na* < *D n*; *D n* < *D* (*Suc na*);
*n* < *psize D* |]
==> *False*
⟨*proof*⟩

**lemma** *psize-const* [*simp*]: *psize* (%*x*. *k*) = *0*
⟨*proof*⟩

**lemma** *lemma-additivity3a*:
[| *partition*(*a*,*b*) *D*; *D na* < *D n*; *D n* < *D* (*Suc na*);
*na* < *psize D* |]
==> *False*
⟨*proof*⟩

**lemma** *better-lemma-psize-right-eq1*:
[| *partition*(*a*,*b*) *D*; *D n* < *b* |] ==> *psize* (%*x*. *D* (*x* + *n*)) ≤ *psize D* − *n*
⟨*proof*⟩

**lemma** *psize-le-n*: *partition* (*a*, *D n*) *D* ==> *psize D* ≤ *n*
⟨*proof*⟩

**lemma** *better-lemma-psize-right-eq1a*:
*partition*(*a*,*D n*) *D* ==> *psize* (%*x*. *D* (*x* + *n*)) ≤ *psize D* − *n*
⟨*proof*⟩

**lemma** *better-lemma-psize-right-eq*:
*partition*(*a*,*b*) *D* ==> *psize* (%*x*. *D* (*x* + *n*)) ≤ *psize D* − *n*
⟨*proof*⟩

**lemma** *lemma-psize-right-eq1*:
[| *partition*(*a*,*b*) *D*; *D n* < *b* |] ==> *psize* (%*x*. *D* (*x* + *n*)) ≤ *psize D*
⟨*proof*⟩

**lemma** *lemma-psize-right-eq1a*:
*partition*(*a*,*D n*) *D* ==> *psize* (%*x*. *D* (*x* + *n*)) ≤ *psize D*
⟨*proof*⟩

**lemma** *lemma-psize-right-eq*:
[| *partition*(*a*,*b*) *D* |] ==> *psize* (%*x*. *D* (*x* + *n*)) ≤ *psize D*
⟨*proof*⟩

**lemma** *tpart-left1*:
[| *a* ≤ *D n*; *tpart* (*a*, *b*) (*D*, *p*) |]
==> *tpart*(*a*, *D n*) (%*x*. *if D x* < *D n then D*(*x*) *else D n*,
%*x*. *if D x* < *D n then p*(*x*) *else D n*)
⟨*proof*⟩

**lemma** *fine-left1*:
[| *a* ≤ *D n*; *tpart* (*a*, *b*) (*D*, *p*); *gauge* (%*x*. *a* ≤ *x* & *x* ≤ *D n*) *g*;
*fine* (%*x*. *if x* < *D n then min* (*g x*) ((*D n* − *x*)/ *2*)
*else if x* = *D n then min* (*g* (*D n*)) (*ga* (*D n*))

*else min* $(ga\ x)\ ((x - D\ n)/\ 2))\ (D,\ p)$ |]
==> *fine g*
(%$x$. *if* $D\ x < D\ n$ *then* $D(x)$ *else* $D\ n$,
%$x$. *if* $D\ x < D\ n$ *then* $p(x)$ *else* $D\ n$)
⟨*proof*⟩

**lemma** *tpart-right1*:
[| $a \le D\ n$; *tpart* $(a,\ b)\ (D,\ p)$ |]
==> *tpart*$(D\ n,\ b)$ (%$x$. $D(x + n)$,%$x$. $p(x + n)$)
⟨*proof*⟩

**lemma** *fine-right1*:
[| $a \le D\ n$; *tpart* $(a,\ b)\ (D,\ p)$; *gauge* (%$x$. $D\ n \le x$ & $x \le b$) *ga*;
*fine* (%$x$. *if* $x < D\ n$ *then min* $(g\ x)\ ((D\ n - x)/\ 2)$
*else if* $x = D\ n$ *then min* $(g\ (D\ n))\ (ga\ (D\ n))$
*else min* $(ga\ x)\ ((x - D\ n)/\ 2))\ (D,\ p)$ |]
==> *fine ga* (%$x$. $D(x + n)$,%$x$. $p(x + n)$)
⟨*proof*⟩

**lemma** *rsum-add*: *rsum* $(D,\ p)$ (%$x$. $f\ x + g\ x$) = *rsum* $(D,\ p)\ f$ + *rsum*$(D,\ p)$
$g$
⟨*proof*⟩

Bartle/Sherbert: Theorem 10.1.5 p. 278

**lemma** *Integral-add-fun*:
[| $a \le b$; *Integral*$(a,b)$ $f$ *k1*; *Integral*$(a,b)$ $g$ *k2* |]
==> *Integral*$(a,b)$ (%$x$. $f\ x + g\ x$) (*k1* + *k2*)
⟨*proof*⟩

**lemma** *partition-lt-gen2*:
[| *partition*$(a,b)$ $D$; $r <$ *psize* $D$ |] ==> $0 < D\ (Suc\ r) - D\ r$
⟨*proof*⟩

**lemma** *lemma-Integral-le*:
[| $\forall x.\ a \le x$ & $x \le b$ --> $f\ x \le g\ x$;
*tpart*$(a,b)$ $(D,p)$
|] ==> $\forall n \le$ *psize* $D$. $f\ (p\ n) \le g\ (p\ n)$
⟨*proof*⟩

**lemma** *lemma-Integral-rsum-le*:
[| $\forall x.\ a \le x$ & $x \le b$ --> $f\ x \le g\ x$;
*tpart*$(a,b)$ $(D,p)$
|] ==> *rsum*$(D,p)$ $f \le$ *rsum*$(D,p)$ $g$
⟨*proof*⟩

**lemma** *Integral-le*:
[| $a \le b$;
$\forall x.\ a \le x$ & $x \le b$ --> $f(x) \le g(x)$;
*Integral*$(a,b)$ $f$ *k1*; *Integral*$(a,b)$ $g$ *k2*

$|] ==> k1 \leq k2$
⟨proof⟩

**lemma** *Integral-imp-Cauchy*:
$(\exists k.\ Integral(a,b)\ f\ k) ==>$
$(\forall e > 0.\ \exists g.\ gauge\ (\%x.\ a \leq x\ \&\ x \leq b)\ g\ \&$
$(\forall D1\ D2\ p1\ p2.$
$tpart(a,b)\ (D1,\ p1)\ \&\ fine\ g\ (D1,p1)\ \&$
$tpart(a,b)\ (D2,\ p2)\ \&\ fine\ g\ (D2,p2)\ -->$
$|rsum(D1,p1)\ f\ -\ rsum(D2,p2)\ f| < e))$
⟨proof⟩

**lemma** *Cauchy-iff2*:
$Cauchy\ X =$
$(\forall j.\ (\exists M.\ \forall m \geq M.\ \forall n \geq M.\ |X\ m\ -\ X\ n| < inverse(real\ (Suc\ j))))$
⟨proof⟩

**lemma** *partition-exists2*:
$[|\ a \leq b;\ \forall n.\ gauge\ (\%x.\ a \leq x\ \&\ x \leq b)\ (fa\ n)\ |]$
$==> \forall n.\ \exists D\ p.\ tpart\ (a,\ b)\ (D,\ p)\ \&\ fine\ (fa\ n)\ (D,\ p)$
⟨proof⟩

**lemma** *monotonic-anti-derivative*:
**fixes** *f g* :: *real* => *real* **shows**
$[|\ a \leq b;\ \forall c.\ a \leq c\ \&\ c \leq b\ --> f'\ c \leq g'\ c;$
$\forall x.\ DERIV\ f\ x :> f'\ x;\ \forall x.\ DERIV\ g\ x :> g'\ x\ |]$
$==> f\ b - f\ a \leq g\ b - g\ a$
⟨proof⟩

**end**

# 44 Log: Logarithms: Standard Version

**theory** *Log*
**imports** *Transcendental*
**begin**

**definition**
*powr* :: [*real*,*real*] => *real* (**infixr** *powr 80*) **where**
— exponentation with real exponent
$x\ powr\ a = exp(a * ln\ x)$

**definition**
*log* :: [*real*,*real*] => *real* **where**
— logarithm of $x$ to base $a$
$log\ a\ x = ln\ x\ /\ ln\ a$

**lemma** *powr-one-eq-one* [*simp*]: *1 powr a = 1*
⟨*proof*⟩

**lemma** *powr-zero-eq-one* [*simp*]: *x powr 0 = 1*
⟨*proof*⟩

**lemma** *powr-one-gt-zero-iff* [*simp*]: (*x powr 1 = x*) = (*0 < x*)
⟨*proof*⟩
**declare** *powr-one-gt-zero-iff* [*THEN iffD2*, *simp*]

**lemma** *powr-mult*:
[| *0 < x*; *0 < y* |] ==> (*x * y*) *powr a* = (*x powr a*) * (*y powr a*)
⟨*proof*⟩

**lemma** *powr-gt-zero* [*simp*]: *0 < x powr a*
⟨*proof*⟩

**lemma** *powr-ge-pzero* [*simp*]: *0 <= x powr y*
⟨*proof*⟩

**lemma** *powr-not-zero* [*simp*]: *x powr a ≠ 0*
⟨*proof*⟩

**lemma** *powr-divide*:
[| *0 < x*; *0 < y* |] ==> (*x / y*) *powr a* = (*x powr a*)/(*y powr a*)
⟨*proof*⟩

**lemma** *powr-divide2*: *x powr a / x powr b = x powr* (*a − b*)
⟨*proof*⟩

**lemma** *powr-add*: *x powr* (*a + b*) = (*x powr a*) * (*x powr b*)
⟨*proof*⟩

**lemma** *powr-powr*: (*x powr a*) *powr b = x powr* (*a * b*)
⟨*proof*⟩

**lemma** *powr-powr-swap*: (*x powr a*) *powr b* = (*x powr b*) *powr a*
⟨*proof*⟩

**lemma** *powr-minus*: *x powr* (−*a*) = *inverse* (*x powr a*)
⟨*proof*⟩

**lemma** *powr-minus-divide*: *x powr* (−*a*) = *1*/(*x powr a*)
⟨*proof*⟩

**lemma** *powr-less-mono*: [| *a < b*; *1 < x* |] ==> *x powr a < x powr b*
⟨*proof*⟩

**lemma** *powr-less-cancel*: $[| \ x \ powr \ a < x \ powr \ b; \ 1 < x \ |] ==> a < b$
⟨*proof*⟩

**lemma** *powr-less-cancel-iff* [*simp*]: $1 < x ==> (x \ powr \ a < x \ powr \ b) = (a < b)$
⟨*proof*⟩

**lemma** *powr-le-cancel-iff* [*simp*]: $1 < x ==> (x \ powr \ a \leq x \ powr \ b) = (a \leq b)$
⟨*proof*⟩

**lemma** *log-ln*: $ln \ x = log \ (exp(1)) \ x$
⟨*proof*⟩

**lemma** *powr-log-cancel* [*simp*]:
$[| \ 0 < a; \ a \neq 1; \ 0 < x \ |] ==> a \ powr \ (log \ a \ x) = x$
⟨*proof*⟩

**lemma** *log-powr-cancel* [*simp*]: $[| \ 0 < a; \ a \neq 1 \ |] ==> log \ a \ (a \ powr \ y) = y$
⟨*proof*⟩

**lemma** *log-mult*:
$[| \ 0 < a; \ a \neq 1; \ 0 < x; \ 0 < y \ |]$
$==> log \ a \ (x * y) = log \ a \ x + log \ a \ y$
⟨*proof*⟩

**lemma** *log-eq-div-ln-mult-log*:
$[| \ 0 < a; \ a \neq 1; \ 0 < b; \ b \neq 1; \ 0 < x \ |]$
$==> log \ a \ x = (ln \ b / ln \ a) * log \ b \ x$
⟨*proof*⟩

Base 10 logarithms

**lemma** *log-base-10-eq1*: $0 < x ==> log \ 10 \ x = (ln \ (exp \ 1) \ / \ ln \ 10) * ln \ x$
⟨*proof*⟩

**lemma** *log-base-10-eq2*: $0 < x ==> log \ 10 \ x = (log \ 10 \ (exp \ 1)) * ln \ x$
⟨*proof*⟩

**lemma** *log-one* [*simp*]: $log \ a \ 1 = 0$
⟨*proof*⟩

**lemma** *log-eq-one* [*simp*]: $[| \ 0 < a; \ a \neq 1 \ |] ==> log \ a \ a = 1$
⟨*proof*⟩

**lemma** *log-inverse*:
$[| \ 0 < a; \ a \neq 1; \ 0 < x \ |] ==> log \ a \ (inverse \ x) = - \ log \ a \ x$
⟨*proof*⟩

**lemma** *log-divide*:
$[|0 < a; \ a \neq 1; \ 0 < x; \ 0 < y|] ==> log \ a \ (x/y) = log \ a \ x - log \ a \ y$
⟨*proof*⟩

**lemma** *log-less-cancel-iff* [*simp*]:
[| *1* < *a*; *0* < *x*; *0* < *y* |] ==> (*log a x* < *log a y*) = (*x* < *y*)
⟨*proof*⟩

**lemma** *log-le-cancel-iff* [*simp*]:
[| *1* < *a*; *0* < *x*; *0* < *y* |] ==> (*log a x* ≤ *log a y*) = (*x* ≤ *y*)
⟨*proof*⟩

**lemma** *powr-realpow*: *0* < *x* ==> *x powr* (*real n*) = *x*ˆ*n*
⟨*proof*⟩

**lemma** *powr-realpow2*: *0* <= *x* ==> *0* < *n* ==> *x*ˆ*n* = (*if* (*x* = *0*) *then 0 else x powr* (*real n*))
⟨*proof*⟩

**lemma** *ln-pwr*: *0* < *x* ==> *0* < *y* ==> *ln*(*x powr y*) = *y* ∗ *ln x*
⟨*proof*⟩

**lemma** *ln-bound*: *1* <= *x* ==> *ln x* <= *x*
⟨*proof*⟩

**lemma** *powr-mono*: *a* <= *b* ==> *1* <= *x* ==> *x powr a* <= *x powr b*
⟨*proof*⟩

**lemma** *ge-one-powr-ge-zero*: *1* <= *x* ==> *0* <= *a* ==> *1* <= *x powr a*
⟨*proof*⟩

**lemma** *powr-less-mono2*: *0* < *a* ==> *0* < *x* ==> *x* < *y* ==> *x powr a* < *y powr a*
⟨*proof*⟩

**lemma** *powr-less-mono2-neg*: *a* < *0* ==> *0* < *x* ==> *x* < *y* ==> *y powr a* < *x powr a*
⟨*proof*⟩

**lemma** *powr-mono2*: *0* <= *a* ==> *0* < *x* ==> *x* <= *y* ==> *x powr a* <= *y powr a*
⟨*proof*⟩

**lemma** *ln-powr-bound*: *1* <= *x* ==> *0* < *a* ==> *ln x* <= (*x powr a*) / *a*
⟨*proof*⟩

**lemma** *ln-powr-bound2*: *1* < *x* ==> *0* < *a* ==> (*ln x*) *powr a* <= (*a powr a*) ∗ *x*
⟨*proof*⟩

**lemma** *LIMSEQ-neg-powr*: *0* < *s* ==> (%*x*. (*real x*) *powr* − *s*) −−−−> *0*

⟨*proof*⟩

**end**

# 45 HLog: Logarithms: Non-Standard Version

**theory** *HLog*
**imports** *Log HTranscendental*
**begin**

**lemma** *epsilon-ge-zero* [*simp*]: *0* ≤ *epsilon*
⟨*proof*⟩

**lemma** *hpfinite-witness*: *epsilon* : {*x*. *0* ≤ *x* & *x* : *HFinite*}
⟨*proof*⟩

**definition**
*powhr* :: [*hypreal*,*hypreal*] => *hypreal* (**infixr** *powhr 80*) **where**
*x powhr a* = *starfun2* (*op powr*) *x a*

**definition**
*hlog* :: [*hypreal*,*hypreal*] => *hypreal* **where**
*hlog a x* = *starfun2 log a x*

**declare** *powhr-def* [*transfer-unfold*]
**declare** *hlog-def* [*transfer-unfold*]

**lemma** *powhr*: (*star-n X*) *powhr* (*star-n Y*) = *star-n* (%*n*. (*X n*) *powr* (*Y n*))
⟨*proof*⟩

**lemma** *powhr-one-eq-one* [*simp*]: !!*a*. *1 powhr a* = *1*
⟨*proof*⟩

**lemma** *powhr-mult*:
!!*a x y*. [| *0* < *x*; *0* < *y* |] ==> (*x* ∗ *y*) *powhr a* = (*x powhr a*) ∗ (*y powhr a*)
⟨*proof*⟩

**lemma** *powhr-gt-zero* [*simp*]: !!*a x*. *0* < *x powhr a*
⟨*proof*⟩

**lemma** *powhr-not-zero* [*simp*]: *x powhr a* ≠ *0*
⟨*proof*⟩

**lemma** *powhr-divide*:
!!*a x y*. [| *0* < *x*; *0* < *y* |] ==> (*x* / *y*) *powhr a* = (*x powhr a*)/(*y powhr a*)

⟨*proof*⟩

**lemma** *powhr-add*: !!$a$ $b$ $x$. $x$ *powhr* $(a + b) = (x$ *powhr* $a) * (x$ *powhr* $b)$
⟨*proof*⟩

**lemma** *powhr-powhr*: !!$a$ $b$ $x$. $(x$ *powhr* $a)$ *powhr* $b = x$ *powhr* $(a * b)$
⟨*proof*⟩

**lemma** *powhr-powhr-swap*: !!$a$ $b$ $x$. $(x$ *powhr* $a)$ *powhr* $b = (x$ *powhr* $b)$ *powhr* $a$
⟨*proof*⟩

**lemma** *powhr-minus*: !!$a$ $x$. $x$ *powhr* $(-a) =$ *inverse* $(x$ *powhr* $a)$
⟨*proof*⟩

**lemma** *powhr-minus-divide*: $x$ *powhr* $(-a) = 1/(x$ *powhr* $a)$
⟨*proof*⟩

**lemma** *powhr-less-mono*: !!$a$ $b$ $x$. [| $a < b$; $1 < x$ |] ==> $x$ *powhr* $a < x$ *powhr* $b$
⟨*proof*⟩

**lemma** *powhr-less-cancel*: !!$a$ $b$ $x$. [| $x$ *powhr* $a < x$ *powhr* $b$; $1 < x$ |] ==> $a < b$
⟨*proof*⟩

**lemma** *powhr-less-cancel-iff* [*simp*]:
  $1 < x$ ==> $(x$ *powhr* $a < x$ *powhr* $b) = (a < b)$
⟨*proof*⟩

**lemma** *powhr-le-cancel-iff* [*simp*]:
  $1 < x$ ==> $(x$ *powhr* $a \leq x$ *powhr* $b) = (a \leq b)$
⟨*proof*⟩

**lemma** *hlog*:
  *hlog* (*star-n* $X$) (*star-n* $Y$) =
   *star-n* (%$n$. *log* ($X$ $n$) ($Y$ $n$))
⟨*proof*⟩

**lemma** *hlog-starfun-ln*: !!$x$. ( $*f*$ *ln*) $x$ = *hlog* (( $*f*$ *exp*) $1$) $x$
⟨*proof*⟩

**lemma** *powhr-hlog-cancel* [*simp*]:
  !!$a$ $x$. [| $0 < a$; $a \neq 1$; $0 < x$ |] ==> $a$ *powhr* (*hlog* $a$ $x$) $= x$
⟨*proof*⟩

**lemma** *hlog-powhr-cancel* [*simp*]:
  !!$a$ $y$. [| $0 < a$; $a \neq 1$ |] ==> *hlog* $a$ ($a$ *powhr* $y$) $= y$
⟨*proof*⟩

**lemma** *hlog-mult*:
  !!$a$ $x$ $y$. [| $0 < a$; $a \neq 1$; $0 < x$; $0 < y$ |]

$==> hlog\ a\ (x * y) = hlog\ a\ x + hlog\ a\ y$
⟨*proof*⟩

**lemma** *hlog-as-starfun*:
$!!a\ x.\ [|\ 0 < a;\ a \neq 1\ |] ==> hlog\ a\ x = (\ *f*\ ln)\ x\ /\ (\ *f*\ ln)\ a$
⟨*proof*⟩

**lemma** *hlog-eq-div-starfun-ln-mult-hlog*:
$!!a\ b\ x.\ [|\ 0 < a;\ a \neq 1;\ 0 < b;\ b \neq 1;\ 0 < x\ |]$
$==> hlog\ a\ x = ((\ *f*\ ln)\ b/(\ *f*ln)\ a) * hlog\ b\ x$
⟨*proof*⟩

**lemma** *powhr-as-starfun*: $!!a\ x.\ x\ powhr\ a = (\ *f*\ exp)\ (a * (\ *f*\ ln)\ x)$
⟨*proof*⟩

**lemma** *HInfinite-powhr*:
$[|\ x : HInfinite;\ 0 < x;\ a : HFinite - Infinitesimal;$
$0 < a\ |] ==> x\ powhr\ a : HInfinite$
⟨*proof*⟩

**lemma** *hlog-hrabs-HInfinite-Infinitesimal*:
$[|\ x : HFinite - Infinitesimal;\ a : HInfinite;\ 0 < a\ |]$
$==> hlog\ a\ (abs\ x) : Infinitesimal$
⟨*proof*⟩

**lemma** *hlog-HInfinite-as-starfun*:
$[|\ a : HInfinite;\ 0 < a\ |] ==> hlog\ a\ x = (\ *f*\ ln)\ x\ /\ (\ *f*\ ln)\ a$
⟨*proof*⟩

**lemma** *hlog-one* [*simp*]: $!!a.\ hlog\ a\ 1 = 0$
⟨*proof*⟩

**lemma** *hlog-eq-one* [*simp*]: $!!a.\ [|\ 0 < a;\ a \neq 1\ |] ==> hlog\ a\ a = 1$
⟨*proof*⟩

**lemma** *hlog-inverse*:
$[|\ 0 < a;\ a \neq 1;\ 0 < x\ |] ==> hlog\ a\ (inverse\ x) = -\ hlog\ a\ x$
⟨*proof*⟩

**lemma** *hlog-divide*:
$[|\ 0 < a;\ a \neq 1;\ 0 < x;\ 0 < y|] ==> hlog\ a\ (x/y) = hlog\ a\ x - hlog\ a\ y$
⟨*proof*⟩

**lemma** *hlog-less-cancel-iff* [*simp*]:
$!!a\ x\ y.\ [|\ 1 < a;\ 0 < x;\ 0 < y\ |] ==> (hlog\ a\ x < hlog\ a\ y) = (x < y)$
⟨*proof*⟩

**lemma** *hlog-le-cancel-iff* [*simp*]:
$[|\ 1 < a;\ 0 < x;\ 0 < y\ |] ==> (hlog\ a\ x \leq hlog\ a\ y) = (x \leq y)$

⟨*proof*⟩

**end**

**theory** *Hyperreal*
**imports** *Ln Poly Taylor Integration HLog*
**begin**

**end**

# 46 Complex-Main: Comprehensive Complex Theory

**theory** *Complex-Main*
**imports** *CLim ../Hyperreal/Hyperreal*
**begin**

**end**