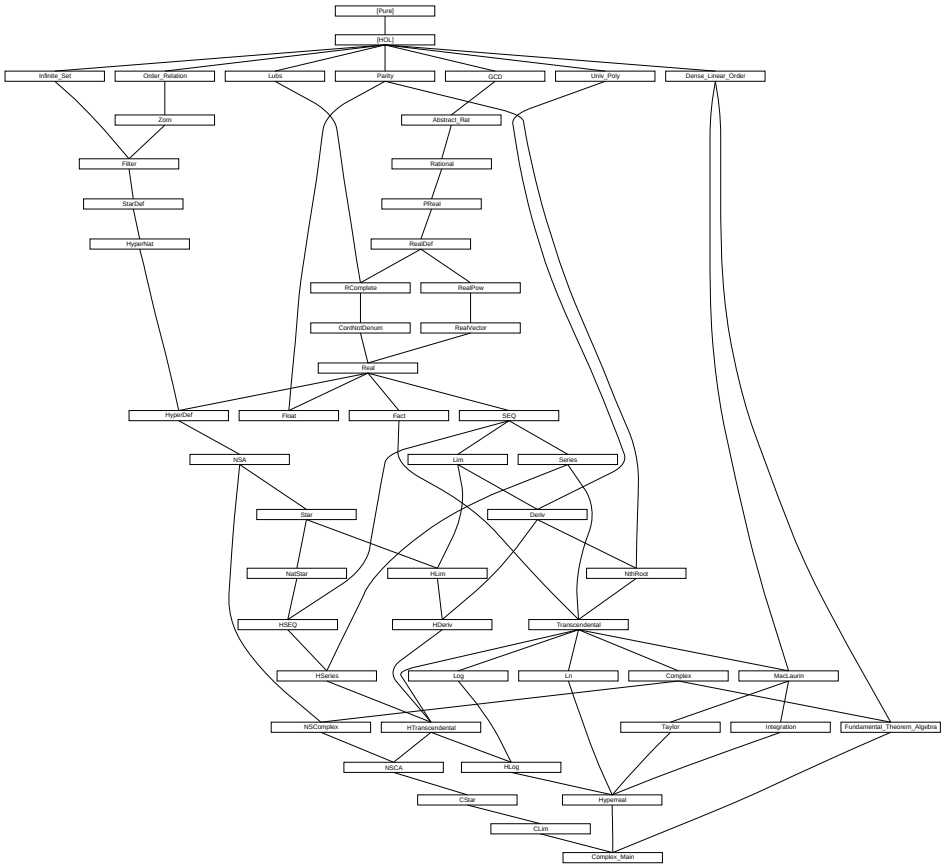
Isabelle/HOL-Complex — Higher-Order Logic with Complex Numbers

June 8, 2008

Contents



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 *ATP-Linkup*
begin

See [**?**].

2.1 Specification of GCD on nats

definition
is-gcd :: *nat* ⇒ *nat* ⇒ *nat* ⇒ *bool* **where** — *gcd* as a relation
is-gcd p m n ⟷ *p dvd m* ∧ *p dvd n* ∧
(∀ *d*. *d dvd m* ⟶ *d dvd n* ⟶ *d dvd p*)

Uniqueness

lemma *is-gcd-unique*: *is-gcd m a b* ⟹ *is-gcd n a b* ⟹ *m* = *n*
⟨*proof*⟩

Connection to divides relation

lemma *is-gcd-dvd*: *is-gcd m a b* ⟹ *k dvd a* ⟹ *k dvd b* ⟹ *k dvd m*
⟨*proof*⟩

Commutativity

lemma *is-gcd-commute*: *is-gcd k m n* = *is-gcd k n m*
⟨*proof*⟩

2.2 GCD on nat by Euclid's algorithm

fun
gcd :: *nat* × *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 ⋀*m*. *P m 0*
and ⋀*m n*. *0* < *n* ⟹ *P n* (*m mod n*) ⟹ *P m n*
shows *P m n*
⟨*proof*⟩

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

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

lemma *gcd-non-0*: *n* > *0* ⟹ *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* ⟹ *k dvd n* ⟹ *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*) ⟷ *k dvd m* ∧ *k dvd n*
⟨*proof*⟩

lemma *gcd-zero*: *gcd* (*m*, *n*) = *0* ⟷ *m* = *0* ∧ *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* ==> *k dvd m* * *n* ==> *k dvd m*
⟨*proof*⟩

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

lemma *gcd-mult-cancel*: *gcd* (*k*, *n*) = *1* ==> *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* ≠ *0* ∨ *b* ≠ *0*
shows *gcd* (*a div gcd*(*a*,*b*), *b div gcd*(*a*,*b*)) = *1*
⟨*proof*⟩

2.4 LCM defined by GCD

definition
lcm :: *nat* × *nat* ⇒ *nat*
where
lcm-prim-def: *lcm* = (λ(*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 ⟹ i dvd k * j ⟹ 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 ≠ 0 ∨ b ≠ 0*
shows *igcd (a div (igcd a b)) (b div (igcd a b)) = 1*
⟨*proof*⟩

definition *ilcm* = (λ*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* $(m,n) > 0$
⟨*proof*⟩

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

end

3 Abstract-Rat: Abstract rational numbers

theory *Abstract-Rat*
imports *GCD*
begin

types *Num* = *int* × *int*

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

abbreviation
  Numi-syn :: *int* ⇒ *Num* (-_N)
where $i_N \equiv (i, 1)$

definition
  isnormNum :: *Num* ⇒ *bool*
where
  isnormNum = (λ(*a*,*b*). (*if a* = 0 *then b* = 0 *else b* > 0 ∧ *igcd a b* = 1))

definition
  normNum :: *Num* ⇒ *Num*
where
  normNum = (λ(*a*,*b*). (*if a*=0 ∨ *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*)
⟨*proof*⟩

Arithmetic over Num

definition
Nadd :: *Num* ⇒ *Num* ⇒ *Num* (**infixl** $+_N$ *60*)
where
Nadd = (λ(*a*,*b*) (*a*′,*b*′). *if a* = *0* ∨ *b* = *0 then normNum*(*a*′,*b*′)
else if a′=*0* ∨ *b*′ = *0 then normNum*(*a*,*b*)
else normNum(*a*∗*b*′ + *b*∗*a*′, *b*∗*b*′))

definition
Nmul :: *Num* ⇒ *Num* ⇒ *Num* (**infixl** $*_N$ *60*)
where
Nmul = (λ(*a*,*b*) (*a*′,*b*′). *let g* = *igcd* (*a*∗*a*′) (*b*∗*b*′)
in (*a*∗*a*′ *div g*, *b*∗*b*′ *div g*))

definition
Nneg :: *Num* ⇒ *Num* ($^{\sim}_N$)
where
Nneg ≡ (λ(*a*,*b*). (−*a*,*b*))

definition
Nsub :: *Num* ⇒ *Num* ⇒ *Num* (**infixl** $-_N$ *60*)
where
Nsub = (λ*a b*. *a* $+_N$ $^{\sim}_N$ *b*)

definition
Ninv :: *Num* ⇒ *Num*
where
Ninv ≡ λ(*a*,*b*). *if a* < *0 then* (−*b*, |*a*|) *else* (*b*,*a*)

definition
Ndiv :: *Num* ⇒ *Num* ⇒ *Num* (**infixl** $\div_N$ *60*)
where
Ndiv ≡ λ*a b*. *a* $*_N$ *Ninv b*

lemma *Nneg-normN*[*simp*]: *isnormNum x* ⟹ *isnormNum* ($^{\sim}_N$ *x*)
⟨*proof*⟩
lemma *Nadd-normN*[*simp*]: *isnormNum* (*x* $+_N$ *y*)
⟨*proof*⟩
lemma *Nsub-normN*[*simp*]: ⟦ *isnormNum y*⟧ ⟹ *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* ⟹ *isnormNum* (*Ninv x*)
⟨*proof*⟩

lemma *isnormNum-int*[*simp*]:
isnormNum 0_N *isnormNum* (*1*::*int*)$_N$ *i* ≠ *0* ⟹ *isnormNum* i_N

⟨*proof*⟩

Relations over Num

definition
Nlt0:: *Num* ⇒ *bool* (*0*>$_N$)
where
Nlt0 = (λ(*a*,*b*). *a* < *0*)

definition
Nle0:: *Num* ⇒ *bool* (*0*≥$_N$)
where
Nle0 = (λ(*a*,*b*). *a* ≤ *0*)

definition
Ngt0:: *Num* ⇒ *bool* (*0*<$_N$)
where
Ngt0 = (λ(*a*,*b*). *a* > *0*)

definition
Nge0:: *Num* ⇒ *bool* (*0*≤$_N$)
where
Nge0 = (λ(*a*,*b*). *a* ≥ *0*)

definition
Nlt :: *Num* ⇒ *Num* ⇒ *bool* (**infix** <$_N$ *55*)
where
Nlt = (λ*a b*. *0*>$_N$ (*a* −$_N$ *b*))

definition
Nle :: *Num* ⇒ *Num* ⇒ *bool* (**infix** ≤$_N$ *55*)
where
Nle = (λ*a b*. *0*≥$_N$ (*a* −$_N$ *b*))

definition
INum = (λ(*a*,*b*). *of-int a* / *of-int b*)

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

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

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

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

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

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

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

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

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

⟨*proof*⟩

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

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

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

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

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

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

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

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

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

lemma *Nmul0*[*simp*]: $c *_N 0_N = 0_N \quad 0_N *_N c = 0_N$
⟨*proof*⟩

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

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

end

4 Rational: Rational numbers

theory *Rational*
imports ~~/*src*/*HOL*/*Library*/*Abstract-Rat*
uses (*rat-arith.ML*)
begin

4.1 Rational numbers

4.1.1 Equivalence of fractions

definition
fraction :: (*int* × *int*) *set* **where**
fraction = {*x*. *snd x* ≠ 0}

definition
ratrel :: ((*int* × *int*) × (*int* × *int*)) *set* **where**
ratrel = {(*x*,*y*). *snd x* ≠ 0 ∧ *snd y* ≠ 0 ∧ *fst x* ∗ *snd y* = *fst y* ∗ *snd x*}

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

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

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* ≠ *0*; *snd y* ≠ *0*⟧
⟹ (*ratrel* `` {*x*} = *ratrel* `` {*y*}) = ((*x*,*y*) ∈ *ratrel*)
⟨*proof*⟩

4.1.2 The type of rational numbers

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

lemma *ratrel-in-Rat* [*simp*]: *snd x* ≠ *0* ⟹ *ratrel*``{*x*} ∈ *Rat*
⟨*proof*⟩

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

definition
Fract :: *int* ⇒ *int* ⇒ *rat* **where**
[*code func del*]: *Fract a b* = *Abs-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* ≠ *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

instantiation *rat* :: {*zero*, *one*, *plus*, *minus*, *uminus*, *times*, *inverse*, *ord*, *abs*, *sgn*}
begin

definition
Zero-rat-def [*code func del*]: $0 = Fract\ 0\ 1$

definition
One-rat-def [*code func del*]: $1 = Fract\ 1\ 1$

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

definition
minus-rat-def [*code func del*]:

$- q = \mathit{Abs\text{-}Rat}\ (\bigcup x \in \mathit{Rep\text{-}Rat}\ q.\ \mathit{ratrel}``\{(-\mathit{fst}\ x,\ \mathit{snd}\ x)\})$

definition
diff-rat-def [*code func del*]: $q - r = q + -\ (r::\mathit{rat})$

definition
mult-rat-def [*code func del*]:
$q * r =$
$\mathit{Abs\text{-}Rat}\ (\bigcup x \in \mathit{Rep\text{-}Rat}\ q.\ \bigcup y \in \mathit{Rep\text{-}Rat}\ r.$
$\mathit{ratrel}``\{(\mathit{fst}\ x * \mathit{fst}\ y,\ \mathit{snd}\ x * \mathit{snd}\ y)\})$

definition
inverse-rat-def [*code func del*]:
$\mathit{inverse}\ q =$
$\mathit{Abs\text{-}Rat}\ (\bigcup x \in \mathit{Rep\text{-}Rat}\ q.$
$\mathit{ratrel}``\{\mathit{if}\ \mathit{fst}\ x{=}0\ \mathit{then}\ (0,1)\ \mathit{else}\ (\mathit{snd}\ x,\ \mathit{fst}\ x)\})$

definition
divide-rat-def [*code func del*]: $q\ /\ r = q * \mathit{inverse}\ (r::\mathit{rat})$

definition
le-rat-def [*code func del*]:
$q \le r \longleftrightarrow \mathit{contents}\ (\bigcup x \in \mathit{Rep\text{-}Rat}\ q.\ \bigcup y \in \mathit{Rep\text{-}Rat}\ r.$
$\{(\mathit{fst}\ x * \mathit{snd}\ y){*}(\mathit{snd}\ x * \mathit{snd}\ y) \le (\mathit{fst}\ y * \mathit{snd}\ x){*}(\mathit{snd}\ x * \mathit{snd}\ y)\})$

definition
less-rat-def [*code func del*]: $z < (w::\mathit{rat}) \longleftrightarrow z \le w \wedge z \neq w$

definition
abs-rat-def: $|q| = (\mathit{if}\ q < 0\ \mathit{then}\ -q\ \mathit{else}\ (q::\mathit{rat}))$

definition
sgn-rat-def: $\mathit{sgn}\ (q::\mathit{rat}) = (\mathit{if}\ q{=}0\ \mathit{then}\ 0\ \mathit{else\ if}\ 0{<}q\ \mathit{then}\ 1\ \mathit{else}\ -\ 1)$

instance ⟨*proof*⟩

end

instantiation *rat* :: *power*
begin

primrec *power-rat*
where
rat-power-0: $q\ \hat{}\ 0 = (1::\mathit{rat})$
| *rat-power-Suc*: $q\ \hat{}\ (\mathit{Suc}\ n) = (q::\mathit{rat}) * (q\ \hat{}\ n)$

instance ⟨*proof*⟩

end

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

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

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

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

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

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

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

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

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

theorem *abs-rat*: $b \neq 0 \Longrightarrow |Fract\ a\ b| = Fract\ |a|\ |b|$
⟨*proof*⟩

4.1.5 The ordered field of rational numbers

instance *rat* :: *field*
⟨*proof*⟩

instance *rat* :: *linorder*
⟨*proof*⟩

instantiation *rat* :: *distrib-lattice*
begin

definition
(*inf* :: *rat* ⇒ *rat* ⇒ *rat*) = *min*

definition
(*sup* :: *rat* ⇒ *rat* ⇒ *rat*) = *max*

instance
⟨*proof*⟩

end

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

instance *rat* :: *division-by-zero*
⟨*proof*⟩

instance *rat* :: *recpower*
⟨*proof*⟩

4.2 Various Other Results

lemma *minus-rat-cancel* [*simp*]: *b* ≠ *0* ==> *Fract* (−*a*) (−*b*) = *Fract a b*
⟨*proof*⟩

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

lemma *zero-less-Fract-iff*:
0 < *b* ==> (*0* < *Fract a b*) = (*0* < *a*)
⟨*proof*⟩

lemma *Fract-add-one*: *n* ≠ *0* ==> *Fract* (*m* + *n*) *n* = *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

instantiation *rat* :: *number-ring*
begin

definition
rat-number-of-def [*code func del*]: *number-of w* = (*of-int w* :: *rat*)

instance
⟨*proof*⟩

end

⟨*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-rat :: *rat* ⇒ 'a::*field-char-0*
where
[*code func del*]: *of-rat q* = *contents* (⋃(*a*,*b*) ∈ *Rep-Rat q*. {*of-int a* / *of-int b*})

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

lemma *of-rat-rat*:
b ≠ *0* ⟹ *of-rat* (*Fract a b*) = *of-int a* / *of-int b*
⟨*proof*⟩

lemma *of-rat-0* [*simp*]: *of-rat 0* = *0*
⟨*proof*⟩

lemma *of-rat-1* [*simp*]: *of-rat 1* = *1*
⟨*proof*⟩

lemma *of-rat-add*: *of-rat* (*a* + *b*) = *of-rat a* + *of-rat b*
⟨*proof*⟩

lemma *of-rat-minus*: *of-rat* (− *a*) = − *of-rat a*
⟨*proof*⟩

lemma *of-rat-diff*: *of-rat* (*a* − *b*) = *of-rat a* − *of-rat b*
⟨*proof*⟩

lemma *of-rat-mult*: *of-rat* (*a* ∗ *b*) = *of-rat a* ∗ *of-rat b*
⟨*proof*⟩

lemma *nonzero-of-rat-inverse*:
a ≠ *0* ⟹ *of-rat* (*inverse a*) = *inverse* (*of-rat a*)
⟨*proof*⟩

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

lemma *nonzero-of-rat-divide*:
b ≠ *0* ⟹ *of-rat* (*a* / *b*) = *of-rat a* / *of-rat b*
⟨*proof*⟩

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

lemma *of-rat-power*:
(*of-rat* (*a* ^ *n*)::′*a*::{*field-char-0*,*recpower*}) = *of-rat a* ^ *n*
⟨*proof*⟩

lemma *of-rat-eq-iff* [*simp*]: (*of-rat a* = *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*]: *of-rat* = (*id* :: *rat* ⇒ *rat*)
⟨*proof*⟩

Collapse nested embeddings

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

lemma *of-rat-of-int-eq* [*simp*]: *of-rat* (*of-int z*) = *of-int z*
⟨*proof*⟩

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

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

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

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

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

4.5 Implementation of rational numbers as pairs of integers

definition
Rational :: *int* × *int* ⇒ *rat*
where
Rational = INum

code-datatype *Rational*

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

lemma *Rational-zero* [*simp*]: *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*⟩
declare *zero-rat-code* [*symmetric*, *code post*]

lemma *one-rat-code* [*code*, *code unfold*]:
1 = *Rational* 1_N ⟨*proof*⟩
declare *one-rat-code* [*symmetric*, *code post*]

lemma [*code unfold*, *symmetric*, *code post*]:
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* ≠ *0*) *k l*
⟨*proof*⟩

lemma [*code*]:
Fract' True k l = (*if l* ≠ *0 then Rational* (*k*, *l*) *else Fract 1 0*)
⟨*proof*⟩

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

instantiation *rat* :: *eq*
begin

definition [*code func del*]: *eq-class.eq* (*r*::*rat*) *s* ⟷ *r* = *s*

instance ⟨*proof*⟩

lemma *rat-eq-code* [*code*]: *eq-class.eq* (*Rational x*) (*Rational y*) ⟷ *eq-class.eq* (*normNum x*) (*normNum y*)
⟨*proof*⟩

end

lemma *rat-less-eq-code* [*code*]: *Rational x* ≤ *Rational y* ⟷ *normNum x* ≤$_N$ *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* (~$_N$ *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* ÷$_N$ *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 @{term op / :: rat ⇒ rat ⇒ rat} $
      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;
    val r = (if one-of [true, false] then p' else ~ p',
      if p' = 0 then 0 else q')
  in
    (r, fn () => term-of-rat r)
  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*: $\exists x.\ a+x = (b::rat)$
⟨*proof*⟩

definition
cut :: *rat set* => *bool* **where**
cut A = ({} ⊂ *A* &
A < {*r*. 0 < *r*} &
(∀ *y* ∈ *A*. ((∀ *z*. 0<*z* & *z* < *y* −−> *z* ∈ *A*) & (∃ *u* ∈ *A*. *y* < *u*))))

lemma *cut-of-rat*:
assumes *q*: 0 < *q* **shows** *cut* {*r*::*rat*. 0 < *r* & *r* < *q*} (**is** *cut ?A*)
⟨*proof*⟩

typedef *preal* = {*A*. *cut A*}
⟨*proof*⟩

definition
preal-of-rat :: *rat* => *preal* **where**
preal-of-rat q = *Abs-preal* {*x*::*rat*. 0 < *x* & *x* < *q*}

definition
psup :: *preal set* => *preal* **where**
psup P = *Abs-preal* (⋃ *X* ∈ *P*. *Rep-preal X*)

definition
add-set :: [*rat set*,*rat set*] => *rat set* **where**
add-set A B = {*w*. ∃ *x* ∈ *A*. ∃ *y* ∈ *B*. *w* = *x* + *y*}

definition
diff-set :: [*rat set*,*rat set*] => *rat set* **where**
diff-set A B = {*w*. ∃ *x*. 0 < *w* & 0 < *x* & *x* ∉ *B* & *x* + *w* ∈ *A*}

definition
mult-set :: [*rat set*,*rat set*] => *rat set* **where**
mult-set A B = {*w*. ∃ *x* ∈ *A*. ∃ *y* ∈ *B*. *w* = *x* ∗ *y*}

definition
inverse-set :: *rat set* => *rat set* **where**
inverse-set A = {*x*. ∃ *y*. 0 < *x* & *x* < *y* & *inverse y* ∉ *A*}

instantiation *preal* :: {*ord*, *plus*, *minus*, *times*, *inverse*, *one*}
begin

definition
preal-less-def:
R < *S* == *Rep-preal R* < *Rep-preal S*

definition
preal-le-def:
R ≤ *S* == *Rep-preal R* ⊆ *Rep-preal S*

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

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

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

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

definition $R\ /\ S = R * inverse\ (S::preal)$

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

instance ⟨*proof*⟩

end

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*⟩

instantiation *preal* :: *distrib-lattice*
begin

definition
$(inf :: preal \Rightarrow preal \Rightarrow preal) = min$

definition
$(sup :: preal \Rightarrow preal \Rightarrow preal) = max$

instance
⟨*proof*⟩

end

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*:
$[|A \in preal;\ B \in preal|] ==> \{\} \subset add\text{-}set\ A\ B$
⟨*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*:
$[|A \in preal; B \in preal|] ==> \exists q{>}0.\ q \notin add\text{-}set\ A\ B$
⟨*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-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-set* A B
⟨*proof*⟩

lemma *mult-set-not-rat-set*:
assumes A: $A \in$ *preal*
and B: $B \in$ *preal*
shows *mult-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-set* A B; $0 < z$; $z < u$|]
==> $z \in$ *mult-set* A B
⟨*proof*⟩

Part 4 of Dedekind sections definition

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

lemma *mem-mult-set*:
[|$A \in$ *preal*; $B \in$ *preal*|] ==> *mult-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*:
$(z \in Rep\text{-}preal(R+S)) = (\exists x \in Rep\text{-}preal\ R.\ \exists y \in Rep\text{-}preal\ S.\ z = x + y)$
⟨*proof*⟩

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

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

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

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

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

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

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

5.6 Existence of Inverse, a Positive Real

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

Part 1 of Dedekind sections definition

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

Part 2 of Dedekind sections definition

lemma *preal-not-mem-inverse-set-Ex*:
assumes *A*: $A \in preal$ **shows** $\exists q.\ 0 < q\ \&\ q \notin inverse\text{-}set\ A$
⟨*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*:
$[|A \in preal;\ u \in inverse\text{-}set\ A;\ 0 < z;\ z < u|] \Longrightarrow z \in inverse\text{-}set\ A$
⟨*proof*⟩

Part 4 of Dedekind sections definition

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

lemma *mem-inverse-set*:
$A \in preal \Longrightarrow inverse\text{-}set\ A \in preal$
⟨*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 $0 \le z$
shows $\exists b \in A.\ b + (of\text{-}int\ z) * u \in A$
⟨*proof*⟩

lemma *Gleason9-34-contra*:
assumes *A*: $A \in preal$
shows $[|\forall x \in A.\ x + u \in A;\ 0 < u;\ 0 < y;\ y \notin A|] \Longrightarrow False$
⟨*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*:
$(z \in Rep\text{-}preal(inverse\ R)) =$
$(0 < z \wedge (\exists\, y.\ z < y \wedge inverse\ y \notin Rep\text{-}preal\ R))$
⟨*proof*⟩

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

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

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

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

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

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

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

Theorems needing *Gleason9-34*

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

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

lemma *Rep-preal-sum-not-eq*: *Rep-preal* $(R + S) \neq$ *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*:
$R < S \Longrightarrow$ *diff-set* (*Rep-preal S*) (*Rep-preal R*) $\in$ *preal*
⟨*proof*⟩

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

proving that $R + D \leq S$

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

lemma *less-add-left-le1*:
$R <$ (*S*::*preal*) $\Longrightarrow R + (S-R) \leq S$
⟨*proof*⟩

5.11 proving that $S \leq R + D$ — trickier

lemma *lemma-sum-mem-Rep-preal-ex*:
$x \in$ *Rep-preal S* $\Longrightarrow \exists e.\ 0 < e\ \&\ x + e \in$ *Rep-preal S*
⟨*proof*⟩

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

lemma *less-add-left-le2*: $R <$ (*S*::*preal*) $\Longrightarrow S \leq R + (S-R)$
⟨*proof*⟩

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

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

lemma *preal-add-less2-mono1*: $R <$ (*S*::*preal*) $\Longrightarrow R + T < S + T$

⟨*proof*⟩

lemma *preal-add-less2-mono2*: $R < (S::preal) \Longrightarrow T + R < T + S$
⟨*proof*⟩

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

lemma *preal-add-left-less-cancel*: $T + R < T + S \Longrightarrow R < (S::preal)$
⟨*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*:
$[\!| x1 < y1; x2 < y2 |\!] \Longrightarrow x1 + x2 < y1 + (y2::preal)$
⟨*proof*⟩

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

lemma *preal-add-left-cancel*: $C + A = C + B \Longrightarrow A = (B::preal)$
⟨*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*:
$P \neq \{\} \Longrightarrow \{\} \subset (\bigcup X \in P.\ \mathit{Rep\text{-}preal}(X))$
⟨*proof*⟩

Part 2 of Dedekind sections definition

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

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

Part 3 of Dedekind sections definition

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

Part 4 of Dedekind sections definition

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

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

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

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

Supremum property

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

5.13 The Embedding from *rat* into *preal*

lemma *preal-of-rat-add-lemma1*:

$[|x < y + z;\ 0 < x;\ 0 < y|] ==> x * y * inverse\ (y + z) < (y::rat)$
⟨*proof*⟩

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

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

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

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

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

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

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

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

lemma *preal-of-rat-eq-iff*:

[| *0* < *x*; *0* < *y*|] ==> (*preal-of-rat x* = *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*)})

instantiation *real* :: {*zero*, *one*, *plus*, *minus*, *uminus*, *times*, *inverse*, *ord*, *abs*, *sgn*}
begin

definition
real-zero-def [*code func del*]: *0* = *Abs-Real*(*realrel*``{(*1*, *1*)})

definition
real-one-def [*code func del*]: *1* = *Abs-Real*(*realrel*``{(*1* + *1*, *1*)})

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

definition
real-minus-def [*code func del*]: − *r* = *contents* (⋃(*x*,*y*) ∈ *Rep-Real*(*r*). { *Abs-Real*(*realrel*``{(*y*,*x*)}) })

definition
real-diff-def [*code func del*]: *r* − (*s*::*real*) = *r* + − *s*

definition
real-mult-def [*code func del*]:
$z * w =$
contents $(\bigcup(x,y) \in$ *Rep-Real*$(z).\ \bigcup(u,v) \in$ *Rep-Real*$(w).$
$\{$ *Abs-Real*(*realrel*``$\{(x*u + y*v,\ x*v + y*u)\}$) $\})$

definition
real-inverse-def [*code func del*]: *inverse* $(R$::*real*$) = (THE\ S.\ (R = 0\ \&\ S = 0)$
$|\ S * R = 1)$

definition
real-divide-def [*code func del*]: R / $(S$::*real*$) = R *$ *inverse* S

definition
real-le-def [*code func del*]: $z \leq (w$::*real*$) \longleftrightarrow$
$(\exists x\ y\ u\ v.\ x+v \leq u+y\ \&\ (x,y) \in$ *Rep-Real* $z\ \&\ (u,v) \in$ *Rep-Real* $w)$

definition
real-less-def [*code func del*]: $x < (y$::*real*$) \longleftrightarrow x \leq y \wedge x \neq y$

definition
real-abs-def: *abs* $(r$::*real*$) = (if\ r < 0\ then\ -\ r\ else\ r)$

definition
real-sgn-def: *sgn* $(x$::*real*$) = (if\ x=0\ then\ 0\ else\ if\ 0<x\ then\ 1\ else\ -\ 1)$

instance ⟨*proof*⟩

end

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-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*))
⟨*proof*⟩

lemma *real-add*:
Abs-Real (*realrel``{(x,y)}*) + *Abs-Real* (*realrel``{(u,v)}*) =
Abs-Real (*realrel``{(x+u*, *y+v)}*)
⟨*proof*⟩

lemma *real-minus*: − *Abs-Real*(*realrel``{(x,y)}*) = *Abs-Real*(*realrel `` {(y,x)}*)
⟨*proof*⟩

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

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*)
⟨*proof*⟩

lemma *real-mult-congruent2*:
(%*p1 p2*.
(%(*x1*,*y1*). (%(*x2*,*y2*).
{ *Abs-Real* (*realrel``{(x1*x2* + *y1*y2*, *x1*y2*+*y1*x2*)}) }) *p2*) *p1*)
respects2 realrel
⟨*proof*⟩

lemma *real-mult*:
Abs-Real((*realrel*``{(*x1*,*y1*)})) ∗ *Abs-Real*((*realrel*``{(*x2*,*y2*)})) =
Abs-Real(*realrel* `` {(*x1*∗*x2*+*y1*∗*y2*,*x1*∗*y2*+*y1*∗*x2*)})
⟨*proof*⟩

lemma *real-mult-commute*: (*z*::*real*) ∗ *w* = *w* ∗ *z*
⟨*proof*⟩

lemma *real-mult-assoc*: ((*z1*::*real*) ∗ *z2*) ∗ *z3* = *z1* ∗ (*z2* ∗ *z3*)
⟨*proof*⟩

lemma *real-mult-1*: (*1*::*real*) ∗ *z* = *z*
⟨*proof*⟩

lemma *real-add-mult-distrib*: ((*z1*::*real*) + *z2*) ∗ *w* = (*z1* ∗ *w*) + (*z2* ∗ *w*)
⟨*proof*⟩

one and zero are distinct

lemma *real-zero-not-eq-one*: *0* ≠ (*1*::*real*)
⟨*proof*⟩

instance *real* :: *comm-ring-1*
⟨*proof*⟩

6.4 Inverse and Division

lemma *real-zero-iff*: *Abs-Real* (*realrel* `` {(*x*, *x*)}) = *0*
⟨*proof*⟩

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* ≠ *0* ==> ∃ *y*. *y*∗*x* = (*1*::*real*)
⟨*proof*⟩

lemma *real-mult-inverse-left*: *x* ≠ *0* ==> *inverse*(*x*)∗*x* = (*1*::*real*)
⟨*proof*⟩

6.5 The Real Numbers form a Field

instance *real* :: *field*
⟨*proof*⟩

Inverse of zero! Useful to simplify certain equations

lemma *INVERSE-ZERO*: *inverse 0* = (*0*::*real*)
⟨*proof*⟩

instance *real* :: *division-by-zero*
⟨*proof*⟩

6.6 The $\leq$ Ordering

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

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 $b \leq (d::preal)$
⟨*proof*⟩

lemma *real-le-lemma*:
assumes *l*: $u1 + v2 \leq u2 + v1$
and $x1 + v1 = u1 + y1$
and $x2 + v2 = u2 + y2$
shows $x1 + y2 \leq x2 + (y1::preal)$
⟨*proof*⟩

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

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

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

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

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

instance *real* :: *order*
⟨*proof*⟩

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

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

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

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

instantiation *real* :: *distrib-lattice*
begin

definition
$(inf :: real \Rightarrow real \Rightarrow real) = min$

definition
$(sup :: real \Rightarrow real \Rightarrow real) = max$

instance
⟨*proof*⟩

end

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*:
∃ *m*. (*x*::*real*) = *real-of-preal m* | *x* = *0* | *x* = −(*real-of-preal m*)
⟨*proof*⟩

lemma *real-of-preal-leD*:
real-of-preal m1 ≤ *real-of-preal m2* ==> *m1* ≤ *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-of-preal m1* ≤ *real-of-preal m2*) = (*m1* ≤ *m2*)
⟨*proof*⟩

lemma *real-of-preal-zero-less*: *0* < *real-of-preal m*
⟨*proof*⟩

lemma *real-of-preal-minus-less-zero*: − *real-of-preal m* < *0*
⟨*proof*⟩

lemma *real-of-preal-not-minus-gt-zero*: ~ *0* < − *real-of-preal m*
⟨*proof*⟩

6.8 Theorems About the Ordering

lemma *real-gt-zero-preal-Ex*: (*0* < *x*) = (∃ *y*. *x* = *real-of-preal y*)
⟨*proof*⟩

lemma *real-gt-preal-preal-Ex*:
real-of-preal z < *x* ==> ∃ *y*. *x* = *real-of-preal y*
⟨*proof*⟩

lemma *real-ge-preal-preal-Ex*:

real-of-preal z $\leq$ *x* ==> $\exists$ *y*. *x* = *real-of-preal y*
⟨*proof*⟩

lemma *real-less-all-preal*: *y* $\leq$ *0* ==> $\forall$ *x*. *y* < *real-of-preal x*
⟨*proof*⟩

lemma *real-less-all-real2*: ~ *0* < *y* ==> $\forall$ *x*. *y* < *real-of-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* $\leq$ *y***z*) = (*x*$\leq$*y*)
⟨*proof*⟩

lemma *real-mult-le-cancel-iff2* [*simp*]: (*0*::*real*) < *z* ==> (*z***x* $\leq$ *z***y*) = (*x*$\leq$*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* $\Rightarrow$ *real*
where
real-of-nat $\equiv$ *of-nat*

abbreviation
real-of-int :: *int* $\Rightarrow$ *real*
where
real-of-int $\equiv$ *of-int*

abbreviation
real-of-rat :: *rat* $\Rightarrow$ *real*
where
real-of-rat $\equiv$ *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*: $n \leq m \Longrightarrow real\ (m - n) = real\ (m::nat) - real\ n$
⟨*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*]: $(real\ (n::nat) \leq 0) = (n = 0)$
⟨*proof*⟩

lemma *not-real-of-nat-less-zero* [*simp*]: $\sim real\ (n::nat) < 0$
⟨*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

instantiation *real* :: *number-ring*
begin

definition
real-number-of-def [*code func del*]: *number-of w* = *real-of-int w*

instance
⟨*proof*⟩

end

lemma [*code unfold*, *symmetric*, *code post*]:
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*) ≤ *x*/*y*) = ((*x* ≤ *0* | *0* ≤ *y*) & (*0* ≤ *x* | *y* ≤ *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* ≤ (*0*::*real*)) = (*y* ≤ −*x*)
⟨*proof*⟩

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

6.13.1 Density of the Reals

lemma *real-lbound-gt-zero*:
[| (*0*::*real*) < *d1*; *0* < *d2* |] ==> ∃ *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*⟩

instance *real* :: *lordered-ring*
⟨*proof*⟩

6.15 Implementation of rational real numbers as pairs of integers

definition
Ratreal :: $int \times int \Rightarrow real$
where
$Ratreal = INum$

code-datatype *Ratreal*

lemma *Ratreal-simp*:
$Ratreal\ (k, l) = real\text{-}of\text{-}int\ k\ /\ real\text{-}of\text{-}int\ l$
⟨*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*⟩
declare *zero-real-code* [*symmetric*, *code post*]

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

instantiation *real* :: *eq*
begin

definition *eq-class.eq* (*x*::*real*) *y* ⟷ *x* = *y*

instance ⟨*proof*⟩

lemma *real-eq-code* [*code*]: *eq-class.eq* (*Ratreal x*) (*Ratreal y*) ⟷ *eq-class.eq* (*normNum x*) (*normNum y*)
⟨*proof*⟩

end

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*;
val r = (*if one-of* [*true*, *false*] *then p′ else* ~ *p′*,
if p′ = *0 then 0 else q′*)
in
(*r*, *fn* () => *term-of-real r*)
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*: $[|\ isLub\ R\ S\ x;\ isLub\ R\ S\ y\ |] ==> x = (y::real)$
⟨*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*:
$0 \leq r \Longrightarrow \exists (n::nat).\ real\ (n - 1) \leq r\ \&\ r < real\ (n)$
⟨*proof*⟩

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

lemma *reals-Archimedean-6b-int*:
$0 \leq r \Longrightarrow \exists\, n::int.\ real\ n \leq r\ \&\ r < real\ (n+1)$
⟨*proof*⟩

lemma *reals-Archimedean-6c-int*:
$r < 0 \Longrightarrow \exists\, n::int.\ real\ n \leq r\ \&\ r < real\ (n+1)$
⟨*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 ($\lfloor$-$\rfloor$) **and**
ceiling ($\lceil$-$\rceil$)

notation (*HTML* **output**)
floor ($\lfloor$-$\rfloor$) **and**
ceiling ($\lceil$-$\rceil$)

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-of n*) ≤ *real* (*m*::*int*)) = (*number-of n* ≤ *m*)
⟨*proof*⟩

lemma *number-of-le-real-of-int-iff2* [*simp*]:
(*real* (*m*::*int*) ≤ (*number-of n*)) = (*m* ≤ *number-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*: ∃ *n*::*int*. *real n* ≤ *r* & *r* < *real* (*n*+*1*)
⟨*proof*⟩

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

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

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

lemma *floor-mono2*: *x* ≤ *y* ==> *floor x* ≤ *floor y*
⟨*proof*⟩

lemma *lemma-floor2*: *real n* < *real* (*x*::*int*) + *1* ==> *n* ≤ *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*]:
$(number\text{-}of\ n <= floor\ x) = (number\text{-}of\ n <= x)$
⟨*proof*⟩

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

lemma *ceiling-eq3*: $[|\ real\ n - 1 < x;\ x \leq real\ n\ |] \Longrightarrow ceiling\ x = n$

⟨*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* ≤ *r*
⟨*proof*⟩

lemma *real-of-int-ceiling-le-add-one* [*simp*]: *real* (*ceiling r*) ≤ *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*]:
$\sim neg((number\text{-}of\ n)::int) ==> 0 <= x ==>$
$(natceiling\ x <= number\text{-}of\ n) = (x <= number\text{-}of\ n)$
⟨*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*]:
$\sim neg\ ((number\text{-}of\ n)::int) ==> 0 <= x ==>$
$natceiling\ (x + number\text{-}of\ n) = natceiling\ x + number\text{-}of\ n$
⟨*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* ⇒ *real* ⇒ *real set* **where**
closed-int x y = {*z*. *x* ≤ *z* ∧ *z* ≤ *y*}

8.2.2 Properties

lemma *closed-int-subset*:
assumes *xy*: *x1* ≥ *x0 y1* ≤ *y0*
shows *closed-int x1 y1* ⊆ *closed-int x0 y0*
⟨*proof*⟩

lemma *closed-int-least*:
assumes *a*: *a* ≤ *b*
shows *a* ∈ *closed-int a b* ∧ (∀ *x* ∈ *closed-int a b*. *a* ≤ *x*)
⟨*proof*⟩

lemma *closed-int-most*:
assumes *a*: *a* ≤ *b*
shows *b* ∈ *closed-int a b* ∧ (∀ *x* ∈ *closed-int a b*. *x* ≤ *b*)
⟨*proof*⟩

lemma *closed-not-empty*:
shows *a* ≤ *b* ⟹ ∃ *x*. *x* ∈ *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* ⇒ *real set*
assumes *subset*: $\forall n.\ f\ (Suc\ n) \subseteq f\ n$
and *closed*: $\forall n.\ \exists a\ b.\ f\ n = closed\text{-}int\ a\ b \wedge a \leq b$
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
$\exists ka\ kb.\ ka < kb \wedge closed\text{-}int\ ka\ kb \subseteq closed\text{-}int\ a\ b \wedge c \notin (closed\text{-}int\ ka\ kb)$
⟨*proof*⟩

8.5 newInt: Interval generation

Given a function f:ℕ⇒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* ⇒ (*nat* ⇒ *real*) ⇒ (*real set*)
primrec
newInt 0 f = *closed-int* (*f 0* + *1*) (*f 0* + *2*)
newInt (*Suc n*) *f* =
(*SOME e*. (∃ *e1 e2*.
e1 < *e2* ∧

$e = \textit{closed-int}\ \textit{e1}\ \textit{e2} \wedge$
$e \subseteq (\textit{newInt}\ n\ f) \wedge$
$(f\ (\textit{Suc}\ n)) \notin e)$
)

8.5.2 Properties

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

lemma *newInt-ex*:
$\exists\, a\ b.\ a < b \wedge$
$\textit{newInt}\ (\textit{Suc}\ n)\ f = \textit{closed-int}\ a\ b \wedge$
$\textit{newInt}\ (\textit{Suc}\ n)\ f \subseteq \textit{newInt}\ n\ f \wedge$
$f\ (\textit{Suc}\ n) \notin \textit{newInt}\ (\textit{Suc}\ n)\ f$
⟨*proof*⟩

lemma *newInt-subset*:
$\textit{newInt}\ (\textit{Suc}\ n)\ f \subseteq \textit{newInt}\ n\ f$
⟨*proof*⟩

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*:
$\forall\, n.\ f\ n \notin (\bigcap n.\ \textit{newInt}\ n\ f)$
⟨*proof*⟩

lemma *newInt-notempty*:
$(\bigcap n.\ \textit{newInt}\ n\ f) \neq \{\}$
⟨*proof*⟩

8.6 Final Theorem

theorem *real-non-denum*:
shows $\neg\ (\exists f{::}\textit{nat}{\Rightarrow}\textit{real}.\ \textit{surj}\ f)$
⟨*proof*⟩

end

9 RealPow: Natural powers theory

theory *RealPow*
imports *RealDef*
begin

declare *abs-mult-self* [*simp*]

instantiation *real* :: *recpower*
begin

primrec *power-real* **where**
realpow-0: $r \hat{} 0 = (1::real)$
| *realpow-Suc*: $r \hat{} Suc\ n = (r::real) * r \hat{} n$

instance ⟨*proof*⟩

end

lemma *two-realpow-ge-one* [*simp*]: $(1::real) \leq 2 \hat{} n$
⟨*proof*⟩

lemma *two-realpow-gt* [*simp*]: $real\ (n::nat) < 2 \hat{} n$
⟨*proof*⟩

lemma *realpow-Suc-le-self*: $[|\ 0 \leq r; r \leq (1::real)\ |] ==> r \hat{} Suc\ n \leq r$
⟨*proof*⟩

lemma *realpow-minus-mult* [*rule-format*]:
$0 < n --> (x::real) \hat{} (n - 1) * x = x \hat{} n$
⟨*proof*⟩

lemma *realpow-two-mult-inverse* [*simp*]:
$r \neq 0 ==> r * inverse\ r \hat{} Suc\ (Suc\ 0) = inverse\ (r::real)$
⟨*proof*⟩

lemma *realpow-two-minus* [*simp*]: $(-x) \hat{} Suc\ (Suc\ 0) = (x::real) \hat{} Suc\ (Suc\ 0)$
⟨*proof*⟩

lemma *realpow-two-diff*:
$(x::real) \hat{} Suc\ (Suc\ 0) - y \hat{} Suc\ (Suc\ 0) = (x - y) * (x + y)$
⟨*proof*⟩

lemma *realpow-two-disj*:
$((x::real) \hat{} Suc\ (Suc\ 0) = y \hat{} Suc\ (Suc\ 0)) = (x = y \mid x = -y)$
⟨*proof*⟩

lemma *realpow-real-of-nat*: $real\ (m::nat) \hat{} n = real\ (m \hat{} n)$
⟨*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*) ^ *n* = *real* (*x* ^ *n*)
⟨*proof*⟩
declare *real-of-int-power* [*symmetric*, *simp*]

lemma *power-real-number-of*:
(*number-of v* :: *real*) ^ *n* = *real* ((*number-of v* :: *int*) ^ *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-ring-strict*
shows $0 \leq x * x + y * y$
⟨*proof*⟩

lemma *not-sum-squares-lt-zero*:
fixes *x y* :: 'a::*ordered-ring-strict*
shows $\neg\ x * x + y * y < 0$
⟨*proof*⟩

lemma *sum-nonneg-eq-zero-iff*:
fixes *x y* :: 'a::*pordered-ab-group-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-ring-strict*
shows $(x * x + y * y = 0) = (x = 0 \wedge y = 0)$
⟨*proof*⟩

lemma *sum-squares-le-zero-iff*:
fixes *x y* :: 'a::*ordered-ring-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-ring-strict*
shows $(0 < x * x + y * y) = (x \neq 0 \vee y \neq 0)$
⟨*proof*⟩

lemma *sum-power2-ge-zero*:
fixes *x y* :: 'a::{*ordered-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 $(x^2 + y^2 = 0) = (x = 0 \wedge y = 0)$
⟨*proof*⟩

lemma *sum-power2-le-zero-iff*:
fixes $x\ y :: 'a::\{ordered\text{-}idom,recpower\}$
shows $(x^2 + y^2 \leq 0) = (x = 0 \wedge y = 0)$
⟨*proof*⟩

lemma *sum-power2-gt-zero-iff*:
fixes $x\ y :: 'a::\{ordered\text{-}idom,recpower\}$
shows $(0 < x^2 + y^2) = (x \neq 0 \vee y \neq 0)$
⟨*proof*⟩

9.3 Squares of Reals

lemma *real-two-squares-add-zero-iff* [*simp*]:
$(x * x + y * y = 0) = ((x::real) = 0 \wedge y = 0)$
⟨*proof*⟩

lemma *real-sum-squares-cancel*: $x * x + y * y = 0 \Longrightarrow x = (0::real)$
⟨*proof*⟩

lemma *real-sum-squares-cancel2*: $x * x + y * y = 0 \Longrightarrow y = (0::real)$
⟨*proof*⟩

lemma *real-mult-self-sum-ge-zero*: $(0::real) \leq x*x + y*y$
⟨*proof*⟩

lemma *real-sum-squares-cancel-a*: $x * x = -(y * y) \Longrightarrow x = (0::real)$ & $y=0$
⟨*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*: $x \neq 0 \Longrightarrow x * x + y * y \neq (0::real)$
⟨*proof*⟩

lemma *real-sum-squares-not-zero2*: $y \sim= 0 \Longrightarrow x * x + y * y \sim= (0::real)$
⟨*proof*⟩

lemma *realpow-two-sum-zero-iff* [*simp*]:
$(x \hat{} 2 + y \hat{} 2 = (0::real)) = (x = 0 \ \& \ y = 0)$
⟨*proof*⟩

lemma *realpow-two-le-add-order* [*simp*]: $(0::real) \leq u \hat{} 2 + v \hat{} 2$
⟨*proof*⟩

lemma *realpow-two-le-add-order2* [*simp*]: $(0::real) \leq u \hat{} 2 + v \hat{} 2 + w \hat{} 2$
⟨*proof*⟩

lemma *real-sum-square-gt-zero*: $x \sim= 0 \Longrightarrow (0::real) < x * x + y * y$
⟨*proof*⟩

lemma *real-sum-square-gt-zero2*: $y \sim= 0 \Longrightarrow (0::real) < x * x + y * y$
⟨*proof*⟩

lemma *real-minus-mult-self-le* [*simp*]: $-(u * u) \leq (x * (x::real))$
⟨*proof*⟩

lemma *realpow-square-minus-le* [*simp*]: $-(u \hat{} 2) \leq (x::real) \hat{} 2$
⟨*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*:
$[|(0::real) < x; \ 0 < x1; \ x1 * y < x * u \ |]$
$\Longrightarrow inverse\ x * y < inverse\ x1 * u$
⟨*proof*⟩

lemma *real-mult-inverse-cancel2*:
$[|(0::real) < x; 0 < x1; \ x1 * y < x * u \ |] \Longrightarrow y * inverse\ x < u * inverse\ x1$
⟨*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

instantiation *real* :: *scaleR*
begin

definition
real-scaleR-def [*simp*]: *scaleR a x* = *a* ∗ *x*

instance ⟨*proof*⟩

end

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* [(λ*a*. *scaleR a x*::*'a*::*real-vector*)]
⟨*proof*⟩

interpretation *scaleR-right*: *additive* [(λ*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 $(scaleR\ a\ x = 0) = (a = 0 \vee x = 0)$
⟨*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 $(scaleR\ a\ x = scaleR\ a\ y) = (x = y \vee a = 0)$
⟨*proof*⟩

lemma *scaleR-cancel-right*:
fixes $x\ y :: 'a$::*real-vector*
shows $(scaleR\ a\ x = scaleR\ b\ x) = (a = b \vee x = 0)$
⟨*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
$of\text{-}real :: real \Rightarrow 'a$::*real-algebra-1* **where**
$of\text{-}real\ r = scaleR\ r\ 1$

lemma *scaleR-conv-of-real*: $scaleR\ r\ x = of\text{-}real\ r * x$
⟨*proof*⟩

lemma *of-real-0* [*simp*]: $of\text{-}real\ 0 = 0$

⟨*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* ^ *n*) = (*of-real x* :: *'a*::{*real-algebra-1*,*recpower*}) ^ *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* ∈ *Reals*; *b* ∈ *Reals*⟧ ⟹ *a* ∗ *b* ∈ *Reals*
⟨*proof*⟩

lemma *nonzero-Reals-inverse*:
fixes *a* :: 'a::*real-div-algebra*
shows ⟦*a* ∈ *Reals*; *a* ≠ *0*⟧ ⟹ *inverse a* ∈ *Reals*
⟨*proof*⟩

lemma *Reals-inverse* [*simp*]:
fixes *a* :: 'a::{*real-div-algebra*,*division-by-zero*}
shows *a* ∈ *Reals* ⟹ *inverse a* ∈ *Reals*
⟨*proof*⟩

lemma *nonzero-Reals-divide*:
fixes *a b* :: 'a::*real-field*
shows ⟦*a* ∈ *Reals*; *b* ∈ *Reals*; *b* ≠ *0*⟧ ⟹ *a* / *b* ∈ *Reals*
⟨*proof*⟩

lemma *Reals-divide* [*simp*]:
fixes *a b* :: 'a::{*real-field*,*division-by-zero*}
shows ⟦*a* ∈ *Reals*; *b* ∈ *Reals*⟧ ⟹ *a* / *b* ∈ *Reals*
⟨*proof*⟩

lemma *Reals-power* [*simp*]:
fixes *a* :: 'a::{*real-algebra-1*,*recpower*}
shows *a* ∈ *Reals* ⟹ *a* ^ *n* ∈ *Reals*
⟨*proof*⟩

lemma *Reals-cases* [*cases set*: *Reals*]:
assumes *q* ∈ ℝ
obtains (*of-real*) *r* **where** *q* = *of-real r*
⟨*proof*⟩

lemma *Reals-induct* [*case-names of-real*, *induct set*: *Reals*]:
q ∈ ℝ ⟹ (⋀*r*. *P* (*of-real r*)) ⟹ *P q*
⟨*proof*⟩

10.5 Real normed vector spaces

class *norm* = *type* +
fixes *norm* :: 'a ⇒ *real*

instantiation *real* :: *norm*
begin

definition
real-norm-def [*simp*]: *norm r* ≡ |*r*|

instance ⟨*proof*⟩

end

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* ≤ *norm x*
  and *norm-eq-zero* [*simp*]: *norm x* = *0* ⟷ *x* = *0*
  and *norm-triangle-ineq*: *norm* (*x* + *y*) ≤ *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*) ≤ *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* ≠ *0*)
⟨*proof*⟩

lemma *norm-not-less-zero* [*simp*]:
  fixes *x* :: '*a*::*real-normed-vector*
  shows ¬ *norm x* < *0*
⟨*proof*⟩

lemma *norm-le-zero-iff* [*simp*]:
  fixes *x* :: '*a*::*real-normed-vector*
  shows (*norm x* ≤ *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* ≠ *0* ⟹ *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* ≠ *0* ⟹ *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*) ≤ *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-normed-vector*) = *0*) = (*x* = *0*)
⟨*proof*⟩

lemma *sgn-minus*: *sgn* (− *x*) = − *sgn*(*x*::'*a*::*real-normed-vector*)
⟨*proof*⟩

lemma *sgn-scaleR*:
sgn (*scaleR r x*) = *scaleR* (*sgn r*) (*sgn*(*x*::'*a*::*real-normed-vector*))
⟨*proof*⟩

lemma *sgn-one* [*simp*]: *sgn* (*1*::'*a*::*real-normed-algebra-1*) = *1*
⟨*proof*⟩

lemma *sgn-of-real*:
sgn (*of-real r*::'*a*::*real-normed-algebra-1*) = *of-real* (*sgn r*)
⟨*proof*⟩

lemma *sgn-mult*:
fixes *x y* :: '*a*::*real-normed-div-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*) ⟹ *sgn x* = *1*
⟨*proof*⟩

lemma *real-sgn-neg*: (*x*::*real*) < *0* ⟹ *sgn x* = −*1*
⟨*proof*⟩

10.7 Bounded Linear and Bilinear Operators

locale *bounded-linear* = *additive* +
constrains *f* :: '*a*::*real-normed-vector* ⇒ '*b*::*real-normed-vector*
assumes *scaleR*: *f* (*scaleR r x*) = *scaleR r* (*f x*)
assumes *bounded*: ∃ *K*. ∀ *x*. *norm* (*f x*) ≤ *norm x* ∗ *K*

lemma (**in** *bounded-linear*) *pos-bounded*:
∃ *K*>*0*. ∀ *x*. *norm* (*f x*) ≤ *norm x* ∗ *K*
⟨*proof*⟩

lemma (**in** *bounded-linear*) *nonneg-bounded*:
$\exists K{\geq}0.\ \forall x.\ norm\ (f\ x) \leq norm\ x * K$
⟨*proof*⟩

locale *bounded-bilinear* =
fixes *prod* :: [′*a*::*real-normed-vector*, ′*b*::*real-normed-vector*]
⇒ ′*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*^(*nat a*)) *else* (*inverse* (*2*^(*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-of-real x = (*SOME y*. *real y* = *x*)

definition
real-is-int :: *real* ⇒ *bool* **where**
real-is-int x = (*EX* (*u*::*int*). *x* = *real u*)

lemma *real-is-int-def2*: *real-is-int x* = (*x* = *real* (*int-of-real x*))
⟨*proof*⟩

lemma *float-transfer*: *real-is-int* ((*real a*)∗(*pow2 c*)) ⟹ *float* (*a*, *b*) = *float* (*int-of-real*

$((real\ a)*(pow2\ c)),\ b\ -\ c)$
⟨*proof*⟩

lemma *pow2-int*: $pow2\ (int\ c) = 2\hat{}c$
⟨*proof*⟩

lemma *float-transfer-nat*: $float\ (a,\ b) = float\ (a\ *\ 2\hat{}c,\ b\ -\ int\ c)$
⟨*proof*⟩

lemma *real-is-int-real*[*simp*]: $real\text{-}is\text{-}int\ (real\ (x::int))$
⟨*proof*⟩

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

lemma *real-int-of-real*[*simp*]: $real\text{-}is\text{-}int\ x \Longrightarrow real\ (int\text{-}of\text{-}real\ x) = x$
⟨*proof*⟩

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

lemma *real-is-int-add*[*simp*]: $real\text{-}is\text{-}int\ a \Longrightarrow real\text{-}is\text{-}int\ b \Longrightarrow real\text{-}is\text{-}int\ (a+b)$
⟨*proof*⟩

lemma *int-of-real-sub*: $real\text{-}is\text{-}int\ a \Longrightarrow real\text{-}is\text{-}int\ b \Longrightarrow (int\text{-}of\text{-}real\ (a-b)) = (int\text{-}of\text{-}real\ a) - (int\text{-}of\text{-}real\ b)$
⟨*proof*⟩

lemma *real-is-int-sub*[*simp*]: $real\text{-}is\text{-}int\ a \Longrightarrow real\text{-}is\text{-}int\ b \Longrightarrow real\text{-}is\text{-}int\ (a-b)$
⟨*proof*⟩

lemma *real-is-int-rep*: $real\text{-}is\text{-}int\ x \Longrightarrow ?!\ (a::int).\ real\ a = x$
⟨*proof*⟩

lemma *int-of-real-mult*:
assumes *real-is-int a real-is-int b*
shows $(int\text{-}of\text{-}real\ (a*b)) = (int\text{-}of\text{-}real\ a) * (int\text{-}of\text{-}real\ b)$
⟨*proof*⟩

lemma *real-is-int-mult*[*simp*]: $real\text{-}is\text{-}int\ a \Longrightarrow real\text{-}is\text{-}int\ b \Longrightarrow real\text{-}is\text{-}int\ (a*b)$
⟨*proof*⟩

lemma *real-is-int-0*[*simp*]: $real\text{-}is\text{-}int\ (0::real)$
⟨*proof*⟩

lemma *real-is-int-1*[*simp*]: $real\text{-}is\text{-}int\ (1::real)$
⟨*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* ⟹ *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* ≠ *0* ⟹ *a* ≠ *−1* ⟹ *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 *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^(*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* ⇒ *real*
where
lbound x = *min 0 x*

definition
ubound :: *real* ⇒ *real*
where
ubound x = *max 0 x*

lemma *lbound*: *lbound x* ≤ *x*
⟨*proof*⟩

lemma *ubound*: *x* ≤ *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*::-::*number-ring*) = *Numeral0* & (*1*::-::*number-ring*) = *Numeral1*
⟨*proof*⟩

lemma *add-left-zero*: *0* + *a* = (*a*::'*a*::*comm-monoid-add*)
⟨*proof*⟩

lemma *add-right-zero*: *a* + *0* = (*a*::'*a*::*comm-monoid-add*)
⟨*proof*⟩

lemma *mult-left-one*: *1* * *a* = (*a*::'*a*::*semiring-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* =
normalize-bin-simps
pred-bin-simps succ-bin-simps
add-bin-simps minus-bin-simps mult-bin-simps

lemma *int-eq-number-of-eq*:
(((*number-of v*)::*int*)=(*number-of w*)) = *iszero* ((*number-of* (*v* + *uminus w*))::*int*)
⟨*proof*⟩

lemma *int-iszero-number-of-Pls*: *iszero* (*Numeral0*::*int*)
⟨*proof*⟩

lemma *int-nonzero-number-of-Min*: ~(*iszero* ((−*1*)::*int*))
⟨*proof*⟩

lemma *int-iszero-number-of-Bit0*: *iszero* ((*number-of* (*Int.Bit0 w*))::*int*) = *iszero* ((*number-of w*)::*int*)
⟨*proof*⟩

lemma *int-iszero-number-of-Bit1*: ¬ *iszero* ((*number-of* (*Int.Bit1 w*))::*int*)
⟨*proof*⟩

lemma *int-less-number-of-eq-neg*: (((*number-of x*)::*int*) < *number-of y*) = *neg* ((*number-of* (*x* + (*uminus y*)))::*int*)
⟨*proof*⟩

lemma *int-not-neg-number-of-Pls*: ¬ (*neg* (*Numeral0*::*int*))
⟨*proof*⟩

lemma *int-neg-number-of-Min*: *neg* (−*1*::*int*)
⟨*proof*⟩

lemma *int-neg-number-of-Bit0*: *neg* ((*number-of* (*Int.Bit0 w*))::*int*) = *neg* ((*number-of w*)::*int*)
⟨*proof*⟩

lemma *int-neg-number-of-Bit1*: *neg* ((*number-of* (*Int.Bit1 w*))::*int*) = *neg* ((*number-of w*)::*int*)
⟨*proof*⟩

lemma *int-le-number-of-eq*: (((*number-of x*)::*int*) ≤ *number-of y*) = (¬ *neg* ((*number-of* (*y* + (*uminus x*)))::*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-Bit0*
lift-bool[*OF int-iszero-number-of-Bit1*] *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-Bit0 int-neg-number-of-Bit1 int-le-number-of-eq

lemma *int-number-of-add-sym*: ((*number-of v*)::*int*) + *number-of w* = *number-of*

$(v + w)$
⟨*proof*⟩

lemma *int-number-of-diff-sym*: $((number\text{-}of\ v)::int) - number\text{-}of\ w = number\text{-}of\ (v + (uminus\ w))$
⟨*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 Univ-Poly: Univariate Polynomials

theory *Univ-Poly*
imports *Main*
begin

Application of polynomial as a function.

primrec (**in** *semiring-0*) *poly* :: *'a list => 'a => 'a* **where**
poly-Nil: *poly [] x = 0*
| *poly-Cons*: *poly (h#t) x = h + x * poly t x*

12.1 Arithmetic Operations on Polynomials

addition

primrec (**in** *semiring-0*) *padd* :: *'a list* ⇒ *'a list* ⇒ *'a list* (**infixl** *+++ 65*)
where
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

primrec (**in** *semiring-0*) *cmult* :: *'a* ⇒ *'a list* ⇒ *'a list* (**infixl** *%** *70*) **where**
cmult-Nil: *c %** [] = []
| *cmult-Cons*: *c %** (*h*#*t*) = (*c* * *h*)#(*c %* t*)

Multiplication by a polynomial

primrec (**in** *semiring-0*) *pmult* :: *'a list* ⇒ *'a list* ⇒ *'a list* (**infixl** ***** *70*)
where
pmult-Nil: [] **** l2* = []
| *pmult-Cons*: (*h*#*t*) **** l2* = (*if t* = [] *then h %* l2*
else (*h %* l2*) *+++* ((*0*) # (*t *** l2*)))

Repeated multiplication by a polynomial

primrec (**in** *semiring-0*) *mulexp* :: *nat* ⇒ *'a list* ⇒ *'a list* ⇒ *'a list* **where**
mulexp-zero: *mulexp 0 p q* = *q*
| *mulexp-Suc*: *mulexp* (*Suc n*) *p q* = *p *** mulexp n p q*

Exponential

primrec (**in** *semiring-1*) *pexp* :: *'a list* ⇒ *nat* ⇒ *'a list* (**infixl** *%^ 80*) **where**
pexp-0: *p %^ 0* = [*1*]
| *pexp-Suc*: *p %^* (*Suc n*) = *p **** (*p %^ n*)

Quotient related value of dividing a polynomial by x + a

primrec (**in** *field*) *pquot* :: *'a list* ⇒ *'a* ⇒ *'a list* **where**
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*))

normalization of polynomials (remove extra 0 coeff)

primrec (**in** *semiring-0*) *pnormalize* :: *'a list* ⇒ *'a list* **where**
pnormalize-Nil: *pnormalize* [] = []
| *pnormalize-Cons*: *pnormalize* (*h*#*p*) = (*if* ((*pnormalize p*) = [])
then (*if* (*h* = *0*) *then* [] *else* [*h*])
else (*h*#(*pnormalize p*)))

definition (**in** *semiring-0*) *pnormal p* = ((*pnormalize p* = *p*) ∧ *p* ≠ [])
definition (**in** *semiring-0*) *nonconstant p* = (*pnormal p* ∧ (∀ *x*. *p* ≠ [*x*]))

Other definitions

definition (**in** *ring-1*)
poly-minus :: *'a list => 'a list* (*-- - [80] 80*) **where**
-- p = (- 1) % p*

definition (**in** *semiring-0*)
divides :: *'a list ⇒ 'a list ⇒ bool* (**infixl** *divides 70*) **where**
*p1 divides p2 = (∃ q. poly p2 = poly(p1 *** q))*

— order of a polynomial
definition (**in** *ring-1*) *order* :: *'a => 'a list => nat* **where**
order a p = (SOME n. ([−a, 1] %^ n) divides p &
~ (([−a, 1] %^ (Suc n)) divides p))

— degree of a polynomial
definition (**in** *semiring-0*) *degree* :: *'a list => nat* **where**
degree p = length (pnormalize p) − 1

— squarefree polynomials — NB with respect to real roots only.
definition (**in** *ring-1*)
rsquarefree :: *'a list => bool* **where**
rsquarefree p = (poly p ≠ poly [] &
(∀ a. (order a p = 0) | (order a p = 1)))

context *semiring-0*
begin

lemma *padd-Nil2*[*simp*]: *p +++ [] = p*
⟨*proof*⟩

lemma *padd-Cons-Cons*: *(h1 # p1) +++ (h2 # p2) = (h1 + h2) # (p1 +++ p2)*
⟨*proof*⟩

lemma *pminus-Nil*[*simp*]: *-- [] = []*
⟨*proof*⟩

lemma *pmult-singleton*: *[h1] *** p1 = h1 %* p1* ⟨*proof*⟩
end

lemma (**in** *semiring-1*) *poly-ident-mult*[*simp*]: *1 %* t = t* ⟨*proof*⟩

lemma (**in** *semiring-0*) *poly-simple-add-Cons*[*simp*]: *[a] +++ ((0)#t) = (a#t)*
⟨*proof*⟩

Handy general properties

lemma (**in** *comm-semiring-0*) *padd-commut*: *b +++ a = a +++ b*
⟨*proof*⟩

lemma (**in** *comm-semiring-0*) *padd-assoc*: *∀ b c. (a +++ b) +++ c = a +++ (b*

+++ c)
⟨*proof*⟩

lemma (**in** *semiring-0*) *poly-cmult-distr*: *a %* (p +++ q*) = (*a %* p +++ a %* q*)
⟨*proof*⟩

lemma (**in** *ring-1*) *pmult-by-x*[*simp*]: [*0*, *1*] **** t* = ((*0*)#*t*)
⟨*proof*⟩

properties of evaluation of polynomials.

lemma (**in** *semiring-0*) *poly-add*: *poly* (*p1 +++ p2*) *x* = *poly p1 x* + *poly p2 x*
⟨*proof*⟩

lemma (**in** *comm-semiring-0*) *poly-cmult*: *poly* (*c %* p*) *x* = *c ** *poly p x*
⟨*proof*⟩

lemma (**in** *comm-semiring-0*) *poly-cmult-map*: *poly* (*map* (*op ** *c*) *p*) *x* = *c*poly p x*
⟨*proof*⟩

lemma (**in** *comm-ring-1*) *poly-minus*: *poly* (*-- p*) *x* = *-* (*poly p x*)
⟨*proof*⟩

lemma (**in** *comm-semiring-0*) *poly-mult*: *poly* (*p1 *** p2*) *x* = *poly p1 x ** *poly p2 x*
⟨*proof*⟩

class *recpower-semiring* = *semiring* + *recpower*
class *recpower-semiring-1* = *semiring-1* + *recpower*
class *recpower-semiring-0* = *semiring-0* + *recpower*
class *recpower-ring* = *ring* + *recpower*
class *recpower-ring-1* = *ring-1* + *recpower*
subclass (**in** *recpower-ring-1*) *recpower-ring* ⟨*proof*⟩
class *recpower-comm-semiring-1* = *recpower* + *comm-semiring-1*
class *recpower-comm-ring-1* = *recpower* + *comm-ring-1*
subclass (**in** *recpower-comm-ring-1*) *recpower-comm-semiring-1* ⟨*proof*⟩
class *recpower-idom* = *recpower* + *idom*
subclass (**in** *recpower-idom*) *recpower-comm-ring-1* ⟨*proof*⟩
class *idom-char-0* = *idom* + *ring-char-0*
class *recpower-idom-char-0* = *recpower* + *idom-char-0*
subclass (**in** *recpower-idom-char-0*) *recpower-idom* ⟨*proof*⟩

lemma (**in** *recpower-comm-ring-1*) *poly-exp*: *poly* (*p %^ n*) *x* = (*poly p x*) *^ n*
⟨*proof*⟩

More Polynomial Evaluation Lemmas

lemma (**in** *semiring-0*) *poly-add-rzero*[*simp*]: *poly* (*a +++* []) *x* = *poly a x*
⟨*proof*⟩

lemma (**in** *comm-semiring-0*) *poly-mult-assoc*: *poly* ((*a* *** *b*) *** *c*) *x* = *poly* (*a* *** (*b* *** *c*)) *x*
⟨*proof*⟩

lemma (**in** *semiring-0*) *poly-mult-Nil2*[*simp*]: *poly* (*p* *** []) *x* = *0*
⟨*proof*⟩

lemma (**in** *comm-semiring-1*) *poly-exp-add*: *poly* (*p* %^ (*n* + *d*)) *x* = *poly*(*p* %^ *n* *** *p* %^ *d*) *x*
⟨*proof*⟩

12.2 Key Property: if *f a* = (*0*::'*a*) then *x* − *a* divides *p x*

lemma (**in** *comm-ring-1*) *lemma-poly-linear-rem*: ∀ *h*. ∃ *q r*. *h*#*t* = [*r*] +++ [−*a*, *1*] *** *q*
⟨*proof*⟩

lemma (**in** *comm-ring-1*) *poly-linear-rem*: ∃ *q r*. *h*#*t* = [*r*] +++ [−*a*, *1*] *** *q*
⟨*proof*⟩

lemma (**in** *comm-ring-1*) *poly-linear-divides*: (*poly p a* = *0*) = ((*p* = []) | (∃ *q*. *p* = [−*a*, *1*] *** *q*))
⟨*proof*⟩

lemma (**in** *semiring-0*) *lemma-poly-length-mult*[*simp*]: ∀ *h k a*. *length* (*k* %* *p* +++ (*h* # (*a* %* *p*))) = *Suc* (*length p*)
⟨*proof*⟩

lemma (**in** *semiring-0*) *lemma-poly-length-mult2*[*simp*]: ∀ *h k*. *length* (*k* %* *p* +++ (*h* # *p*)) = *Suc* (*length p*)
⟨*proof*⟩

lemma (**in** *ring-1*) *poly-length-mult*[*simp*]: *length*([−*a*,*1*] *** *q*) = *Suc* (*length q*)
⟨*proof*⟩

12.3 Polynomial length

lemma (**in** *semiring-0*) *poly-cmult-length*[*simp*]: *length* (*a* %* *p*) = *length p*
⟨*proof*⟩

lemma (**in** *semiring-0*) *poly-add-length*: *length* (*p1* +++ *p2*) = *max* (*length p1*) (*length p2*)
⟨*proof*⟩

lemma (**in** *semiring-0*) *poly-root-mult-length*[*simp*]: *length*([*a*,*b*] *** *p*) = *Suc* (*length p*)
⟨*proof*⟩

lemma (**in** *idom*) *poly-mult-not-eq-poly-Nil*[*simp*]:
poly (*p* *** *q*) *x* ≠ *poly* [] *x* ⟷ *poly p x* ≠ *poly* [] *x* ∧ *poly q x* ≠ *poly* [] *x*
⟨*proof*⟩

lemma (**in** *idom*) *poly-mult-eq-zero-disj*: *poly* (*p* *** *q*) *x* = *0* ⟷ *poly p x* = *0*
∨ *poly q x* = *0*
⟨*proof*⟩

Normalisation Properties

lemma (**in** *semiring-0*) *poly-normalized-nil*: (*pnormalize p* = []) −−> (*poly p x* =
0)
⟨*proof*⟩

A nontrivial polynomial of degree n has no more than n roots

lemma (**in** *idom*) *poly-roots-index-lemma*:
assumes *p*: *poly p x* ≠ *poly* [] *x* **and** *n*: *length p* = *n*
shows ∃ *i*. ∀ *x*. *poly p x* = *0* ⟶ (∃ *m*≤*n*. *x* = *i m*)
⟨*proof*⟩

lemma (**in** *idom*) *poly-roots-index-length*: *poly p x* ≠ *poly* [] *x* ==>
∃ *i*. ∀ *x*. (*poly p x* = *0*) −−> (∃ *n*. *n* ≤ *length p* & *x* = *i n*)
⟨*proof*⟩

lemma (**in** *idom*) *poly-roots-finite-lemma1*: *poly p x* ≠ *poly* [] *x* ==>
∃ *N i*. ∀ *x*. (*poly p x* = *0*) −−> (∃ *n*. (*n*::*nat*) < *N* & *x* = *i n*)
⟨*proof*⟩

lemma (**in** *idom*) *idom-finite-lemma*:
assumes *P*: ∀ *x*. *P x* −−> (∃ *n*. *n* < *length j* & *x* = *j*!*n*)
shows *finite* {*x*. *P x*}
⟨*proof*⟩

lemma (**in** *idom*) *poly-roots-finite-lemma2*: *poly p x* ≠ *poly* [] *x* ==>
∃ *i*. ∀ *x*. (*poly p x* = *0*) −−> *x* ∈ *set i*
⟨*proof*⟩

lemma *UNIV-nat-infinite*: ¬ *finite* (*UNIV* :: *nat set*)
⟨*proof*⟩

lemma (**in** *ring-char-0*) *UNIV-ring-char-0-infinte*:
¬ (*finite* (*UNIV*:: *'a set*))
⟨*proof*⟩

lemma (**in** *idom-char-0*) *poly-roots-finite*: (*poly p* ≠ *poly* []) =
finite {*x*. *poly p x* = *0*}
⟨*proof*⟩

Entirety and Cancellation for polynomials

lemma (**in** *idom-char-0*) *poly-entire-lemma2*:
assumes *p0*: *poly p* ≠ *poly* [] **and** *q0*: *poly q* ≠ *poly* []
shows *poly* (*p*∗∗∗*q*) ≠ *poly* []
⟨*proof*⟩

lemma (**in** *idom-char-0*) *poly-entire*:
poly (*p* ∗∗∗ *q*) = *poly* [] ⟷ *poly p* = *poly* [] ∨ *poly q* = *poly* []
⟨*proof*⟩

lemma (**in** *idom-char-0*) *poly-entire-neg*: (*poly* (*p* ∗∗∗ *q*) ≠ *poly* []) = ((*poly p* ≠ *poly* []) & (*poly q* ≠ *poly* []))
⟨*proof*⟩

lemma *fun-eq*: (*f* = *g*) = (∀ *x*. *f x* = *g x*)
⟨*proof*⟩

lemma (**in** *comm-ring-1*) *poly-add-minus-zero-iff*: (*poly* (*p* +++ −− *q*) = *poly* []) = (*poly p* = *poly q*)
⟨*proof*⟩

lemma (**in** *comm-ring-1*) *poly-add-minus-mult-eq*: *poly* (*p* ∗∗∗ *q* +++ −−(*p* ∗∗∗ *r*)) = *poly* (*p* ∗∗∗ (*q* +++ −− *r*))
⟨*proof*⟩

subclass (**in** *idom-char-0*) *comm-ring-1* ⟨*proof*⟩
lemma (**in** *idom-char-0*) *poly-mult-left-cancel*: (*poly* (*p* ∗∗∗ *q*) = *poly* (*p* ∗∗∗ *r*)) = (*poly p* = *poly* [] | *poly q* = *poly r*)
⟨*proof*⟩

lemma (**in** *recpower-idom*) *poly-exp-eq-zero*[*simp*]:
(*poly* (*p* %^ *n*) = *poly* []) = (*poly p* = *poly* [] & *n* ≠ *0*)
⟨*proof*⟩

lemma (**in** *semiring-1*) *one-neq-zero*[*simp*]: *1* ≠ *0* ⟨*proof*⟩
lemma (**in** *comm-ring-1*) *poly-prime-eq-zero*[*simp*]: *poly* [*a*,*1*] ≠ *poly* []
⟨*proof*⟩

lemma (**in** *recpower-idom*) *poly-exp-prime-eq-zero*: (*poly* ([*a*, *1*] %^ *n*) ≠ *poly* [])
⟨*proof*⟩

A more constructive notion of polynomials being trivial

lemma (**in** *idom-char-0*) *poly-zero-lemma'*: *poly* (*h* # *t*) = *poly* [] ==> *h* = *0* & *poly t* = *poly* []
⟨*proof*⟩

lemma (**in** *idom-char-0*) *poly-zero*: (*poly p* = *poly* []) = *list-all* (%*c*. *c* = *0*) *p*
⟨*proof*⟩

lemma (**in** *idom-char-0*) *poly-0*: *list-all* ($\lambda c.\ c = 0$) $p \Longrightarrow$ *poly* $p\ x = 0$
⟨*proof*⟩

Basics of divisibility.

lemma (**in** *idom*) *poly-primes*: ([a, *1*] *divides* (p ***** q)) = ([a, *1*] *divides* p | [a, *1*] *divides* q)
⟨*proof*⟩

lemma (**in** *comm-semiring-1*) *poly-divides-refl*[*simp*]: p *divides* p
⟨*proof*⟩

lemma (**in** *comm-semiring-1*) *poly-divides-trans*: [| p *divides* q; q *divides* r |] ==> p *divides* r
⟨*proof*⟩

lemma (**in** *recpower-comm-semiring-1*) *poly-divides-exp*: $m \leq n$ ==> (p % ^ m) *divides* (p % ^ n)
⟨*proof*⟩

lemma (**in** *recpower-comm-semiring-1*) *poly-exp-divides*: [| (p % ^ n) *divides* q; $m \leq n$ |] ==> (p % ^ m) *divides* q
⟨*proof*⟩

lemma (**in** *comm-semiring-0*) *poly-divides-add*:
[| p *divides* q; p *divides* r |] ==> p *divides* (q *+++* r)
⟨*proof*⟩

lemma (**in** *comm-ring-1*) *poly-divides-diff*:
[| p *divides* q; p *divides* (q *+++* r) |] ==> p *divides* r
⟨*proof*⟩

lemma (**in** *comm-ring-1*) *poly-divides-diff2*: [| p *divides* r; p *divides* (q *+++* r) |] ==> p *divides* q
⟨*proof*⟩

lemma (**in** *semiring-0*) *poly-divides-zero*: *poly* p = *poly* [] ==> q *divides* p
⟨*proof*⟩

lemma (**in** *semiring-0*) *poly-divides-zero2*[*simp*]: q *divides* []
⟨*proof*⟩

At last, we can consider the order of a root.

lemma (**in** *idom-char-0*) *poly-order-exists-lemma*:
assumes *lp*: *length* $p = d$ **and** p: *poly* $p \neq$ *poly* []
shows $\exists n\ q.\ p = mulexp\ n\ [-a,\ 1]\ q \wedge$ *poly* $q\ a \neq 0$
⟨*proof*⟩

lemma (**in** *recpower-comm-semiring-1*) *poly-mulexp*: *poly* (*mulexp n p q*) *x* = (*poly p x*) ^ *n* * *poly q x*
⟨*proof*⟩

lemma (**in** *comm-semiring-1*) *divides-left-mult*:
assumes *d*:(*p****q*) *divides r* **shows** *p divides r* ∧ *q divides r*
⟨*proof*⟩

lemma (**in** *recpower-semiring-1*)
zero-power-iff: *0* ^ *n* = (*if n* = *0 then 1 else 0*)
⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *poly-order-exists*:
assumes *lp*: *length p* = *d* **and** *p0*: *poly p* ≠ *poly* []
shows ∃ *n*. ([−*a*, *1*] % ^ *n*) *divides p* & ~(([−*a*, *1*] % ^ (*Suc n*)) *divides p*)
⟨*proof*⟩

lemma (**in** *semiring-1*) *poly-one-divides*[*simp*]: [*1*] *divides p*
⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *poly-order*: *poly p* ≠ *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 (**in** *recpower-idom-char-0*) *order*:
(([−*a*, *1*] % ^ *n*) *divides p* &
~(([−*a*, *1*] % ^ (*Suc n*)) *divides p*)) =
((*n* = *order a p*) & ~(*poly p* = *poly* []))
⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *order2*: [| *poly p* ≠ *poly* [] |]
==> ([−*a*, *1*] % ^ (*order a p*)) *divides p* &
~(([−*a*, *1*] % ^ (*Suc*(*order a p*))) *divides p*)
⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *order-unique*: [| *poly p* ≠ *poly* []; ([−*a*, *1*] % ^ *n*) *divides p*;
~(([−*a*, *1*] % ^ (*Suc n*)) *divides p*)
|] ==> (*n* = *order a p*)

⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *order-unique-lemma*: (*poly p* ≠ *poly* [] & ([−*a*, *1*] %^ *n*) *divides p* &
~(([−*a*, *1*] %^ (*Suc n*)) *divides p*))
==> (*n* = *order a p*)
⟨*proof*⟩

lemma (**in** *ring-1*) *order-poly*: *poly p* = *poly q* ==> *order a p* = *order a q*
⟨*proof*⟩

lemma (**in** *semiring-1*) *pexp-one*[*simp*]: *p* %^ (*Suc 0*) = *p*
⟨*proof*⟩

lemma (**in** *comm-ring-1*) *lemma-order-root*:
0 < *n* & [− *a*, *1*] %^ *n* *divides p* & ~ [− *a*, *1*] %^ (*Suc n*) *divides p*
⟹ *poly p a* = *0*
⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *order-root*: (*poly p a* = *0*) = ((*poly p* = *poly* []) | *order a p* ≠ *0*)
⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *order-divides*: (([−*a*, *1*] %^ *n*) *divides p*) = ((*poly p* = *poly* []) | *n* ≤ *order a p*)
⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *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*::'*a*::{*recpower-idom-char-0*}) *q*
⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *order-mult*:
assumes *pq0*: *poly* (*p* *** *q*) ≠ *poly* []
shows *order a* (*p* *** *q*) = *order a p* + *order a q*
⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *order-root2*: *poly p* ≠ *poly* [] ==> (*poly p a* = *0*) = (*order a p* ≠ *0*)
⟨*proof*⟩

lemma (**in** *semiring-1*) *pmult-one*[*simp*]: [*1*] *** *p* = *p* ⟨*proof*⟩

lemma (**in** *semiring-0*) *poly-Nil-zero*: *poly* [] = *poly* [*0*]
⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *rsquarefree-decomp*:
[| *rsquarefree p*; *poly p a* = *0* |]
==> ∃ *q*. (*poly p* = *poly* ([−*a*, *1*] *** *q*)) & *poly q a* ≠ *0*
⟨*proof*⟩

Normalization of a polynomial.

lemma (**in** *semiring-0*) *poly-normalize*[*simp*]: *poly* (*pnormalize p*) = *poly p*
⟨*proof*⟩

The degree of a polynomial.

lemma (**in** *semiring-0*) *lemma-degree-zero*:
list-all (%*c*. *c* = *0*) *p* ⟷ *pnormalize p* = []
⟨*proof*⟩

lemma (**in** *idom-char-0*) *degree-zero*:
assumes *pN*: *poly p* = *poly* [] **shows***degree p* = *0*
⟨*proof*⟩

lemma (**in** *semiring-0*) *pnormalize-sing*: (*pnormalize* [*x*] = [*x*]) ⟷ *x* ≠ *0* ⟨*proof*⟩
lemma (**in** *semiring-0*) *pnormalize-pair*: *y* ≠ *0* ⟷ (*pnormalize* [*x*, *y*] = [*x*, *y*])
⟨*proof*⟩
lemma (**in** *semiring-0*) *pnormal-cons*: *pnormal p* ⟹ *pnormal* (*c*#*p*)
⟨*proof*⟩
lemma (**in** *semiring-0*) *pnormal-tail*: *p*≠[] ⟹ *pnormal* (*c*#*p*) ⟹ *pnormal p*
⟨*proof*⟩

lemma (**in** *semiring-0*) *pnormal-last-nonzero*: *pnormal p* ==> *last p* ≠ *0*
⟨*proof*⟩

lemma (**in** *semiring-0*) *pnormal-length*: *pnormal p* ⟹ *0* < *length p*
⟨*proof*⟩

lemma (**in** *semiring-0*) *pnormal-last-length*: ⟦*0* < *length p* ; *last p* ≠ *0*⟧ ⟹ *pnormal p*
⟨*proof*⟩

lemma (**in** *semiring-0*) *pnormal-id*: *pnormal p* ⟷ (*0* < *length p* ∧ *last p* ≠ *0*)
⟨*proof*⟩

lemma (**in** *idom-char-0*) *poly-Cons-eq*: *poly* (*c*#*cs*) = *poly* (*d*#*ds*) ⟷ *c*=*d* ∧ *poly cs* = *poly ds* (**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

lemma (**in** *idom-char-0*) *pnormalize-unique*: *poly p* = *poly q* ⟹ *pnormalize p* =

pnormalize q
⟨*proof*⟩

lemma (**in** *idom-char-0*) *degree-unique*: **assumes** *pq*: *poly p* = *poly q*
shows *degree p* = *degree q*
⟨*proof*⟩

lemma (**in** *semiring-0*) *pnormalize-length*: *length* (*pnormalize p*) ≤ *length p* ⟨*proof*⟩

lemma (**in** *semiring-0*) *last-linear-mul-lemma*:
last ((*a* %∗ *p*) +++ (*x*#(*b* %∗ *p*))) = (*if p*=[] *then x else b*∗*last p*)

⟨*proof*⟩

lemma (**in** *semiring-1*) *last-linear-mul*: **assumes** *p*:*p*≠[] **shows** *last* ([*a*,*1*] ∗∗∗ *p*)
= *last p*
⟨*proof*⟩

lemma (**in** *semiring-0*) *pnormalize-eq*: *last p* ≠ *0* ⟹ *pnormalize p* = *p*
⟨*proof*⟩

lemma (**in** *semiring-0*) *last-pnormalize*: *pnormalize p* ≠ [] ⟹ *last* (*pnormalize p*) ≠ *0*
⟨*proof*⟩

lemma (**in** *semiring-0*) *pnormal-degree*: *last p* ≠ *0* ⟹ *degree p* = *length p* − *1*
⟨*proof*⟩

lemma (**in** *semiring-0*) *poly-Nil-ext*: *poly* [] = (λ*x*. *0*) ⟨*proof*⟩

lemma (**in** *idom-char-0*) *linear-mul-degree*: **assumes** *p*: *poly p* ≠ *poly* []
shows *degree* ([*a*,*1*] ∗∗∗ *p*) = *degree p* + *1*
⟨*proof*⟩

lemma (**in** *idom-char-0*) *linear-pow-mul-degree*:
degree([*a*,*1*] %^*n* ∗∗∗ *p*) = (*if poly p* = *poly* [] *then 0 else degree p* + *n*)
⟨*proof*⟩

lemma (**in** *recpower-idom-char-0*) *order-degree*:
assumes *p0*: *poly p* ≠ *poly* []
shows *order a p* ≤ *degree p*
⟨*proof*⟩

Tidier versions of finiteness of roots.

lemma (**in** *idom-char-0*) *poly-roots-finite-set*: *poly p* ≠ *poly* [] ==> *finite* {*x*. *poly p x* = *0*}
⟨*proof*⟩

bound for polynomial.

lemma *poly-mono*: $abs(x) \leq k \Longrightarrow abs(poly\ p\ (x::'a::\{ordered\text{-}idom\})) \leq poly\ (map\ abs\ p)\ k$
⟨*proof*⟩

lemma (**in** *semiring-0*) *poly-Sing*: $poly\ [c]\ x = c$ ⟨*proof*⟩

end

13 Dense-Linear-Order: Dense linear order without endpoints and a quantifier elimination procedure in Ferrante and Rackoff style

theory *Dense-Linear-Order*
imports *Arith-Tools*
uses
~~/*src/HOL/Tools/Qelim/qelim.ML*
~~/*src/HOL/Tools/Qelim/langford-data.ML*
~~/*src/HOL/Tools/Qelim/ferrante-rackoff-data.ML*
(~~/*src/HOL/Tools/Qelim/langford.ML*)
(~~/*src/HOL/Tools/Qelim/ferrante-rackoff.ML*)
begin

⟨*ML*⟩

context *linorder*
begin

lemma *less-not-permute*: $\neg\ (x < y \wedge y < x)$ ⟨*proof*⟩

lemma *gather-simps*:
shows
$(\exists x.\ (\forall y \in L.\ y < x) \wedge (\forall y \in U.\ x < y) \wedge x < u \wedge P\ x) \longleftrightarrow (\exists x.\ (\forall y \in L.\ y < x) \wedge (\forall y \in (insert\ u\ U).\ x < y) \wedge P\ x)$
and $(\exists x.\ (\forall y \in L.\ y < x) \wedge (\forall y \in U.\ x < y) \wedge l < x \wedge P\ x) \longleftrightarrow (\exists x.\ (\forall y \in (insert\ l\ L).\ y < x) \wedge (\forall y \in U.\ x < y) \wedge P\ x)$
$(\exists x.\ (\forall y \in L.\ y < x) \wedge (\forall y \in U.\ x < y) \wedge x < u) \longleftrightarrow (\exists x.\ (\forall y \in L.\ y < x) \wedge (\forall y \in (insert\ u\ U).\ x < y))$
and $(\exists x.\ (\forall y \in L.\ y < x) \wedge (\forall y \in U.\ x < y) \wedge l < x) \longleftrightarrow (\exists x.\ (\forall y \in (insert\ l\ L).\ y < x) \wedge (\forall y \in U.\ x < y))$ ⟨*proof*⟩

lemma
gather-start: $(\exists x.\ P\ x) \equiv (\exists x.\ (\forall y \in \{\}.\ y < x) \wedge (\forall y \in \{\}.\ x < y) \wedge P\ x)$
⟨*proof*⟩

Theorems for $\exists z.\ \forall x.\ x < z \longrightarrow (P\ x \longleftrightarrow P_{-\infty})$

lemma *minf-lt*: $\exists z\ .\ \forall x.\ x < z \longrightarrow (x < t \longleftrightarrow True)$ ⟨*proof*⟩
lemma *minf-gt*: $\exists z\ .\ \forall x.\ x < z \longrightarrow (t < x \longleftrightarrow False)$

⟨*proof*⟩

lemma *minf-le*: $\exists\, z.\ \forall\, x.\ x < z \longrightarrow (x \leq t \longleftrightarrow \mathit{True})$ ⟨*proof*⟩
lemma *minf-ge*: $\exists\, z.\ \forall\, x.\ x < z \longrightarrow (t \leq x \longleftrightarrow \mathit{False})$
⟨*proof*⟩
lemma *minf-eq*: $\exists\, z.\ \forall\, x.\ x < z \longrightarrow (x = t \longleftrightarrow \mathit{False})$ ⟨*proof*⟩
lemma *minf-neq*: $\exists\, z.\ \forall\, x.\ x < z \longrightarrow (x \neq t \longleftrightarrow \mathit{True})$ ⟨*proof*⟩
lemma *minf-P*: $\exists\, z.\ \forall\, x.\ x < z \longrightarrow (P \longleftrightarrow P)$ ⟨*proof*⟩

Theorems for $\exists\, z.\ \forall\, x.\ x < z \longrightarrow (P\ x \longleftrightarrow P_{+\infty})$

lemma *pinf-gt*: $\exists\, z\ .\ \forall\, x.\ z < x \longrightarrow (t < x \longleftrightarrow \mathit{True})$ ⟨*proof*⟩
lemma *pinf-lt*: $\exists\, z\ .\ \forall\, x.\ z < x \longrightarrow (x < t \longleftrightarrow \mathit{False})$
⟨*proof*⟩

lemma *pinf-ge*: $\exists\, z.\ \forall\, x.\ z < x \longrightarrow (t \leq x \longleftrightarrow \mathit{True})$ ⟨*proof*⟩
lemma *pinf-le*: $\exists\, z.\ \forall\, x.\ z < x \longrightarrow (x \leq t \longleftrightarrow \mathit{False})$
⟨*proof*⟩
lemma *pinf-eq*: $\exists\, z.\ \forall\, x.\ z < x \longrightarrow (x = t \longleftrightarrow \mathit{False})$ ⟨*proof*⟩
lemma *pinf-neq*: $\exists\, z.\ \forall\, x.\ z < x \longrightarrow (x \neq t \longleftrightarrow \mathit{True})$ ⟨*proof*⟩
lemma *pinf-P*: $\exists\, z.\ \forall\, x.\ z < x \longrightarrow (P \longleftrightarrow P)$ ⟨*proof*⟩

lemma *nmi-lt*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{True} \wedge x < t \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)$ ⟨*proof*⟩
lemma *nmi-gt*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{False} \wedge t < x \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)$
⟨*proof*⟩
lemma *nmi-le*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{True} \wedge x{\leq}\ t \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)$ ⟨*proof*⟩
lemma *nmi-ge*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{False} \wedge t{\leq}\ x \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)$ ⟨*proof*⟩
lemma *nmi-eq*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{False} \wedge\ x = t \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)$ ⟨*proof*⟩
lemma *nmi-neq*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{True} \wedge x \neq t \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)$ ⟨*proof*⟩
lemma *nmi-P*: $\forall\ x.\ {}^{\sim}P \wedge P \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)$ ⟨*proof*⟩
lemma *nmi-conj*: $[\![\forall\, x.\ \neg P1\,' \wedge P1\ x \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)\ ;$
$\forall\, x.\ \neg P2\,' \wedge P2\ x \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)]\!] \Longrightarrow$
$\forall\, x.\ \neg(P1\,' \wedge P2\,') \wedge (P1\ x \wedge P2\ x) \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)$ ⟨*proof*⟩
lemma *nmi-disj*: $[\![\forall\, x.\ \neg P1\,' \wedge P1\ x \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)\ ;$
$\forall\, x.\ \neg P2\,' \wedge P2\ x \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)]\!] \Longrightarrow$
$\forall\, x.\ \neg(P1\,' \vee P2\,') \wedge (P1\ x \vee P2\ x) \longrightarrow (\exists\ u{\in}\ U.\ u \leq x)$ ⟨*proof*⟩

lemma *npi-lt*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{False} \wedge\ x < t \longrightarrow (\exists\ u{\in}\ U.\ x \leq u)$ ⟨*proof*⟩
lemma *npi-gt*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{True} \wedge t < x \longrightarrow (\exists\ u{\in}\ U.\ x \leq u)$ ⟨*proof*⟩
lemma *npi-le*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{False} \wedge\ x \leq t \longrightarrow (\exists\ u{\in}\ U.\ x \leq u)$ ⟨*proof*⟩
lemma *npi-ge*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{True} \wedge t \leq x \longrightarrow (\exists\ u{\in}\ U.\ x \leq u)$ ⟨*proof*⟩
lemma *npi-eq*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{False} \wedge\ x = t \longrightarrow (\exists\ u{\in}\ U.\ x \leq u)$ ⟨*proof*⟩
lemma *npi-neq*: $t \in U \Longrightarrow \forall\, x.\ \neg \mathit{True} \wedge x \neq t \longrightarrow (\exists\ u{\in}\ U.\ x \leq u\ )$ ⟨*proof*⟩
lemma *npi-P*: $\forall\ x.\ {}^{\sim}P \wedge P \longrightarrow (\exists\ u{\in}\ U.\ x \leq u)$ ⟨*proof*⟩
lemma *npi-conj*: $[\![\forall\, x.\ \neg P1\,' \wedge P1\ x \longrightarrow (\exists\ u{\in}\ U.\ x \leq u)\ ;\ \forall\, x.\ \neg P2\,' \wedge P2\ x$
$\longrightarrow (\exists\ u{\in}\ U.\ x \leq u)]\!]$
$\Longrightarrow\ \forall\, x.\ \neg(P1\,' \wedge P2\,') \wedge (P1\ x \wedge P2\ x) \longrightarrow (\exists\ u{\in}\ U.\ x \leq u)$ ⟨*proof*⟩
lemma *npi-disj*: $[\![\forall\, x.\ \neg P1\,' \wedge P1\ x \longrightarrow (\exists\ u{\in}\ U.\ x \leq u)\ ; \forall\, x.\ \neg P2\,' \wedge P2\ x$
$\longrightarrow (\exists\ u{\in}\ U.\ x \leq u)]\!]$
$\Longrightarrow \forall\, x.\ \neg(P1\,' \vee P2\,') \wedge (P1\ x \vee P2\ x) \longrightarrow (\exists\ u{\in}\ U.\ x \leq u)$ ⟨*proof*⟩

lemma *lin-dense-lt*: $t \in U \Longrightarrow \forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge x < t \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow y < t)$
⟨*proof*⟩

lemma *lin-dense-gt*: $t \in U \Longrightarrow \forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge t < x \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow t < y)$
⟨*proof*⟩

lemma *lin-dense-le*: $t \in U \Longrightarrow \forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge x \leq t \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow y \leq t)$
⟨*proof*⟩

lemma *lin-dense-ge*: $t \in U \Longrightarrow \forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge t \leq x \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow t \leq y)$
⟨*proof*⟩
lemma *lin-dense-eq*: $t \in U \Longrightarrow \forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge x = t \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow y = t)$ ⟨*proof*⟩
lemma *lin-dense-neq*: $t \in U \Longrightarrow \forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge x \neq t \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow y \neq t)$ ⟨*proof*⟩
lemma *lin-dense-P*: $\forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge P \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow P)$ ⟨*proof*⟩

lemma *lin-dense-conj*:
$[\![\forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge P1\ x \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow P1\ y)\ ;$
$\forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge P2\ x \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow P2\ y)]\!] \Longrightarrow$
$\forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge (P1\ x \wedge P2\ x) \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow (P1\ y \wedge P2\ y))$
⟨*proof*⟩
lemma *lin-dense-disj*:
$[\![\forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge P1\ x \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow P1\ y)\ ;$
$\forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge P2\ x \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow P2\ y)]\!] \Longrightarrow$
$\forall x\ l\ u.\ (\forall\ t.\ l < t \wedge t < u \longrightarrow t \notin U) \wedge l < x \wedge x < u \wedge (P1\ x \vee P2\ x) \longrightarrow (\forall\ y.\ l < y \wedge y < u \longrightarrow (P1\ y \vee P2\ y))$
⟨*proof*⟩

lemma *npmibnd*: $[\![\forall x.\ \neg\ MP \wedge P\ x \longrightarrow (\exists\ u \in U.\ u \leq x); \forall x.\ \neg PP \wedge P\ x \longrightarrow (\exists\ u \in U.\ x \leq u)]\!]$
$\Longrightarrow \forall x.\ \neg\ MP \wedge \neg PP \wedge P\ x \longrightarrow (\exists\ u \in U.\ \exists\ u' \in U.\ u \leq x \wedge x \leq u')$
⟨*proof*⟩

lemma *finite-set-intervals*:
assumes *px*: $P\ x$ **and** *lx*: $l \leq x$ **and** *xu*: $x \leq u$ **and** *linS*: $l \in S$
and *uinS*: $u \in S$ **and** *fS*:*finite* S **and** *lS*: $\forall\ x \in S.\ l \leq x$ **and** *Su*: $\forall\ x \in S.\ x \leq u$

shows $\exists\ a \in S.\ \exists\ b \in S.\ (\forall\ y.\ a < y \wedge y < b \longrightarrow y \notin S) \wedge a \leq x \wedge x \leq b \wedge P\ x$
⟨*proof*⟩

lemma *finite-set-intervals2*:
assumes *px*: $P\ x$ **and** *lx*: $l \leq x$ **and** *xu*: $x \leq u$ **and** *linS*: $l \in S$
and *uinS*: $u \in S$ **and** *fS*:*finite S* **and** *lS*: $\forall\ x \in S.\ l \leq x$ **and** *Su*: $\forall\ x \in S.\ x \leq u$
shows $(\exists\ s \in S.\ P\ s) \vee (\exists\ a \in S.\ \exists\ b \in S.\ (\forall\ y.\ a < y \wedge y < b \longrightarrow y \notin S) \wedge a < x \wedge x < b \wedge P\ x)$
⟨*proof*⟩

end

14 The classical QE after Langford for dense linear orders

context *dense-linear-order*
begin

lemma *dlo-qe-bnds*:
assumes *ne*: $L \neq \{\}$ **and** *neU*: $U \neq \{\}$ **and** *fL*: *finite L* **and** *fU*: *finite U*
shows $(\exists x.\ (\forall y \in L.\ y < x) \wedge (\forall y \in U.\ x < y)) \equiv (\forall\ l \in L.\ \forall u \in U.\ l < u)$
⟨*proof*⟩

lemma *dlo-qe-noub*:
assumes *ne*: $L \neq \{\}$ **and** *fL*: *finite L*
shows $(\exists x.\ (\forall y \in L.\ y < x) \wedge (\forall y \in \{\}.\ x < y)) \equiv \mathit{True}$
⟨*proof*⟩

lemma *dlo-qe-nolb*:
assumes *ne*: $U \neq \{\}$ **and** *fU*: *finite U*
shows $(\exists x.\ (\forall y \in \{\}.\ y < x) \wedge (\forall y \in U.\ x < y)) \equiv \mathit{True}$
⟨*proof*⟩

lemma *exists-neq*: $\exists (x::'a).\ x \neq t\ \exists (x::'a).\ t \neq x$
⟨*proof*⟩

lemmas *dlo-simps* = *order-refl less-irrefl not-less not-le exists-neq le-less neq-iff linear less-not-permute*

lemma *axiom*: *dense-linear-order* $(op \leq)$ $(op <)$ ⟨*proof*⟩
lemma *atoms*:
includes *meta-term-syntax*
shows *TERM* (*less* :: $'a \Rightarrow$ -)
and *TERM* (*less-eq* :: $'a \Rightarrow$ -)
and *TERM* $(op = :: 'a \Rightarrow$ -) ⟨*proof*⟩

declare *axiom*[*langford qe*: *dlo-qe-bnds dlo-qe-nolb dlo-qe-noub gather*: *gather-start gather-simps atoms*: *atoms*]
declare *dlo-simps*[*langfordsimp*]

end

lemma *dnf*:
(*P* & (*Q* | *R*)) = ((*P*&*Q*) | (*P*&*R*))
((*Q* | *R*) & *P*) = ((*Q*&*P*) | (*R*&*P*))
⟨*proof*⟩

lemmas *weak-dnf-simps* = *simp-thms dnf*

lemma *nnf-simps*:
(¬(*P* ∧ *Q*)) = (¬*P* ∨ ¬*Q*) (¬(*P* ∨ *Q*)) = (¬*P* ∧ ¬*Q*) (*P* ⟶ *Q*) = (¬*P* ∨ *Q*)
(*P* = *Q*) = ((*P* ∧ *Q*) ∨ (¬*P* ∧ ¬ *Q*)) (¬ ¬(*P*)) = *P*
⟨*proof*⟩

lemma *ex-distrib*: (∃ *x*. *P x* ∨ *Q x*) ⟷ ((∃ *x*. *P x*) ∨ (∃ *x*. *Q x*)) ⟨*proof*⟩

lemmas *dnf-simps* = *weak-dnf-simps nnf-simps ex-distrib*

⟨*ML*⟩

15 Contructive dense linear orders yield QE for linear arithmetic over ordered Fields – see *Arith-Tools.thy*

Linear order without upper bounds

locale *linorder-stupid-syntax* = *linorder*
begin
notation
less-eq (*op* ⊑) **and**
less-eq ((-/ ⊑ -) [*51*, *51*] *50*) **and**
less (*op* ⊏) **and**
less ((-/ ⊏ -) [*51*, *51*] *50*)

end

locale *linorder-no-ub* = *linorder-stupid-syntax* +
assumes *gt-ex*: ∃ *y*. *less x y*
begin
lemma *ge-ex*: ∃ *y*. *x* ⊑ *y* ⟨*proof*⟩

Theorems for ∃ *z*. ∀ *x*. *z* ⊏ *x* ⟶ (*P x* ⟷ $P_{+\infty}$)

lemma *pinf-conj*:
assumes *ex1*: ∃ *z1*. ∀ *x*. *z1* ⊏ *x* ⟶ (*P1 x* ⟷ *P1'*)

and *ex2*: $\exists\, z2.\ \forall\, x.\ z2 \sqsubset x \longrightarrow (P2\ x \longleftrightarrow P2')$
shows $\exists\, z.\ \forall\, x.\ z \sqsubset\ x \longrightarrow ((P1\ x \wedge P2\ x) \longleftrightarrow (P1' \wedge P2'))$
⟨*proof*⟩

lemma *pinf-disj*:
assumes *ex1*: $\exists\, z1.\ \forall\, x.\ z1 \sqsubset x \longrightarrow (P1\ x \longleftrightarrow P1')$
and *ex2*: $\exists\, z2.\ \forall\, x.\ z2 \sqsubset x \longrightarrow (P2\ x \longleftrightarrow P2')$
shows $\exists\, z.\ \forall\, x.\ z \sqsubset\ x \longrightarrow ((P1\ x \vee P2\ x) \longleftrightarrow (P1' \vee P2'))$
⟨*proof*⟩

lemma *pinf-ex*: **assumes** *ex*:$\exists\, z.\ \forall\, x.\ z \sqsubset x \longrightarrow (P\ x \longleftrightarrow P1)$ **and** *p1*: *P1* **shows** $\exists\ x.\ P\ x$
⟨*proof*⟩

end

Linear order without upper bounds

locale *linorder-no-lb* = *linorder-stupid-syntax* +
assumes *lt-ex*: $\exists\, y.$ *less* $y\ x$
begin
lemma *le-ex*: $\exists\, y.\ y \sqsubseteq x$ ⟨*proof*⟩

Theorems for $\exists\, z.\ \forall\, x.\ x \sqsubset z \longrightarrow (P\ x \longleftrightarrow P_{-\infty})$

lemma *minf-conj*:
assumes *ex1*: $\exists\, z1.\ \forall\, x.\ x \sqsubset z1 \longrightarrow (P1\ x \longleftrightarrow P1')$
and *ex2*: $\exists\, z2.\ \forall\, x.\ x \sqsubset z2 \longrightarrow (P2\ x \longleftrightarrow P2')$
shows $\exists\, z.\ \forall\, x.\ x \sqsubset\ z \longrightarrow ((P1\ x \wedge P2\ x) \longleftrightarrow (P1' \wedge P2'))$
⟨*proof*⟩

lemma *minf-disj*:
assumes *ex1*: $\exists\, z1.\ \forall\, x.\ x \sqsubset z1 \longrightarrow (P1\ x \longleftrightarrow P1')$
and *ex2*: $\exists\, z2.\ \forall\, x.\ x \sqsubset z2 \longrightarrow (P2\ x \longleftrightarrow P2')$
shows $\exists\, z.\ \forall\, x.\ x \sqsubset\ z \longrightarrow ((P1\ x \vee P2\ x) \longleftrightarrow (P1' \vee P2'))$
⟨*proof*⟩

lemma *minf-ex*: **assumes** *ex*:$\exists\, z.\ \forall\, x.\ x \sqsubset z \longrightarrow (P\ x \longleftrightarrow P1)$ **and** *p1*: *P1* **shows** $\exists\ x.\ P\ x$
⟨*proof*⟩

end

locale *constr-dense-linear-order* = *linorder-no-lb* + *linorder-no-ub* +
fixes *between*
assumes *between-less*: *less* $x\ y \Longrightarrow$ *less* x (*between* $x\ y$) $\wedge$ *less* (*between* $x\ y$) y
and *between-same*: *between* $x\ x = x$

interpretation *constr-dense-linear-order* < *dense-linear-order*
⟨*proof*⟩

context *constr-dense-linear-order*
begin

lemma *rinf-U*:
assumes *fU*: *finite U*
and *lin-dense*: $\forall x\ l\ u.\ (\forall\ t.\ l \sqsubset t \wedge t \sqsubset u \longrightarrow t \notin U) \wedge l \sqsubset x \wedge x \sqsubset u \wedge P\ x \longrightarrow (\forall\ y.\ l \sqsubset y \wedge y \sqsubset u \longrightarrow P\ y\ )$
and *nmpiU*: $\forall x.\ \neg\ MP \wedge \neg PP \wedge P\ x \longrightarrow (\exists\ u{\in}\ U.\ \exists\ u' \in U.\ u \sqsubseteq x \wedge x \sqsubseteq u')$
and *nmi*: $\neg\ MP$ **and** *npi*: $\neg\ PP$ **and** *ex*: $\exists\ x.\ \ P\ x$
shows $\exists\ u{\in}\ U.\ \exists\ u' \in U.\ P\ (between\ u\ u')$
⟨*proof*⟩
term *linorder.Min less-eq*
⟨*proof*⟩

theorem *fr-eq*:
assumes *fU*: *finite U*
and *lin-dense*: $\forall x\ l\ u.\ (\forall\ t.\ l \sqsubset t \wedge t \sqsubset u \longrightarrow t \notin U) \wedge l \sqsubset x \wedge x \sqsubset u \wedge P\ x \longrightarrow (\forall\ y.\ l \sqsubset y \wedge y \sqsubset u \longrightarrow P\ y\ )$
and *nmibnd*: $\forall x.\ \neg\ MP \wedge P\ x \longrightarrow (\exists\ u{\in}\ U.\ u \sqsubseteq x)$
and *npibnd*: $\forall x.\ \neg PP \wedge P\ x \longrightarrow (\exists\ u{\in}\ U.\ x \sqsubseteq u)$
and *mi*: $\exists z.\ \forall x.\ x \sqsubset z \longrightarrow (P\ x = MP)$ **and** *pi*: $\exists z.\ \forall x.\ z \sqsubset x \longrightarrow (P\ x = PP)$
shows $(\exists\ x.\ P\ x) \equiv (MP \vee PP \vee (\exists\ u \in U.\ \exists\ u'{\in}\ U.\ P\ (between\ u\ u')))$
(**is** - ≡ (- ∨ - ∨ *?F*) **is** *?E* ≡ *?D*)
⟨*proof*⟩

lemmas *minf-thms* = *minf-conj minf-disj minf-eq minf-neq minf-lt minf-le minf-gt minf-ge minf-P*
lemmas *pinf-thms* = *pinf-conj pinf-disj pinf-eq pinf-neq pinf-lt pinf-le pinf-gt pinf-ge pinf-P*

lemmas *nmi-thms* = *nmi-conj nmi-disj nmi-eq nmi-neq nmi-lt nmi-le nmi-gt nmi-ge nmi-P*
lemmas *npi-thms* = *npi-conj npi-disj npi-eq npi-neq npi-lt npi-le npi-gt npi-ge npi-P*
lemmas *lin-dense-thms* = *lin-dense-conj lin-dense-disj lin-dense-eq lin-dense-neq lin-dense-lt lin-dense-le lin-dense-gt lin-dense-ge lin-dense-P*

lemma *ferrack-axiom*: *constr-dense-linear-order less-eq less between*
⟨*proof*⟩
lemma *atoms*:
includes *meta-term-syntax*
shows *TERM* (*less* :: *'a* ⇒ -)
and *TERM* (*less-eq* :: *'a* ⇒ -)
and *TERM* (*op* = :: *'a* ⇒ -) ⟨*proof*⟩

declare *ferrack-axiom* [*ferrack minf*: *minf-thms pinf*: *pinf-thms*

nmi: *nmi-thms npi*: *npi-thms lindense*:
lin-dense-thms qe: *fr-eq atoms*: *atoms*]

⟨*ML*⟩

end

⟨*ML*⟩

15.1 Ferrante and Rackoff algorithm over ordered fields

lemma *neg-prod-lt*:$(c::'a::ordered\text{-}field) < 0 \Longrightarrow ((c*x < 0) == (x > 0))$
⟨*proof*⟩

lemma *pos-prod-lt*:$(c::'a::ordered\text{-}field) > 0 \Longrightarrow ((c*x < 0) == (x < 0))$
⟨*proof*⟩

lemma *neg-prod-sum-lt*: $(c::'a::ordered\text{-}field) < 0 \Longrightarrow ((c*x + t< 0) == (x > (-1/c)*t))$
⟨*proof*⟩

lemma *pos-prod-sum-lt*:$(c::'a::ordered\text{-}field) > 0 \Longrightarrow ((c*x + t < 0) == (x < (-1/c)*t))$
⟨*proof*⟩

lemma *sum-lt*:$((x::'a::pordered\text{-}ab\text{-}group\text{-}add) + t < 0) == (x < - t)$
⟨*proof*⟩

lemma *neg-prod-le*:$(c::'a::ordered\text{-}field) < 0 \Longrightarrow ((c*x <= 0) == (x >= 0))$
⟨*proof*⟩

lemma *pos-prod-le*:$(c::'a::ordered\text{-}field) > 0 \Longrightarrow ((c*x <= 0) == (x <= 0))$
⟨*proof*⟩

lemma *neg-prod-sum-le*: $(c::'a::ordered\text{-}field) < 0 \Longrightarrow ((c*x + t <= 0) == (x >= (- 1/c)*t))$
⟨*proof*⟩

lemma *pos-prod-sum-le*:$(c::'a::ordered\text{-}field) > 0 \Longrightarrow ((c*x + t <= 0) == (x <= (- 1/c)*t))$
⟨*proof*⟩

lemma *sum-le*:$((x::'a::pordered\text{-}ab\text{-}group\text{-}add) + t <= 0) == (x <= - t)$
⟨*proof*⟩

lemma *nz-prod-eq*:$(c::'a::ordered\text{-}field) \neq 0 \Longrightarrow ((c*x = 0) == (x = 0))$ ⟨*proof*⟩
lemma *nz-prod-sum-eq*: $(c::'a::ordered\text{-}field) \neq 0 \Longrightarrow ((c*x + t = 0) == (x = (- 1/c)*t))$
⟨*proof*⟩

lemma *sum-eq*:((*x*::'*a*::*pordered-ab-group-add*) + *t* = *0*) == (*x* = − *t*)
⟨*proof*⟩

interpretation *class-ordered-field-dense-linear-order*: *constr-dense-linear-order*
[*op* <= *op* <
λ *x y*. *1/2* ∗ ((*x*::'*a*::{*ordered-field*,*recpower*,*number-ring*}) + *y*)]
⟨*proof*⟩

⟨*ML*⟩

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*
⟨*proof*⟩

lemma *fact-not-eq-zero* [*simp*]: *fact n* ≠ *0*
⟨*proof*⟩

lemma *real-of-nat-fact-not-zero* [*simp*]: *real* (*fact n*) ≠ *0*
⟨*proof*⟩

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

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

lemma *fact-ge-one* [*simp*]: *1* ≤ *fact n*
⟨*proof*⟩

lemma *fact-mono*: *m* ≤ *n* ==> *fact m* ≤ *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* ≤ *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 SEQ: Sequences and Convergence

theory *SEQ*
imports *../Real/Real*
begin

definition
Zseq :: [*nat* ⇒ *'a*::*real-normed-vector*] ⇒ *bool* **where**
— Standard definition of sequence converging to zero
Zseq X = (∀ *r*>*0*. ∃ *no*. ∀ *n*≥*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* −−−−> *L*)

definition
convergent :: (*nat* => *'a*::*real-normed-vector*) => *bool* **where**
— Standard definition of convergence
convergent X = (∃ *L*. *X* −−−−> *L*)

definition
Bseq :: (*nat* => *'a*::*real-normed-vector*) => *bool* **where**
— Standard definition for bounded sequence
Bseq X = (∃ *K*>*0*.∀ *n*. *norm* (*X n*) ≤ *K*)

definition
monoseq :: (*nat*=>*real*)=>*bool* **where**
— Definition for monotonicity
monoseq X = ((∀ *m*. ∀ *n*≥*m*. *X m* ≤ *X n*) | (∀ *m*. ∀ *n*≥*m*. *X n* ≤ *X m*))

definition
subseq :: (*nat* => *nat*) => *bool* **where**
— Definition of subsequence
subseq f = (∀ *m*. ∀ *n*>*m*. (*f m*) < (*f n*))

definition
Cauchy :: (*nat* => *'a*::*real-normed-vector*) => *bool* **where**
— Standard definition of the Cauchy condition
Cauchy X = (∀ *e*>*0*. ∃ *M*. ∀ *m* ≥ *M*. ∀ *n* ≥ *M*. *norm* (*X m* − *X n*) < *e*)

17.1 Bounded Sequences

lemma *BseqI'*: **assumes** *K*: ⋀*n*. *norm* (*X n*) ≤ *K* **shows** *Bseq X*
⟨*proof*⟩

lemma *BseqE*: ⟦*Bseq X*; ⋀*K*. ⟦*0* < *K*; ∀ *n*. *norm* (*X n*) ≤ *K*⟧ ⟹ *Q*⟧ ⟹ *Q*
⟨*proof*⟩

lemma *BseqI2'*: **assumes** *K*: ∀ *n*≥*N*. *norm* (*X n*) ≤ *K* **shows** *Bseq X*
⟨*proof*⟩

lemma *Bseq-ignore-initial-segment*: *Bseq X* ⟹ *Bseq* (λ*n*. *X* (*n* + *k*))
⟨*proof*⟩

lemma *Bseq-offset*: *Bseq* (λ*n*. *X* (*n* + *k*)) ⟹ *Bseq X*
⟨*proof*⟩

17.2 Sequences That Converge to Zero

lemma *ZseqI*:
(⋀*r*. *0* < *r* ⟹ ∃ *no*. ∀ *n*≥*no*. *norm* (*X n*) < *r*) ⟹ *Zseq X*

⟨*proof*⟩

lemma *ZseqD*:
  ⟦*Zseq X*; *0* < *r*⟧ ⟹ ∃ *no*. ∀ *n*≥*no*. *norm* (*X n*) < *r*
⟨*proof*⟩

lemma *Zseq-zero*: *Zseq* (λ*n*. *0*)
⟨*proof*⟩

lemma *Zseq-const-iff*: *Zseq* (λ*n*. *k*) = (*k* = *0*)
⟨*proof*⟩

lemma *Zseq-norm-iff*: *Zseq* (λ*n*. *norm* (*X n*)) = *Zseq* (λ*n*. *X n*)
⟨*proof*⟩

lemma *Zseq-imp-Zseq*:
  assumes *X*: *Zseq X*
  assumes *Y*: ⋀*n*. *norm* (*Y n*) ≤ *norm* (*X n*) ∗ *K*
  shows *Zseq* (λ*n*. *Y n*)
⟨*proof*⟩

lemma *Zseq-le*: ⟦*Zseq Y*; ∀ *n*. *norm* (*X n*) ≤ *norm* (*Y n*)⟧ ⟹ *Zseq X*
⟨*proof*⟩

lemma *Zseq-add*:
  assumes *X*: *Zseq X*
  assumes *Y*: *Zseq Y*
  shows *Zseq* (λ*n*. *X n* + *Y n*)
⟨*proof*⟩

lemma *Zseq-minus*: *Zseq X* ⟹ *Zseq* (λ*n*. − *X n*)
⟨*proof*⟩

lemma *Zseq-diff*: ⟦*Zseq X*; *Zseq Y*⟧ ⟹ *Zseq* (λ*n*. *X n* − *Y n*)
⟨*proof*⟩

lemma (**in** *bounded-linear*) *Zseq*:
  assumes *X*: *Zseq X*
  shows *Zseq* (λ*n*. *f* (*X n*))
⟨*proof*⟩

lemma (**in** *bounded-bilinear*) *Zseq*:
  assumes *X*: *Zseq X*
  assumes *Y*: *Zseq Y*
  shows *Zseq* (λ*n*. *X n* ∗∗ *Y n*)
⟨*proof*⟩

lemma (**in** *bounded-bilinear*) *Zseq-prod-Bseq*:
  assumes *X*: *Zseq X*

assumes Y: *Bseq* Y
shows *Zseq* $(\lambda n.\ X\ n\ **\ Y\ n)$
⟨*proof*⟩

lemma (**in** *bounded-bilinear*) *Bseq-prod-Zseq*:
assumes X: *Bseq* X
assumes Y: *Zseq* Y
shows *Zseq* $(\lambda n.\ X\ n\ **\ Y\ n)$
⟨*proof*⟩

lemma (**in** *bounded-bilinear*) *Zseq-left*:
Zseq $X \Longrightarrow$ *Zseq* $(\lambda n.\ X\ n\ **\ a)$
⟨*proof*⟩

lemma (**in** *bounded-bilinear*) *Zseq-right*:
Zseq $X \Longrightarrow$ *Zseq* $(\lambda n.\ a\ **\ X\ n)$
⟨*proof*⟩

lemmas *Zseq-mult* = *mult.Zseq*
lemmas *Zseq-mult-right* = *mult.Zseq-right*
lemmas *Zseq-mult-left* = *mult.Zseq-left*

17.3 Limits of Sequences

lemma *LIMSEQ-iff*:
$(X\ ----->\ L) = (\forall r{>}0.\ \exists no.\ \forall n \geq no.\ norm\ (X\ n - L) < r)$
⟨*proof*⟩

lemma *LIMSEQ-Zseq-iff*: $((\lambda n.\ X\ n)\ ----->\ L) = Zseq\ (\lambda n.\ X\ n - L)$
⟨*proof*⟩

lemma *LIMSEQ-I*:
$(\bigwedge r.\ 0 < r \Longrightarrow \exists no.\ \forall n{\geq}no.\ norm\ (X\ n - L) < r) \Longrightarrow X\ ----->\ L$
⟨*proof*⟩

lemma *LIMSEQ-D*:
$[\![X\ ----->\ L;\ 0 < r]\!] \Longrightarrow \exists no.\ \forall n{\geq}no.\ norm\ (X\ n - L) < r$
⟨*proof*⟩

lemma *LIMSEQ-const*: $(\lambda n.\ k)\ ----->\ k$
⟨*proof*⟩

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*:

$f \longrightarrow a \Longrightarrow (\lambda n.\ f\ (n + k)) \longrightarrow a$
⟨*proof*⟩

lemma *LIMSEQ-offset*:
$(\lambda n.\ f\ (n + k)) \longrightarrow a \Longrightarrow f \longrightarrow a$
⟨*proof*⟩

lemma *LIMSEQ-Suc*: $f \longrightarrow l \Longrightarrow (\lambda n.\ f\ (Suc\ n)) \longrightarrow l$
⟨*proof*⟩

lemma *LIMSEQ-imp-Suc*: $(\lambda n.\ f\ (Suc\ n)) \longrightarrow l \Longrightarrow f \longrightarrow l$
⟨*proof*⟩

lemma *LIMSEQ-Suc-iff*: $(\lambda n.\ f\ (Suc\ n)) \longrightarrow l = f \longrightarrow l$
⟨*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*: $[\![X \longrightarrow a;\ Y \longrightarrow b]\!] \Longrightarrow (\lambda n.\ X\ n + Y\ n) \longrightarrow a + b$
⟨*proof*⟩

lemma *LIMSEQ-minus*: $X \longrightarrow a \Longrightarrow (\lambda n.\ -\ X\ n) \longrightarrow -\ a$
⟨*proof*⟩

lemma *LIMSEQ-minus-cancel*: $(\lambda n.\ -\ X\ n) \longrightarrow -\ a \Longrightarrow X \longrightarrow a$
⟨*proof*⟩

lemma *LIMSEQ-diff*: $[\![X \longrightarrow a;\ Y \longrightarrow b]\!] \Longrightarrow (\lambda n.\ X\ n - Y\ n) \longrightarrow a - b$
⟨*proof*⟩

lemma *LIMSEQ-unique*: $[\![X \longrightarrow a;\ X \longrightarrow b]\!] \Longrightarrow a = b$
⟨*proof*⟩

lemma (**in** *bounded-linear*) *LIMSEQ*:
$X \longrightarrow a \Longrightarrow (\lambda n.\ f\ (X\ n)) \longrightarrow f\ a$
⟨*proof*⟩

lemma (**in** *bounded-bilinear*) *LIMSEQ*:
$[\![X \longrightarrow a;\ Y \longrightarrow b]\!] \Longrightarrow (\lambda n.\ X\ n\ **\ Y\ n) \longrightarrow a\ **\ b$

⟨*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-ring*) ≠ *0*; *b* ≠ *0*⟧
⟹ *inverse a* − *inverse b* = − (*inverse a* * (*a* − *b*) * *inverse b*)
⟨*proof*⟩

lemma *Bseq-inverse-lemma*:
fixes *x* :: *'a*::*real-normed-div-algebra*
shows ⟦*r* ≤ *norm x*; *0* < *r*⟧ ⟹ *norm* (*inverse x*) ≤ *inverse r*
⟨*proof*⟩

lemma *Bseq-inverse*:
fixes *a* :: *'a*::*real-normed-div-algebra*
assumes *X*: *X* ----> *a*
assumes *a*: *a* ≠ *0*
shows *Bseq* (λ*n*. *inverse* (*X n*))
⟨*proof*⟩

lemma *LIMSEQ-inverse-lemma*:
fixes *a* :: *'a*::*real-normed-div-algebra*
shows ⟦*X* ----> *a*; *a* ≠ *0*; ∀ *n*. *X n* ≠ *0*⟧
⟹ (λ*n*. *inverse* (*X n*)) ----> *inverse a*
⟨*proof*⟩

lemma *LIMSEQ-inverse*:
fixes *a* :: *'a*::*real-normed-div-algebra*
assumes *X*: *X* ----> *a*
assumes *a*: *a* ≠ *0*
shows (λ*n*. *inverse* (*X n*)) ----> *inverse a*
⟨*proof*⟩

lemma *LIMSEQ-divide*:
fixes *a b* :: *'a*::*real-normed-field*
shows ⟦*X* ----> *a*; *Y* ----> *b*; *b* ≠ *0*⟧ ⟹ (λ*n*. *X n* / *Y n*) ----> *a* / *b*
⟨*proof*⟩

lemma *LIMSEQ-pow*:
fixes *a* :: *'a*::{*real-normed-algebra*,*recpower*}
shows *X* ----> *a* ⟹ (λ*n*. (*X n*) ^ *m*) ----> *a* ^ *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 $L :: 'a \Rightarrow 'b{::}\{real\text{-}normed\text{-}algebra, comm\text{-}ring\text{-}1\}$
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*:
$(\%n.\ r + inverse(real(Suc\ n))) \longrightarrow r$
⟨*proof*⟩

lemma *LIMSEQ-inverse-real-of-nat-add-minus*:
$(\%n.\ r + -inverse(real(Suc\ n))) \longrightarrow r$
⟨*proof*⟩

lemma *LIMSEQ-inverse-real-of-nat-add-minus-mult*:
$(\%n.\ r*(\ 1 + -inverse(real(Suc\ n)))) \longrightarrow r$
⟨*proof*⟩

lemma *LIMSEQ-le-const*:
$[\![X \longrightarrow (x::real);\ \exists N.\ \forall n{\geq}N.\ a \leq X\ n]\!] \Longrightarrow a \leq x$
⟨*proof*⟩

lemma *LIMSEQ-le-const2*:
$[\![X \longrightarrow (x::real);\ \exists N.\ \forall n{\geq}N.\ X\ n \leq a]\!] \Longrightarrow x \leq a$
⟨*proof*⟩

lemma *LIMSEQ-le*:
$[\![X \longrightarrow x;\ Y \longrightarrow y;\ \exists N.\ \forall n{\geq}N.\ X\ n \leq Y\ n]\!] \Longrightarrow x \leq (y::real)$
⟨*proof*⟩

17.4 Convergence

lemma *limI*: $X \longrightarrow L \Longrightarrow lim\ X = L$
⟨*proof*⟩

lemma *convergentD*: $convergent\ X \Longrightarrow \exists L.\ (X \longrightarrow L)$
⟨*proof*⟩

lemma *convergentI*: $(X \longrightarrow L) \Longrightarrow convergent\ X$
⟨*proof*⟩

lemma *convergent-LIMSEQ-iff*: $convergent\ X = (X \longrightarrow lim\ X)$
⟨*proof*⟩

lemma *convergent-minus-iff*: $(convergent\ X) = (convergent\ (\%n.\ -(X\ n)))$
⟨*proof*⟩

17.5 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*:
$monoseq\ X = ((\forall n.\ X\ n \leq X\ (Suc\ n))$
$\quad | (\forall n.\ X\ (Suc\ n) \leq X\ n))$
⟨*proof*⟩

lemma *monoI1*: $\forall m.\ \forall\ n \geq m.\ X\ m \leq X\ n \Longrightarrow monoseq\ X$
⟨*proof*⟩

lemma *monoI2*: $\forall m.\ \forall\ n \geq m.\ X\ n \leq X\ m \Longrightarrow monoseq\ X$
⟨*proof*⟩

lemma *mono-SucI1*: $\forall n.\ X\ n \leq X\ (Suc\ n) \Longrightarrow monoseq\ X$
⟨*proof*⟩

lemma *mono-SucI2*: $\forall n.\ X\ (Suc\ n) \leq X\ n \Longrightarrow monoseq\ X$
⟨*proof*⟩

Bounded Sequence

lemma *BseqD*: $Bseq\ X \Longrightarrow \exists K.\ 0 < K\ \&\ (\forall n.\ norm\ (X\ n) \leq K)$
⟨*proof*⟩

lemma *BseqI*: $[|\ 0 < K; \forall n.\ norm\ (X\ n) \leq K\ |] \Longrightarrow Bseq\ X$
⟨*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*⟩

17.5.1 Upper Bounds and Lubs of Bounded Sequences

lemma *Bseq-isUb*:
$!!(X::nat=>real).\ Bseq\ X \Longrightarrow \exists U.\ isUb\ (UNIV::real\ set)\ \{x.\ \exists n.\ X\ n = x\}\ U$
⟨*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*⟩

17.5.2 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*⟩

17.5.3 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*⟩

17.6 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*⟩

17.6.1 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*⟩

17.6.2 Cauchy Sequences are Convergent

axclass *banach* $\subseteq$ *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 :: $nat \Rightarrow real$
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 :: $nat \Rightarrow real$
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*⟩

17.7 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*: $|c| < (1::real) ==> (\%n.\ c \hat{} n) ----> 0$
⟨*proof*⟩

end

18 Series: Finite Summation and Infinite Series

theory *Series*
imports *SEQ*
begin

definition
  sums :: (*nat* ⇒ *'a*::*real-normed-vector*) ⇒ *'a* ⇒ *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 = (∃ *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* (∑ -. - [*0*, *10*] *10*)
translations
  ∑ *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* ⟹ *f*(*p*) ≤ *K*)
     ⟹ *setsum f* {*0*..<*n*::*nat*} ≤ *real n* ∗ *K*
⟨*proof*⟩

lemma *sumr-minus-one-realpow-zero* [*simp*]:
  (∑ *i*=*0*..<*2*∗*n*. (−*1*) ^ *Suc i*) = (*0*::*real*)
⟨*proof*⟩

lemma *sumr-one-lb-realpow-zero* [*simp*]:
  (∑ *n*=*Suc 0*..<*n*. *f*(*n*) ∗ (*0*::*real*) ^ *n*) = *0*
⟨*proof*⟩

lemma *sumr-group*:
    (∑ *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} = (∑ m=0..<n. f (m+k)) + setsum f {0..<k}
⟨*proof*⟩

lemma *sumr-offset*:
fixes *f :: nat ⇒ 'a::ab-group-add*
shows *(∑ m=0..<n. f(m+k)) = setsum f {0..<n+k} − setsum f {0..<k}*
⟨*proof*⟩

lemma *sumr-offset2*:
∀ f. (∑ m=0..<n::nat. f(m+k)::real) = setsum f {0..<n+k} − setsum f {0..<k}
⟨*proof*⟩

lemma *sumr-offset4*:
∀ n f. setsum f {0::nat..<n+k} = (∑ m=0..<n. f (m+k)::real) + setsum f {0..<k}
⟨*proof*⟩

18.1 Infinite Sums, by the Properties of Limits

lemma *sums-summable*: *f sums l ==> summable f*
⟨*proof*⟩

lemma *summable-sums*: *summable f ==> f sums (suminf f)*
⟨*proof*⟩

lemma *summable-sumr-LIMSEQ-suminf*:
summable f ==> (%n. setsum f {0..<n}) ----> (suminf f)
⟨*proof*⟩

lemma *sums-unique*: *f sums s ==> (s = suminf f)*
⟨*proof*⟩

lemma *sums-split-initial-segment*: *f sums s ==>*
(%n. f(n + k)) sums (s − (SUM i = 0..< k. f i))
⟨*proof*⟩

lemma *summable-ignore-initial-segment*: *summable f ==>*
summable (%n. f(n + k))
⟨*proof*⟩

lemma *suminf-minus-initial-segment*: *summable f ==>*
suminf f = s ==> suminf (%n. f(n + k)) = s − (SUM i = 0..< k. f i)
⟨*proof*⟩

lemma *suminf-split-initial-segment*: *summable f ==>*
suminf f = (SUM i = 0..< k. f i) + suminf (%n. f(n + k))
⟨*proof*⟩

lemma *series-zero*:
(∀ *m*. *n* ≤ *m* −−> *f*(*m*) = *0*) ==> *f sums* (*setsum f* {*0*..<*n*})
⟨*proof*⟩

lemma *sums-zero*: (λ*n*. *0*) *sums 0*
⟨*proof*⟩

lemma *summable-zero*: *summable* (λ*n*. *0*)
⟨*proof*⟩

lemma *suminf-zero*: *suminf* (λ*n*. *0*) = *0*
⟨*proof*⟩

lemma (**in** *bounded-linear*) *sums*:
(λ*n*. *X n*) *sums a* ⟹ (λ*n*. *f* (*X n*)) *sums* (*f a*)
⟨*proof*⟩

lemma (**in** *bounded-linear*) *summable*:
summable (λ*n*. *X n*) ⟹ *summable* (λ*n*. *f* (*X n*))
⟨*proof*⟩

lemma (**in** *bounded-linear*) *suminf*:
summable (λ*n*. *X n*) ⟹ *f* (∑ *n*. *X n*) = (∑ *n*. *f* (*X n*))
⟨*proof*⟩

lemma *sums-mult*:
fixes *c* :: *'a*::*real-normed-algebra*
shows *f sums a* ⟹ (λ*n*. *c* ∗ *f n*) *sums* (*c* ∗ *a*)
⟨*proof*⟩

lemma *summable-mult*:
fixes *c* :: *'a*::*real-normed-algebra*
shows *summable f* ⟹ *summable* (%*n*. *c* ∗ *f n*)
⟨*proof*⟩

lemma *suminf-mult*:
fixes *c* :: *'a*::*real-normed-algebra*
shows *summable f* ⟹ *suminf* (λ*n*. *c* ∗ *f n*) = *c* ∗ *suminf f*
⟨*proof*⟩

lemma *sums-mult2*:
fixes *c* :: *'a*::*real-normed-algebra*
shows *f sums a* ⟹ (λ*n*. *f n* ∗ *c*) *sums* (*a* ∗ *c*)
⟨*proof*⟩

lemma *summable-mult2*:
fixes *c* :: *'a*::*real-normed-algebra*
shows *summable f* ⟹ *summable* (λ*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* ⇒ *real*
shows ⟦*summable f*; ∀ *m*≥*n*. *0* ≤ *f m*⟧ ⟹ *setsum f* {*0*..<*n*} ≤ *suminf f*
⟨*proof*⟩

lemma *series-pos-less*:
fixes *f* :: *nat* ⇒ *real*
shows ⟦*summable f*; ∀ *m*≥*n*. *0* < *f m*⟧ ⟹ *setsum f* {*0*..<*n*} < *suminf f*
⟨*proof*⟩

lemma *suminf-gt-zero*:
fixes *f* :: *nat* ⇒ *real*
shows ⟦*summable f*; ∀ *n*. *0* < *f n*⟧ ⟹ *0* < *suminf f*
⟨*proof*⟩

lemma *suminf-ge-zero*:
fixes *f* :: *nat* ⇒ *real*
shows ⟦*summable f*; ∀ *n*. *0* ≤ *f n*⟧ ⟹ *0* ≤ *suminf f*
⟨*proof*⟩

lemma *sumr-pos-lt-pair*:
fixes *f* :: *nat* ⇒ *real*
shows ⟦*summable f*;
∀ *d*. *0* < *f* (*k* + (*Suc*(*Suc 0*) ∗ *d*)) + *f* (*k* + ((*Suc*(*Suc 0*) ∗ *d*) + *1*))⟧
⟹ *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-normed-field*,*recpower*}
shows *norm x* < *1* ⟹ (λ*n*. *x* ^ *n*) *sums* (*1* / (*1* − *x*))
⟨*proof*⟩

lemma *summable-geometric*:
fixes *x* :: *'a*::{*real-normed-field*,*recpower*}
shows *norm x* < *1* ⟹ *summable* (λ*n*. *x* ^ *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 ⟹ f ----> 0*
⟨*proof*⟩

lemma *summable-Cauchy*:
summable (f::nat ⇒ 'a::banach) =
(∀ e > 0. ∃ N. ∀ m ≥ N. ∀ n. norm (setsum f {m..<n}) < e)
⟨*proof*⟩

Comparison test

lemma *norm-setsum*:
fixes *f* :: *'a ⇒ 'b::real-normed-vector*
shows *norm (setsum f A) ≤ (∑ i∈A. norm (f i))*
⟨*proof*⟩

lemma *summable-comparison-test*:
fixes *f* :: *nat ⇒ 'a::banach*
shows *⟦∃ N. ∀ n≥N. norm (f n) ≤ g n; summable g⟧ ⟹ summable f*
⟨*proof*⟩

lemma *summable-norm-comparison-test*:
fixes *f* :: *nat ⇒ 'a::banach*
shows *⟦∃ N. ∀ n≥N. norm (f n) ≤ g n; summable g⟧*
⟹ summable (λn. norm (f n))
⟨*proof*⟩

lemma *summable-rabs-comparison-test*:
fixes *f* :: *nat ⇒ real*
shows *⟦∃ N. ∀ n≥N. |f n| ≤ g n; summable g⟧ ⟹ summable (λn. |f n|)*
⟨*proof*⟩

Summability of geometric series for real algebras

lemma *complete-algebra-summable-geometric*:
fixes *x* :: *'a::{real-normed-algebra-1,banach,recpower}*
shows *norm x < 1 ⟹ summable (λn. x ^ n)*
⟨*proof*⟩

Limit comparison property for series (c.f. jrh)

lemma *summable-le*:
fixes *f g* :: *nat ⇒ real*
shows *⟦∀ n. f n ≤ g n; summable f; summable g⟧ ⟹ suminf f ≤ suminf g*
⟨*proof*⟩

lemma *summable-le2*:

fixes $f\ g :: nat \Rightarrow real$
shows $[\![\forall n.\ |f\ n| \leq g\ n;\ summable\ g]\!] \Longrightarrow summable\ f \wedge suminf\ f \leq suminf\ g$
⟨*proof*⟩

lemma *suminf-0-le*:
fixes f::$nat \Rightarrow real$
assumes *gt0*: $\forall n.\ 0 \leq f\ n$ **and** *sm*: $summable\ f$
shows $0 \leq suminf\ f$
⟨*proof*⟩

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*⟩

18.2 The Ratio Test

lemma *norm-ratiotest-lemma*:
fixes $x\ y :: 'a$::*real-normed-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 $[\![\forall n \geq N.\ norm\ (f\ (Suc\ n)) \leq c * norm\ (f\ n)]\!] \Longrightarrow 0 < c \vee summable\ f$
⟨*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*⟩

18.3 Cauchy Product Formula

lemma *setsum-triangle-reindex*:
fixes $n :: nat$
shows $(\sum (i,j) \in \{(i,j).\ i+j < n\}.\ f\ i\ j) = (\sum k=0..<n.\ \sum i=0..k.\ f\ i\ (k - i))$
⟨*proof*⟩

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 $(\lambda k.\ \sum i=0..k.\ a\ i * b\ (k - i))\ sums\ ((\sum k.\ a\ k) * (\sum k.\ b\ k))$
⟨*proof*⟩

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 $(\sum k.\ a\ k) * (\sum k.\ b\ k) = (\sum k.\ \sum i=0..k.\ a\ i * b\ (k - i))$
⟨*proof*⟩

end

19 Lim: Limits and Continuity

theory *Lim*
imports *SEQ*
begin

Standard Definitions

definition
$LIM :: ['a::real\text{-}normed\text{-}vector \Rightarrow 'b::real\text{-}normed\text{-}vector, 'a, 'b] \Rightarrow bool$
(((-)/ –– (-)/ –––> (-)) [60, 0, 60] 60) **where**
f –– a –––> L =
$(\forall r > 0.\ \exists s > 0.\ \forall x.\ x \neq a\ \&\ norm\ (x - a) < s$
–––> $norm\ (f\ x - L) < r)$

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*)

19.1 Limits of Functions

19.1.1 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* ⟹ (λ*x*. *f* (*x* + *k*)) –– *a* − *k* –––> *L*
⟨*proof*⟩

lemma *LIM-offset-zero*: *f* –– *a* –––> *L* ⟹ (λ*h*. *f* (*a* + *h*)) –– *0* –––> *L*
⟨*proof*⟩

lemma *LIM-offset-zero-cancel*: (λ*h*. *f* (*a* + *h*)) –– *0* –––> *L* ⟹ *f* –– *a* –––> *L*
⟨*proof*⟩

lemma *LIM-const* [*simp*]: (%*x*. *k*) –– *x* –––> *k*
⟨*proof*⟩

lemma *LIM-add*:
fixes *f g* :: '*a*::*real-normed-vector* ⇒ '*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*⟧ ⟹ (λ*x*. *f x* + *g x*) –– *a* –––> *0*
⟨*proof*⟩

lemma *minus-diff-minus*:
fixes $a\ b :: 'a::ab\text{-}group\text{-}add$
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*: $f \dashrightarrow a \longrightarrow (0::real) \Longrightarrow (\lambda x.\ |f\ x|) \dashrightarrow a \longrightarrow 0$
⟨*proof*⟩

lemma *LIM-rabs-zero-cancel*: $(\lambda x.\ |f\ x|) \dashrightarrow a \longrightarrow (0::real) \Longrightarrow f \dashrightarrow a \longrightarrow 0$
⟨*proof*⟩

lemma *LIM-rabs-zero-iff*: $(\lambda x.\ |f\ x|) \dashrightarrow a \longrightarrow (0::real) = f \dashrightarrow a \longrightarrow 0$
⟨*proof*⟩

lemma *LIM-const-not-eq*:
fixes $a :: 'a::real\text{-}normed\text{-}algebra\text{-}1$
shows $k \neq L \Longrightarrow \neg\ (\lambda x.\ k) \dashrightarrow a \longrightarrow L$
⟨*proof*⟩

lemmas *LIM-not-zero* = *LIM-const-not-eq* [**where** $L = 0$]

lemma *LIM-const-eq*:
fixes $a :: 'a::real\text{-}normed\text{-}algebra\text{-}1$
shows $(\lambda x.\ k) \dashrightarrow a \longrightarrow L \Longrightarrow k = L$
⟨*proof*⟩

lemma *LIM-unique*:
fixes $a :: 'a::real\text{-}normed\text{-}algebra\text{-}1$
shows $[\![f \dashrightarrow a \longrightarrow L; f \dashrightarrow a \longrightarrow M]\!] \Longrightarrow L = M$
⟨*proof*⟩

lemma *LIM-ident* [*simp*]: $(\lambda x.\ x) \dashrightarrow a \longrightarrow a$
⟨*proof*⟩

Limits are equal for functions equal except at limit point

lemma *LIM-equal*:
$[|\ \forall x.\ x \neq a \longrightarrow (f\ x = g\ x)\ |] ==> (f \dashrightarrow a \longrightarrow l) = (g \dashrightarrow a \longrightarrow l)$
⟨*proof*⟩

lemma *LIM-cong*:
$[\![a = b; \bigwedge x.\ x \neq b \Longrightarrow f\ x = g\ x; l = m]\!]$
$\Longrightarrow ((\lambda x.\ f\ x) \dashrightarrow a \longrightarrow l) = ((\lambda x.\ g\ x) \dashrightarrow b \longrightarrow m)$
⟨*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 $g \dashrightarrow a \longrightarrow l \Longrightarrow f \dashrightarrow a \longrightarrow l$
⟨*proof*⟩

Two uses in Hyperreal/Transcendental.ML

lemma *LIM-trans*:
$[|\ (\%x.\ f(x) + -g(x)) \dashrightarrow a \longrightarrow 0;\ \ g \dashrightarrow a \longrightarrow l\ |] ==> f \dashrightarrow a \longrightarrow l$

⟨*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* ⇒ *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*:
$f \mathrel{--} a \mathrel{-->} 0 \Longrightarrow (\lambda x.\ c ** f\ x) \mathrel{--} a \mathrel{-->} 0$
⟨*proof*⟩

lemma (**in** *bounded-bilinear*) *LIM*:
$[\![f \mathrel{--} a \mathrel{-->} L;\ g \mathrel{--} a \mathrel{-->} M]\!] \Longrightarrow (\lambda x.\ f\ x ** g\ x) \mathrel{--} a \mathrel{-->} L ** M$
⟨*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*: $f \mathrel{--} a \mathrel{-->} l$
shows $(\lambda x.\ f\ x\ \hat{}\ n) \mathrel{--} a \mathrel{-->} l\ \hat{}\ n$
⟨*proof*⟩

19.1.2 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 $inverse \mathrel{--} a \mathrel{-->} inverse\ a$
⟨*proof*⟩

lemma *LIM-inverse*:
fixes $L :: 'a$::*real-normed-div-algebra*
shows $[\![f \mathrel{--} a \mathrel{-->} L;\ L \neq 0]\!] \Longrightarrow (\lambda x.\ inverse\ (f\ x)) \mathrel{--} a \mathrel{-->} inverse\ L$
⟨*proof*⟩

19.2 Continuity

19.2.1 Purely standard proofs

lemma *LIM-isCont-iff*: $(f \mathrel{--} a \mathrel{-->} f\ a) = ((\lambda h.\ f\ (a + h)) \mathrel{--} 0 \mathrel{-->} f\ a)$

⟨*proof*⟩

lemma *isCont-iff*: *isCont f x* = (λ*h*. *f* (*x* + *h*)) −− *0* −−> *f x*
⟨*proof*⟩

lemma *isCont-ident* [*simp*]: *isCont* (λ*x*. *x*) *a*
⟨*proof*⟩

lemma *isCont-const* [*simp*]: *isCont* (λ*x*. *k*) *a*
⟨*proof*⟩

lemma *isCont-norm*: *isCont f a* ⟹ *isCont* (λ*x*. *norm* (*f x*)) *a*
⟨*proof*⟩

lemma *isCont-rabs*: *isCont f a* ⟹ *isCont* (λ*x*. |*f x* :: *real*|) *a*
⟨*proof*⟩

lemma *isCont-add*: ⟦*isCont f a*; *isCont g a*⟧ ⟹ *isCont* (λ*x*. *f x* + *g x*) *a*
⟨*proof*⟩

lemma *isCont-minus*: *isCont f a* ⟹ *isCont* (λ*x*. − *f x*) *a*
⟨*proof*⟩

lemma *isCont-diff*: ⟦*isCont f a*; *isCont g a*⟧ ⟹ *isCont* (λ*x*. *f x* − *g x*) *a*
⟨*proof*⟩

lemma *isCont-mult*:
fixes *f g* :: *'a*::*real-normed-vector* ⇒ *'b*::*real-normed-algebra*
shows ⟦*isCont f a*; *isCont g a*⟧ ⟹ *isCont* (λ*x*. *f x* ∗ *g x*) *a*
⟨*proof*⟩

lemma *isCont-inverse*:
fixes *f* :: *'a*::*real-normed-vector* ⇒ *'b*::*real-normed-div-algebra*
shows ⟦*isCont f a*; *f a* ≠ *0*⟧ ⟹ *isCont* (λ*x*. *inverse* (*f x*)) *a*
⟨*proof*⟩

lemma *isCont-LIM-compose*:
⟦*isCont g l*; *f* −− *a* −−> *l*⟧ ⟹ (λ*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*: ∃ *d*>*0*. ∀ *x*. *x* ≠ *a* ∧ *norm* (*x* − *a*) < *d* ⟶ *f x* ≠ *f a*
shows (λ*x*. *g* (*f x*)) −− *a* −−> *l*
⟨*proof*⟩

lemma *isCont-o2*: ⟦*isCont f a*; *isCont g* (*f a*)⟧ ⟹ *isCont* (λ*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* ⇒ *'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*⟩

19.3 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*⟩

19.4 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

20 Deriv: Differentiation

theory *Deriv*
imports *Lim Univ-Poly*
begin

Standard Definitions

definition
deriv :: $['a::real\text{-}normed\text{-}field \Rightarrow 'a, 'a, 'a] \Rightarrow bool$
— 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*))

20.1 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-group-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*:
$[\![DERIV\ f\ x :> D;\ DERIV\ g\ x :> E]\!] \Longrightarrow DERIV\ (\lambda x.\ f\ x + -\ g\ x)\ x :> D + -\ E$
⟨*proof*⟩

lemma *DERIV-isCont*: $DERIV\ f\ x :> D \Longrightarrow isCont\ f\ x$
⟨*proof*⟩

lemma *DERIV-mult-lemma*:
fixes $a\ b\ c\ d :: 'a$::*real-field*
shows $(a * b - c * d)\ /\ h = a * ((b - d)\ /\ h) + ((a - c)\ /\ h) * d$
⟨*proof*⟩

lemma *DERIV-mult'*:
assumes *f*: $DERIV\ f\ x :> D$
assumes *g*: $DERIV\ g\ x :> E$
shows $DERIV\ (\lambda x.\ f\ x * g\ x)\ x :> f\ x * E + D * g\ x$
⟨*proof*⟩

lemma *DERIV-mult*:
$[|\ DERIV\ f\ x :> Da;\ DERIV\ g\ x :> Db\ |]$
$\Longrightarrow DERIV\ (\%x.\ f\ x * g\ x)\ x :> (Da * g(x)) + (Db * f(x))$
⟨*proof*⟩

lemma *DERIV-unique*:

[| *DERIV f x :> D*; *DERIV f x :> E* |] ==> *D = E*
⟨*proof*⟩

Differentiation of finite sum

lemma *DERIV-sumr* [*rule-format* (*no-asm*)]:
(∀ *r*. *m* ≤ *r* & *r* < (*m* + *n*) --> *DERIV* (%*x*. *f r x*) *x* :> (*f' r x*))
--> *DERIV* (%*x*. ∑ *n*=*m*..<*n*::*nat*. *f n x* :: *real*) *x* :> (∑ *r*=*m*..<*n*. *f' r x*)
⟨*proof*⟩

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*)
⟨*proof*⟩

lemma *DERIV-iff2*: (*DERIV f x :> D*) = ((%*z*. (*f*(*z*) − *f*(*x*)) / (*z*−*x*)) -- *x* --> *D*)
⟨*proof*⟩

lemma *inverse-diff-inverse*:
⟦(*a*::'*a*::*division-ring*) ≠ *0*; *b* ≠ *0*⟧
⟹ *inverse a* − *inverse b* = − (*inverse a* ∗ (*a* − *b*) ∗ *inverse b*)
⟨*proof*⟩

lemma *DERIV-inverse-lemma*:
⟦*a* ≠ *0*; *b* ≠ (*0*::'*a*::*real-normed-field*)⟧
⟹ (*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* ≠ *0*
shows *DERIV* (λ*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* ≠ *0*⟧
⟹ *DERIV* (λ*x*. *f x* / *g x*) *x* :> (*D* ∗ *g x* − *f x* ∗ *E*) / (*g x* ∗ *g x*)
⟨*proof*⟩

lemma *DERIV-power-Suc*:
fixes *f* :: '*a* ⇒ '*a*::{*real-normed-field*,*recpower*}
assumes *f*: *DERIV f x :> D*
shows *DERIV* (λ*x*. *f x* ^ *Suc n*) *x* :> (*1* + *of-nat n*) ∗ (*D* ∗ *f x* ^ *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*⟩

20.2 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*⟩

20.3 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*)]: $(\forall n.\ (f{::}nat{=>}real)\ n \leq f\ (Suc\ n)) \dashrightarrow f\ m \leq f(m + no)$
⟨*proof*⟩

lemma *f-inc-g-dec-Beq-f*: $[|\ \forall n.\ f(n) \leq f(Suc\ n);$
$\forall n.\ g(Suc\ n) \leq g(n);$
$\forall n.\ f(n) \leq g(n)\ |] ==> Bseq\ (f :: nat \Rightarrow real)$
⟨*proof*⟩

lemma *f-inc-g-dec-Beq-g*: $[|\ \forall n.\ f(n) \leq f(Suc\ n);$
$\forall n.\ g(Suc\ n) \leq g(n);$
$\forall n.\ f(n) \leq g(n)\ |] ==> Bseq\ (g :: nat \Rightarrow real)$
⟨*proof*⟩

lemma *f-inc-imp-le-lim*:
fixes $f :: nat \Rightarrow real$
shows $[\![\forall n.\ f\ n \leq f\ (Suc\ n);\ convergent\ f]\!] \Longrightarrow f\ n \leq lim\ f$
⟨*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*: $[|\ \forall n.\ f(n) \leq f(Suc\ n);$
$\forall n.\ g(Suc\ n) \leq g(n);$
$\forall n.\ f(n) \leq g(n)\ |] ==> \exists l\ m :: real.\ l \leq m\ \&\ ((\forall n.\ f(n) \leq l)\ \&\ f \dashrightarrow l)\ \&$
$((\forall n.\ m \leq g(n))\ \&\ g \dashrightarrow m)$
⟨*proof*⟩

lemma *lemma-nest-unique*: $[|\ \forall n.\ f(n) \leq f(Suc\ n);$
$\forall n.\ g(Suc\ n) \leq g(n);$
$\forall n.\ f(n) \leq g(n);$
$(\%n.\ f(n) - g(n)) \dashrightarrow 0\ |] ==> \exists l{::}real.\ ((\forall n.\ f(n) \leq l)\ \&\ f \dashrightarrow l)\ \&$
$((\forall n.\ l \leq g(n))\ \&\ g \dashrightarrow l)$
⟨*proof*⟩

The universal quantifiers below are required for the declaration of *Bolzano-nest-unique*

below.

lemma *Bolzano-bisect-le*:
$a \leq b \Longrightarrow \forall n.\ fst\ (Bolzano\text{-}bisect\ P\ a\ b\ n) \leq snd\ (Bolzano\text{-}bisect\ P\ a\ b\ n)$
⟨*proof*⟩

lemma *Bolzano-bisect-fst-le-Suc*: $a \leq b \Longrightarrow$
$\forall n.\ fst(Bolzano\text{-}bisect\ P\ a\ b\ n) \leq fst\ (Bolzano\text{-}bisect\ P\ a\ b\ (Suc\ n))$
⟨*proof*⟩

lemma *Bolzano-bisect-Suc-le-snd*: $a \leq b \Longrightarrow$
$\forall n.\ snd(Bolzano\text{-}bisect\ P\ a\ b\ (Suc\ n)) \leq snd\ (Bolzano\text{-}bisect\ P\ a\ b\ n)$
⟨*proof*⟩

lemma *eq-divide-2-times-iff*: $((x::real) = y\ /\ (2 * z)) = (2 * x = y/z)$
⟨*proof*⟩

lemma *Bolzano-bisect-diff*:
$a \leq b \Longrightarrow$
$snd(Bolzano\text{-}bisect\ P\ a\ b\ n) - fst(Bolzano\text{-}bisect\ P\ a\ b\ n) =$
$(b-a)\ /\ (2\ \hat{}\ n)$
⟨*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*: $!!a\ b\ c.\ [|\ P(a,b);\ P(b,c);\ a \leq b;\ b \leq c|] \Longrightarrow P(a,c)$
and *notP*: $\sim P(a,b)$
and *le*: $a \leq b$
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'*:
$[|\ \forall a\ b\ c.\ P(a,b)\ \&\ P(b,c)\ \&\ a \leq b\ \&\ b \leq c \longrightarrow P(a,c);$
$\sim P(a,b);\ a \leq b\ |] \Longrightarrow$
$\forall n.\ \sim P(fst(Bolzano\text{-}bisect\ P\ a\ b\ n),\ snd(Bolzano\text{-}bisect\ P\ a\ b\ n))$
⟨*proof*⟩

lemma *lemma-BOLZANO*:
$[|\ \forall a\ b\ c.\ P(a,b)\ \&\ P(b,c)\ \&\ a \leq b\ \&\ b \leq c \longrightarrow P(a,c);$
$\forall x.\ \exists d::real.\ 0 < d\ \&$
$(\forall a\ b.\ a \leq x\ \&\ x \leq b\ \&\ (b-a) < d \longrightarrow P(a,b));$
$a \leq b\ |]$
$\Longrightarrow P(a,b)$

⟨*proof*⟩

lemma *lemma-BOLZANO2*: $((\forall\, a\ b\ c.\ (a \le b\ \&\ b \le c\ \&\ P(a,b)\ \&\ P(b,c)) \longrightarrow P(a,c))\ \&$
$(\forall\, x.\ \exists\, d{::}real.\ 0 < d\ \&$
$(\forall\, a\ b.\ a \le x\ \&\ x \le b\ \&\ (b-a) < d \longrightarrow P(a,b))))$
$\longrightarrow (\forall\, a\ b.\ a \le b \longrightarrow P(a,b))$
⟨*proof*⟩

20.4 Intermediate Value Theorem

Prove Contrapositive by Bisection

lemma *IVT*: $[|\ f(a{::}real) \le (y{::}real);\ y \le f(b);$
$a \le b;$
$(\forall\, x.\ a \le x\ \&\ x \le b \longrightarrow isCont\ f\ x)\ |]$
$\Longrightarrow \exists\, x.\ a \le x\ \&\ x \le b\ \&\ f(x) = y$
⟨*proof*⟩

lemma *IVT2*: $[|\ f(b{::}real) \le (y{::}real);\ y \le f(a);$
$a \le b;$
$(\forall\, x.\ a \le x\ \&\ x \le b \longrightarrow isCont\ f\ x)$
$|] \Longrightarrow \exists\, x.\ a \le x\ \&\ x \le b\ \&\ f(x) = y$
⟨*proof*⟩

lemma *IVT-objl*: $(f(a{::}real) \le (y{::}real)\ \&\ y \le f(b)\ \&\ a \le b\ \&$
$(\forall\, x.\ a \le x\ \&\ x \le b \longrightarrow isCont\ f\ x))$
$\longrightarrow (\exists\, x.\ a \le x\ \&\ x \le b\ \&\ f(x) = y)$
⟨*proof*⟩

lemma *IVT2-objl*: $(f(b{::}real) \le (y{::}real)\ \&\ y \le f(a)\ \&\ a \le b\ \&$
$(\forall\, x.\ a \le x\ \&\ x \le b \longrightarrow isCont\ f\ x))$
$\longrightarrow (\exists\, x.\ a \le x\ \&\ x \le b\ \&\ f(x) = y)$
⟨*proof*⟩

By bisection, function continuous on closed interval is bounded above

lemma *isCont-bounded*:
$[|\ a \le b;\ \forall\, x.\ a \le x\ \&\ x \le b \longrightarrow isCont\ f\ x\ |]$
$\Longrightarrow \exists\, M{::}real.\ \forall\, x{::}real.\ a \le x\ \&\ x \le b \longrightarrow f(x) \le M$
⟨*proof*⟩

Refine the above to existence of least upper bound

lemma *lemma-reals-complete*: $((\exists\, x.\ x \in S)\ \&\ (\exists\, y.\ isUb\ UNIV\ S\ (y{::}real))) \longrightarrow$
$(\exists\, t.\ isLub\ UNIV\ S\ t)$
⟨*proof*⟩

lemma *isCont-has-Ub*: $[|\ a \le b;\ \forall\, x.\ a \le x\ \&\ x \le b \longrightarrow isCont\ f\ x\ |]$

$$\Longrightarrow \exists\, M{::}real.\ (\forall\, x{::}real.\ a \le x\ \&\ x \le b \longrightarrow f(x) \le M)\ \&\ (\forall\, N.\ N < M \longrightarrow (\exists\, x.\ a \le x\ \&\ x \le b\ \&\ N < f(x)))$$

⟨*proof*⟩

Now show that it attains its upper bound

lemma *isCont-eq-Ub*:
assumes *le*: $a \le b$
and *con*: $\forall\, x{::}real.\ a \le x\ \&\ x \le b \longrightarrow isCont\ f\ x$
shows $\exists\, M{::}real.\ (\forall\, x.\ a \le x\ \&\ x \le b \longrightarrow f(x) \le M)\ \&\ (\exists\, x.\ a \le x\ \&\ x \le b\ \&\ f(x) = M)$
⟨*proof*⟩

Same theorem for lower bound

lemma *isCont-eq-Lb*: $[|\ a \le b;\ \forall\, x.\ a \le x\ \&\ x \le b \longrightarrow isCont\ f\ x\ |]$
$\Longrightarrow \exists\, M{::}real.\ (\forall\, x{::}real.\ a \le x\ \&\ x \le b \longrightarrow M \le f(x))\ \&\ (\exists\, x.\ a \le x\ \&\ x \le b\ \&\ f(x) = M)$
⟨*proof*⟩

Another version.

lemma *isCont-Lb-Ub*: $[|a \le b;\ \forall\, x.\ a \le x\ \&\ x \le b \longrightarrow isCont\ f\ x\ |]$
$\Longrightarrow \exists\, L\ M{::}real.\ (\forall\, x{::}real.\ a \le x\ \&\ x \le b \longrightarrow L \le f(x)\ \&\ f(x) \le M)\ \&\ (\forall\, y.\ L \le y\ \&\ y \le M \longrightarrow (\exists\, x.\ a \le x\ \&\ x \le b\ \&\ (f(x) = y)))$
⟨*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 $\exists\, d > 0.\ \forall\, h > 0.\ h < d \longrightarrow f(x) < f(x + h)$
⟨*proof*⟩

lemma *DERIV-left-dec*:
fixes f :: *real* => *real*
assumes *der*: $DERIV\ f\ x :> l$
and *l*: $l < 0$
shows $\exists\, d > 0.\ \forall\, h > 0.\ h < d \longrightarrow f(x) < f(x-h)$
⟨*proof*⟩

lemma *DERIV-local-max*:
fixes f :: *real* => *real*
assumes *der*: $DERIV\ f\ x :> l$
and *d*: $0 < d$
and *le*: $\forall\, y.\ |x-y| < d \longrightarrow f(y) \le f(x)$
shows $l = 0$
⟨*proof*⟩

Similar theorem for a local minimum

lemma *DERIV-local-min*:
fixes f :: *real* => *real*
shows [| *DERIV* $f\ x$:> l; $0 < d$; $\forall y.\ |x-y| < d$ --> $f(x) \leq f(y)$ |] ==> $l = 0$
⟨*proof*⟩

In particular, if a function is locally flat

lemma *DERIV-local-const*:
fixes f :: *real* => *real*
shows [| *DERIV* $f\ x$:> l; $0 < d$; $\forall y.\ |x-y| < d$ --> $f(x) = f(y)$ |] ==> $l = 0$
⟨*proof*⟩

Lemma about introducing open ball in open interval

lemma *lemma-interval-lt*:
$[|\ a < x;\ \ x < b\ |]$
$==> \exists d::real.\ 0 < d\ \&\ (\forall y.\ |x-y| < d$ --> $a < y\ \&\ y < b)$
⟨*proof*⟩

lemma *lemma-interval*: $[|\ a < x;\ \ x < b\ |] ==>$
$\exists d::real.\ 0 < d\ \&\ \ (\forall y.\ |x-y| < d$ --> $a \leq y\ \&\ y \leq b)$
⟨*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*: $\forall x.\ a \leq x\ \&\ x \leq b$ --> *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*⟩

20.5 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*: $\forall x.\ a \leq x\ \&\ x \leq b$ --> *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 [| $a < b$;
$\forall x.\ a \leq x\ \&\ x \leq b$ --> *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 [| $a < b$;
$\forall x.\ a \leq x\ \&\ x \leq b$ --> *isCont* $f\ x$;
$\forall x.\ a < x\ \&\ x < b$ --> *DERIV* $f\ x$:> 0 |]
==> $\forall x.\ a \leq x\ \&\ x \leq b$ --> $f\ x = f\ a$
⟨*proof*⟩

lemma *DERIV-isconst2*:
fixes f :: *real* => *real*
shows [| $a < b$;
$\forall x.\ a \leq x\ \&\ x \leq b$ --> *isCont* $f\ x$;
$\forall x.\ a < x\ \&\ x < b$ --> *DERIV* $f\ x$:> 0;
$a \leq x$; $x \leq b$ |]
==> $f\ x = f\ a$
⟨*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 $f :: real \Rightarrow real$
assumes *d*: $0 < d$
and *inj* [*rule-format*]: $\forall z.\ |z-x| \leq d \longrightarrow g(f\ z) = z$
and *cont*: $\forall z.\ |z-x| \leq d \longrightarrow isCont\ f\ z$
shows $\exists z.\ |z-x| \leq d\ \&\ f\ x < f\ z$
⟨*proof*⟩

Similar version for lower bound.

lemma *lemma-isCont-inj2*:
fixes $f\ g :: real \Rightarrow real$
shows $[|0 < d; \forall z.\ |z-x| \leq d \longrightarrow g(f\ z) = z;$
$\forall z.\ |z-x| \leq d \longrightarrow isCont\ f\ z\ |]$
$\Longrightarrow \exists z.\ |z-x| \leq d\ \&\ f\ z < f\ x$
⟨*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*: $\forall z.\ |z-x| \leq d \longrightarrow g(f\ z) = z$
and *cont*: $\forall z.\ |z-x| \leq d \longrightarrow isCont\ f\ z$
shows $\exists e{>}0.\ \forall y.\ |y - f\ x| \leq e \longrightarrow (\exists z.\ |z-x| \leq d\ \&\ f\ z = y)$
⟨*proof*⟩

Continuity of inverse function

lemma *isCont-inverse-function*:
fixes $f\ g :: real \Rightarrow real$
assumes *d*: $0 < d$
and *inj*: $\forall z.\ |z-x| \leq d \longrightarrow g(f\ z) = z$
and *cont*: $\forall z.\ |z-x| \leq d \longrightarrow isCont\ f\ z$
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*⟩

20.6 Derivatives of univariate polynomials

primrec *pderiv-aux* :: *nat => real list => real list* **where**
  pderiv-aux-Nil: *pderiv-aux n* [] = []
| *pderiv-aux-Cons*: *pderiv-aux n* (*h*#*t*) =
    (*real n* * *h*)#(*pderiv-aux* (*Suc n*) *t*)

definition
  pderiv :: *real list => real list* **where**
  pderiv p = (*if p* = [] *then* [] *else pderiv-aux 1* (*tl p*))

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* ^ *Suc n*) *x :> real* (*Suc n*) * (*x* ^ *n*)
⟨*proof*⟩

declare *DERIV-pow2* [*simp*] *DERIV-pow* [*simp*]

lemma *lemma-DERIV-poly1*: $\forall n.\ DERIV\ (\%x.\ (x\ \hat{}\ (Suc\ n) * poly\ p\ x))\ x :>$
$x\ \hat{}\ n * poly\ (pderiv\text{-}aux\ (Suc\ n)\ p)\ x$
⟨*proof*⟩

lemma *lemma-DERIV-poly*: $DERIV\ (\%x.\ (x\ \hat{}\ (Suc\ n) * poly\ p\ x))\ x :>$
$x\ \hat{}\ n * poly\ (pderiv\text{-}aux\ (Suc\ n)\ p)\ x$
⟨*proof*⟩

lemma *DERIV-add-const*: $DERIV\ f\ x :> D ==>\ DERIV\ (\%x.\ a + f\ x :: real)\ x :> D$
⟨*proof*⟩

lemma *poly-DERIV*[*simp*]: $DERIV\ (\%x.\ poly\ p\ x)\ x :> poly\ (pderiv\ p)\ x$
⟨*proof*⟩

Consequences of the derivative theorem above

lemma *poly-differentiable*[*simp*]: $(\%x.\ poly\ p\ x)\ differentiable\ (x::real)$
⟨*proof*⟩

lemma *poly-isCont*[*simp*]: $isCont\ (\%x.\ poly\ p\ x)\ (x::real)$
⟨*proof*⟩

lemma *poly-IVT-pos*: $[|\ a < b;\ poly\ p\ (a::real) < 0;\ 0 < poly\ p\ b\ |]$
$==> \exists x.\ a < x\ \&\ x < b\ \&\ (poly\ p\ x = 0)$
⟨*proof*⟩

lemma *poly-IVT-neg*: $[|\ (a::real) < b;\ 0 < poly\ p\ a;\ poly\ p\ b < 0\ |]$
$==> \exists x.\ a < x\ \&\ x < b\ \&\ (poly\ p\ x = 0)$
⟨*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-aux n* (*c* %∗ *p*)) *x* = *poly* (*c* %∗ *pderiv-aux n p*) *x*
⟨*proof*⟩

lemma *poly-pderiv-aux-minus*:
poly (*pderiv-aux n* (−− *p*)) *x* = *poly* (−− *pderiv-aux n p*) *x*
⟨*proof*⟩

lemma *lemma-poly-pderiv-aux-mult1*: ∀ *n*. *poly* (*pderiv-aux* (*Suc n*) *p*) *x* = *poly* ((*pderiv-aux n p*) +++ *p*) *x*
⟨*proof*⟩

lemma *lemma-poly-pderiv-aux-mult*: *poly* (*pderiv-aux* (*Suc n*) *p*) *x* = *poly* ((*pderiv-aux n p*) +++ *p*) *x*
⟨*proof*⟩

lemma *lemma-poly-pderiv-add*: ∀ *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*: ∀ *q*. *poly* (*pderiv* (*p* ∗∗∗ *q*)) *x* =
poly (*p* ∗∗∗ (*pderiv q*) +++ *q* ∗∗∗ (*pderiv p*)) *x*
⟨*proof*⟩

lemma *poly-pderiv-exp*: *poly* (*pderiv* (*p* %^ (*Suc n*))) *x* =
poly ((*real* (*Suc n*)) %∗ (*p* %^ *n*) ∗∗∗ *pderiv p*) *x*
⟨*proof*⟩

lemma *poly-pderiv-exp-prime*: *poly* (*pderiv* ([−*a*, *1*] %^ (*Suc n*))) *x* =
poly (*real* (*Suc n*) %∗ ([−*a*, *1*] %^ *n*)) *x*
⟨*proof*⟩

lemma *real-mult-zero-disj-iff* [*simp*]: (*x* ∗ *y* = *0*) = (*x* = (*0*::*real*) | *y* = *0*)
⟨*proof*⟩

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*⟩

instance *real*:: *idom-char-0*
⟨*proof*⟩

instance *real*:: *recpower-idom-char-0*
⟨*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*⟩

lemma *lemma-order-pderiv* [*rule-format*]:
∀ *p q a*. *0* < *n* &
poly (*pderiv p*) ≠ *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*) ≠ *poly* []; *order a p* ≠ *0* |]
==> (*order a p* = *Suc* (*order a* (*pderiv p*)))
⟨*proof*⟩

Now justify the standard squarefree decomposition, i.e. f / gcd(f,f').

lemma *poly-squarefree-decomp-order*: [| *poly* (*pderiv p*) ≠ *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*) ≠ *poly* [];
poly p = *poly* (*q* *** *d*);
poly (*pderiv p*) = *poly* (*e* *** *d*);
poly d = *poly* (*r* *** *p* +++ *s* *** *pderiv p*)
|] ==> ∀ *a*. *order a q* = (*if order a p* = *0 then 0 else 1*)
⟨*proof*⟩

lemma *order-pderiv2*: [| *poly* (*pderiv p*) ≠ *poly* []; *order a p* ≠ *0* |]
==> (*order a* (*pderiv p*) = *n*) = (*order a p* = *Suc n*)
⟨*proof*⟩

lemma *rsquarefree-roots*:
rsquarefree p = (∀ *a*. ~(*poly p a* = *0* & *poly* (*pderiv p*) *a* = *0*))
⟨*proof*⟩

lemma *poly-squarefree-decomp*: [| *poly* (*pderiv p*) ≠ *poly* [];
poly p = *poly* (*q* *** *d*);
poly (*pderiv p*) = *poly* (*e* *** *d*);
poly d = *poly* (*r* *** *p* +++ *s* *** *pderiv p*)
|] ==> *rsquarefree q* & (∀ *a*. (*poly q a* = *0*) = (*poly p a* = *0*))
⟨*proof*⟩

end

21 NthRoot: Nth Roots of Real Numbers

theory *NthRoot*
imports ~~/*src*/*HOL*/*Library*/*Parity* ../*Hyperreal*/*Deriv*
begin

21.1 Existence of Nth Root

Existence follows from the Intermediate Value Theorem

lemma *realpow-pos-nth*:
assumes *n*: *0* < *n*
assumes *a*: *0* < *a*
shows ∃ *r*>*0*. *r* ^ *n* = (*a*::*real*)
⟨*proof*⟩

lemma *realpow-pos-nth2*: (*0*::*real*) < *a* ⟹ ∃ *r*>*0*. *r* ^ *Suc n* = *a*
⟨*proof*⟩

Uniqueness of nth positive root

lemma *realpow-pos-nth-unique*:

$[\![0 < n;\ 0 < a]\!] \Longrightarrow \exists!r.\ 0 < r \wedge r\ \hat{}\ n = (a::real)$
⟨*proof*⟩

21.2 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**
$root\ n\ x = (if\ 0 < x\ then\ (THE\ u.\ 0 < u \wedge u\ \hat{}\ n = x)\ else$
$\quad if\ x < 0\ then\ -\ (THE\ u.\ 0 < u \wedge u\ \hat{}\ n = -\ x)\ else\ 0)$

lemma *real-root-zero* [*simp*]: $root\ n\ 0 = 0$
⟨*proof*⟩

lemma *real-root-minus*: $0 < n \Longrightarrow root\ n\ (-\ x) = -\ root\ n\ x$
⟨*proof*⟩

lemma *real-root-gt-zero*: $[\![0 < n;\ 0 < x]\!] \Longrightarrow 0 < root\ n\ x$
⟨*proof*⟩

lemma *real-root-pow-pos*:
$[\![0 < n;\ 0 < x]\!] \Longrightarrow root\ n\ x\ \hat{}\ n = x$
⟨*proof*⟩

lemma *real-root-pow-pos2* [*simp*]:
$[\![0 < n;\ 0 \leq x]\!] \Longrightarrow root\ n\ x\ \hat{}\ n = x$
⟨*proof*⟩

lemma *odd-real-root-pow*: $odd\ n \Longrightarrow root\ n\ x\ \hat{}\ n = x$
⟨*proof*⟩

lemma *real-root-ge-zero*: $[\![0 < n;\ 0 \leq x]\!] \Longrightarrow 0 \leq root\ n\ x$
⟨*proof*⟩

lemma *real-root-power-cancel*: $[\![0 < n;\ 0 \leq x]\!] \Longrightarrow root\ n\ (x\ \hat{}\ n) = x$
⟨*proof*⟩

lemma *odd-real-root-power-cancel*: $odd\ n \Longrightarrow root\ n\ (x\ \hat{}\ n) = x$
⟨*proof*⟩

lemma *real-root-pos-unique*:
$[\![0 < n;\ 0 \leq y;\ y\ \hat{}\ n = x]\!] \Longrightarrow root\ n\ x = y$
⟨*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*:
$[\![0 < n;\ 0 \leq x;\ x < y]\!] \Longrightarrow root\ n\ x < root\ n\ y$
⟨*proof*⟩

lemma *real-root-less-mono*: $[\![0 < n;\ x < y]\!] \Longrightarrow root\ n\ x < root\ n\ y$
⟨*proof*⟩

lemma *real-root-le-mono*: $[\![0 < n;\ x \leq y]\!] \Longrightarrow root\ n\ x \leq root\ n\ y$
⟨*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*]:
$0 < n \Longrightarrow (root\ n\ x \leq root\ n\ y) = (x \leq y)$
⟨*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*⟧ ⟹ *root* (*m* * *n*) *x* = *root m* (*root n x*)
⟨*proof*⟩

lemma *real-root-mult-exp*:
⟦0 < *m*; 0 < *n*⟧ ⟹ *root* (*m* * *n*) *x* = *root m* (*root n x*)
⟨*proof*⟩

lemma *real-root-commute*:
⟦0 < *m*; 0 < *n*⟧ ⟹ *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*⟧ ⟹ *root N x* < *root n x*
⟨*proof*⟩

lemma *real-root-strict-increasing*:
⟦0 < *n*; *n* < *N*; 0 < *x*; *x* < 1⟧ ⟹ *root n x* < *root N x*
⟨*proof*⟩

lemma *real-root-decreasing*:
⟦0 < *n*; *n* < *N*; 1 ≤ *x*⟧ ⟹ *root N x* ≤ *root n x*
⟨*proof*⟩

lemma *real-root-increasing*:
⟦0 < *n*; *n* < *N*; 0 ≤ *x*; *x* ≤ 1⟧ ⟹ *root n x* ≤ *root N x*
⟨*proof*⟩

Roots of multiplication and division

lemma *real-root-mult-lemma*:
⟦0 < *n*; 0 ≤ *x*; 0 ≤ *y*⟧ ⟹ *root n* (*x* * *y*) = *root n x* * *root n y*
⟨*proof*⟩

lemma *real-root-inverse-lemma*:
⟦0 < *n*; 0 ≤ *x*⟧ ⟹ *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 ⟹ root n (x / y) = root n x / root n y
⟨*proof*⟩

lemma *real-root-power*:
0 < n ⟹ root n (x ^ k) = root n x ^ k
⟨*proof*⟩

lemma *real-root-abs*: *0 < n ⟹ 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⟧ ⟹ isCont (root n) x
⟨*proof*⟩

lemma *isCont-root-zero*:
0 < n ⟹ isCont (root n) 0
⟨*proof*⟩

lemma *isCont-real-root*: *0 < n ⟹ 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 ^ (n − Suc 0))*
⟨*proof*⟩

lemma *DERIV-odd-real-root*:
assumes *n*: *odd n*
assumes *x*: *x ≠ 0*
shows *DERIV (root n) x :> inverse (real n * root n x ^ (n − Suc 0))*
⟨*proof*⟩

21.3 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*: $sqrt\ (x\ \hat{}\ k) = sqrt\ x\ \hat{}\ k$
⟨*proof*⟩

lemma *real-sqrt-gt-zero*: $0 < x \Longrightarrow 0 < sqrt\ x$
⟨*proof*⟩

lemma *real-sqrt-ge-zero*: $0 \leq x \Longrightarrow 0 \leq sqrt\ x$
⟨*proof*⟩

lemma *real-sqrt-less-mono*: $x < y \Longrightarrow sqrt\ x < sqrt\ y$
⟨*proof*⟩

lemma *real-sqrt-le-mono*: $x \leq y \Longrightarrow sqrt\ x \leq sqrt\ y$
⟨*proof*⟩

lemma *real-sqrt-less-iff* [*simp*]: $(sqrt\ x < sqrt\ y) = (x < y)$
⟨*proof*⟩

lemma *real-sqrt-le-iff* [*simp*]: $(sqrt\ x \leq sqrt\ y) = (x \leq y)$
⟨*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*:
$0 < x \Longrightarrow DERIV\ sqrt\ x :> inverse\ (sqrt\ x)\ /\ 2$
⟨*proof*⟩

lemma *not-real-square-gt-zero* [*simp*]: $(\sim\ (0::real) < x*x) = (x = 0)$
⟨*proof*⟩

lemma *real-sqrt-abs2* [*simp*]: $sqrt(x*x) = |x|$
⟨*proof*⟩

lemma *real-sqrt-pow2-gt-zero*: $0 < x \Longrightarrow 0 < (sqrt\ x)^2$
⟨*proof*⟩

lemma *real-sqrt-not-eq-zero*: $0 < x \Longrightarrow sqrt\ x \neq 0$
⟨*proof*⟩

lemma *real-inv-sqrt-pow2*: $0 < x \Longrightarrow inverse\ (sqrt(x))\ \hat{}\ 2 = inverse\ x$
⟨*proof*⟩

lemma *real-sqrt-eq-zero-cancel*: $[|\ 0 \leq x;\ sqrt(x) = 0|] \Longrightarrow x = 0$
⟨*proof*⟩

lemma *real-sqrt-ge-one*: $1 \leq x \Longrightarrow 1 \leq sqrt\ x$
⟨*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*: $0 < u \Longrightarrow u\ /\ sqrt\ 2 < u$
⟨*proof*⟩

lemma *four-x-squared*:
fixes $x::real$
shows $4 * x^2 = (2 * x)^2$
⟨*proof*⟩

21.4 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-ring*,*recpower*}
shows $(x + y)^2 = x^2 + y^2 + 2 * x * y$
⟨*proof*⟩

lemma *power2-diff*:
fixes $x\ y :: 'a$::{*number-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*:
$[\![|x| < u\ /\ sqrt\ 2; |y| < u\ /\ sqrt\ 2]\!] \Longrightarrow sqrt\ (x^2 + y^2) < u$
⟨*proof*⟩

Needed for the infinitely close relation over the nonstandard complex numbers

lemma *lemma-sqrt-hcomplex-capprox*:
[| $0 < u$; $x < u/2$; $y < u/2$; $0 \leq x$; $0 \leq y$ |] ==> $sqrt\ (x^2 + y^2) < u$
⟨*proof*⟩

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*: $0 < x \Longrightarrow root\ (Suc\ n)\ (x\ \hat{}\ (Suc\ n)) = x$
⟨*proof*⟩

end

22 Transcendental: Power Series, Transcendental Functions etc.

theory *Transcendental*
imports *Fact Series Deriv NthRoot*
begin

22.1 Properties of Power Series

lemma *lemma-realpow-diff*:
fixes $y :: 'a::recpower$
shows $p \leq n \Longrightarrow y \hat{} (Suc\ n - p) = (y \hat{} (n - p)) * y$
⟨*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*⟩

22.2 Term-by-Term Differentiability of Power Series

definition
diffs :: (*nat* => *'a*::*ring-1*) => *nat* => *'a* **where**
diffs c = (%*n*. *of-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*:
(∑ *n*=*0*..<*n*. (*diffs c*)(*n*) * (*x* ^ *n*)) =
(∑ *n*=*0*..<*n*. *of-nat n* * *c*(*n*) * (*x* ^ (*n* − *Suc 0*))) +
(*of-nat n* * *c*(*n*) * *x* ^ (*n* − *Suc 0*))
⟨*proof*⟩

lemma *lemma-diffs2*:
(∑ *n*=*0*..<*n*. *of-nat n* * *c*(*n*) * (*x* ^ (*n* − *Suc 0*))) =
(∑ *n*=*0*..<*n*. (*diffs c*)(*n*) * (*x* ^ *n*)) −
(*of-nat n* * *c*(*n*) * *x* ^ (*n* − *Suc 0*))
⟨*proof*⟩

lemma *diffs-equiv*:
summable (%*n*. (*diffs c*)(*n*) * (*x* ^ *n*)) ==>
(%*n*. *of-nat n* * *c*(*n*) * (*x* ^ (*n* − *Suc 0*))) *sums*
(∑ *n*. (*diffs c*)(*n*) * (*x* ^ *n*))
⟨*proof*⟩

lemma *lemma-termdiff1*:
fixes *z* :: *'a* :: {*recpower*,*comm-ring*} **shows**
(∑ *p*=*0*..<*m*. (((*z* + *h*) ^ (*m* − *p*)) * (*z* ^ *p*)) − (*z* ^ *m*)) =
(∑ *p*=*0*..<*m*. (*z* ^ *p*) * (((*z* + *h*) ^ (*m* − *p*)) − (*z* ^ (*m* − *p*))))
⟨*proof*⟩

lemma *less-add-one*: *m* < *n* ==> (∃ *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-nat n* * (*r*::*'a*::*ring-1*) = (∑ *i* = *0*..<*n*. *f i* − *r*)
⟨*proof*⟩

lemma *lemma-termdiff2*:
fixes *h* :: *'a* :: {*recpower*,*field*}
assumes *h*: *h* ≠ *0* **shows**
((*z* + *h*) ^ *n* − *z* ^ *n*) / *h* − *of-nat n* * *z* ^ (*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-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*⟩

22.3 Exponential Function

definition
$exp :: 'a \Rightarrow 'a::\{recpower,real\text{-}normed\text{-}field,banach\}$ **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 $x :: 'a::\{real\text{-}normed\text{-}algebra\text{-}1,recpower,banach\}$
defines *S-def*: $S \equiv \lambda n.\ x\ \hat{}\ n\ /_R\ real\ (fact\ n)$
shows *summable S*
⟨*proof*⟩

lemma *summable-norm-exp*:
fixes $x :: 'a::\{real\text{-}normed\text{-}algebra\text{-}1,recpower,banach\}$
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 ^ ((n − Suc 0) div 2)/(real (fact n))) ∗ 0 ^ 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*:
(∑ n=1..<n. if even n then −1 ^ (n div 2)/(real (fact n)) ∗ 0 ^ n
else 0) = 0
⟨*proof*⟩

lemma *exp-converges*: *(λn. 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*⟩

22.4 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 (λn. of-real (f n)) = (λn. 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* = (*λx.* $\sum$ *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*⟩

22.5 Properties of the Exponential Function

lemma *powser-zero*:
fixes *f* :: *nat* ⇒ *'a*::{*real-normed-algebra-1*,*recpower*}
shows (∑ *n*. *f n* ∗ *0* ^ *n*) = *f 0*
⟨*proof*⟩

lemma *exp-zero* [*simp*]: *exp 0* = *1*
⟨*proof*⟩

lemma *setsum-head2*:
m ≤ *n* ⟹ *setsum f* {*m*..*n*} = *f m* + *setsum f* {*Suc m*..*n*}
⟨*proof*⟩

lemma *setsum-cl-ivl-Suc2*:
(∑ *i*=*m*..*Suc n*. *f i*) = (*if Suc n* < *m then 0 else f m* + (∑ *i*=*m*..*n*. *f* (*Suc i*)))
⟨*proof*⟩

lemma *exp-series-add*:
fixes *x y* :: *'a*::{*real-field*,*recpower*}
defines *S-def*: *S* ≡ λ*x n*. *x* ^ *n* /$_R$ *real* (*fact n*)
shows *S* (*x* + *y*) *n* = (∑ *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-real x*) = *of-real* (*exp x*)
⟨*proof*⟩

lemma *exp-ge-add-one-self-aux*: *0* ≤ (*x*::*real*) ==> (*1* + *x*) ≤ *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 ≤ exp (x::real)*
⟨*proof*⟩

lemma *exp-not-eq-zero* [*simp*]: *exp x ≠ 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*: $exp\ (x::real) < exp\ y \Longrightarrow x < y$
⟨*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*: $1 \leq y \Longrightarrow \exists x.\ 0 \leq x\ \&\ x \leq y - 1\ \&\ exp(x::real) = y$
⟨*proof*⟩

lemma *exp-total*: $0 < (y::real) \Longrightarrow \exists x.\ exp\ x = y$
⟨*proof*⟩

22.6 Properties of the Logarithmic Function

definition
$ln :: real \Rightarrow real$ **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*: $[|\ 0 < x;\ 0 < y\ |] \Longrightarrow ln(x * y) = ln(x) + ln(y)$
⟨*proof*⟩

lemma *ln-inj-iff*[*simp*]: $[|\ 0 < x;\ 0 < y\ |] \Longrightarrow (ln\ x = ln\ y) = (x = y)$
⟨*proof*⟩

lemma *ln-one*[*simp*]: $ln\ 1 = 0$
⟨*proof*⟩

lemma *ln-inverse*: $0 < x \Longrightarrow ln(inverse\ x) = -\ ln\ x$

⟨*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* ≤ *ln y*) = (*x* ≤ *y*)
⟨*proof*⟩

lemma *ln-realpow*: *0 < x* ==> *ln*(*x* ^ *n*) = *real n* ∗ *ln*(*x*)
⟨*proof*⟩

lemma *ln-add-one-self-le-self* [*simp*]: *0* ≤ *x* ==> *ln*(*1* + *x*) ≤ *x*
⟨*proof*⟩

lemma *ln-less-self* [*simp*]: *0 < x* ==> *ln x < x*
⟨*proof*⟩

lemma *ln-ge-zero* [*simp*]:
  assumes *x*: *1* ≤ *x* **shows** *0* ≤ *ln x*
⟨*proof*⟩

lemma *ln-ge-zero-imp-ge-one*:
  assumes *ln*: *0* ≤ *ln x*
      and *x*: *0 < x*
  shows *1* ≤ *x*
⟨*proof*⟩

lemma *ln-ge-zero-iff* [*simp*]: *0 < x* ==> (*0* ≤ *ln x*) = (*1* ≤ *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*]: $0 < x \Longrightarrow (ln\ x = 0) = (x = 1)$
⟨*proof*⟩

lemma *ln-less-zero*: $[|\ 0 < x; x < 1\ |] \Longrightarrow ln\ x < 0$
⟨*proof*⟩

lemma *exp-ln-eq*: $exp\ u = x \Longrightarrow ln\ x = u$
⟨*proof*⟩

lemma *isCont-ln*: $0 < x \Longrightarrow isCont\ ln\ x$
⟨*proof*⟩

lemma *DERIV-ln*: $0 < x \Longrightarrow DERIV\ ln\ x :> inverse\ x$
⟨*proof*⟩

22.7 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*: [| $1 < x$; $0 \leq y$ |] ==> $1 < x + (y::real)$
⟨*proof*⟩

lemma *abs-sin-le-one* [*simp*]: $|sin\ x| \leq 1$
⟨*proof*⟩

lemma *sin-ge-minus-one* [*simp*]: $-1 \leq sin\ x$
⟨*proof*⟩

lemma *sin-le-one* [*simp*]: $sin\ x \leq 1$
⟨*proof*⟩

lemma *abs-cos-le-one* [*simp*]: $|cos\ x| \leq 1$
⟨*proof*⟩

lemma *cos-ge-minus-one* [*simp*]: $-1 \leq cos\ x$
⟨*proof*⟩

lemma *cos-le-one* [*simp*]: $cos\ x \leq 1$
⟨*proof*⟩

lemma *DERIV-fun-pow*: $DERIV\ g\ x :> m ==>$
$DERIV\ (\%x.\ (g\ x)\ \hat{}\ n)\ x :> real\ n * (g\ x)\ \hat{}\ (n - 1) * m$
⟨*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*:
$\forall x.$
$DERIV\ (\%x.\ (sin\ (x + y) - (sin\ x * cos\ y + cos\ x * sin\ y))\ \hat{}\ 2 +$
$(cos\ (x + y) - (cos\ x * cos\ y - sin\ x * sin\ y))\ \hat{}\ 2)\ x :> 0$
⟨*proof*⟩

lemma *sin-cos-add* [*simp*]:
$(sin\ (x + y) - (sin\ x * cos\ y + cos\ x * sin\ y))\ \hat{}\ 2 +$
$(cos\ (x + y) - (cos\ x * cos\ y - sin\ x * sin\ y))\ \hat{}\ 2 = 0$
⟨*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*⟩

22.8 The Constant Pi

definition
$pi :: real$ **where**
$pi = 2 * (THE\ x.\ 0 \leq (x::real)\ \&\ x \leq 2\ \&\ cos\ x = 0)$

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*⟩

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 ≤ x* & *x ≤ 2* & *cos x* = *0*
⟨*proof*⟩

lemma *pi-half*: *pi*/*2* = (*THE x*. *0 ≤ x* & *x ≤ 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 ≤ 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* ^ *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*) ≤ *x*; *x* ≤ *pi*/*2* |] ==> *0* ≤ *cos x*
⟨*proof*⟩

lemma *sin-gt-zero-pi*: [| *0* < *x*; *x* < *pi* |] ==> *0* < *sin x*
⟨*proof*⟩

lemma *sin-ge-zero*: [| *0* ≤ *x*; *x* ≤ *pi* |] ==> *0* ≤ *sin x*
⟨*proof*⟩

lemma *cos-total*: [| −*1* ≤ *y*; *y* ≤ *1* |] ==> *EX*! *x*. *0* ≤ *x* & *x* ≤ *pi* & (*cos x* = *y*)
⟨*proof*⟩

lemma *sin-total*:
[| −*1* ≤ *y*; *y* ≤ *1* |] ==> *EX*! *x*. −(*pi*/*2*) ≤ *x* & *x* ≤ *pi*/*2* & (*sin x* = *y*)
⟨*proof*⟩

lemma *reals-Archimedean4*:
[| 0 < y; 0 ≤ x |] ==> ∃ n. real n * y ≤ x & x < real (Suc n) * y
⟨*proof*⟩

lemma *cos-zero-lemma*:
[| 0 ≤ x; cos x = 0 |] ==>
∃ n::nat. ~even n & x = real n * (pi/2)
⟨*proof*⟩

lemma *sin-zero-lemma*:
[| 0 ≤ x; sin x = 0 |] ==>
∃ n::nat. even n & x = real n * (pi/2)
⟨*proof*⟩

lemma *cos-zero-iff*:
(cos x = 0) =
((∃ n::nat. ~even n & (x = real n * (pi/2))) |
(∃ n::nat. ~even n & (x = -(real n * (pi/2)))))
⟨*proof*⟩

lemma *sin-zero-iff*:
(sin x = 0) =
((∃ n::nat. even n & (x = real n * (pi/2))) |
(∃ n::nat. even n & (x = -(real n * (pi/2)))))
⟨*proof*⟩

22.9 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*⟩

22.10 Inverse Trigonometric Functions

definition
$arcsin :: real \Rightarrow real$ **where**
$arcsin\ y = (THE\ x.\ -(pi/2) \leq x\ \&\ x \leq pi/2\ \&\ sin\ x = y)$

definition
$arccos :: real \Rightarrow real$ **where**
$arccos\ y = (THE\ x.\ 0 \leq x\ \&\ x \leq pi\ \&\ cos\ x = y)$

definition
$arctan :: real \Rightarrow real$ **where**
$arctan\ y = (THE\ x.\ -(pi/2) < x\ \&\ x < pi/2\ \&\ tan\ x = y)$

lemma *arcsin*:
$[|\ -1 \leq y;\ y \leq 1\ |] \Longrightarrow -(pi/2) \leq arcsin\ y\ \&\ arcsin\ y \leq pi/2\ \&\ sin(arcsin\ y) = y$
⟨*proof*⟩

lemma *arcsin-pi*:
$[|\ -1 \leq y;\ y \leq 1\ |] \Longrightarrow -(pi/2) \leq arcsin\ y\ \&\ arcsin\ y \leq pi\ \&\ sin(arcsin\ y) = y$
⟨*proof*⟩

lemma *sin-arcsin* [*simp*]: $[|\ -1 \leq y;\ y \leq 1\ |] \Longrightarrow sin(arcsin\ y) = y$
⟨*proof*⟩

lemma *arcsin-bounded*:
$[|\ -1 \leq y;\ y \leq 1\ |] \Longrightarrow -(pi/2) \leq arcsin\ y\ \&\ arcsin\ y \leq pi/2$
⟨*proof*⟩

lemma *arcsin-lbound*: $[|\ -1 \leq y;\ y \leq 1\ |] \Longrightarrow -(pi/2) \leq arcsin\ y$
⟨*proof*⟩

lemma *arcsin-ubound*: $[|\ -1 \leq y;\ y \leq 1\ |] \Longrightarrow arcsin\ y \leq pi/2$
⟨*proof*⟩

lemma *arcsin-lt-bounded*:
$[|\ -1 < y;\ y < 1\ |] \Longrightarrow -(pi/2) < arcsin\ y\ \&\ arcsin\ y < pi/2$
⟨*proof*⟩

lemma *arcsin-sin*: $[|-(pi/2) \leq x;\ x \leq pi/2\ |] \Longrightarrow arcsin(sin\ x) = x$
⟨*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*]: [| $-1 \leq y$; $y \leq 1$ |] ==> $\cos(\arccos\ y) = y$
⟨*proof*⟩

lemma *arccos-bounded*: [| $-1 \leq y$; $y \leq 1$ |] ==> $0 \leq \arccos\ y$ & $\arccos\ y \leq pi$
⟨*proof*⟩

lemma *arccos-lbound*: [| $-1 \leq y$; $y \leq 1$ |] ==> $0 \leq \arccos\ y$
⟨*proof*⟩

lemma *arccos-ubound*: [| $-1 \leq y$; $y \leq 1$ |] ==> $\arccos\ y \leq pi$
⟨*proof*⟩

lemma *arccos-lt-bounded*:
[| $-1 < y$; $y < 1$ |]
==> $0 < \arccos\ y$ & $\arccos\ y < pi$
⟨*proof*⟩

lemma *arccos-cos*: [|$0 \leq x$; $x \leq pi$ |] ==> $\arccos(\cos\ x) = x$
⟨*proof*⟩

lemma *arccos-cos2*: [|$x \leq 0$; $-pi \leq x$ |] ==> $\arccos(\cos\ x) = -x$
⟨*proof*⟩

lemma *cos-arcsin*: $[\![-1 \leq x; x \leq 1]\!] \Longrightarrow \cos\ (\arcsin\ x) = sqrt\ (1 - x^2)$
⟨*proof*⟩

lemma *sin-arccos*: $[\![-1 \leq x; x \leq 1]\!] \Longrightarrow \sin\ (\arccos\ x) = sqrt\ (1 - x^2)$
⟨*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*⟩

22.11 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*⟩

22.12 Existence of Polar Coordinates

lemma *cos-x-y-le-one*: $|x \mathbin{/} sqrt\ (x^2 + y^2)| \leq 1$
⟨*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 ==> ∃ r a. x = r * cos a & y = r * sin a*
⟨*proof*⟩

lemma *polar-ex2*:
*y < 0 ==> ∃ 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*⟩

22.13 Theorems about Limits

lemma *isCont-inv-fun*:
fixes $f\ g :: real \Rightarrow real$
shows $[|\ 0 < d; \forall z.\ |z - x| \leq d \longrightarrow g(f(z)) = z;$
$\forall z.\ |z - x| \leq d \longrightarrow isCont\ f\ z\ |]$
$\Longrightarrow isCont\ g\ (f\ x)$
⟨*proof*⟩

lemma *isCont-inv-fun-inv*:
fixes $f\ g :: real \Rightarrow real$
shows $[|\ 0 < d;$
$\forall z.\ |z - x| \leq d \longrightarrow g(f(z)) = z;$
$\forall z.\ |z - x| \leq d \longrightarrow isCont\ f\ z\ |]$
$\Longrightarrow \exists e.\ 0 < e\ \&$
$(\forall y.\ 0 < |y - f(x)|\ \&\ |y - f(x)| < e \longrightarrow f(g(y)) = y)$
⟨*proof*⟩

Bartle/Sherbert: Introduction to Real Analysis, Theorem 4.2.9, p. 110

lemma *LIM-fun-gt-zero*:
$[|\ f \dashrightarrow c \longrightarrow (l::real);\ 0 < l\ |]$
$\Longrightarrow \exists r.\ 0 < r\ \&\ (\forall x::real.\ x \neq c\ \&\ |c - x| < r \longrightarrow 0 < f\ x)$
⟨*proof*⟩

lemma *LIM-fun-less-zero*:
$[|\ f \dashrightarrow c \longrightarrow (l::real);\ l < 0\ |]$
$\Longrightarrow \exists r.\ 0 < r\ \&\ (\forall x::real.\ x \neq c\ \&\ |c - x| < r \longrightarrow f\ x < 0)$
⟨*proof*⟩

lemma *LIM-fun-not-zero*:
$[|\ f \dashrightarrow c \longrightarrow (l::real);\ l \neq 0\ |]$
$\Longrightarrow \exists r.\ 0 < r\ \&\ (\forall x::real.\ x \neq c\ \&\ |c - x| < r \longrightarrow f\ x \neq 0)$
⟨*proof*⟩

end

23 Complex: Complex Numbers: Rectangular and Polar Representations

theory *Complex*
imports *../Real/Real ../Hyperreal/Transcendental*
begin

datatype *complex* = *Complex real real*

primrec
Re :: *complex* ⇒ *real*
where
Re: *Re* (*Complex x y*) = *x*

primrec
Im :: *complex* ⇒ *real*
where
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*⟧ ⟹ *x* = *y*
⟨*proof*⟩

lemma *expand-complex-eq*: *x* = *y* ⟷ *Re x* = *Re y* ∧ *Im x* = *Im y*
⟨*proof*⟩

lemmas *complex-Re-Im-cancel-iff* = *expand-complex-eq*

23.1 Addition and Subtraction

instantiation *complex* :: *ab-group-add*
begin

definition
complex-zero-def: *0* = *Complex 0 0*

definition
complex-add-def: *x* + *y* = *Complex* (*Re x* + *Re y*) (*Im x* + *Im y*)

definition
complex-minus-def: − *x* = *Complex* (− *Re x*) (− *Im x*)

definition
complex-diff-def: *x* − (*y*::*complex*) = *x* + − *y*

lemma *Complex-eq-0* [*simp*]: *Complex a b* = *0* ⟷ *a* = *0* ∧ *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
⟨*proof*⟩

end

23.2 Multiplication and Division

instantiation *complex* :: {*field*, *division-by-zero*}
begin

definition
complex-one-def: *1 = Complex 1 0*

definition
complex-mult-def: *x * y =*
*Complex (Re x * Re y − Im x * Im y) (Re x * Im y + Im x * Re y)*

definition

complex-inverse-def: *inverse x* =
Complex (*Re x* / ((*Re x*)^2 + (*Im x*)^2)) (− *Im x* / ((*Re x*)^2 + (*Im x*)^2))

definition
complex-divide-def: *x* / (*y*::*complex*) = *x* ∗ *inverse y*

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
⟨*proof*⟩

end

23.3 Exponentiation

instantiation *complex* :: *recpower*
begin

primrec *power-complex* **where**
complexpow-0: *z* ^ *0* = (*1*::*complex*)

| *complexpow-Suc*: $z \hat{} Suc\ n = (z::complex) * z \hat{} n$

instance ⟨*proof*⟩

end

23.4 Numerals and Arithmetic

instantiation *complex* :: *number-ring*
begin

definition *number-of-complex* **where**
complex-number-of-def: *number-of w* = (*of-int w* :: *complex*)

instance
⟨*proof*⟩

end

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

lemma *complex-Im-of-nat* [*simp*]: *Im* (*of-nat n*) = *0*
⟨*proof*⟩

lemma *complex-Re-of-int* [*simp*]: *Re* (*of-int z*) = *of-int z*
⟨*proof*⟩

lemma *complex-Im-of-int* [*simp*]: *Im* (*of-int z*) = *0*
⟨*proof*⟩

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

lemma *complex-Im-number-of* [*simp*]: *Im* (*number-of v*) = *0*
⟨*proof*⟩

lemma *Complex-eq-number-of* [*simp*]:
(*Complex a b* = *number-of w*) = (*a* = *number-of w* ∧ *b* = *0*)
⟨*proof*⟩

23.5 Scalar Multiplication

instantiation *complex* :: *real-field*
begin

definition
complex-scaleR-def: *scaleR r x* = *Complex* (*r* ∗ *Re x*) (*r* ∗ *Im x*)

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
⟨*proof*⟩

end

23.6 Properties of Embedding from Reals

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

lemma *complex-of-real-def*: *complex-of-real r = Complex r 0*
⟨*proof*⟩

lemma *Re-complex-of-real* [*simp*]: *Re (complex-of-real z) = z*
⟨*proof*⟩

lemma *Im-complex-of-real* [*simp*]: *Im (complex-of-real z) = 0*
⟨*proof*⟩

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

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

lemma *Complex-mult-complex-of-real*:
*Complex x y * complex-of-real r = Complex (x*r) (y*r)*
⟨*proof*⟩

lemma *complex-of-real-mult-Complex*:
*complex-of-real r * Complex x y = Complex (r*x) (r*y)*
⟨*proof*⟩

23.7 Vector Norm

instantiation *complex* :: *real-normed-field*
begin

definition
complex-norm-def: $norm\ z = sqrt\ ((Re\ z)^2 + (Im\ z)^2)$

abbreviation
$cmod :: complex \Rightarrow real$ **where**
$cmod \equiv norm$

definition
complex-sgn-def: $sgn\ x = x\ /_R\ cmod\ x$

lemmas *cmod-def* = *complex-norm-def*

lemma *complex-norm* [*simp*]: $cmod\ (Complex\ x\ y) = sqrt\ (x^2 + y^2)$
⟨*proof*⟩

instance
⟨*proof*⟩

end

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*]

lemma *abs-Re-le-cmod*: $|Re\ x| \leq cmod\ x$
⟨*proof*⟩

lemma *abs-Im-le-cmod*: $|Im\ x| \leq cmod\ x$
⟨*proof*⟩

23.8 Completeness of the Complexes

interpretation *Re*: *bounded-linear* [*Re*]
⟨*proof*⟩

interpretation *Im*: *bounded-linear* [*Im*]
⟨*proof*⟩

lemma *LIMSEQ-Complex*:
⟦*X* −−−−> *a*; *Y* −−−−> *b*⟧ ⟹ (λ*n*. *Complex* (*X n*) (*Y n*)) −−−−> *Complex a b*
⟨*proof*⟩

instance *complex* :: *banach*
⟨*proof*⟩

23.9 The Complex Number i

definition
ii :: *complex* (i) **where**
i-def: *ii* ≡ *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* ∧ *y = 1*)
⟨*proof*⟩

lemma *complex-i-not-zero* [*simp*]: *ii* ≠ *0*
⟨*proof*⟩

lemma *complex-i-not-one* [*simp*]: *ii* ≠ *1*
⟨*proof*⟩

lemma *complex-i-not-number-of* [*simp*]: *ii* ≠ *number-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-of-real r* = *Complex 0 r*
⟨*proof*⟩

lemma *complex-of-real-i* [*simp*]: *complex-of-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*⟩

23.10 Complex Conjugation

definition
cnj :: *complex* ⇒ *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* ^ *n*) = *cnj x* ^ *n*
⟨*proof*⟩

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

lemma *complex-cnj-of-int* [*simp*]: *cnj* (*of-int z*) = *of-int z*
⟨*proof*⟩

lemma *complex-cnj-number-of* [*simp*]: *cnj* (*number-of w*) = *number-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-real x*) = *of-real x*
⟨*proof*⟩

lemma *complex-cnj-i* [*simp*]: *cnj ii* = − *ii*
⟨*proof*⟩

lemma *complex-add-cnj*: *z* + *cnj z* = *complex-of-real* (*2* ∗ *Re z*)
⟨*proof*⟩

lemma *complex-diff-cnj*: *z* − *cnj z* = *complex-of-real* (*2* ∗ *Im z*) ∗ *ii*
⟨*proof*⟩

lemma *complex-mult-cnj*: *z* ∗ *cnj z* = *complex-of-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*⟩

23.11 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* ≤ *pi*)

lemma *sgn-eq*: *sgn z* = *z* / *complex-of-real* (*cmod z*)
⟨*proof*⟩

lemma *i-mult-eq*: *ii* ∗ *ii* = *complex-of-real* (−*1*)
⟨*proof*⟩

lemma *i-mult-eq2* [*simp*]: *ii* ∗ *ii* = −(*1*::*complex*)
⟨*proof*⟩

lemma *complex-eq-cancel-iff2* [*simp*]:
(*Complex x y* = *complex-of-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-of-real x* + *ii* ∗ *complex-of-real y*) =
complex-of-real(*x*/(*x* ^ *2* + *y* ^ *2*)) −
ii ∗ *complex-of-real*(*y*/(*x* ^ *2* + *y* ^ *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 ≠ *0* ==> *cos* (*arg*(*Complex 0 y*)) = *0*
⟨*proof*⟩

23.12 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-of-real r* ∗ *cis a*

definition

expi :: *complex* => *complex* **where**
expi z = *complex-of-real*(*exp* (*Re z*)) ∗ *cis* (*Im z*)

lemma *complex-split-polar*:
∃ *r a*. *z* = *complex-of-real r* ∗ (*Complex* (*cos a*) (*sin a*))
⟨*proof*⟩

lemma *rcis-Ex*: ∃ *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-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*) ^ *n* = *cis* (*real n* ∗ *a*)
⟨*proof*⟩

lemma *DeMoivre2*: (*rcis r a*) ^ *n* = *rcis* (*r* ^ *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* ^ *n*)
⟨*proof*⟩

lemma *sin-n-Im-cis-pow-n*: *sin* (*real n* ∗ *a*) = *Im*(*cis a* ^ *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*: ∃ *a r*. *z* = *complex-of-real r* ∗ *expi a*
⟨*proof*⟩

lemma *expi-two-pi-i* [*simp*]: *expi*((*2*::*complex*) ∗ *complex-of-real pi* ∗ *ii*) = *1*
⟨*proof*⟩

end

24 Fundamental-Theorem-Algebra: Fundamental Theorem of Algebra

theory *Fundamental-Theorem-Algebra*
imports *Univ-Poly Dense-Linear-Order Complex*
begin

25 Square root of complex numbers

definition *csqrt* :: *complex* ⇒ *complex* **where**
csqrt z = (*if Im z* = *0 then*
if 0 ≤ *Re z then Complex* (*sqrt*(*Re z*)) *0*
else Complex 0 (*sqrt*(− *Re z*))
else Complex (*sqrt*((*cmod z* + *Re z*) */2*))
((*Im z* / *abs*(*Im z*)) ∗ *sqrt*((*cmod z* − *Re z*) */2*)))

lemma *csqrt*: *csqrt z* ^ *2* = *z*
⟨*proof*⟩

26 More lemmas about module of complex numbers

lemma *complex-of-real-power*: *complex-of-real x* ^ *n* = *complex-of-real* (*x*^*n*)
⟨*proof*⟩

lemma *cmod-pos*: *cmod z* ≥ *0* ⟨*proof*⟩
lemma *complex-mod-triangle-ineq*: *cmod* (*z* + *w*) ≤ *cmod z* + *cmod w*
⟨*proof*⟩

lemma *cmod-mult*: *cmod* (*z*∗*w*) = *cmod z* ∗ *cmod w*
⟨*proof*⟩

lemma *cmod-divide*: *cmod* (*z*/*w*) = *cmod z* / *cmod w*
⟨*proof*⟩

lemma *cmod-inverse*: *cmod* (*inverse z*) = *inverse* (*cmod z*)
⟨*proof*⟩

lemma *cmod-uminus*: *cmod* (− *z*) = *cmod z*
⟨*proof*⟩
lemma *cmod-abs-norm*: |*cmod w* − *cmod z*| ≤ *cmod* (*w* − *z*)
⟨*proof*⟩

lemma *cmod-power*: *cmod* (*z* ^*n*) = *cmod z* ^ *n* ⟨*proof*⟩
lemma *real-down2*: (*0*::*real*) < *d1* ⟹ *0* < *d2* ==> *EX e*. *0* < *e* & *e* < *d1* & *e* < *d2*
⟨*proof*⟩

lemma *cmod-complex-of-real*: *cmod* (*complex-of-real x*) = |*x*|
⟨*proof*⟩

The triangle inequality for cmod

lemma *complex-mod-triangle-sub*: *cmod w* ≤ *cmod* (*w* + *z*) + *norm z*
⟨*proof*⟩

27 Basic lemmas about complex polynomials

lemma *poly-bound-exists*:
shows ∃ *m*. *m* > *0* ∧ (∀ *z*. *cmod z* <= *r* ⟶ *cmod* (*poly p z*) ≤ *m*)
⟨*proof*⟩

Offsetting the variable in a polynomial gives another of same degree

lemma *poly-offset-lemma*:
shows ∃ *b q*. (*length q* = *length p*) ∧ (∀ *x*. *poly* (*b*#*q*) (*x*::*complex*) = (*a* + *x*) ∗ *poly p x*)
⟨*proof*⟩

lemma *poly-offset*: ∃ *q*. *length q* = *length p* ∧ (∀ *x*. *poly q* (*x*::*complex*) = *poly p* (*a* + *x*))
⟨*proof*⟩

An alternative useful formulation of completeness of the reals

lemma *real-sup-exists*: **assumes** *ex*: $\exists\, x.\ P\ x$ **and** *bz*: $\exists\, z.\ \forall\, x.\ P\ x \longrightarrow x < z$
shows $\exists\, (s{::}real).\ \forall\, y.\ (\exists\, x.\ P\ x \wedge y < x) \longleftrightarrow y < s$
⟨*proof*⟩

28 Some theorems about Sequences

Given a binary function $f{::}\ nat \Rightarrow 'a \Rightarrow 'a$, its values are uniquely determined by a function g

lemma *num-Axiom*: $EX!\ g.\ g\ 0 = e \wedge (\forall\, n.\ g\ (Suc\ n) = f\ n\ (g\ n))$
⟨*proof*⟩

An equivalent formulation of monotony – Not used here, but might be useful

lemma *mono-Suc*: $mono\ f = (\forall\, n.\ (f\ n :: 'a :: order) \leq f\ (Suc\ n))$
⟨*proof*⟩

for any sequence, there is a mootonic subsequence

lemma *seq-monosub*: $\exists f.\ subseq\ f \wedge monoseq\ (\lambda\ n.\ (s\ (f\ n)))$
⟨*proof*⟩

lemma *seq-suble*: **assumes** *sf*: *subseq f* **shows** $n \leq f\ n$
⟨*proof*⟩

29 Fundamental theorem of algebra

lemma *unimodular-reduce-norm*:
assumes *md*: *cmod z = 1*
shows *cmod (z + 1) < 1 ∨ cmod (z − 1) < 1 ∨ cmod (z + ii) < 1 ∨ cmod (z − ii) < 1*
⟨*proof*⟩

Hence we can always reduce modulus of *1 + b z^n* if nonzero

lemma *reduce-poly-simple*:
assumes *b*: *b ≠ 0* **and** *n*: *n≠0*
shows $\exists\, z.\ cmod\ (1 + b * z\hat{}n) < 1$
⟨*proof*⟩

Bolzano-Weierstrass type property for closed disc in complex plane.

lemma *metric-bound-lemma*: *cmod (x − y) <= |Re x − Re y| + |Im x − Im y|*
⟨*proof*⟩

lemma *bolzano-weierstrass-complex-disc*:
assumes *r*: $\forall\, n.\ cmod\ (s\ n) \leq r$
shows $\exists f\ z.\ subseq\ f \wedge (\forall\, e > 0.\ \exists\, N.\ \forall\, n \geq N.\ cmod\ (s\ (f\ n) - z) < e)$
⟨*proof*⟩

Polynomial is continuous.

lemma *poly-cont*:
assumes *ep*: $e > 0$
shows $\exists\, d > 0.\ \forall\, w.\ 0 < cmod\ (w - z) \wedge cmod\ (w - z) < d \longrightarrow cmod\ (poly\ p\ w - poly\ p\ z) < e$
⟨*proof*⟩

Hence a polynomial attains minimum on a closed disc in the complex plane.

lemma *poly-minimum-modulus-disc*:
$\exists\, z.\ \forall\, w.\ cmod\ w \leq r \longrightarrow cmod\ (poly\ p\ z) \leq cmod\ (poly\ p\ w)$
⟨*proof*⟩

lemma $(rcis\ (sqrt\ (abs\ r))\ (a/2))\ \hat{}\ 2 = rcis\ (abs\ r)\ a$
⟨*proof*⟩

lemma *cispi*: $cis\ pi = -1$
⟨*proof*⟩

lemma $(rcis\ (sqrt\ (abs\ r))\ ((pi + a)/2))\ \hat{}\ 2 = rcis\ (-\ abs\ r)\ a$
⟨*proof*⟩

Nonzero polynomial in z goes to infinity as z does.

instance *complex*::*idom-char-0* ⟨*proof*⟩
instance *complex* :: *recpower-idom-char-0* ⟨*proof*⟩

lemma *poly-infinity*:
assumes *ex*: $list\text{-}ex\ (\lambda c.\ c \neq 0)\ p$
shows $\exists\, r.\ \forall\, z.\ r \leq cmod\ z \longrightarrow d \leq cmod\ (poly\ (a\#p)\ z)$
⟨*proof*⟩

Hence polynomial's modulus attains its minimum somewhere.

lemma *poly-minimum-modulus*:
$\exists\, z.\forall\, w.\ cmod\ (poly\ p\ z) \leq cmod\ (poly\ p\ w)$
⟨*proof*⟩

Constant function (non-syntactic characterization).

definition $constant\ f = (\forall\, x\ y.\ f\ x = f\ y)$

lemma *nonconstant-length*: $\neg\ (constant\ (poly\ p)) \Longrightarrow length\ p \geq 2$
⟨*proof*⟩

lemma *poly-replicate-append*:
$poly\ ((replicate\ n\ 0)@p)\ (x::'a::\{recpower,\ comm\text{-}ring\}) = x\hat{}n * poly\ p\ x$
⟨*proof*⟩

Decomposition of polynomial, skipping zero coefficients after the first.

lemma *poly-decompose-lemma*:
assumes *nz*: $\neg(\forall\, z.\ z \neq 0 \longrightarrow poly\ p\ z = (0::'a::\{recpower,idom\}))$
shows $\exists\, k\ a\ q.\ a \neq 0 \wedge Suc\ (length\ q + k) = length\ p \wedge$

$(\forall z.\ poly\ p\ z = z\hat{}k * poly\ (a\#q)\ z)$
⟨*proof*⟩

lemma *poly-decompose*:
assumes *nc*: ~*constant*(*poly p*)
shows $\exists k\ a\ q.\ a{\neq}(0::'a::\{recpower,idom\}) \wedge k{\neq}0 \wedge$
$length\ q + k + 1 = length\ p \wedge$
$(\forall z.\ poly\ p\ z = poly\ p\ 0 + z\hat{}k * poly\ (a\#q)\ z)$
⟨*proof*⟩

Fundamental theorem of algebral

lemma *fundamental-theorem-of-algebra*:
assumes *nc*: ~*constant*(*poly p*)
shows $\exists z::complex.\ poly\ p\ z = 0$
⟨*proof*⟩

Alternative version with a syntactic notion of constant polynomial.

lemma *fundamental-theorem-of-algebra-alt*:
assumes *nc*: ~($\exists a\ l.\ a{\neq}\ 0 \wedge$ *list-all*($\lambda b.\ b = 0$) $l \wedge p = a\#l$)
shows $\exists z.\ poly\ p\ z = (0::complex)$
⟨*proof*⟩

30 Nullstellenstatz, degrees and divisibility of polynomials

lemma *nullstellensatz-lemma*:
fixes *p* :: *complex list*
assumes $\forall x.\ poly\ p\ x = 0 \longrightarrow poly\ q\ x = 0$
and *degree p* = *n* **and** $n \neq 0$
shows *p divides* (*pexp q n*)
⟨*proof*⟩

lemma *nullstellensatz-univariate*:
$(\forall x.\ poly\ p\ x = (0::complex) \longrightarrow poly\ q\ x = 0) \longleftrightarrow$
p divides ($q\ \%\hat{}\ (degree\ p)$) $\vee$ (*poly p* = *poly* [] $\wedge$ *poly q* = *poly* [])
⟨*proof*⟩

Useful lemma

lemma (**in** *idom-char-0*) *constant-degree*: *constant* (*poly p*) $\longleftrightarrow$ *degree p* = *0* (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *divides-degree-lemma*: **assumes** *dpn*: *degree* (*p*::*complex list*) = *n*
shows $n \leq degree\ (p\ ***\ q) \vee poly\ (p\ ***\ q) = poly\ []$
⟨*proof*⟩

lemma *divides-degree*: **assumes** *pq*: *p divides* (*q*:: *complex list*)
shows *degree p* ≤ *degree q* ∨ *poly q* = *poly* []
⟨*proof*⟩

lemma *mpoly-base-conv*:
(*0*::*complex*) ≡ *poly* [] *x c* ≡ *poly* [*c*] *x x* ≡ *poly* [*0*,*1*] *x* ⟨*proof*⟩

lemma *mpoly-norm-conv*:
poly [*0*] (*x*::*complex*) ≡ *poly* [] *x poly* [*poly* [] *y*] *x* ≡ *poly* [] *x* ⟨*proof*⟩

lemma *mpoly-sub-conv*:
poly p (*x*::*complex*) − *poly q x* ≡ *poly p x* + −*1* ∗ *poly q x*
⟨*proof*⟩

lemma *poly-pad-rule*: *poly p x* = *0* ==> *poly* (*0*#*p*) *x* = (*0*::*complex*) ⟨*proof*⟩

lemma *poly-cancel-eq-conv*: *p* = (*0*::*complex*) ⟹ *a* ≠ *0* ⟹ (*q* = *0*) ≡ (*a* ∗ *q* − *b* ∗ *p* = *0*) ⟨*proof*⟩

lemma *resolve-eq-raw*: *poly* [] *x* ≡ *0 poly* [*c*] *x* ≡ (*c*::*complex*) ⟨*proof*⟩
lemma *resolve-eq-then*: (*P* ⟹ (*Q* ≡ *Q1*)) ⟹ (¬*P* ⟹ (*Q* ≡ *Q2*))
⟹ *Q* ≡ *P* ∧ *Q1* ∨ ¬*P*∧ *Q2* ⟨*proof*⟩
lemma *expand-ex-beta-conv*: *list-ex P* [*c*] ≡ *P c* ⟨*proof*⟩

lemma *poly-divides-pad-rule*:
fixes *p q* :: *complex list*
assumes *pq*: *p divides q*
shows *p divides* ((*0*::*complex*)#*q*)
⟨*proof*⟩

lemma *poly-divides-pad-const-rule*:
fixes *p q* :: *complex list*
assumes *pq*: *p divides q*
shows *p divides* (*a* %∗ *q*)
⟨*proof*⟩

lemma *poly-divides-conv0*:
fixes *p* :: *complex list*
assumes *lgpq*: *length q* < *length p* **and** *lq*:*last p* ≠ *0*
shows *p divides q* ≡ (¬ (*list-ex* (λ*c*. *c* ≠ *0*) *q*)) (**is** *?lhs* ≡ *?rhs*)
⟨*proof*⟩

lemma *poly-divides-conv1*:
assumes *a0*: *a*≠ (*0*::*complex*) **and** *pp'*: (*p*::*complex list*) *divides p'*
and *qrp'*: ⋀*x*. *a* ∗ *poly q x* − *poly p' x* ≡ *poly r x*
shows *p divides q* ≡ *p divides* (*r*::*complex list*) (**is** *?lhs* ≡ *?rhs*)

⟨*proof*⟩

lemma *basic-cqe-conv1*:
(∃ *x*. *poly p x* = *0* ∧ *poly* [] *x* ≠ *0*) ≡ *False*
(∃ *x*. *poly* [] *x* ≠ *0*) ≡ *False*
(∃ *x*. *poly* [*c*] *x* ≠ *0*) ≡ *c*≠*0*
(∃ *x*. *poly* [] *x* = *0*) ≡ *True*
(∃ *x*. *poly* [*c*] *x* = *0*) ≡ *c* = *0* ⟨*proof*⟩

lemma *basic-cqe-conv2*:
assumes *l*:*last* (*a*#*b*#*p*) ≠ *0*
shows (∃ *x*. *poly* (*a*#*b*#*p*) *x* = (*0*::*complex*)) ≡ *True*
⟨*proof*⟩

lemma *basic-cqe-conv-2b*: (∃ *x*. *poly p x* ≠ (*0*::*complex*)) ≡ (*list-ex* (λ*c*. *c* ≠ *0*) *p*)
⟨*proof*⟩

lemma *basic-cqe-conv3*:
fixes *p q* :: *complex list*
assumes *l*: *last* (*a*#*p*) ≠ *0*
shows (∃ *x*. *poly* (*a*#*p*) *x* =*0* ∧ *poly q x* ≠ *0*) ≡ ¬ ((*a*#*p*) *divides* (*q* %^ (*length p*)))
⟨*proof*⟩

lemma *basic-cqe-conv4*:
fixes *p q* :: *complex list*
assumes *h*: ⋀*x*. *poly* (*q* %^ *n*) *x* ≡ *poly r x*
shows *p divides* (*q* %^ *n*) ≡ *p divides r*
⟨*proof*⟩

lemma *pmult-Cons-Cons*: ((*a*::*complex*)#*b*#*p*) *** *q* = (*a* %**q*) +++ (*0*#((*b*#*p*) *** *q*))
⟨*proof*⟩

lemma *elim-neg-conv*: − *z* ≡ (−*1*) * (*z*::*complex*) ⟨*proof*⟩
lemma *eqT-intr*: *PROP P* ⟹ (*True* ⟹ *PROP P*) *PROP P* ⟹ *True* ⟨*proof*⟩
lemma *negate-negate-rule*: *Trueprop P* ≡ ¬ *P* ≡ *False* ⟨*proof*⟩
lemma *last-simps*: *last* [*x*] = *x last* (*x*#*y*#*ys*) = *last* (*y*#*ys*) ⟨*proof*⟩
lemma *length-simps*: *length* [] = *0 length* (*x*#*y*#*xs*) = *length xs* + *2 length* [*x*] = *1* ⟨*proof*⟩

lemma *complex-entire*: (*z*::*complex*) ≠ *0* ∧ *w* ≠ *0* ≡ *z***w* ≠ *0* ⟨*proof*⟩
lemma *resolve-eq-ne*: (*P* ≡ *True*) ≡ (¬*P* ≡ *False*) (*P* ≡ *False*) ≡ (¬*P* ≡ *True*)
⟨*proof*⟩
lemma *cqe-conv1*: *poly* [] *x* = *0* ⟷ *True* ⟨*proof*⟩
lemma *cqe-conv2*: (*p* ⟹ (*q* ≡ *r*)) ≡ ((*p* ∧ *q*) ≡ (*p* ∧ *r*)) (**is** *?l* ≡ *?r*)
⟨*proof*⟩
lemma *poly-const-conv*: *poly* [*c*] (*x*::*complex*) = *y* ⟷ *c* = *y* ⟨*proof*⟩

end

31 Order-Relation: Orders as Relations

theory *Order-Relation*
imports *ATP-Linkup Hilbert-Choice*
begin

This prelude could be moved to theory Relation:

definition *irrefl r* ≡ ∀ *x*. (*x*,*x*) ∉ *r*

definition *total-on A r* ≡ ∀ *x*∈*A*.∀ *y*∈*A*. *x*≠*y* ⟶ (*x*,*y*)∈*r* ∨ (*y*,*x*)∈*r*

abbreviation *total* ≡ *total-on UNIV*

lemma *total-on-empty*[*simp*]: *total-on* {} *r*
⟨*proof*⟩

lemma *refl-on-converse*[*simp*]: *refl A* (*r*^−*1*) = *refl A r*
⟨*proof*⟩

lemma *total-on-converse*[*simp*]: *total-on A* (*r*^−*1*) = *total-on A r*
⟨*proof*⟩

lemma *irrefl-diff-Id*[*simp*]: *irrefl*(*r*−*Id*)
⟨*proof*⟩

declare [[*simp-depth-limit* = *2*]]
lemma *trans-diff-Id*: *trans r* ⟹ *antisym r* ⟹ *trans* (*r*−*Id*)
⟨*proof*⟩
declare [[*simp-depth-limit* = *50*]]

lemma *total-on-diff-Id*[*simp*]: *total-on A* (*r*−*Id*) = *total-on A r*
⟨*proof*⟩

31.1 Orders on a set

definition *preorder-on A r* ≡ *refl A r* ∧ *trans r*

definition *partial-order-on A r* ≡ *preorder-on A r* ∧ *antisym r*

definition *linear-order-on A r* ≡ *partial-order-on A r* ∧ *total-on A r*

definition *strict-linear-order-on A r* ≡ *trans r* ∧ *irrefl r* ∧ *total-on A r*

definition *well-order-on A r* ≡ *linear-order-on A r* ∧ *wf*(*r* − *Id*)

lemmas *order-on-defs* =
preorder-on-def partial-order-on-def linear-order-on-def
strict-linear-order-on-def well-order-on-def

lemma *preorder-on-empty*[*simp*]: *preorder-on* {} {}
⟨*proof*⟩

lemma *partial-order-on-empty*[*simp*]: *partial-order-on* {} {}
⟨*proof*⟩

lemma *lnear-order-on-empty*[*simp*]: *linear-order-on* {} {}
⟨*proof*⟩

lemma *well-order-on-empty*[*simp*]: *well-order-on* {} {}
⟨*proof*⟩

lemma *preorder-on-converse*[*simp*]: *preorder-on A* (*r*ˆ−*1*) = *preorder-on A r*
⟨*proof*⟩

lemma *partial-order-on-converse*[*simp*]:
partial-order-on A (*r*ˆ−*1*) = *partial-order-on A r*
⟨*proof*⟩

lemma *linear-order-on-converse*[*simp*]:
linear-order-on A (*r*ˆ−*1*) = *linear-order-on A r*
⟨*proof*⟩

lemma *strict-linear-order-on-diff-Id*:
linear-order-on A r ⟹ *strict-linear-order-on A* (*r*−*Id*)
⟨*proof*⟩

31.2 Orders on the field

abbreviation *Refl r* ≡ *refl* (*Field r*) *r*

abbreviation *Preorder r* ≡ *preorder-on* (*Field r*) *r*

abbreviation *Partial-order r* ≡ *partial-order-on* (*Field r*) *r*

abbreviation *Total r* ≡ *total-on* (*Field r*) *r*

abbreviation *Linear-order r* ≡ *linear-order-on* (*Field r*) *r*

abbreviation *Well-order r* ≡ *well-order-on* (*Field r*) *r*

lemma *subset-Image-Image-iff*:
⟦ *Preorder r*; *A* ⊆ *Field r*; *B* ⊆ *Field r*⟧ ⟹
r `` *A* ⊆ *r* `` *B* ⟷ (∀ *a*∈*A*.∃ *b*∈*B*. (*b*,*a*):*r*)
⟨*proof*⟩

lemma *subset-Image1-Image1-iff*:
⟦ *Preorder r*; *a* : *Field r*; *b* : *Field r*⟧ ⟹ *r* `` {*a*} ⊆ *r* `` {*b*} ⟷ (*b*,*a*):*r*
⟨*proof*⟩

lemma *Refl-antisym-eq-Image1-Image1-iff*:
⟦*Refl r*; *antisym r*; *a*:*Field r*; *b*:*Field r*⟧ ⟹ *r* `` {*a*} = *r* `` {*b*} ⟷ *a*=*b*
⟨*proof*⟩

lemma *Partial-order-eq-Image1-Image1-iff*:
⟦*Partial-order r*; *a*:*Field r*; *b*:*Field r*⟧ ⟹ *r* `` {*a*} = *r* `` {*b*} ⟷ *a*=*b*
⟨*proof*⟩

31.3 Orders on a type

abbreviation *strict-linear-order* ≡ *strict-linear-order-on UNIV*

abbreviation *linear-order* ≡ *linear-order-on UNIV*

abbreviation *well-order r* ≡ *well-order-on UNIV*

end

32 Zorn: Zorn's Lemma

theory *Zorn*
imports *Order-Relation*
begin

definition *chain-subset* :: *'a set set* ⇒ *bool* (*chain*$_{\subseteq}$) **where**
chain$_{\subseteq}$ *C* ≡ ∀ *A*∈*C*.∀ *B*∈*C*. *A* ⊆ *B* ∨ *B* ⊆ *A*

The lemma and section numbers refer to an unpublished article [**?**].

definition
chain :: *'a set set* => *'a set set set* **where**
chain S = {*F*. *F* ⊆ *S* & *chain*$_{\subseteq}$ *F*}

definition
super :: [*'a set set*,*'a set set*] => *'a set set set* **where**
super S c = {*d*. *d* ∈ *chain S* & *c* ⊂ *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
$succI:\ x \in TFin\ S ==> succ\ S\ x \in TFin\ S$
$\mid Pow\text{-}UnionI:\ Y \in Pow(TFin\ S) ==> Union(Y) \in TFin\ S$

32.1 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 $TFin\text{-}UnionI = TFin.Pow\text{-}UnionI\ [OF\ PowI]$

lemma *TFin-induct*:
assumes $H: n \in TFin\ S$
and $I: !!x.\ x \in TFin\ S ==> P\ x ==> P\ (succ\ S\ x)$
$!!Y.\ Y \subseteq TFin\ S ==> Ball\ Y\ P ==> P(Union\ Y)$
shows $P\ n$ ⟨*proof*⟩

lemma *succ-trans*: $x \subseteq y ==> x \subseteq succ\ S\ y$
⟨*proof*⟩

Lemma 1 of section 3.1

lemma *TFin-linear-lemma1*:
$[|\ n \in TFin\ S;\ m \in TFin\ S;$
$\forall x \in TFin\ S.\ x \subseteq m --> x = m \mid succ\ S\ x \subseteq m$
$|] ==> n \subseteq m \mid succ\ S\ m \subseteq n$
⟨*proof*⟩

Lemma 2 of section 3.2

lemma *TFin-linear-lemma2*:
$m \in TFin\ S ==> \forall n \in TFin\ S.\ n \subseteq m --> n=m \mid succ\ S\ n \subseteq m$
⟨*proof*⟩

Re-ordering the premises of Lemma 2

lemma *TFin-subsetD*:
[| $n \subseteq m$; $m \in TFin\ S$; $n \in TFin\ S$ |] ==> $n=m \mid succ\ S\ n \subseteq m$
⟨*proof*⟩

Consequences from section 3.3 – Property 3.2, the ordering is total

lemma *TFin-subset-linear*: [| $m \in TFin\ S$; $n \in TFin\ S$|] ==> $n \subseteq m \mid m \subseteq n$
⟨*proof*⟩

Lemma 3 of section 3.3

lemma *eq-succ-upper*: [| $n \in TFin\ S$; $m \in TFin\ S$; $m = succ\ S\ m$ |] ==> $n \subseteq m$
⟨*proof*⟩

Property 3.3 of section 3.3

lemma *equal-succ-Union*: $m \in TFin\ S$ ==> $(m = succ\ S\ m) = (m = Union(TFin\ S))$
⟨*proof*⟩

32.2 Hausdorff's Theorem: Every Set Contains a Maximal Chain.

NB: We assume the partial ordering is $\subseteq$, the subset relation!

lemma *empty-set-mem-chain*: $(\{\} :: 'a\ set\ set) \in chain\ S$
⟨*proof*⟩

lemma *super-subset-chain*: $super\ S\ c \subseteq chain\ S$
⟨*proof*⟩

lemma *maxchain-subset-chain*: $maxchain\ S \subseteq chain\ S$
⟨*proof*⟩

lemma *mem-super-Ex*: $c \in chain\ S - maxchain\ S$ ==> $EX\ d.\ d \in super\ S\ c$
⟨*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*: $c \in TFin\ S \Longrightarrow (c :: 'a\ set\ set)$: *chain S*
⟨*proof*⟩

theorem *Hausdorff*: $\exists\, c.\ (c :: 'a\ set\ set)$: *maxchain S*
⟨*proof*⟩

32.3 Zorn's Lemma: If All Chains Have Upper Bounds Then There Is a Maximal Element

lemma *chain-extend*:
$[|\ c \in chain\ S;\ z \in S;\ \forall x \in c.\ x \subseteq (z :: 'a\ set)\ |] \Longrightarrow \{z\}\ Un\ c \in chain\ S$
⟨*proof*⟩

lemma *chain-Union-upper*: $[|\ c \in chain\ S;\ x \in c\ |] \Longrightarrow x \subseteq Union(c)$
⟨*proof*⟩

lemma *chain-ball-Union-upper*: $c \in chain\ S \Longrightarrow \forall x \in c.\ x \subseteq Union(c)$
⟨*proof*⟩

lemma *maxchain-Zorn*:
$[|\ c \in maxchain\ S;\ u \in S;\ Union(c) \subseteq u\ |] \Longrightarrow Union(c) = u$
⟨*proof*⟩

theorem *Zorn-Lemma*:
$\forall c \in chain\ S.\ Union(c)$: $S \Longrightarrow \exists y \in S.\ \forall z \in S.\ y \subseteq z \longrightarrow y = z$
⟨*proof*⟩

32.4 Alternative version of Zorn's Lemma

lemma *Zorn-Lemma2*:
$\forall c \in chain\ S.\ \exists y \in S.\ \forall x \in c.\ x \subseteq y$
$\Longrightarrow \exists y \in S.\ \forall x \in S.\ (y :: 'a\ set) \subseteq x \longrightarrow y = x$
⟨*proof*⟩

Various other lemmas

lemma *chainD*: $[|\ c \in chain\ S;\ x \in c;\ y \in c\ |] \Longrightarrow x \subseteq y \mid y \subseteq x$
⟨*proof*⟩

lemma *chainD2*: $!!(c :: 'a\ set\ set).\ c \in chain\ S \Longrightarrow c \subseteq S$
⟨*proof*⟩

definition *Chain* :: $('a * 'a)set \Rightarrow 'a\ set\ set$ **where**
$Chain\ r \equiv \{A.\ \forall a{\in}A. \forall b{\in}A.\ (a,b) : r \vee (b,a) \in r\}$

lemma *mono-Chain*: $r \subseteq s \Longrightarrow Chain\ r \subseteq Chain\ s$
⟨*proof*⟩

Zorn's lemma for partial orders:

lemma *Zorns-po-lemma*:
assumes *po*: *Partial-order r* **and** *u*: $\forall\, C{\in}Chain\ r.\ \exists\, u{\in}Field\ r.\ \forall\, a{\in}C.\ (a,u){:}r$
shows $\exists\, m{\in}Field\ r.\ \forall\, a{\in}Field\ r.\ (m,a){:}r \longrightarrow a{=}m$
⟨*proof*⟩

definition *init-seg-of* :: $((\prime a{*}\prime a)set * (\prime a{*}\prime a)set)set$ **where**
init-seg-of == $\{(r,s).\ r \subseteq s \wedge (\forall\, a\ b\ c.\ (a,b){:}s \wedge (b,c){:}r \longrightarrow (a,b){:}r)\}$

abbreviation *initialSegmentOf* :: $(\prime a{*}\prime a)set \Rightarrow (\prime a{*}\prime a)set \Rightarrow bool$
(**infix** *initial′-segment′-of 55*) **where**
r initial-segment-of s == $(r,s){:}$*init-seg-of*

lemma *refl-init-seg-of*[*simp*]: *r initial-segment-of r*
⟨*proof*⟩

lemma *trans-init-seg-of*:
r initial-segment-of s $\Longrightarrow$ *s initial-segment-of t* $\Longrightarrow$ *r initial-segment-of t*
⟨*proof*⟩

lemma *antisym-init-seg-of*:
r initial-segment-of s $\Longrightarrow$ *s initial-segment-of r* $\Longrightarrow$ $r{=}s$
⟨*proof*⟩

lemma *Chain-init-seg-of-Union*:
$R \in$ *Chain init-seg-of* $\Longrightarrow$ $r{\in}R \Longrightarrow$ *r initial-segment-of* $\bigcup R$
⟨*proof*⟩

lemma *chain-subset-trans-Union*:
$chain_{\subseteq}\ R \Longrightarrow \forall\, r{\in}R.\ trans\ r \Longrightarrow trans(\bigcup R)$
⟨*proof*⟩

lemma *chain-subset-antisym-Union*:
$chain_{\subseteq}\ R \Longrightarrow \forall\, r{\in}R.\ antisym\ r \Longrightarrow antisym(\bigcup R)$
⟨*proof*⟩

lemma *chain-subset-Total-Union*:
assumes $chain_{\subseteq}\ R\ \forall\, r{\in}R.\ Total\ r$
shows *Total* $(\bigcup R)$
⟨*proof*⟩

lemma *wf-Union-wf-init-segs*:
assumes $R \in$ *Chain init-seg-of* **and** $\forall\, r{\in}R.$ *wf r* **shows** *wf*$(\bigcup R)$
⟨*proof*⟩

lemma *Chain-inits-DiffI*:
$R \in$ *Chain init-seg-of* $\Longrightarrow$ $\{r - s\ |r.\ r \in R\} \in$ *Chain init-seg-of*
⟨*proof*⟩

theorem *well-ordering*: $\exists r::('a*'a)set.\ Well\text{-}order\ r \wedge Field\ r = UNIV$
⟨*proof*⟩

corollary *well-order-on*: $\exists r::('a*'a)set.\ well\text{-}order\text{-}on\ A\ r$
⟨*proof*⟩

end

33 Filter: Filters and Ultrafilters

theory *Filter*
imports ~~/*src/HOL/Library/Zorn* ~~/*src/HOL/Library/Infinite-Set*
begin

33.1 Definitions and basic properties

33.1.1 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*⟩

33.1.2 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*: $(x \cup y \in F) = (x \in F \vee y \in F)$
⟨*proof*⟩

33.1.3 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*]: $(\mathit{insert}\ x\ A \in F) = (A \in F)$
⟨*proof*⟩

lemma (**in** *freeultrafilter*) *filter*: *filter* F ⟨*proof*⟩

lemma (**in** *freeultrafilter*) *ultrafilter*: *ultrafilter* F
⟨*proof*⟩

33.2 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*:
$(\{n.\ P\ n \wedge Q\ n\} \in F) = (\{n.\ P\ n\} \in F \wedge \{n.\ Q\ n\} \in F)$
⟨*proof*⟩

lemma (**in** *ultrafilter*) *Collect-not*:
$(\{n.\ \neg\ P\ n\} \in F) = (\{n.\ P\ n\} \notin F)$
⟨*proof*⟩

lemma (**in** *ultrafilter*) *Collect-disj*:
$(\{n.\ P\ n \vee Q\ n\} \in F) = (\{n.\ P\ n\} \in F \vee \{n.\ Q\ n\} \in F)$
⟨*proof*⟩

lemma (**in** *ultrafilter*) *Collect-all*:
$(\{n.\ \forall x.\ P\ n\ x\} \in F) = (\forall X.\ \{n.\ P\ n\ (X\ n)\} \in F)$
⟨*proof*⟩

33.3 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*⟩

33.4 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* ≡ {*A*. *finite* (− *A*)}
  and *superfrechet-def*: *superfrechet* ≡ {*G*. *filter G* ∧ *frechet* ⊆ *G*}

lemma (**in** *UFT*) *superfrechetI*:
  [*filter G*; *frechet* ⊆ *G*] ⟹ *G* ∈ *superfrechet*
⟨*proof*⟩

lemma (**in** *UFT*) *superfrechetD1*:
  G ∈ *superfrechet* ⟹ *filter G*
⟨*proof*⟩

lemma (**in** *UFT*) *superfrechetD2*:
$G \in \mathit{superfrechet} \Longrightarrow \mathit{frechet} \subseteq G$
⟨*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*: $\mathit{frechet} \in \mathit{superfrechet}$
⟨*proof*⟩

lemma (**in** *UFT*) *lemma-mem-chain-filter*:
$[\![c \in \mathit{chain\ superfrechet};\ x \in c]\!] \Longrightarrow \mathit{filter}\ x$
⟨*proof*⟩

33.4.1 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*:
$[\![c \in \mathit{chain\ superfrechet};\ c \neq \{\}]\!] \Longrightarrow \mathit{UNIV} \in \bigcup c$
⟨*proof*⟩

lemma (**in** *UFT*) *Union-chain-empty*:
$c \in \mathit{chain\ superfrechet} \Longrightarrow \{\} \notin \bigcup c$
⟨*proof*⟩

lemma (**in** *UFT*) *Union-chain-Int*:
$[\![c \in \mathit{chain\ superfrechet};\ u \in \bigcup c;\ v \in \bigcup c]\!] \Longrightarrow u \cap v \in \bigcup c$
⟨*proof*⟩

lemma (**in** *UFT*) *Union-chain-subset*:
$[\![c \in \mathit{chain\ superfrechet};\ u \in \bigcup c;\ u \subseteq v]\!] \Longrightarrow v \in \bigcup c$
⟨*proof*⟩

lemma (**in** *UFT*) *Union-chain-filter*:
assumes *chain*: $c \in \mathit{chain\ superfrechet}$ **and** *nonempty*: $c \neq \{\}$

shows *filter* $(\bigcup c)$
⟨*proof*⟩

lemma (**in** *UFT*) *lemma-mem-chain-frechet-subset*:
$[\![c \in \mathit{chain}\ \mathit{superfrechet};\ x \in c]\!] \Longrightarrow \mathit{frechet} \subseteq x$
⟨*proof*⟩

lemma (**in** *UFT*) *Union-chain-superfrechet*:
$[\![c \neq \{\};\ c \in \mathit{chain}\ \mathit{superfrechet}]\!] \Longrightarrow \bigcup c \in \mathit{superfrechet}$
⟨*proof*⟩

33.4.2 Existence of free ultrafilter

lemma (**in** *UFT*) *max-cofinite-filter-Ex*:
$\exists\, U{\in}\mathit{superfrechet}.\ \forall\, G{\in}\mathit{superfrechet}.\ U \subseteq G \longrightarrow U = G$
⟨*proof*⟩

lemma (**in** *UFT*) *mem-superfrechet-all-infinite*:
$[\![U \in \mathit{superfrechet};\ A \in U]\!] \Longrightarrow \mathit{infinite}\ A$
⟨*proof*⟩

There exists a free ultrafilter on any infinite set

lemma (**in** *UFT*) *freeultrafilter-ex*:
$\exists\, U{::}'a\ \mathit{set}\ \mathit{set}.\ \mathit{freeultrafilter}\ U$
⟨*proof*⟩

lemmas *freeultrafilter-Ex* = *UFT.freeultrafilter-ex*

hide (**open**) *const filter*

end

34 StarDef: Construction of Star Types Using Ultrafilters

theory *StarDef*
imports *Filter*
uses (*transfer.ML*)
begin

34.1 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*} ∈ $\mathcal{U}$; {*n*. *P n* ⟶ *Q n*} ∈ $\mathcal{U}$⟧ ⟹ {*n*. *Q n*} ∈ $\mathcal{U}$
⟨*proof*⟩

34.2 Definition of *star* type constructor

definition
starrel :: (('*nat* ⇒ '*a*) × ('*nat* ⇒ '*a*)) *set* **where**
starrel = {(*X*, *Y*). {*n*. *X n* = *Y n*} ∈ $\mathcal{U}$}

typedef '*a star* = (*UNIV* :: ('*nat* ⇒ '*a*) *set*) // *starrel*
⟨*proof*⟩

definition
star-n :: ('*nat* ⇒ '*a*) ⇒ '*a star* **where**
star-n X = *Abs-star* (*starrel* `` {*X*})

theorem *star-cases* [*case-names star-n*, *cases type*: *star*]:
(⋀*X*. *x* = *star-n X* ⟹ *P*) ⟹ *P*
⟨*proof*⟩

lemma *all-star-eq*: (∀ *x*. *P x*) = (∀ *X*. *P* (*star-n X*))
⟨*proof*⟩

lemma *ex-star-eq*: (∃ *x*. *P x*) = (∃ *X*. *P* (*star-n X*))
⟨*proof*⟩

Proving that *starrel* is an equivalence relation

lemma *starrel-iff* [*iff*]: ((*X*, *Y*) ∈ *starrel*) = ({*n*. *X n* = *Y n*} ∈ $\mathcal{U}$)
⟨*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*} ∈ *star*
⟨*proof*⟩

lemma *star-n-eq-iff*: (*star-n X* = *star-n Y*) = ({*n*. *X n* = *Y n*} ∈ $\mathcal{U}$)
⟨*proof*⟩

34.3 Transfer principle

This introduction rule starts each transfer proof.

lemma *transfer-start*:
$P \equiv \{n.\ Q\} \in \mathcal{U} \Longrightarrow \mathit{Trueprop}\ P \equiv \mathit{Trueprop}\ Q$
⟨*proof*⟩

Initialize transfer tactic.

⟨*ML*⟩
Transfer introduction rules.

lemma *transfer-ex* [*transfer-intro*]:
$[\![\bigwedge X.\ p\ (\mathit{star\text{-}n}\ X) \equiv \{n.\ P\ n\ (X\ n)\} \in \mathcal{U}]\!]$
$\Longrightarrow \exists x{::}'a\ \mathit{star}.\ p\ x \equiv \{n.\ \exists x.\ P\ n\ x\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-all* [*transfer-intro*]:
$[\![\bigwedge X.\ p\ (\mathit{star\text{-}n}\ X) \equiv \{n.\ P\ n\ (X\ n)\} \in \mathcal{U}]\!]$
$\Longrightarrow \forall x{::}'a\ \mathit{star}.\ p\ x \equiv \{n.\ \forall x.\ P\ n\ x\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-not* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U}]\!] \Longrightarrow \neg\ p \equiv \{n.\ \neg\ P\ n\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-conj* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ q \equiv \{n.\ Q\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow p \wedge q \equiv \{n.\ P\ n \wedge Q\ n\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-disj* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ q \equiv \{n.\ Q\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow p \vee q \equiv \{n.\ P\ n \vee Q\ n\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-imp* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ q \equiv \{n.\ Q\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow p \longrightarrow q \equiv \{n.\ P\ n \longrightarrow Q\ n\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-iff* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ q \equiv \{n.\ Q\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow p = q \equiv \{n.\ P\ n = Q\ n\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-if-bool* [*transfer-intro*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ x \equiv \{n.\ X\ n\} \in \mathcal{U};\ y \equiv \{n.\ Y\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow (\mathit{if}\ p\ \mathit{then}\ x\ \mathit{else}\ y) \equiv \{n.\ \mathit{if}\ P\ n\ \mathit{then}\ X\ n\ \mathit{else}\ Y\ n\} \in \mathcal{U}$
⟨*proof*⟩

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*⟩

34.4 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*⟩

34.5 Internal functions

definition
Ifun :: $('a \Rightarrow 'b)\ \textit{star} \Rightarrow 'a\ \textit{star} \Rightarrow 'b\ \textit{star}$ (- ⋆ - [*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 (λ*F X*. *starrel*‘‘{λ*n*. *F n* (*X n*)})
⟨*proof*⟩

lemma *Ifun-star-n*: *star-n F* ⋆ *star-n X* = *star-n* (λ*n*. *F n* (*X n*))
⟨*proof*⟩

Transfer tactic should remove occurrences of *Ifun*

⟨*ML*⟩

lemma *transfer-Ifun* [*transfer-intro*]:
⟦*f* ≡ *star-n F*; *x* ≡ *star-n X*⟧ ⟹ *f* ⋆ *x* ≡ *star-n* (λ*n*. *F n* (*X n*))
⟨*proof*⟩

lemma *Ifun-star-of* [*simp*]: *star-of f* ⋆ *star-of x* = *star-of* (*f x*)
⟨*proof*⟩

lemma *Standard-Ifun* [*simp*]:
⟦*f* ∈ *Standard*; *x* ∈ *Standard*⟧ ⟹ *f* ⋆ *x* ∈ *Standard*
⟨*proof*⟩

Nonstandard extensions of functions

definition
starfun :: (*'a* ⇒ *'b*) ⇒ (*'a star* ⇒ *'b star*) (∗*f*∗ - [*80*] *80*) **where**
starfun f == λ*x*. *star-of f* ⋆ *x*

definition
starfun2 :: (*'a* ⇒ *'b* ⇒ *'c*) ⇒ (*'a star* ⇒ *'b star* ⇒ *'c star*)
(∗*f2*∗ - [*80*] *80*) **where**
starfun2 f == λ*x y*. *star-of f* ⋆ *x* ⋆ *y*

declare *starfun-def* [*transfer-unfold*]
declare *starfun2-def* [*transfer-unfold*]

lemma *starfun-star-n*: (∗*f*∗ *f*) (*star-n X*) = *star-n* (λ*n*. *f* (*X n*))
⟨*proof*⟩

lemma *starfun2-star-n*:
(∗*f2*∗ *f*) (*star-n X*) (*star-n Y*) = *star-n* (λ*n*. *f* (*X n*) (*Y n*))
⟨*proof*⟩

lemma *starfun-star-of* [*simp*]: (∗*f*∗ *f*) (*star-of x*) = *star-of* (*f x*)
⟨*proof*⟩

lemma *starfun2-star-of* [*simp*]: (∗*f2*∗ *f*) (*star-of x*) = ∗*f*∗ *f x*
⟨*proof*⟩

lemma *Standard-starfun* [*simp*]: *x* ∈ *Standard* ⟹ *starfun f x* ∈ *Standard*

⟨*proof*⟩

lemma *Standard-starfun2* [*simp*]:
⟦$x \in$ *Standard*; $y \in$ *Standard*⟧ $\Longrightarrow$ *starfun2 f x y* $\in$ *Standard*
⟨*proof*⟩

lemma *Standard-starfun-iff*:
assumes *inj*: $\bigwedge x\ y.\ f\ x = f\ y \Longrightarrow x = y$
shows (*starfun f x* $\in$ *Standard*) = ($x \in$ *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 (*starfun2 f x y* $\in$ *Standard*) = ($x \in$ *Standard* $\wedge$ $y \in$ *Standard*)
⟨*proof*⟩

34.6 Internal predicates

definition
unstar :: *bool star* $\Rightarrow$ *bool* **where**
unstar b = (*b* = *star-of True*)

lemma *unstar-star-n*: *unstar* (*star-n P*) = ($\{n.\ P\ n\} \in \mathcal{U}$)
⟨*proof*⟩

lemma *unstar-star-of* [*simp*]: *unstar* (*star-of p*) = *p*
⟨*proof*⟩

Transfer tactic should remove occurrences of *unstar*

⟨*ML*⟩

lemma *transfer-unstar* [*transfer-intro*]:
$p \equiv$ *star-n P* $\Longrightarrow$ *unstar p* $\equiv \{n.\ P\ n\} \in \mathcal{U}$
⟨*proof*⟩

definition
starP :: ('*a* $\Rightarrow$ *bool*) $\Rightarrow$ '*a star* $\Rightarrow$ *bool* (∗*p*∗ - [*80*] *80*) **where**
∗*p*∗ *P* = ($\lambda x.$ *unstar* (*star-of P* $\star$ *x*))

definition
starP2 :: ('*a* $\Rightarrow$ '*b* $\Rightarrow$ *bool*) $\Rightarrow$ '*a star* $\Rightarrow$ '*b star* $\Rightarrow$ *bool* (∗*p2*∗ - [*80*] *80*) **where**
∗*p2*∗ *P* = ($\lambda x\ y.$ *unstar* (*star-of P* $\star$ *x* $\star$ *y*))

declare *starP-def* [*transfer-unfold*]
declare *starP2-def* [*transfer-unfold*]

lemma *starP-star-n*: (∗*p*∗ *P*) (*star-n X*) = ($\{n.\ P\ (X\ n)\} \in \mathcal{U}$)
⟨*proof*⟩

lemma *starP2-star-n*:
($*p2*$ *P*) (*star-n X*) (*star-n Y*) = ({*n*. *P* (*X n*) (*Y n*)} ∈ $\mathcal{U}$)
⟨*proof*⟩

lemma *starP-star-of* [*simp*]: ($*p*$ *P*) (*star-of x*) = *P x*
⟨*proof*⟩

lemma *starP2-star-of* [*simp*]: ($*p2*$ *P*) (*star-of x*) = $*p*$ *P x*
⟨*proof*⟩

34.7 Internal sets

definition
Iset :: *'a set star* ⇒ *'a star set* **where**
Iset A = {*x*. ($*p2*$ *op* ∈) *x A*}

lemma *Iset-star-n*:
(*star-n X* ∈ *Iset* (*star-n A*)) = ({*n*. *X n* ∈ *A n*} ∈ $\mathcal{U}$)
⟨*proof*⟩

Transfer tactic should remove occurrences of *Iset*

⟨*ML*⟩

lemma *transfer-mem* [*transfer-intro*]:
⟦*x* ≡ *star-n X*; *a* ≡ *Iset* (*star-n A*)⟧
⟹ *x* ∈ *a* ≡ {*n*. *X n* ∈ *A n*} ∈ $\mathcal{U}$
⟨*proof*⟩

lemma *transfer-Collect* [*transfer-intro*]:
⟦⋀*X*. *p* (*star-n X*) ≡ {*n*. *P n* (*X n*)} ∈ $\mathcal{U}$⟧
⟹ *Collect p* ≡ *Iset* (*star-n* (λ*n*. *Collect* (*P n*)))
⟨*proof*⟩

lemma *transfer-set-eq* [*transfer-intro*]:
⟦*a* ≡ *Iset* (*star-n A*); *b* ≡ *Iset* (*star-n B*)⟧
⟹ *a* = *b* ≡ {*n*. *A n* = *B n*} ∈ $\mathcal{U}$
⟨*proof*⟩

lemma *transfer-ball* [*transfer-intro*]:
⟦*a* ≡ *Iset* (*star-n A*); ⋀*X*. *p* (*star-n X*) ≡ {*n*. *P n* (*X n*)} ∈ $\mathcal{U}$⟧
⟹ ∀ *x*∈*a*. *p x* ≡ {*n*. ∀ *x*∈*A n*. *P n x*} ∈ $\mathcal{U}$
⟨*proof*⟩

lemma *transfer-bex* [*transfer-intro*]:
⟦*a* ≡ *Iset* (*star-n A*); ⋀*X*. *p* (*star-n X*) ≡ {*n*. *P n* (*X n*)} ∈ $\mathcal{U}$⟧
⟹ ∃ *x*∈*a*. *p x* ≡ {*n*. ∃ *x*∈*A n*. *P n x*} ∈ $\mathcal{U}$
⟨*proof*⟩

lemma *transfer-Iset* [*transfer-intro*]:

$[\![a \equiv \mathit{star\text{-}n}\ A]\!] \Longrightarrow \mathit{Iset}\ a \equiv \mathit{Iset}\ (\mathit{star\text{-}n}\ (\lambda n.\ A\ n))$
⟨*proof*⟩

Nonstandard extensions of sets.

definition
$\mathit{starset} :: 'a\ \mathit{set} \Rightarrow 'a\ \mathit{star}\ \mathit{set}\ (*s*\ \text{-}\ [80]\ 80)$ **where**
$\mathit{starset}\ A = \mathit{Iset}\ (\mathit{star\text{-}of}\ A)$

declare *starset-def* [*transfer-unfold*]

lemma *starset-mem*: $(\mathit{star\text{-}of}\ x \in *s*\ A) = (x \in A)$
⟨*proof*⟩

lemma *starset-UNIV*: $*s*\ (\mathit{UNIV}::'a\ \mathit{set}) = (\mathit{UNIV}::'a\ \mathit{star}\ \mathit{set})$
⟨*proof*⟩

lemma *starset-empty*: $*s*\ \{\} = \{\}$
⟨*proof*⟩

lemma *starset-insert*: $*s*\ (\mathit{insert}\ x\ A) = \mathit{insert}\ (\mathit{star\text{-}of}\ x)\ (\ *s*\ A)$
⟨*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*: $*s*\ (f\ `\ A) = (\ *f*\ f)\ `\ (\ *s*\ A)$
⟨*proof*⟩

lemma *starset-vimage*: $*s*\ (f\ -`\ A) = (\ *f*\ f)\ -`\ (\ *s*\ A)$
⟨*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*

34.8 Syntactic classes

instantiation *star* :: (*zero*) *zero*
begin

definition
star-zero-def: $0 \equiv$ *star-of 0*

instance ⟨*proof*⟩

end

instantiation *star* :: (*one*) *one*
begin

definition
star-one-def: $1 \equiv$ *star-of 1*

instance ⟨*proof*⟩

end

instantiation *star* :: (*plus*) *plus*
begin

definition
star-add-def: $(op +) \equiv *f2*\ (op +)$

instance ⟨*proof*⟩

end

instantiation *star* :: (*times*) *times*
begin

definition
star-mult-def: $(op *) \equiv *f2*\ (op *)$

instance ⟨*proof*⟩

end

instantiation *star* :: (*uminus*) *uminus*
begin

definition
star-minus-def: $uminus \equiv *f* uminus$

instance ⟨*proof*⟩

end

instantiation *star* :: (*minus*) *minus*
begin

definition
star-diff-def: $(op\ -) \equiv *f2* (op\ -)$

instance ⟨*proof*⟩

end

instantiation *star* :: (*abs*) *abs*
begin

definition
star-abs-def: $abs \equiv *f* abs$

instance ⟨*proof*⟩

end

instantiation *star* :: (*sgn*) *sgn*
begin

definition
star-sgn-def: $sgn \equiv *f* sgn$

instance ⟨*proof*⟩

end

instantiation *star* :: (*inverse*) *inverse*
begin

definition
star-divide-def: $(op\ /) \equiv *f2* (op\ /)$

definition
star-inverse-def: $inverse \equiv *f* inverse$

instance ⟨*proof*⟩

end

instantiation *star* :: (*number*) *number*
begin

definition
star-number-def: *number-of b* ≡ *star-of* (*number-of b*)

instance ⟨*proof*⟩

end

instantiation *star* :: (*Divides.div*) *Divides.div*
begin

definition
star-div-def: (*op div*) ≡ ∗*f2*∗ (*op div*)

definition
star-mod-def: (*op mod*) ≡ ∗*f2*∗ (*op mod*)

instance ⟨*proof*⟩

end

instantiation *star* :: (*power*) *power*
begin

definition
star-power-def: (*op* ^) ≡ λ*x n*. (∗*f*∗ (λ*x*. *x* ^ *n*)) *x*

instance ⟨*proof*⟩

end

instantiation *star* :: (*ord*) *ord*
begin

definition
star-le-def: (*op* ≤) ≡ ∗*p2*∗ (*op* ≤)

definition
star-less-def: (*op* <) ≡ ∗*p2*∗ (*op* <)

instance ⟨*proof*⟩

end

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-of* $(x + y) = star\text{-}of\ x + star\text{-}of\ y$
⟨*proof*⟩

lemma *star-of-diff*: *star-of* $(x - y) = star\text{-}of\ x - star\text{-}of\ y$
⟨*proof*⟩

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

lemma *star-of-mult*: *star-of* $(x * y) = star\text{-}of\ x * star\text{-}of\ y$
⟨*proof*⟩

lemma *star-of-divide*: *star-of* $(x\ /\ y) = star\text{-}of\ x\ /\ star\text{-}of\ y$
⟨*proof*⟩

lemma *star-of-inverse*: *star-of* $(inverse\ x) = inverse\ (star\text{-}of\ x)$
⟨*proof*⟩

lemma *star-of-div*: *star-of* $(x\ div\ y) = star\text{-}of\ x\ div\ star\text{-}of\ y$
⟨*proof*⟩

lemma *star-of-mod*: *star-of* $(x\ mod\ y) = star\text{-}of\ x\ mod\ star\text{-}of\ y$
⟨*proof*⟩

lemma *star-of-power*: *star-of* $(x\ \hat{}\ n) = star\text{-}of\ x\ \hat{}\ n$
⟨*proof*⟩

lemma *star-of-abs*: *star-of* $(abs\ x) = abs\ (star\text{-}of\ x)$
⟨*proof*⟩

star-of preserves numerals

lemma *star-of-zero*: *star-of 0 = 0*
⟨*proof*⟩

lemma *star-of-one*: *star-of 1 = 1*
⟨*proof*⟩

lemma *star-of-number-of*: *star-of* $(number\text{-}of\ x) = number\text{-}of\ x$
⟨*proof*⟩

star-of preserves orderings

lemma *star-of-less*: $(star\text{-}of\ x < star\text{-}of\ y) = (x < y)$
⟨*proof*⟩

lemma *star-of-le*: $(\mathit{star\text{-}of}\ x \leq \mathit{star\text{-}of}\ y) = (x \leq y)$
⟨*proof*⟩

lemma *star-of-eq*: $(\mathit{star\text{-}of}\ x = \mathit{star\text{-}of}\ y) = (x = y)$
⟨*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*

34.9 Ordering and lattice classes

instance *star* :: (*order*) *order*
⟨*proof*⟩

instantiation *star* :: (*lower-semilattice*) *lower-semilattice*
begin

definition
star-inf-def [*transfer-unfold*]: *inf* ≡ ∗*f2*∗ *inf*

instance
⟨*proof*⟩

end

instantiation *star* :: (*upper-semilattice*) *upper-semilattice*
begin

definition
star-sup-def [*transfer-unfold*]: *sup* ≡ ∗*f2*∗ *sup*

instance
⟨*proof*⟩

end

instance *star* :: (*lattice*) *lattice* ⟨*proof*⟩

instance *star* :: (*distrib-lattice*) *distrib-lattice*
⟨*proof*⟩

lemma *Standard-inf* [*simp*]:
⟦*x* ∈ *Standard*; *y* ∈ *Standard*⟧ ⟹ *inf x y* ∈ *Standard*
⟨*proof*⟩

lemma *Standard-sup* [*simp*]:
⟦*x* ∈ *Standard*; *y* ∈ *Standard*⟧ ⟹ *sup x y* ∈ *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* ∈ *Standard*; *y* ∈ *Standard*⟧ ⟹ *max x y* ∈ *Standard*
⟨*proof*⟩

lemma *Standard-min* [*simp*]:
⟦*x* ∈ *Standard*; *y* ∈ *Standard*⟧ ⟹ *min x y* ∈ *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*⟩

34.10 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*⟩

34.11 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*⟩

34.12 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*⟩

34.13 Number classes

lemma *star-of-nat-def* [*transfer-unfold*]: *of-nat n* = *star-of* (*of-nat n*)
⟨*proof*⟩

lemma *Standard-of-nat* [*simp*]: *of-nat n* ∈ *Standard*
⟨*proof*⟩

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

lemma *star-of-int-def* [*transfer-unfold*]: *of-int z* = *star-of* (*of-int z*)
⟨*proof*⟩

lemma *Standard-of-int* [*simp*]: *of-int z* ∈ *Standard*
⟨*proof*⟩

lemma *star-of-of-int* [*simp*]: *star-of* (*of-int z*) = *of-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*⟩

34.14 Finite class

lemma *starset-finite*: *finite A* ⟹ ∗*s*∗ *A* = *star-of* ' *A*
⟨*proof*⟩

instance *star* :: (*finite*) *finite*
⟨*proof*⟩

end

35 HyperNat: Hypernatural numbers

theory *HyperNat*
imports *StarDef*
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*

35.1 Properties Transferred from Naturals

lemma *hSuc-not-zero* [*iff*]: ⋀*m*. *hSuc m* ≠ *0*
⟨*proof*⟩

lemma *zero-not-hSuc* [*iff*]: ⋀*m*. *0* ≠ *hSuc m*
⟨*proof*⟩

lemma *hSuc-hSuc-eq* [*iff*]: ⋀*m n*. (*hSuc m* = *hSuc n*) = (*m* = *n*)
⟨*proof*⟩

lemma *zero-less-hSuc* [*iff*]: ⋀*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*. ~ *m*<*n* ==> *n*+(*m*−*n*) = (*m*::*hypnat*)
⟨*proof*⟩

lemma *hypnat-le-add-diff-inverse* [*simp*]: !!*m n*. *n* ≤ *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*⟩

35.2 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*⟩

35.3 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*⟩

35.3.1 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*: $x \in HNatInfinite \Longrightarrow 0 < x$
⟨*proof*⟩

lemma *one-less-HNatInfinite*: $x \in HNatInfinite \Longrightarrow 1 < x$
⟨*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*: $x \in HNatInfinite \Longrightarrow \exists y.\ x = y + (1::hypnat)$
⟨*proof*⟩

35.4 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* ∈ *HNatInfinite*
⟨*proof*⟩

lemma *lemma-unbounded-set* [*simp*]: {*n*::*nat*. *m* < *n*} ∈ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *Compl-Collect-le*: − {*n*::*nat*. *N* ≤ *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*. ∃ *N*. *n* = *hypnat-of-nat N*}
⟨*proof*⟩

lemma *Nats-less-whn*: *n* ∈ *Nats* ⟹ *n* < *whn*
⟨*proof*⟩

lemma *Nats-le-whn*: *n* ∈ *Nats* ⟹ *n* ≤ *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* ≤ *whn*
⟨*proof*⟩

lemma *hypnat-zero-less-hypnat-omega* [*simp*]: *0* < *whn*
⟨*proof*⟩

lemma *hypnat-one-less-hypnat-omega* [*simp*]: *1* < *whn*
⟨*proof*⟩

35.4.1 Alternative characterization of the set of infinite hypernaturals

HNatInfinite = {*N*. ∀ *n*∈ℕ. *n* < *N*}

lemma *HNatInfinite-FreeUltrafilterNat-lemma*:
∀ *N*::*nat*. {*n*. *f n* ≠ *N*} ∈ *FreeUltrafilterNat*
==> {*n*. *N* < *f n*} ∈ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *HNatInfinite-iff*: *HNatInfinite* = {*N*. ∀ *n* ∈ *Nats*. *n* < *N*}
⟨*proof*⟩

35.4.2 Alternative Characterization of *HNatInfinite* using Free Ultrafilter

lemma *HNatInfinite-FreeUltrafilterNat*:
star-n X ∈ *HNatInfinite* ==> ∀ *u*. {*n*. *u* < *X n*}: *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-HNatInfinite*:
∀ *u*. {*n*. *u* < *X n*}: *FreeUltrafilterNat* ==> *star-n X* ∈ *HNatInfinite*
⟨*proof*⟩

lemma *HNatInfinite-FreeUltrafilterNat-iff*:
(*star-n X* ∈ *HNatInfinite*) = (∀ *u*. {*n*. *u* < *X n*}: *FreeUltrafilterNat*)
⟨*proof*⟩

35.5 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*: ⋀*m*. *of-hypnat* (*hSuc m*) = *1* + *of-hypnat m*
⟨*proof*⟩

lemma *of-hypnat-add* [*simp*]:
⋀*m n*. *of-hypnat* (*m* + *n*) = *of-hypnat m* + *of-hypnat n*
⟨*proof*⟩

lemma *of-hypnat-mult* [*simp*]:
⋀*m n*. *of-hypnat* (*m* ∗ *n*) = *of-hypnat m* ∗ *of-hypnat n*
⟨*proof*⟩

lemma *of-hypnat-less-iff* [*simp*]:
⋀*m n*. (*of-hypnat m* < (*of-hypnat n*::*'a*::*ordered-semidom star*)) = (*m* < *n*)
⟨*proof*⟩

lemma *of-hypnat-0-less-iff* [*simp*]:
⋀*n*. (*0* < (*of-hypnat n*::*'a*::*ordered-semidom star*)) = (*0* < *n*)
⟨*proof*⟩

lemma *of-hypnat-less-0-iff* [*simp*]:
⋀*m*. ¬ (*of-hypnat m*::*'a*::*ordered-semidom star*) < *0*
⟨*proof*⟩

lemma *of-hypnat-le-iff* [*simp*]:
$\bigwedge m\ n.\ (of\text{-}hypnat\ m \leq (of\text{-}hypnat\ n::'a::ordered\text{-}semidom\ star)) = (m \leq n)$
⟨*proof*⟩

lemma *of-hypnat-0-le-iff* [*simp*]:
$\bigwedge n.\ 0 \leq (of\text{-}hypnat\ n::'a::ordered\text{-}semidom\ star)$
⟨*proof*⟩

lemma *of-hypnat-le-0-iff* [*simp*]:
$\bigwedge m.\ ((of\text{-}hypnat\ m::'a::ordered\text{-}semidom\ star) \leq 0) = (m = 0)$
⟨*proof*⟩

lemma *of-hypnat-eq-iff* [*simp*]:
$\bigwedge m\ n.\ (of\text{-}hypnat\ m = (of\text{-}hypnat\ n::'a::ordered\text{-}semidom\ star)) = (m = n)$
⟨*proof*⟩

lemma *of-hypnat-eq-0-iff* [*simp*]:
$\bigwedge m.\ ((of\text{-}hypnat\ m::'a::ordered\text{-}semidom\ star) = 0) = (m = 0)$
⟨*proof*⟩

lemma *HNatInfinite-of-hypnat-gt-zero*:
$N \in HNatInfinite \Longrightarrow (0::'a::ordered\text{-}semidom\ star) < of\text{-}hypnat\ N$
⟨*proof*⟩

end

36 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* (λ*n*. *real* (*Suc n*))

definition
epsilon :: *hypreal* **where**
— an infinitesimal number = [<*1*,*1*/*2*,*1*/*3*,...>]
epsilon = *star-n* (λ*n*. *inverse* (*real* (*Suc n*)))

notation (*xsymbols*)
omega (ω) **and**
epsilon (ε)

notation (*HTML* **output**)
omega (ω) **and**
epsilon (ε)

36.1 Real vector class instances

instantiation *star* :: (*scaleR*) *scaleR*
begin

definition
star-scaleR-def [*transfer-unfold*]: *scaleR r* ≡ ∗*f*∗ (*scaleR r*)

instance ⟨*proof*⟩

end

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*⟩

36.2 Injection from *hypreal*

definition
of-hypreal :: *hypreal* ⇒ 'a::*real-algebra-1 star* **where**
of-hypreal = *f* *of-real*

declare *of-hypreal-def* [*transfer-unfold*]

lemma *Standard-of-hypreal* [*simp*]:
r ∈ *Standard* ⟹ *of-hypreal r* ∈ *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*]:
⋀*x y*. *of-hypreal* (*x* + *y*) = *of-hypreal x* + *of-hypreal y*
⟨*proof*⟩

lemma *of-hypreal-minus* [*simp*]: ⋀*x*. *of-hypreal* (− *x*) = − *of-hypreal x*
⟨*proof*⟩

lemma *of-hypreal-diff* [*simp*]:
⋀*x y*. *of-hypreal* (*x* − *y*) = *of-hypreal x* − *of-hypreal y*
⟨*proof*⟩

lemma *of-hypreal-mult* [*simp*]:
⋀*x y*. *of-hypreal* (*x* * *y*) = *of-hypreal x* * *of-hypreal y*
⟨*proof*⟩

lemma *of-hypreal-inverse* [*simp*]:
⋀*x*. *of-hypreal* (*inverse x*) =
inverse (*of-hypreal x* :: 'a::{*real-div-algebra*,*division-by-zero*} *star*)
⟨*proof*⟩

lemma *of-hypreal-divide* [*simp*]:
$\bigwedge x\ y.\ \mathit{of\text{-}hypreal}\ (x\ /\ y) =$
$(\mathit{of\text{-}hypreal}\ x\ /\ \mathit{of\text{-}hypreal}\ y :: 'a{::}\{\mathit{real\text{-}field}, \mathit{division\text{-}by\text{-}zero}\}\ \mathit{star})$
⟨*proof*⟩

lemma *of-hypreal-eq-iff* [*simp*]:
$\bigwedge x\ y.\ (\mathit{of\text{-}hypreal}\ x = \mathit{of\text{-}hypreal}\ y) = (x = y)$
⟨*proof*⟩

lemma *of-hypreal-eq-0-iff* [*simp*]:
$\bigwedge x.\ (\mathit{of\text{-}hypreal}\ x = 0) = (x = 0)$
⟨*proof*⟩

36.3 Properties of *starrel*

lemma *lemma-starrel-refl* [*simp*]: $x \in \mathit{starrel}\ ``\ \{x\}$
⟨*proof*⟩

lemma *starrel-in-hypreal* [*simp*]: $\mathit{starrel}``\{x\}{:}\mathit{star}$
⟨*proof*⟩

declare *Abs-star-inject* [*simp*] *Abs-star-inverse* [*simp*]
declare *equiv-starrel* [*THEN eq-equiv-class-iff*, *simp*]

36.4 *hypreal-of-real*: the Injection from *real* to *hypreal*

lemma *inj-star-of*: *inj star-of*
⟨*proof*⟩

lemma *mem-Rep-star-iff*: $(X \in \mathit{Rep\text{-}star}\ x) = (x = \mathit{star\text{-}n}\ X)$
⟨*proof*⟩

lemma *Rep-star-star-n-iff* [*simp*]:
$(X \in \mathit{Rep\text{-}star}\ (\mathit{star\text{-}n}\ Y)) = (\{n.\ Y\ n = X\ n\} \in \mathcal{U})$
⟨*proof*⟩

lemma *Rep-star-star-n*: $X \in \mathit{Rep\text{-}star}\ (\mathit{star\text{-}n}\ X)$
⟨*proof*⟩

36.5 Properties of *star-n*

lemma *star-n-add*:
$\mathit{star\text{-}n}\ X + \mathit{star\text{-}n}\ Y = \mathit{star\text{-}n}\ (\%n.\ X\ n + Y\ n)$
⟨*proof*⟩

lemma *star-n-minus*:
$-\ \mathit{star\text{-}n}\ X = \mathit{star\text{-}n}\ (\%n.\ -(X\ n))$
⟨*proof*⟩

lemma *star-n-diff*:

star-n X − star-n Y = star-n (%n. X n − Y n)
⟨*proof*⟩

lemma *star-n-mult*:
star-n X ∗ star-n Y = star-n (%n. X n ∗ Y n)
⟨*proof*⟩

lemma *star-n-inverse*:
inverse (star-n X) = star-n (%n. inverse(X n))
⟨*proof*⟩

lemma *star-n-le*:
star-n X ≤ star-n Y =
({n. X n ≤ Y n} ∈ FreeUltrafilterNat)
⟨*proof*⟩

lemma *star-n-less*:
star-n X < star-n Y = ({n. X n < Y n} ∈ FreeUltrafilterNat)
⟨*proof*⟩

lemma *star-n-zero-num*: *0 = star-n (%n. 0)*
⟨*proof*⟩

lemma *star-n-one-num*: *1 = star-n (%n. 1)*
⟨*proof*⟩

lemma *star-n-abs*:
abs (star-n X) = star-n (%n. abs (X n))
⟨*proof*⟩

36.6 Misc Others

lemma *hypreal-not-refl2*: !!(*x*::*hypreal*). *x < y ==> x ≠ y*
⟨*proof*⟩

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

lemma *hypreal-mult-left-cancel*: (*c*::*hypreal*) ≠ *0 ==>* (*c∗a=c∗b*) = (*a=b*)
⟨*proof*⟩

lemma *hypreal-mult-right-cancel*: (*c*::*hypreal*) ≠ *0 ==>* (*a∗c=b∗c*) = (*a=b*)
⟨*proof*⟩

lemma *hypreal-omega-gt-zero* [*simp*]: *0 < omega*
⟨*proof*⟩

36.7 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*: $\{n::nat.\ x = real\ n\} = \{\}\ |$
$(\exists y.\ \{n::nat.\ x = real\ n\} = \{y\})$
⟨*proof*⟩

lemma *lemma-finite-omega-set*: *finite* $\{n::nat.\ x = real\ n\}$
⟨*proof*⟩

lemma *not-ex-hypreal-of-real-eq-omega*:
$\sim (\exists x.\ hypreal\text{-}of\text{-}real\ x = omega)$
⟨*proof*⟩

lemma *hypreal-of-real-not-eq-omega*: $hypreal\text{-}of\text{-}real\ x \neq omega$
⟨*proof*⟩

Existence of infinitesimal number also not corresponding to any real number

lemma *lemma-epsilon-empty-singleton-disj*:
$\{n::nat.\ x = inverse(real(Suc\ n))\} = \{\}\ |$
$(\exists y.\ \{n::nat.\ x = inverse(real(Suc\ n))\} = \{y\})$
⟨*proof*⟩

lemma *lemma-finite-epsilon-set*: *finite* $\{n.\ x = inverse(real(Suc\ n))\}$
⟨*proof*⟩

lemma *not-ex-hypreal-of-real-eq-epsilon*: $\sim (\exists x.\ hypreal\text{-}of\text{-}real\ x = epsilon)$
⟨*proof*⟩

lemma *hypreal-of-real-not-eq-epsilon*: $hypreal\text{-}of\text{-}real\ x \neq epsilon$
⟨*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*⟩

36.8 Absolute Value Function for the Hyperreals

lemma *hrabs-add-less*:
$[|\ abs\ x < r;\ abs\ y < s\ |] ==> abs(x+y) < r + (s::hypreal)$
⟨*proof*⟩

lemma *hrabs-less-gt-zero*: *abs x < r ==> (0::hypreal) < r*
⟨*proof*⟩

lemma *hrabs-disj*: *abs x = (x::'a::abs-if) | abs x = −x*
⟨*proof*⟩

lemma *hrabs-add-lemma-disj*: *(y::hypreal) + − x + (y + − z) = abs (x + − z) ==> y = z | x = y*
⟨*proof*⟩

36.9 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*⟩

36.10 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*]:
$(0::hypreal) \leq u \ \hat{}\ Suc\ (Suc\ 0) + v \ \hat{}\ Suc\ (Suc\ 0)$
⟨*proof*⟩

lemma *hrealpow-two-le-add-order2* [*simp*]:
$(0::hypreal) \leq u \ \hat{}\ Suc\ (Suc\ 0) + v \ \hat{}\ Suc\ (Suc\ 0) + w \ \hat{}\ Suc\ (Suc\ 0)$
⟨*proof*⟩

lemma *hypreal-add-nonneg-eq-0-iff*:
$[|\ 0 \leq x;\ 0 \leq y\ |] ==> (x+y = 0) = (x = 0\ \&\ y = (0::hypreal))$
⟨*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-of-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*]

36.11 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* ^) *R N*

lemma *Standard-hyperpow* [*simp*]:
⟦*r* ∈ *Standard*; *n* ∈ *Standard*⟧ ⟹ *r pow n* ∈ *Standard*
⟨*proof*⟩

lemma *hyperpow*: *star-n X pow star-n Y* = *star-n* (%*n*. *X n* ^ *Y n*)
⟨*proof*⟩

lemma *hyperpow-zero* [*simp*]:
⋀*n*. (*0*::*'a*::{*recpower*,*semiring-0*} *star*) *pow* (*n* + (*1*::*hypnat*)) = *0*
⟨*proof*⟩

lemma *hyperpow-not-zero*:
⋀*r n*. *r* ≠ (*0*::*'a*::{*recpower*,*field*} *star*) ==> *r pow n* ≠ *0*
⟨*proof*⟩

lemma *hyperpow-inverse*:
⋀*r n*. *r* ≠ (*0*::*'a*::{*recpower*,*division-by-zero*,*field*} *star*)
⟹ *inverse* (*r pow n*) = (*inverse r*) *pow n*
⟨*proof*⟩

lemma *hyperpow-hrabs*:
⋀*r n*. *abs* (*r*::*'a*::{*recpower*,*ordered-idom*} *star*) *pow n* = *abs* (*r pow n*)
⟨*proof*⟩

lemma *hyperpow-add*:
⋀*r n m*. (*r*::*'a*::*recpower star*) *pow* (*n* + *m*) = (*r pow n*) ∗ (*r pow m*)
⟨*proof*⟩

lemma *hyperpow-one* [*simp*]:
⋀*r*. (*r*::*'a*::*recpower star*) *pow* (*1*::*hypnat*) = *r*
⟨*proof*⟩

lemma *hyperpow-two*:
⋀*r*. (*r*::*'a*::*recpower star*) *pow* ((*1*::*hypnat*) + (*1*::*hypnat*)) = *r* ∗ *r*
⟨*proof*⟩

lemma *hyperpow-gt-zero*:
⋀*r n*. (*0*::*'a*::{*recpower*,*ordered-semidom*} *star*) < *r* ⟹ *0* < *r pow n*
⟨*proof*⟩

lemma *hyperpow-ge-zero*:
⋀*r n*. (*0*::*'a*::{*recpower*,*ordered-semidom*} *star*) ≤ *r* ⟹ *0* ≤ *r pow n*

⟨*proof*⟩

lemma *hyperpow-le*:
⋀*x y n*. ⟦(*0*::*'a*::{*recpower*,*ordered-semidom*} *star*) < *x*; *x* ≤ *y*⟧
⟹ *x pow n* ≤ *y pow n*
⟨*proof*⟩

lemma *hyperpow-eq-one* [*simp*]:
⋀*n*. *1 pow n* = (*1*::*'a*::*recpower star*)
⟨*proof*⟩

lemma *hrabs-hyperpow-minus-one* [*simp*]:
⋀*n*. *abs*(−*1 pow n*) = (*1*::*'a*::{*number-ring*,*recpower*,*ordered-idom*} *star*)
⟨*proof*⟩

lemma *hyperpow-mult*:
⋀*r s n*. (*r* ∗ *s*::*'a*::{*comm-monoid-mult*,*recpower*} *star*) *pow n*
= (*r pow n*) ∗ (*s pow n*)
⟨*proof*⟩

lemma *hyperpow-two-le* [*simp*]:
(*0*::*'a*::{*recpower*,*ordered-ring-strict*} *star*) ≤ *r pow* (*1* + *1*)
⟨*proof*⟩

lemma *hrabs-hyperpow-two* [*simp*]:
abs(*x pow* (*1* + *1*)) =
(*x*::*'a*::{*recpower*,*ordered-ring-strict*} *star*) *pow* (*1* + *1*)
⟨*proof*⟩

lemma *hyperpow-two-hrabs* [*simp*]:
abs(*x*::*'a*::{*recpower*,*ordered-idom*} *star*) *pow* (*1* + *1*) = *x pow* (*1* + *1*)
⟨*proof*⟩

The precondition could be weakened to (*0*::*'a*) ≤ *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*:
⋀*r*::*'a*::{*recpower*,*ordered-semidom*} *star*. *1* < *r* ⟹ *1* < *r pow* (*1* + *1*)
⟨*proof*⟩

lemma *hyperpow-two-ge-one*:
⋀*r*::*'a*::{*recpower*,*ordered-semidom*} *star*. *1* ≤ *r* ⟹ *1* ≤ *r pow* (*1* + *1*)
⟨*proof*⟩

lemma *two-hyperpow-ge-one* [*simp*]: (*1*::*hypreal*) ≤ *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*) ≤ *r*; *r* ≤ *1*; *n* < *N*|] ==> *r pow N* ≤ *r pow n*
⟨*proof*⟩

lemma *hyperpow-SHNat-le*:
[| *0* ≤ *r*; *r* ≤ (*1*::*hypreal*); *N* ∈ *HNatInfinite* |]
==> *ALL n*: *Nats*. *r pow N* ≤ *r pow n*
⟨*proof*⟩

lemma *hyperpow-realpow*:
(*hypreal-of-real r*) *pow* (*hypnat-of-nat n*) = *hypreal-of-real* (*r* ^ *n*)
⟨*proof*⟩

lemma *hyperpow-SReal* [*simp*]:
(*hypreal-of-real r*) *pow* (*hypnat-of-nat n*) ∈ *Reals*
⟨*proof*⟩

lemma *hyperpow-zero-HNatInfinite* [*simp*]:
N ∈ *HNatInfinite* ==> (*0*::*hypreal*) *pow N* = *0*
⟨*proof*⟩

lemma *hyperpow-le-le*:
[| (*0*::*hypreal*) ≤ *r*; *r* ≤ *1*; *n* ≤ *N* |] ==> *r pow N* ≤ *r pow n*
⟨*proof*⟩

lemma *hyperpow-Suc-le-self2*:
[| (*0*::*hypreal*) ≤ *r*; *r* < *1* |] ==> *r pow* (*n* + (*1*::*hypnat*)) ≤ *r*
⟨*proof*⟩

lemma *hyperpow-hypnat-of-nat*: ⋀*x*. *x pow hypnat-of-nat n* = *x* ^ *n*
⟨*proof*⟩

lemma *of-hypreal-hyperpow*:
⋀*x n*. *of-hypreal* (*x pow n*) =
(*of-hypreal x*::'*a*::{*real-algebra-1*,*recpower*} *star*) *pow n*
⟨*proof*⟩

end

37 NSA: Infinite Numbers, Infinitesimals, Infinitely Close Relation

theory *NSA*

imports *HyperDef ../Real/RComplete*
begin

definition
hnorm :: *'a*::*norm star* ⇒ *real star* **where**
hnorm = *f* *norm*

definition
Infinitesimal :: (*'a*::*real-normed-vector*) *star set* **where**
Infinitesimal = {*x*. ∀ *r* ∈ *Reals*. 0 < *r* −−> *hnorm x* < *r*}

definition
HFinite :: (*'a*::*real-normed-vector*) *star set* **where**
HFinite = {*x*. ∃ *r* ∈ *Reals*. *hnorm x* < *r*}

definition
HInfinite :: (*'a*::*real-normed-vector*) *star set* **where**
HInfinite = {*x*. ∀ *r* ∈ *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*) ∈ *Infinitesimal*)

definition
st :: *hypreal* => *hypreal* **where**
— the standard part of a hyperreal
st = (%*x*. @*r*. *x* ∈ *HFinite* & *r* ∈ *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*) ∈ *HFinite*}

notation (*xsymbols*)
approx (**infixl** ≈ *50*)

notation (*HTML* **output**)
approx (**infixl** ≈ *50*)

lemma *SReal-def*: *Reals* == {*x*. ∃ *r*. *x* = *hypreal-of-real r*}
⟨*proof*⟩

37.1 Nonstandard Extension of the Norm Function

definition

scaleHR :: *real star* ⇒ *'a star* ⇒ *'a*::*real-normed-vector star* **where**
scaleHR = *starfun2 scaleR*

declare *hnorm-def* [*transfer-unfold*]
declare *scaleHR-def* [*transfer-unfold*]

lemma *Standard-hnorm* [*simp*]: $x \in \mathit{Standard} \Longrightarrow \mathit{hnorm}\ x \in \mathit{Standard}$
⟨*proof*⟩

lemma *star-of-norm* [*simp*]: $\mathit{star\text{-}of}\ (\mathit{norm}\ x) = \mathit{hnorm}\ (\mathit{star\text{-}of}\ x)$
⟨*proof*⟩

lemma *hnorm-ge-zero* [*simp*]:
$\bigwedge x::'a::\mathit{real\text{-}normed\text{-}vector\ star}.\ 0 \leq \mathit{hnorm}\ x$
⟨*proof*⟩

lemma *hnorm-eq-zero* [*simp*]:
$\bigwedge x::'a::\mathit{real\text{-}normed\text{-}vector\ star}.\ (\mathit{hnorm}\ x = 0) = (x = 0)$
⟨*proof*⟩

lemma *hnorm-triangle-ineq*:
$\bigwedge x\ y::'a::\mathit{real\text{-}normed\text{-}vector\ star}.\ \mathit{hnorm}\ (x + y) \leq \mathit{hnorm}\ x + \mathit{hnorm}\ y$
⟨*proof*⟩

lemma *hnorm-triangle-ineq3*:
$\bigwedge x\ y::'a::\mathit{real\text{-}normed\text{-}vector\ star}.\ |\mathit{hnorm}\ x - \mathit{hnorm}\ y| \leq \mathit{hnorm}\ (x - y)$
⟨*proof*⟩

lemma *hnorm-scaleR*:
$\bigwedge x::'a::\mathit{real\text{-}normed\text{-}vector\ star}.$
$\mathit{hnorm}\ (a *_R x) = |\mathit{star\text{-}of}\ a| * \mathit{hnorm}\ x$
⟨*proof*⟩

lemma *hnorm-scaleHR*:
$\bigwedge a\ (x::'a::\mathit{real\text{-}normed\text{-}vector\ star}).$
$\mathit{hnorm}\ (\mathit{scaleHR}\ a\ x) = |a| * \mathit{hnorm}\ x$
⟨*proof*⟩

lemma *hnorm-mult-ineq*:
$\bigwedge x\ y::'a::\mathit{real\text{-}normed\text{-}algebra\ star}.\ \mathit{hnorm}\ (x * y) \leq \mathit{hnorm}\ x * \mathit{hnorm}\ y$
⟨*proof*⟩

lemma *hnorm-mult*:
$\bigwedge x\ y::'a::\mathit{real\text{-}normed\text{-}div\text{-}algebra\ star}.\ \mathit{hnorm}\ (x * y) = \mathit{hnorm}\ x * \mathit{hnorm}\ y$
⟨*proof*⟩

lemma *hnorm-hyperpow*:
$\bigwedge (x::'a::\{\mathit{real\text{-}normed\text{-}div\text{-}algebra,recpower}\}\ \mathit{star})\ n.$
$\mathit{hnorm}\ (x\ \mathit{pow}\ n) = \mathit{hnorm}\ x\ \mathit{pow}\ n$

⟨*proof*⟩

lemma *hnorm-one* [*simp*]:
hnorm (*1*::'*a*::*real-normed-div-algebra star*) = *1*
⟨*proof*⟩

lemma *hnorm-zero* [*simp*]:
hnorm (*0*::'*a*::*real-normed-vector star*) = *0*
⟨*proof*⟩

lemma *zero-less-hnorm-iff* [*simp*]:
⋀*x*::'*a*::*real-normed-vector star*. (*0* < *hnorm x*) = (*x* ≠ *0*)
⟨*proof*⟩

lemma *hnorm-minus-cancel* [*simp*]:
⋀*x*::'*a*::*real-normed-vector star*. *hnorm* (− *x*) = *hnorm x*
⟨*proof*⟩

lemma *hnorm-minus-commute*:
⋀*a b*::'*a*::*real-normed-vector star*. *hnorm* (*a* − *b*) = *hnorm* (*b* − *a*)
⟨*proof*⟩

lemma *hnorm-triangle-ineq2*:
⋀*a b*::'*a*::*real-normed-vector star*. *hnorm a* − *hnorm b* ≤ *hnorm* (*a* − *b*)
⟨*proof*⟩

lemma *hnorm-triangle-ineq4*:
⋀*a b*::'*a*::*real-normed-vector star*. *hnorm* (*a* − *b*) ≤ *hnorm a* + *hnorm b*
⟨*proof*⟩

lemma *abs-hnorm-cancel* [*simp*]:
⋀*a*::'*a*::*real-normed-vector star*. |*hnorm a*| = *hnorm a*
⟨*proof*⟩

lemma *hnorm-of-hypreal* [*simp*]:
⋀*r*. *hnorm* (*of-hypreal r*::'*a*::*real-normed-algebra-1 star*) = |*r*|
⟨*proof*⟩

lemma *nonzero-hnorm-inverse*:
⋀*a*::'*a*::*real-normed-div-algebra star*.
a ≠ *0* ⟹ *hnorm* (*inverse a*) = *inverse* (*hnorm a*)
⟨*proof*⟩

lemma *hnorm-inverse*:
⋀*a*::'*a*::{*real-normed-div-algebra*,*division-by-zero*} *star*.
hnorm (*inverse a*) = *inverse* (*hnorm a*)
⟨*proof*⟩

lemma *hnorm-divide*:

$\bigwedge a\ b::'a::\{real\text{-}normed\text{-}field,division\text{-}by\text{-}zero\}\ star.$
$hnorm\ (a\ /\ b) = hnorm\ a\ /\ hnorm\ b$
⟨*proof*⟩

lemma *hypreal-hnorm-def* [*simp*]:
$\bigwedge r::hypreal.\ hnorm\ r \equiv |r|$
⟨*proof*⟩

lemma *hnorm-add-less*:
$\bigwedge (x::'a::real\text{-}normed\text{-}vector\ star)\ y\ r\ s.$
$[\![hnorm\ x < r;\ hnorm\ y < s]\!] \Longrightarrow hnorm\ (x + y) < r + s$
⟨*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*⟩

37.2 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*: $(x::hypreal) \in Reals \Longrightarrow abs\ x \in Reals$
⟨*proof*⟩

lemma *SReal-hypreal-of-real* [*simp*]: $hypreal\text{-}of\text{-}real\ x \in Reals$
⟨*proof*⟩

lemma *SReal-divide-number-of*: $r \in Reals \Longrightarrow r/(number\text{-}of\ w::hypreal) \in Reals$
⟨*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-of-real* ‘$(UNIV::real\ set) = Reals$
⟨*proof*⟩

lemma *inv-hypreal-of-real-image*: *inv hypreal-of-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*:
$P \subseteq Reals ==> ((\exists x \in P.\ y < x) =$
$(\exists X.\ hypreal\text{-}of\text{-}real\ X \in P\ \&\ y < hypreal\text{-}of\text{-}real\ X))$
⟨*proof*⟩

lemma *SReal-sup-lemma2*:
$[|\ P \subseteq Reals; \exists x.\ x \in P; \exists y \in Reals.\ \forall x \in P.\ x < y\ |]$
$==> (\exists X.\ X \in \{w.\ hypreal\text{-}of\text{-}real\ w \in P\})\ \&$
$(\exists Y.\ \forall X \in \{w.\ hypreal\text{-}of\text{-}real\ w \in P\}.\ X < Y)$
⟨*proof*⟩

37.3 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 $x\ y :: 'a::real\text{-}normed\text{-}algebra\ star$
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*]: $star\text{-}of\ x \in HFinite$
⟨*proof*⟩

lemma *SReal-subset-HFinite*: (*Reals*::*hypreal set*) ⊆ *HFinite*
⟨*proof*⟩

lemma *HFiniteD*: *x* ∈ *HFinite* ==> ∃ *t* ∈ *Reals*. *hnorm x* < *t*
⟨*proof*⟩

lemma *HFinite-hrabs-iff* [*iff*]: (*abs* (*x*::*hypreal*) ∈ *HFinite*) = (*x* ∈ *HFinite*)
⟨*proof*⟩

lemma *HFinite-hnorm-iff* [*iff*]:
(*hnorm* (*x*::*hypreal*) ∈ *HFinite*) = (*x* ∈ *HFinite*)
⟨*proof*⟩

lemma *HFinite-number-of* [*simp*]: *number-of w* ∈ *HFinite*
⟨*proof*⟩

lemma *HFinite-0* [*simp*]: *0* ∈ *HFinite*
⟨*proof*⟩

lemma *HFinite-1* [*simp*]: *1* ∈ *HFinite*
⟨*proof*⟩

lemma *hrealpow-HFinite*:
fixes *x* :: *'a*::{*real-normed-algebra*,*recpower*} *star*
shows *x* ∈ *HFinite* ==> *x* ^ *n* ∈ *HFinite*
⟨*proof*⟩

lemma *HFinite-bounded*:
[|(*x*::*hypreal*) ∈ *HFinite*; *y* ≤ *x*; *0* ≤ *y* |] ==> *y* ∈ *HFinite*
⟨*proof*⟩

37.4 Set of Infinitesimals is a Subring of the Hyperreals

lemma *InfinitesimalI*:
(⋀*r*. ⟦*r* ∈ ℝ; *0* < *r*⟧ ⟹ *hnorm x* < *r*) ⟹ *x* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *InfinitesimalD*:
x ∈ *Infinitesimal* ==> ∀ *r* ∈ *Reals*. *0* < *r* --> *hnorm x* < *r*
⟨*proof*⟩

lemma *InfinitesimalI2*:
(⋀*r*. *0* < *r* ⟹ *hnorm x* < *star-of r*) ⟹ *x* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *InfinitesimalD2*:

$[\![x \in \mathit{Infinitesimal};\ 0 < r]\!] \Longrightarrow \mathit{hnorm}\ x < \mathit{star\text{-}of}\ r$
⟨*proof*⟩

lemma *Infinitesimal-zero* [*iff*]: $0 \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *hypreal-sum-of-halves*: $x/(2{::}\mathit{hypreal}) + x/(2{::}\mathit{hypreal}) = x$
⟨*proof*⟩

lemma *Infinitesimal-add*:
$[\!|\ x \in \mathit{Infinitesimal};\ y \in \mathit{Infinitesimal}\ |\!] ==> (x+y) \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *Infinitesimal-minus-iff* [*simp*]: $(-x{:}\mathit{Infinitesimal}) = (x{:}\mathit{Infinitesimal})$
⟨*proof*⟩

lemma *Infinitesimal-hnorm-iff*:
$(\mathit{hnorm}\ x \in \mathit{Infinitesimal}) = (x \in \mathit{Infinitesimal})$
⟨*proof*⟩

lemma *Infinitesimal-hrabs-iff* [*iff*]:
$(\mathit{abs}\ (x{::}\mathit{hypreal}) \in \mathit{Infinitesimal}) = (x \in \mathit{Infinitesimal})$
⟨*proof*⟩

lemma *Infinitesimal-of-hypreal-iff* [*simp*]:
$((\mathit{of\text{-}hypreal}\ x{::}'a{::}\mathit{real\text{-}normed\text{-}algebra\text{-}1\ star}) \in \mathit{Infinitesimal}) =$
$(x \in \mathit{Infinitesimal})$
⟨*proof*⟩

lemma *Infinitesimal-diff*:
$[\!|\ x \in \mathit{Infinitesimal};\ \ y \in \mathit{Infinitesimal}\ |\!] ==> x-y \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *Infinitesimal-mult*:
fixes $x\ y :: 'a{::}\mathit{real\text{-}normed\text{-}algebra\ star}$
shows $[\!|x \in \mathit{Infinitesimal};\ y \in \mathit{Infinitesimal}|\!] ==> (x * y) \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *Infinitesimal-HFinite-mult*:
fixes $x\ y :: 'a{::}\mathit{real\text{-}normed\text{-}algebra\ star}$
shows $[\!|\ x \in \mathit{Infinitesimal};\ y \in \mathit{HFinite}\ |\!] ==> (x * y) \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *Infinitesimal-HFinite-scaleHR*:
$[\!|\ x \in \mathit{Infinitesimal};\ y \in \mathit{HFinite}\ |\!] ==> \mathit{scaleHR}\ x\ y \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *Infinitesimal-HFinite-mult2*:
fixes $x\ y :: 'a{::}\mathit{real\text{-}normed\text{-}algebra\ star}$

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) ∈ 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 ∈ HFinite
⟨proof⟩

lemma *not-Infinitesimal-not-zero*: x ∉ Infinitesimal ==> x ≠ 0
⟨proof⟩

lemma *not-Infinitesimal-not-zero2*: x ∈ HFinite − Infinitesimal ==> x ≠ 0
⟨proof⟩

lemma *HFinite-diff-Infinitesimal-hrabs*:
(x::hypreal) ∈ HFinite − Infinitesimal ==> abs x ∈ HFinite − Infinitesimal
⟨proof⟩

lemma *hnorm-le-Infinitesimal*:
⟦e ∈ Infinitesimal; hnorm x ≤ e⟧ ⟹ x ∈ Infinitesimal
⟨proof⟩

lemma *hnorm-less-Infinitesimal*:
⟦e ∈ Infinitesimal; hnorm x < e⟧ ⟹ x ∈ Infinitesimal
⟨proof⟩

lemma *hrabs-le-Infinitesimal*:
[| e ∈ Infinitesimal; abs (x::hypreal) ≤ e |] ==> x ∈ Infinitesimal
⟨proof⟩

lemma *hrabs-less-Infinitesimal*:
[| e ∈ Infinitesimal; abs (x::hypreal) < e |] ==> x ∈ Infinitesimal
⟨proof⟩

lemma *Infinitesimal-interval*:
[| e ∈ Infinitesimal; e' ∈ Infinitesimal; e' < x ; x < e |]
==> (x::hypreal) ∈ Infinitesimal
⟨proof⟩

lemma *Infinitesimal-interval2*:
[| e ∈ Infnitesimal; e' ∈ Infinitesimal;
e' ≤ x ; x ≤ e |] ==> (x::hypreal) ∈ Infinitesimal
⟨proof⟩

lemma *lemma-Infinitesimal-hyperpow*:
[| (x::hypreal) ∈ Infinitesimal; 0 < N |] ==> abs (x pow N) ≤ abs x
⟨proof⟩

lemma *Infinitesimal-hyperpow*:
[| (*x*::*hypreal*) ∈ *Infinitesimal*; *0* < *N* |] ==> *x pow N* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *hrealpow-hyperpow-Infinitesimal-iff*:
(*x* ^ *n* ∈ *Infinitesimal*) = (*x pow* (*hypnat-of-nat n*) ∈ *Infinitesimal*)
⟨*proof*⟩

lemma *Infinitesimal-hrealpow*:
[| (*x*::*hypreal*) ∈ *Infinitesimal*; *0* < *n* |] ==> *x* ^ *n* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *not-Infinitesimal-mult*:
fixes *x y* :: *'a*::*real-normed-div-algebra star*
shows [| *x* ∉ *Infinitesimal*; *y* ∉ *Infinitesimal*|] ==> (*x*∗*y*) ∉*Infinitesimal*
⟨*proof*⟩

lemma *Infinitesimal-mult-disj*:
fixes *x y* :: *'a*::*real-normed-div-algebra star*
shows *x*∗*y* ∈ *Infinitesimal* ==> *x* ∈ *Infinitesimal* | *y* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-Infinitesimal-not-zero*: *x* ∈ *HFinite*−*Infinitesimal* ==> *x* ≠ *0*
⟨*proof*⟩

lemma *HFinite-Infinitesimal-diff-mult*:
fixes *x y* :: *'a*::*real-normed-div-algebra star*
shows [| *x* ∈ *HFinite* − *Infinitesimal*;
y ∈ *HFinite* − *Infinitesimal*
|] ==> (*x*∗*y*) ∈ *HFinite* − *Infinitesimal*
⟨*proof*⟩

lemma *Infinitesimal-subset-HFinite*:
Infinitesimal ⊆ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-star-of-mult*:
fixes *x* :: *'a*::*real-normed-algebra star*
shows *x* ∈ *Infinitesimal* ==> *x* ∗ *star-of r* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *Infinitesimal-star-of-mult2*:
fixes *x* :: *'a*::*real-normed-algebra star*
shows *x* ∈ *Infinitesimal* ==> *star-of r* ∗ *x* ∈ *Infinitesimal*
⟨*proof*⟩

37.5 The Infinitely Close Relation

lemma *mem-infmal-iff*: $(x \in Infinitesimal) = (x \mathbin{@=} 0)$
⟨*proof*⟩

lemma *approx-minus-iff*: $(x \mathbin{@=} y) = (x - y \mathbin{@=} 0)$
⟨*proof*⟩

lemma *approx-minus-iff2*: $(x \mathbin{@=} y) = (-y + x \mathbin{@=} 0)$
⟨*proof*⟩

lemma *approx-refl* [*iff*]: $x \mathbin{@=} x$
⟨*proof*⟩

lemma *hypreal-minus-distrib1*: $-(y + -(x::'a::ab\text{-}group\text{-}add)) = x + -y$
⟨*proof*⟩

lemma *approx-sym*: $x \mathbin{@=} y \Longrightarrow y \mathbin{@=} x$
⟨*proof*⟩

lemma *approx-trans*: $[\!|\, x \mathbin{@=} y;\ y \mathbin{@=} z \,|\!] \Longrightarrow x \mathbin{@=} z$
⟨*proof*⟩

lemma *approx-trans2*: $[\!|\, r \mathbin{@=} x;\ s \mathbin{@=} x \,|\!] \Longrightarrow r \mathbin{@=} s$
⟨*proof*⟩

lemma *approx-trans3*: $[\!|\, x \mathbin{@=} r;\ x \mathbin{@=} s|\!] \Longrightarrow r \mathbin{@=} s$
⟨*proof*⟩

lemma *number-of-approx-reorient*: $(number\text{-}of\ w \mathbin{@=} x) = (x \mathbin{@=} number\text{-}of\ w)$
⟨*proof*⟩

lemma *zero-approx-reorient*: $(0 \mathbin{@=} x) = (x \mathbin{@=} 0)$
⟨*proof*⟩

lemma *one-approx-reorient*: $(1 \mathbin{@=} x) = (x \mathbin{@=} 1)$
⟨*proof*⟩

⟨*ML*⟩

lemma *Infinitesimal-approx-minus*: $(x-y \in Infinitesimal) = (x \mathbin{@=} y)$
⟨*proof*⟩

lemma *approx-monad-iff*: $(x \mathbin{@=} y) = (monad(x)=monad(y))$
⟨*proof*⟩

lemma *Infinitesimal-approx*:
$[\!|\, x \in Infinitesimal;\ y \in Infinitesimal \,|\!] \Longrightarrow x \mathbin{@=} y$
⟨*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 ∈ *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 ∈ *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 ∈ *Infinitesimal* ==> $-x$ @= x
⟨*proof*⟩

lemma *bex-Infinitesimal-iff*: $(\exists\, y \in \mathit{Infinitesimal}.\ x - z = y) = (x\ @{=}\ z)$
⟨*proof*⟩

lemma *bex-Infinitesimal-iff2*: $(\exists\, y \in \mathit{Infinitesimal}.\ x = z + y) = (x\ @{=}\ z)$
⟨*proof*⟩

lemma *Infinitesimal-add-approx*: $[|\ y \in \mathit{Infinitesimal};\ x + y = z\ |] ==> x\ @{=}\ z$
⟨*proof*⟩

lemma *Infinitesimal-add-approx-self*: $y \in \mathit{Infinitesimal} ==> x\ @{=}\ x + y$
⟨*proof*⟩

lemma *Infinitesimal-add-approx-self2*: $y \in \mathit{Infinitesimal} ==> x\ @{=}\ y + x$
⟨*proof*⟩

lemma *Infinitesimal-add-minus-approx-self*: $y \in \mathit{Infinitesimal} ==> x\ @{=}\ x + -y$
⟨*proof*⟩

lemma *Infinitesimal-add-cancel*: $[|\ y \in \mathit{Infinitesimal};\ x+y\ @{=}\ z|] ==> x\ @{=}\ z$
⟨*proof*⟩

lemma *Infinitesimal-add-right-cancel*:
$[|\ y \in \mathit{Infinitesimal};\ x\ @{=}\ z + y|] ==> x\ @{=}\ z$
⟨*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*: $[|\ x \in \mathit{HFinite};\ x\ @{=}\ y\ |] ==> y \in \mathit{HFinite}$
⟨*proof*⟩

lemma *approx-star-of-HFinite*: $x\ @{=}\ \mathit{star\text{-}of}\ D ==> x \in \mathit{HFinite}$
⟨*proof*⟩

lemma *approx-mult-HFinite*:
fixes *a b c d* :: *'a::real-normed-algebra star*
shows [|*a* @= *b*; *c* @= *d*; *b*: *HFinite*; *d*: *HFinite*|] ==> *a*∗*c* @= *b*∗*d*
⟨*proof*⟩

lemma *scaleHR-left-diff-distrib*:
⋀*a b x*. *scaleHR* (*a* − *b*) *x* = *scaleHR a x* − *scaleHR b x*
⟨*proof*⟩

lemma *approx-scaleR1*:
[| *a* @= *star-of b*; *c*: *HFinite*|] ==> *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* ≠ *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* ≠ *0* |] ==> (*a*∗*x* @= *0*) = (*x* @= *0*)
⟨*proof*⟩

lemma *approx-SReal-mult-cancel*:
[| (*a*::*hypreal*) ∈ *Reals*; *a* ≠ *0*; *a*∗ *w* @= *a*∗*z* |] ==> *w* @= *z*
⟨*proof*⟩

lemma *approx-SReal-mult-cancel-iff1* [*simp*]:
[| (*a*::*hypreal*) ∈ *Reals*; *a* ≠ *0*|] ==> (*a*∗ *w* @= *a*∗*z*) = (*w* @= *z*)
⟨*proof*⟩

lemma *approx-le-bound*: [| (*z*::*hypreal*) ≤ *f*; *f* @= *g*; *g* ≤ *z* |] ==> *f* @= *z*
⟨*proof*⟩

lemma *approx-hnorm*:
fixes *x y* :: 'a::*real-normed-vector star*
shows *x* ≈ *y* ⟹ *hnorm x* ≈ *hnorm y*
⟨*proof*⟩

37.6 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*) ∈ *Infinitesimal* ==> ∀ *r* ∈ *Reals*. *0* < *r* --> *y* < *r*
⟨*proof*⟩

lemma *SReal-not-Infinitesimal*:
[| *0* < *y*; (*y*::*hypreal*) ∈ *Reals*|] ==> *y* ∉ *Infinitesimal*
⟨*proof*⟩

lemma *SReal-minus-not-Infinitesimal*:
[| *y* < *0*; (*y*::*hypreal*) ∈ *Reals* |] ==> *y* ∉ *Infinitesimal*
⟨*proof*⟩

lemma *SReal-Int-Infinitesimal-zero*: *Reals Int Infinitesimal* = {*0*::*hypreal*}
⟨*proof*⟩

lemma *SReal-Infinitesimal-zero*:
[| (*x*::*hypreal*) ∈ *Reals*; *x* ∈ *Infinitesimal*|] ==> *x* = *0*
⟨*proof*⟩

lemma *SReal-HFinite-diff-Infinitesimal*:
[| (*x*::*hypreal*) ∈ *Reals*; *x* ≠ *0* |] ==> *x* ∈ *HFinite* − *Infinitesimal*
⟨*proof*⟩

lemma *hypreal-of-real-HFinite-diff-Infinitesimal*:
hypreal-of-real x ≠ *0* ==> *hypreal-of-real x* ∈ *HFinite* − *Infinitesimal*
⟨*proof*⟩

lemma *star-of-Infinitesimal-iff-0* [*iff*]:
(*star-of x* ∈ *Infinitesimal*) = (*x* = *0*)
⟨*proof*⟩

lemma *star-of-HFinite-diff-Infinitesimal*:
x ≠ *0* ==> *star-of x* ∈ *HFinite* − *Infinitesimal*
⟨*proof*⟩

lemma *number-of-not-Infinitesimal* [*simp*]:
number-of w ≠ (*0*::*hypreal*) ==> (*number-of w* :: *hypreal*) ∉ *Infinitesimal*
⟨*proof*⟩

lemma *one-not-Infinitesimal* [*simp*]:
(*1*::'*a*::{*real-normed-vector*,*zero-neq-one*} *star*) ∉ *Infinitesimal*
⟨*proof*⟩

lemma *approx-SReal-not-zero*:
[| (*y*::*hypreal*) ∈ *Reals*; *x* @= *y*; *y*≠ *0* |] ==> *x* ≠ *0*
⟨*proof*⟩

lemma *HFinite-diff-Infinitesimal-approx*:
[| *x* @= *y*; *y* ∈ *HFinite* − *Infinitesimal* |]
==> *x* ∈ *HFinite* − *Infinitesimal*
⟨*proof*⟩

lemma *Infinitesimal-ratio*:
fixes *x y* :: '*a*::{*real-normed-div-algebra*,*field*} *star*
shows [| *y* ≠ *0*; *y* ∈ *Infinitesimal*; *x*/*y* ∈ *HFinite* |]
==> *x* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *Infinitesimal-SReal-divide*:
[| (*x*::*hypreal*) ∈ *Infinitesimal*; *y* ∈ *Reals* |] ==> *x*/*y* ∈ *Infinitesimal*
⟨*proof*⟩

37.7 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*) ∈ *Reals*; *y* ∈ *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*⟩

37.8 Existence of Unique Real Infinitely Close

37.8.1 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*:
$isLub\ Reals\ P\ Y \Longrightarrow \exists\, Yo.\ isLub\ Reals\ P\ (hypreal\text{-}of\text{-}real\ Yo)$
⟨*proof*⟩

lemma *lemma-isLub-hypreal-of-real2*:
$\exists\, Yo.\ isLub\ Reals\ P\ (hypreal\text{-}of\text{-}real\ Yo) \Longrightarrow \exists\, Y.\ isLub\ Reals\ P\ Y$
⟨*proof*⟩

lemma *SReal-complete*:
$[|\ P \subseteq Reals;\ \exists\, x.\ x \in P;\ \exists\, Y.\ isUb\ Reals\ P\ Y\ |]$
$\Longrightarrow \exists\, t{::}hypreal.\ isLub\ Reals\ P\ t$
⟨*proof*⟩

lemma *hypreal-isLub-unique*:
$[|\ isLub\ R\ S\ x;\ isLub\ R\ S\ y\ |] \Longrightarrow x = (y{::}hypreal)$
⟨*proof*⟩

lemma *lemma-st-part-ub*:
$(x{::}hypreal) \in HFinite \Longrightarrow \exists\, u.\ isUb\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ u$
⟨*proof*⟩

lemma *lemma-st-part-nonempty*:
$(x{::}hypreal) \in HFinite \Longrightarrow \exists\, y.\ y \in \{s.\ s \in Reals\ \&\ s < x\}$
⟨*proof*⟩

lemma *lemma-st-part-subset*: $\{s.\ s \in Reals\ \&\ s < x\} \subseteq Reals$
⟨*proof*⟩

lemma *lemma-st-part-lub*:
$(x{::}hypreal) \in HFinite \Longrightarrow \exists\, t.\ isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t$
⟨*proof*⟩

lemma *lemma-hypreal-le-left-cancel*: $((t{::}hypreal) + r \leq t) = (r \leq 0)$
⟨*proof*⟩

lemma *lemma-st-part-le1*:
$[|\ (x{::}hypreal) \in HFinite;\ isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ 0 < r\ |] \Longrightarrow x \leq t + r$
⟨*proof*⟩

lemma *hypreal-setle-less-trans*:
$[|\ S *<= (x{::}hypreal);\ x < y\ |] \Longrightarrow S *<= y$
⟨*proof*⟩

lemma *hypreal-gt-isUb*:
$[|\ isUb\ R\ S\ (x{::}hypreal);\ x < y;\ y \in R\ |] \Longrightarrow isUb\ R\ S\ y$
⟨*proof*⟩

lemma *lemma-st-part-gt-ub*:
[| (*x*::*hypreal*) ∈ *HFinite*; *x* < *y*; *y* ∈ *Reals* |]
==> *isUb Reals* {*s*. *s* ∈ *Reals* & *s* < *x*} *y*
⟨*proof*⟩

lemma *lemma-minus-le-zero*: *t* ≤ *t* + −*r* ==> *r* ≤ (*0*::*hypreal*)
⟨*proof*⟩

lemma *lemma-st-part-le2*:
[| (*x*::*hypreal*) ∈ *HFinite*;
isLub Reals {*s*. *s* ∈ *Reals* & *s* < *x*} *t*;
r ∈ *Reals*; *0* < *r* |]
==> *t* + −*r* ≤ *x*
⟨*proof*⟩

lemma *lemma-st-part1a*:
[| (*x*::*hypreal*) ∈ *HFinite*;
isLub Reals {*s*. *s* ∈ *Reals* & *s* < *x*} *t*;
r ∈ *Reals*; *0* < *r* |]
==> *x* + −*t* ≤ *r*
⟨*proof*⟩

lemma *lemma-st-part2a*:
[| (*x*::*hypreal*) ∈ *HFinite*;
isLub Reals {*s*. *s* ∈ *Reals* & *s* < *x*} *t*;
r ∈ *Reals*; *0* < *r* |]
==> −(*x* + −*t*) ≤ *r*
⟨*proof*⟩

lemma *lemma-SReal-ub*:
(*x*::*hypreal*) ∈ *Reals* ==> *isUb Reals* {*s*. *s* ∈ *Reals* & *s* < *x*} *x*
⟨*proof*⟩

lemma *lemma-SReal-lub*:
(*x*::*hypreal*) ∈ *Reals* ==> *isLub Reals* {*s*. *s* ∈ *Reals* & *s* < *x*} *x*
⟨*proof*⟩

lemma *lemma-st-part-not-eq1*:
[| (*x*::*hypreal*) ∈ *HFinite*;
isLub Reals {*s*. *s* ∈ *Reals* & *s* < *x*} *t*;
r ∈ *Reals*; *0* < *r* |]
==> *x* + −*t* ≠ *r*
⟨*proof*⟩

lemma *lemma-st-part-not-eq2*:
[| (*x*::*hypreal*) ∈ *HFinite*;
isLub Reals {*s*. *s* ∈ *Reals* & *s* < *x*} *t*;
r ∈ *Reals*; *0* < *r* |]

$==> -(x + -t) \neq r$
⟨*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*⟩

37.9 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*:
$x \notin Infinitesimal \Longrightarrow x \in HInfinite \mid x \in HFinite - Infinitesimal$
⟨*proof*⟩

lemma *HFinite-inverse*:
fixes $x :: 'a$::*real-normed-div-algebra star*
shows $[\!| x \in HFinite; x \notin Infinitesimal |\!] \Longrightarrow inverse\ x \in HFinite$
⟨*proof*⟩

lemma *HFinite-inverse2*:
fixes $x :: 'a$::*real-normed-div-algebra star*
shows $x \in HFinite - Infinitesimal \Longrightarrow inverse\ x \in HFinite$
⟨*proof*⟩

lemma *Infinitesimal-inverse-HFinite*:
fixes $x :: 'a$::*real-normed-div-algebra star*
shows $x \notin Infinitesimal \Longrightarrow inverse(x) \in HFinite$
⟨*proof*⟩

lemma *HFinite-not-Infinitesimal-inverse*:
fixes $x :: 'a$::*real-normed-div-algebra star*
shows $x \in HFinite - Infinitesimal \Longrightarrow inverse\ x \in HFinite - Infinitesimal$
⟨*proof*⟩

lemma *approx-inverse*:
fixes $x\ y :: 'a$::*real-normed-div-algebra star*
shows
$[\!| x \mathbin{@=} y; y \in HFinite - Infinitesimal |\!] \Longrightarrow inverse\ x \mathbin{@=} inverse\ y$
⟨*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
$[\!| x \in HFinite - Infinitesimal; h \in Infinitesimal |\!] \Longrightarrow inverse(x + h) \mathbin{@=} inverse\ x$
⟨*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*$) \in HInfinite$; $x \leq y$; $0 \leq x$ |] ==> $y \in HInfinite$
⟨*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 | *abs* x @= $-x$
⟨*proof*⟩

37.10 Theorems about Monads

lemma *monad-hrabs-Un-subset*: *monad* (*abs* x) $\leq$ *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*⟩

37.11 Proof that $x \approx y$ implies $|x| \approx |y|$

lemma *approx-subset-monad*: $x\ @=\ y ==> \{x,y\} \leq monad\ x$
⟨*proof*⟩

lemma *approx-subset-monad2*: $x\ @=\ y ==> \{x,y\} \leq monad\ y$
⟨*proof*⟩

lemma *mem-monad-approx*: $u \in monad\ x ==> x\ @=\ u$
⟨*proof*⟩

lemma *approx-mem-monad*: $x\ @=\ u ==> u \in monad\ x$
⟨*proof*⟩

lemma *approx-mem-monad2*: $x\ @=\ u ==> x \in monad\ u$
⟨*proof*⟩

lemma *approx-mem-monad-zero*: $[|\ x\ @=\ y;x \in monad\ 0\ |] ==> y \in monad\ 0$
⟨*proof*⟩

lemma *Infinitesimal-approx-hrabs*:
$[|\ x\ @=\ y;\ (x::hypreal) \in Infinitesimal\ |] ==> abs\ x\ @=\ abs\ y$
⟨*proof*⟩

lemma *less-Infinitesimal-less*:
$[|\ 0 < x;\ \ (x::hypreal) \notin Infinitesimal;\ \ e :Infinitesimal\ |] ==> e < x$
⟨*proof*⟩

lemma *Ball-mem-monad-gt-zero*:
$[|\ 0 < (x::hypreal);\ \ x \notin Infinitesimal;\ u \in monad\ x\ |] ==> 0 < u$
⟨*proof*⟩

lemma *Ball-mem-monad-less-zero*:
$[|\ (x::hypreal) < 0;\ x \notin Infinitesimal;\ u \in monad\ x\ |] ==> u < 0$
⟨*proof*⟩

lemma *lemma-approx-gt-zero*:
$[|0 < (x::hypreal);\ x \notin Infinitesimal;\ x\ @=\ y|] ==> 0 < y$
⟨*proof*⟩

lemma *lemma-approx-less-zero*:
$[|(x::hypreal) < 0;\ x \notin Infinitesimal;\ x\ @=\ y|] ==> y < 0$
⟨*proof*⟩

theorem *approx-hrabs*: $(x::hypreal)\ @=\ y \Longrightarrow abs\ x\ @=\ abs\ y$
⟨*proof*⟩

lemma *approx-hrabs-zero-cancel*: $abs(x::hypreal)\ @=\ 0 \Longrightarrow x\ @=\ 0$
⟨*proof*⟩

lemma *approx-hrabs-add-Infinitesimal*:
$(e::hypreal) \in Infinitesimal \Longrightarrow abs\ x\ @=\ abs(x+e)$
⟨*proof*⟩

lemma *approx-hrabs-add-minus-Infinitesimal*:
$(e::hypreal) \in Infinitesimal \Longrightarrow abs\ x\ @=\ abs(x + -e)$
⟨*proof*⟩

lemma *hrabs-add-Infinitesimal-cancel*:
$[|\ (e::hypreal) \in Infinitesimal;\ e' \in Infinitesimal;$
$abs(x+e) = abs(y+e')|] \Longrightarrow abs\ x\ @=\ abs\ y$
⟨*proof*⟩

lemma *hrabs-add-minus-Infinitesimal-cancel*:
$[|\ (e::hypreal) \in Infinitesimal;\ e' \in Infinitesimal;$
$abs(x + -e) = abs(y + -e')|] \Longrightarrow abs\ x\ @=\ abs\ y$
⟨*proof*⟩

37.12 More *HFinite* and *Infinitesimal* Theorems

lemma *Infinitesimal-add-hypreal-of-real-less*:
$[|\ x < y;\ \ u \in Infinitesimal\ |]$
$\Longrightarrow hypreal\text{-}of\text{-}real\ x + u < hypreal\text{-}of\text{-}real\ y$
⟨*proof*⟩

lemma *Infinitesimal-add-hrabs-hypreal-of-real-less*:
$[|\ x \in Infinitesimal;\ abs(hypreal\text{-}of\text{-}real\ r) < hypreal\text{-}of\text{-}real\ y\ |]$
$\Longrightarrow abs\ (hypreal\text{-}of\text{-}real\ r + x) < hypreal\text{-}of\text{-}real\ y$
⟨*proof*⟩

lemma *Infinitesimal-add-hrabs-hypreal-of-real-less2*:
$[|\ x \in Infinitesimal;\ \ abs(hypreal\text{-}of\text{-}real\ r) < hypreal\text{-}of\text{-}real\ y\ |]$
$\Longrightarrow abs\ (x + hypreal\text{-}of\text{-}real\ r) < hypreal\text{-}of\text{-}real\ y$
⟨*proof*⟩

lemma *hypreal-of-real-le-add-Infininitesimal-cancel*:
$[|\ u \in Infinitesimal;\ v \in Infinitesimal;$
$hypreal\text{-}of\text{-}real\ x + u \leq hypreal\text{-}of\text{-}real\ y + v\ |]$
$\Longrightarrow hypreal\text{-}of\text{-}real\ x \leq hypreal\text{-}of\text{-}real\ y$
⟨*proof*⟩

lemma *hypreal-of-real-le-add-Infininitesimal-cancel2*:

[| $u \in$ *Infinitesimal*; $v \in$ *Infinitesimal*;
hypreal-of-real $x + u \leq$ *hypreal-of-real* $y + v$ |]
==> $x \leq y$
⟨*proof*⟩

lemma *hypreal-of-real-less-Infinitesimal-le-zero*:
[| *hypreal-of-real* $x < e$; $e \in$ *Infinitesimal* |] ==> *hypreal-of-real* $x \leq 0$
⟨*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*)$*x + y*y \in$ *HFinite* ==> $x*x \in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-square-cancel2* [*simp*]:
(x::*hypreal*)$*x + y*y \in$ *Infinitesimal* ==> $y*y \in$ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-square-cancel2* [*simp*]:
(x::*hypreal*)$*x + y*y \in$ *HFinite* ==> $y*y \in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-sum-square-cancel* [*simp*]:
(x::*hypreal*)$*x + y*y + z*z \in$ *Infinitesimal* ==> $x*x \in$ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-sum-square-cancel* [*simp*]:
(x::*hypreal*)$*x + y*y + z*z \in$ *HFinite* ==> $x*x \in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-sum-square-cancel2* [*simp*]:
(y::*hypreal*)$*y + x*x + z*z \in$ *Infinitesimal* ==> $x*x \in$ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-sum-square-cancel2* [*simp*]:
(y::*hypreal*)$*y + x*x + z*z \in$ *HFinite* ==> $x*x \in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-sum-square-cancel3* [*simp*]:
(z::*hypreal*)$*z + y*y + x*x \in$ *Infinitesimal* ==> $x*x \in$ *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*⟩

37.13 Theorems about Standard Part

lemma *st-approx-self*: *x* ∈ *HFinite* ==> *st x* @= *x*
⟨*proof*⟩

lemma *st-SReal*: *x* ∈ *HFinite* ==> *st x* ∈ *Reals*
⟨*proof*⟩

lemma *st-HFinite*: *x* ∈ *HFinite* ==> *st x* ∈ *HFinite*
⟨*proof*⟩

lemma *st-unique*: ⟦*r* ∈ ℝ; *r* ≈ *x*⟧ ⟹ *st x* = *r*
⟨*proof*⟩

lemma *st-SReal-eq*: *x* ∈ *Reals* ==> *st x* = *x*
⟨*proof*⟩

lemma *st-hypreal-of-real* [*simp*]: *st* (*hypreal-of-real x*) = *hypreal-of-real x*
⟨*proof*⟩

lemma *st-eq-approx*: [| *x* ∈ *HFinite*; *y* ∈ *HFinite*; *st x* = *st y* |] ==> *x* @= *y*
⟨*proof*⟩

lemma *approx-st-eq*:
assumes *x* ∈ *HFinite* **and** *y* ∈ *HFinite* **and** *x* @= *y*
shows *st x* = *st y*
⟨*proof*⟩

lemma *st-eq-approx-iff*:
[| *x* ∈ *HFinite*; *y* ∈ *HFinite*|]
==> (*x* @= *y*) = (*st x* = *st y*)
⟨*proof*⟩

lemma *st-Infinitesimal-add-SReal*:
[| *x* ∈ *Reals*; *e* ∈ *Infinitesimal* |] ==> *st*(*x* + *e*) = *x*

⟨*proof*⟩

lemma *st-Infinitesimal-add-SReal2*:
[| $x \in Reals$; $e \in Infinitesimal$ |] ==> $st(e + x) = x$
⟨*proof*⟩

lemma *HFinite-st-Infinitesimal-add*:
$x \in HFinite$ ==> $\exists e \in Infinitesimal.\ x = st(x) + e$
⟨*proof*⟩

lemma *st-add*: $[\![x \in HFinite; y \in HFinite]\!] \Longrightarrow st\ (x + y) = st\ x + st\ y$
⟨*proof*⟩

lemma *st-number-of* [*simp*]: $st\ (number\text{-}of\ w) = number\text{-}of\ w$
⟨*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*⟩

37.14 Alternative Definitions using Free Ultrafilter

37.14.1 *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*⟩

37.14.2 *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*:
$\{n.\ norm\ (X\ n) = u\} \in FreeUltrafilterNat \Longrightarrow star\text{-}n\ X \in HFinite$
⟨*proof*⟩

lemma *HInfinite-FreeUltrafilterNat*:
$star\text{-}n\ X \in HInfinite \Longrightarrow \forall u.\ \{n.\ u < norm\ (X\ n)\} \in FreeUltrafilterNat$
⟨*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*:
$\forall u.\ \{n.\ u < norm\ (X\ n)\} \in FreeUltrafilterNat \Longrightarrow star\text{-}n\ X \in HInfinite$
⟨*proof*⟩

lemma *HInfinite-FreeUltrafilterNat-iff*:
$(star\text{-}n\ X \in HInfinite) = (\forall u.\ \{n.\ u < norm\ (X\ n)\} \in FreeUltrafilterNat)$
⟨*proof*⟩

37.14.3 *Infinitesimal*

lemma *ball-SReal-eq*: $(\forall x::hypreal \in Reals.\ P\ x) = (\forall x::real.\ P\ (star\text{-}of\ x))$
⟨*proof*⟩

lemma *Infinitesimal-FreeUltrafilterNat*:
$star\text{-}n\ X \in Infinitesimal \Longrightarrow \forall u{>}0.\ \{n.\ norm\ (X\ n) < u\} \in \mathcal{U}$
⟨*proof*⟩

lemma *FreeUltrafilterNat-Infinitesimal*:
$\forall u{>}0.\ \{n.\ norm\ (X\ n) < u\} \in \mathcal{U} \Longrightarrow star\text{-}n\ X \in Infinitesimal$
⟨*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*:
$(\forall r.\ 0 < r \longrightarrow x < r) = (\forall n.\ x < inverse(real\ (Suc\ n)))$
⟨*proof*⟩

lemma *lemma-Infinitesimal2*:
$(\forall r \in Reals.\ 0 < r \longrightarrow x < r) =$
$(\forall n.\ x < inverse(hypreal\text{-}of\text{-}nat\ (Suc\ n)))$
⟨*proof*⟩

lemma *Infinitesimal-hypreal-of-nat-iff*:
$Infinitesimal = \{x.\ \forall n.\ hnorm\ x < inverse\ (hypreal\text{-}of\text{-}nat\ (Suc\ n))\}$
⟨*proof*⟩

37.15 Proof that ω 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*:
$\{n.\ f\ n \leq u\} = \{n.\ f\ n < u\}\ Un\ \{n.\ u = (f\ n :: real)\}$
⟨*proof*⟩

lemma *finite-real-of-nat-le-real*: *finite* $\{n{::}nat.\ real\ n \leq u\}$
⟨*proof*⟩

lemma *finite-rabs-real-of-nat-le-real*: *finite* $\{n{::}nat.\ abs(real\ n) \leq u\}$
⟨*proof*⟩

lemma *rabs-real-of-nat-le-real-FreeUltrafilterNat*:
$\{n.\ abs(real\ n) \leq u\} \notin FreeUltrafilterNat$
⟨*proof*⟩

lemma *FreeUltrafilterNat-nat-gt-real*: $\{n.\ u < real\ n\} \in FreeUltrafilterNat$
⟨*proof*⟩

lemma *Compl-real-le-eq*: $-\{n::nat.\ real\ n \leq u\} = \{n.\ u < real\ n\}$
⟨*proof*⟩

ω 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*:

$0 < u \Longrightarrow \{n.\ u \leq inverse(real(Suc\ n))\} \notin FreeUltrafilterNat$
⟨*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*:
$0 < u \Longrightarrow$
$\{n.\ inverse(real(Suc\ n)) < u\} \in FreeUltrafilterNat$
⟨*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*:
$\forall n.\ norm(X\ n - x) < inverse(real(Suc\ n))$
$\Longrightarrow star\text{-}n\ X - star\text{-}of\ x \in Infinitesimal$
⟨*proof*⟩

lemma *real-seq-to-hypreal-approx*:
$\forall n.\ norm(X\ n - x) < inverse(real(Suc\ n))$
$\Longrightarrow star\text{-}n\ X\ @=\ star\text{-}of\ x$
⟨*proof*⟩

lemma *real-seq-to-hypreal-approx2*:
$\forall n.\ norm(x - X\ n) < inverse(real(Suc\ n))$
$\Longrightarrow star\text{-}n\ X\ @=\ star\text{-}of\ x$
⟨*proof*⟩

lemma *real-seq-to-hypreal-Infinitesimal2*:
$\forall n.\ norm(X\ n - Y\ n) < inverse(real(Suc\ n))$
$\Longrightarrow star\text{-}n\ X - star\text{-}n\ Y \in Infinitesimal$
⟨*proof*⟩

end

38 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* ∈ *Standard* ⟹ *hRe x* ∈ *Standard*
⟨*proof*⟩

lemma *Standard-hIm* [*simp*]: *x* ∈ *Standard* ⟹ *hIm x* ∈ *Standard*
⟨*proof*⟩

lemma *Standard-iii* [*simp*]: *iii* ∈ *Standard*
⟨*proof*⟩

lemma *Standard-hcnj* [*simp*]: *x* ∈ *Standard* ⟹ *hcnj x* ∈ *Standard*
⟨*proof*⟩

lemma *Standard-hsgn* [*simp*]: *x* ∈ *Standard* ⟹ *hsgn x* ∈ *Standard*
⟨*proof*⟩

lemma *Standard-harg* [*simp*]: *x* ∈ *Standard* ⟹ *harg x* ∈ *Standard*
⟨*proof*⟩

lemma *Standard-hcis* [*simp*]: *r* ∈ *Standard* ⟹ *hcis r* ∈ *Standard*

⟨*proof*⟩

lemma *Standard-hexpi* [*simp*]: $x \in Standard \Longrightarrow hexpi\ x \in Standard$
⟨*proof*⟩

lemma *Standard-hrcis* [*simp*]:
$[\![r \in Standard;\ s \in Standard]\!] \Longrightarrow hrcis\ r\ s \in Standard$
⟨*proof*⟩

lemma *Standard-HComplex* [*simp*]:
$[\![r \in Standard;\ s \in Standard]\!] \Longrightarrow HComplex\ r\ s \in Standard$
⟨*proof*⟩

lemma *hcmod-def*: $hcmod = *f* cmod$
⟨*proof*⟩

38.1 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?*]:
$!!z\ w.\ hRe\ z = hRe\ w \Longrightarrow hIm\ z = hIm\ w \Longrightarrow z = w$
⟨*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*⟩

38.2 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*⟩

38.3 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*⟩

38.4 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*⟩

38.5 Subraction and Division

lemma *hcomplex-diff-eq-eq* [*simp*]: $((x::hcomplex) - y = z) = (x = z + y)$

⟨*proof*⟩

38.6 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-of-hypreal epsilon $\neq 0$
⟨*proof*⟩

38.7 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*]:
(⋀*x y*. *P* (*HComplex x y*)) ==> *P z*
⟨*proof*⟩

38.8 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-of-hypreal r* = *HComplex 0 r*
⟨*proof*⟩

lemma *hcomplex-of-hypreal-i* [*simp*]:
!!*r*. *hcomplex-of-hypreal r* ∗ *iii* = *HComplex 0 r*
⟨*proof*⟩

38.9 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-of-hypreal x*) = *hcomplex-of-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-of-hypreal* (*hRe*(*z*) ^ *2* + *hIm*(*z*) ^ *2*)
⟨*proof*⟩

38.10 More Theorems about the Function *hcmod*

lemma *hcmod-hcomplex-of-hypreal-of-nat* [*simp*]:
hcmod (*hcomplex-of-hypreal*(*hypreal-of-nat n*)) = *hypreal-of-nat n*
⟨*proof*⟩

lemma *hcmod-hcomplex-of-hypreal-of-hypnat* [*simp*]:
hcmod (*hcomplex-of-hypreal*(*hypreal-of-hypnat n*)) = *hypreal-of-hypnat n*
⟨*proof*⟩

lemma *hcmod-mult-hcnj*: !!*z*. *hcmod*(*z* ∗ *hcnj*(*z*)) = *hcmod*(*z*) ^ *2*
⟨*proof*⟩

lemma *hcmod-triangle-ineq2* [*simp*]:
!!*a b*. *hcmod*(*b* + *a*) − *hcmod b* ≤ *hcmod a*
⟨*proof*⟩

lemma *hcmod-diff-ineq* [*simp*]: !!*a b*. *hcmod*(*a*) − *hcmod*(*b*) ≤ *hcmod*(*a* + *b*)
⟨*proof*⟩

38.11 Exponentiation

lemma *hcomplexpow-0* [*simp*]: *z* ^ *0* = (*1*::*hcomplex*)
⟨*proof*⟩

lemma *hcomplexpow-Suc* [*simp*]: *z* ^ (*Suc n*) = (*z*::*hcomplex*) ∗ (*z* ^ *n*)
⟨*proof*⟩

lemma *hcomplexpow-i-squared* [*simp*]: *iii* ^ *2* = −*1*
⟨*proof*⟩

lemma *hcomplex-of-hypreal-pow*:
!!x. *hcomplex-of-hypreal* $(x \hat{} n) = (\textit{hcomplex-of-hypreal}\ x) \hat{} n$
⟨*proof*⟩

lemma *hcomplex-hcnj-pow*: !!z. $\textit{hcnj}(z \hat{} n) = \textit{hcnj}(z) \hat{} n$
⟨*proof*⟩

lemma *hcmod-hcomplexpow*: !!x. $\textit{hcmod}(x \hat{} n) = \textit{hcmod}(x) \hat{} n$
⟨*proof*⟩

lemma *hcpow-minus*:
!!$x\ n$. $(-x::\textit{hcomplex})\ \textit{pow}\ n =$
$(\textit{if}\ (\ *p*\ \textit{even})\ n\ \textit{then}\ (x\ \textit{pow}\ n)\ \textit{else}\ -(x\ \textit{pow}\ n))$
⟨*proof*⟩

lemma *hcpow-mult*:
!!$r\ s\ n$. $((r::\textit{hcomplex}) * s)\ \textit{pow}\ n = (r\ \textit{pow}\ n) * (s\ \textit{pow}\ n)$
⟨*proof*⟩

lemma *hcpow-zero2* [*simp*]:
$\bigwedge n.\ 0\ \textit{pow}\ (\textit{hSuc}\ n) = (0::'a::\{\textit{recpower},\textit{semiring-0}\}\ \textit{star})$
⟨*proof*⟩

lemma *hcpow-not-zero* [*simp*,*intro*]:
!!$r\ n$. $r \neq 0 \Longrightarrow r\ \textit{pow}\ n \neq (0::\textit{hcomplex})$
⟨*proof*⟩

lemma *hcpow-zero-zero*: $r\ \textit{pow}\ n = (0::\textit{hcomplex}) \Longrightarrow r = 0$
⟨*proof*⟩

38.12 The Function *hsgn*

lemma *hsgn-zero* [*simp*]: $\textit{hsgn}\ 0 = 0$
⟨*proof*⟩

lemma *hsgn-one* [*simp*]: $\textit{hsgn}\ 1 = 1$
⟨*proof*⟩

lemma *hsgn-minus*: !!z. $\textit{hsgn}\ (-z) = -\ \textit{hsgn}(z)$
⟨*proof*⟩

lemma *hsgn-eq*: !!z. $\textit{hsgn}\ z = z\ /\ \textit{hcomplex-of-hypreal}\ (\textit{hcmod}\ z)$
⟨*proof*⟩

lemma *hcmod-i*: !!$x\ y$. $\textit{hcmod}\ (\textit{HComplex}\ x\ y) = (\ *f*\ \textit{sqrt})\ (x \hat{} 2 + y \hat{} 2)$
⟨*proof*⟩

lemma *hcomplex-eq-cancel-iff1* [*simp*]:
$(\textit{hcomplex-of-hypreal}\ xa = \textit{HComplex}\ x\ y) = (xa = x\ \&\ y = 0)$

⟨*proof*⟩

lemma *hcomplex-eq-cancel-iff2* [*simp*]:
(*HComplex x y* = *hcomplex-of-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-of-hypreal x* + *iii* ∗ *hcomplex-of-hypreal y*) =
hcomplex-of-hypreal(*x*/(*x* ^ *2* + *y* ^ *2*)) −
iii ∗ *hcomplex-of-hypreal*(*y*/(*x* ^ *2* + *y* ^ *2*))
⟨*proof*⟩

lemma *HComplex-inverse*:
!!*x y*. *inverse* (*HComplex x y*) =
HComplex (*x*/(*x* ^ *2* + *y* ^ *2*)) (−*y*/(*x* ^ *2* + *y* ^ *2*))
⟨*proof*⟩

lemma *hRe-mult-i-eq*[*simp*]:
!!*y*. *hRe* (*iii* ∗ *hcomplex-of-hypreal y*) = *0*
⟨*proof*⟩

lemma *hIm-mult-i-eq* [*simp*]:
!!*y*. *hIm* (*iii* ∗ *hcomplex-of-hypreal y*) = *y*
⟨*proof*⟩

lemma *hcmod-mult-i* [*simp*]: !!*y*. *hcmod* (*iii* ∗ *hcomplex-of-hypreal y*) = *abs y*
⟨*proof*⟩

lemma *hcmod-mult-i2* [*simp*]: !!*y*. *hcmod* (*hcomplex-of-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*]:
$!!y.\ (hcomplex\text{-}of\text{-}hypreal\ y = 0) = (y = 0)$
⟨*proof*⟩

38.13 Polar Form for Nonstandard Complex Numbers

lemma *complex-split-polar2*:
$\forall n.\ \exists r\ a.\ (z\ n) =\ complex\text{-}of\text{-}real\ r * (Complex\ (cos\ a)\ (sin\ a))$
⟨*proof*⟩

lemma *hcomplex-split-polar*:
$!!z.\ \exists r\ a.\ z = hcomplex\text{-}of\text{-}hypreal\ r * (HComplex((\ *f*\ cos)\ a)((\ *f*\ sin)\ a))$
⟨*proof*⟩

lemma *hcis-eq*:
$!!a.\ hcis\ a =$
$(hcomplex\text{-}of\text{-}hypreal((\ *f*\ cos)\ a) +$
$iii * hcomplex\text{-}of\text{-}hypreal((\ *f*\ sin)\ a))$
⟨*proof*⟩

lemma *hrcis-Ex*: $!!z.\ \exists r\ a.\ z = hrcis\ r\ a$
⟨*proof*⟩

lemma *hRe-hcomplex-polar* [*simp*]:
$!!r\ a.\ hRe\ (hcomplex\text{-}of\text{-}hypreal\ r * HComplex\ ((\ *f*\ cos)\ a)\ ((\ *f*\ sin)\ a)) =$
$r * (\ *f*\ cos)\ a$
⟨*proof*⟩

lemma *hRe-hrcis* [*simp*]: $!!r\ a.\ hRe(hrcis\ r\ a) = r * (\ *f*\ cos)\ a$
⟨*proof*⟩

lemma *hIm-hcomplex-polar* [*simp*]:
$!!r\ a.\ hIm\ (hcomplex\text{-}of\text{-}hypreal\ r * HComplex\ ((\ *f*\ cos)\ a)\ ((\ *f*\ sin)\ a)) =$

$r * (*f* sin)\ a$
⟨*proof*⟩

lemma *hIm-hrcis* [*simp*]: !!$r\ a$. $hIm(hrcis\ r\ a) = r * (*f* sin)\ a$
⟨*proof*⟩

lemma *hcmod-unit-one* [*simp*]:
!!a. $hcmod\ (HComplex\ ((*f* cos)\ a)\ ((*f* sin)\ a)) = 1$
⟨*proof*⟩

lemma *hcmod-complex-polar* [*simp*]:
!!$r\ a$. $hcmod\ (hcomplex\text{-}of\text{-}hypreal\ r * HComplex\ ((*f* cos)\ a)\ ((*f* sin)\ a)) = abs\ r$
⟨*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. $hrcis\ r\ 0 = hcomplex\text{-}of\text{-}hypreal\ r$
⟨*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. $hcis\ (hypreal\text{-}of\text{-}nat\ (Suc\ n) * a) =$
$hcis\ a * hcis\ (hypreal\text{-}of\text{-}nat\ n * a)$
⟨*proof*⟩

lemma *NSDeMoivre*: !!a. $(hcis\ a)\ \hat{}\ n = hcis\ (hypreal\text{-}of\text{-}nat\ n * a)$
⟨*proof*⟩

lemma *hcis-hypreal-of-hypnat-Suc-mult*:
!! $a\ n$. $hcis\ (hypreal\text{-}of\text{-}hypnat\ (n + 1) * a) =$
$hcis\ a * hcis\ (hypreal\text{-}of\text{-}hypnat\ n * a)$
⟨*proof*⟩

lemma *NSDeMoivre-ext*:
!!$a\ n$. $(hcis\ a)\ pow\ n = hcis\ (hypreal\text{-}of\text{-}hypnat\ n * a)$
⟨*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*⟩

38.14 *hcomplex-of-complex*: the Injection from type *complex* to to *hcomplex*

lemma *inj-hcomplex-of-complex*: *inj*(*hcomplex-of-complex*)

⟨*proof*⟩

lemma *hcomplex-of-complex-i*: *iii* = *hcomplex-of-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*⟩

38.15 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

39 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*. ∃ *As*. *X* = ∗*sn*∗ *As*}

definition

is-starext :: [*'a star* => *'a star*, *'a* => *'a*] => *bool* **where**
is-starext F f = (∀ *x y*. ∃ *X* ∈ *Rep-star*(*x*). ∃ *Y* ∈ *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*. ∃ *F*. *X* = ∗*fn*∗ *F*}

lemma *no-choice*: $\forall x.\ \exists y.\ Q\ x\ y \Longrightarrow \exists (f :: 'a \Rightarrow nat).\ \forall x.\ Q\ x\ (f\ x)$
⟨*proof*⟩

39.1 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*: $*s*\ X\ Int\ Reals = hypreal\text{-}of\text{-}real$ ‘ X
⟨*proof*⟩

lemma *STAR-star-of-Int*: $*s*\ X\ Int\ Standard = star\text{-}of$ ‘ X
⟨*proof*⟩

lemma *lemma-not-hyprealA*: $x \notin hypreal\text{-}of\text{-}real$ ‘ $A \Longrightarrow \forall y \in A.\ x \neq hypreal\text{-}of\text{-}real\ y$
⟨*proof*⟩

lemma *lemma-not-starA*: $x \notin star\text{-}of$ ‘ $A \Longrightarrow \forall y \in A.\ x \neq star\text{-}of\ y$
⟨*proof*⟩

lemma *lemma-Compl-eq*: $-\{n.\ X\ n = xa\} = \{n.\ X\ n \neq xa\}$
⟨*proof*⟩

lemma *STAR-real-seq-to-hypreal*:
$\forall n.\ (X\ n) \notin M \Longrightarrow star\text{-}n\ X \notin *s*\ M$
⟨*proof*⟩

lemma *STAR-singleton*: $*s*\ \{x\} = \{star\text{-}of\ x\}$
⟨*proof*⟩

lemma *STAR-not-mem*: $x \notin F \Longrightarrow star\text{-}of\ x \notin *s*\ F$
⟨*proof*⟩

lemma *STAR-subset-closed*: $[|\ x : *s*\ A;\ A <= B\ |] \Longrightarrow x : *s*\ B$
⟨*proof*⟩

Nonstandard extension of a set (defined using a constant sequence) as a special case of an internal set

lemma *starset-n-starset*: $\forall n.\ (As\ n = A) \Longrightarrow *sn*\ As = *s*\ A$
⟨*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-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 @= *star-of a* ==> (∗*f*∗ (%*x*. *x*)) *x* @= *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*: (*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*) (*star-of a*) = *star-of* (*f a*)
⟨*proof*⟩

lemma *starfun-approx*: (∗*f*∗ *f*) (*star-of a*) @= *star-of* (*f a*)
⟨*proof*⟩

lemma *starfun-lambda-cancel*:
!!*x'*. (∗*f*∗ (%*h*. *f* (*x* + *h*))) *x'* = (∗*f*∗ *f*) (*star-of x* + *x'*)
⟨*proof*⟩

lemma *starfun-lambda-cancel2*:
($*f*$ (%h. $f(g(x + h))$)) $x' = (*f* (f \circ g))$ (*star-of* $x + x'$)
⟨*proof*⟩

lemma *starfun-mult-HFinite-approx*:
fixes $l\ m$ :: $'a$::*real-normed-algebra star*
shows [| ($*f*$ f) x @= l; ($*f*$ g) x @= m;
l: *HFinite*; m: *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-n* X) = *star-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

40 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*: $Nats \leq *s*\ (UNIV :: nat\ set)$
⟨*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*⟩

40.1 Nonstandard Extensions of Functions

Example of transfer of a property from reals to hyperreals — used for limit comparison of sequences

lemma *starfun-le-mono*:
$\forall n.\ N \leq n \dashrightarrow f\ n \leq g\ n$
$\Longrightarrow \forall n.\ hypnat\text{-}of\text{-}nat\ N \leq n \dashrightarrow (\ *f*\ f)\ n \leq (\ *f*\ g)\ n$
⟨*proof*⟩

lemma *starfun-less-mono*:
$\forall n.\ N \leq n \dashrightarrow f\ n < g\ n$
$\Longrightarrow \forall n.\ hypnat\text{-}of\text{-}nat\ N \leq n \dashrightarrow (\ *f*\ f)\ n < (\ *f*\ g)\ n$
⟨*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*:
$N \in HNatInfinite$
$\Longrightarrow (\ *f*\ (\%x::nat.\ inverse(real\ x)))\ N = inverse(hypreal\text{-}of\text{-}hypnat\ N)$
⟨*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*]:
$N \in HNatInfinite \Longrightarrow (*f* \; (\%x. \; inverse \; (real \; x))) \; N \in Infinitesimal$
⟨*proof*⟩

40.2 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*:
$!!n. \; [| \; (*p* \; P) \; (0::hypnat); \; !!n. \; (*p* \; P)(n) \Longrightarrow (*p* \; P)(n + 1)|] \Longrightarrow (*p* \; P)(n)$
⟨*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 ∈ (*S* :: *nat set*) ==> (*LEAST n*. *n* ∈ *S*) ∈ *S*
⟨*proof*⟩

lemma *nonempty-set-star-has-least*:
!!*S*::*nat set star*. *Iset S* ≠ {} ==> ∃ *n* ∈ *Iset S*. ∀ *m* ∈ *Iset S*. *n* ≤ *m*
⟨*proof*⟩

lemma *nonempty-InternalNatSet-has-least*:
[| (*S*::*hypnat set*) ∈ *InternalSets*; *S* ≠ {} |] ==> ∃ *n* ∈ *S*. ∀ *m* ∈ *S*. *n* ≤ *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

41 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 = (∀ *M* ∈ *HNatInfinite*. ∀ *N* ∈ *HNatInfinite*. (∗*f*∗ *X*) *M* ≈ (∗*f*∗ *X*) *N*)

41.1 Limits of Sequences

lemma *NSLIMSEQ-iff*:
(*X* −−−−*NS*> *L*) = (∀ *N* ∈ *HNatInfinite*. (∗*f*∗ *X*) *N* ≈ *star-of L*)
⟨*proof*⟩

lemma *NSLIMSEQ-I*:
(⋀*N*. *N* ∈ *HNatInfinite* ⟹ *starfun X N* ≈ *star-of L*) ⟹ *X* −−−−*NS*> *L*
⟨*proof*⟩

lemma *NSLIMSEQ-D*:
⟦*X* −−−−*NS*> *L*; *N* ∈ *HNatInfinite*⟧ ⟹ *starfun X N* ≈ *star-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-normed-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) = ($*f*$ (λx. *norm* ($f\ x$))) x
⟨*proof*⟩

lemma *NSLIMSEQ-norm*: X ----$NS>$ a $\Longrightarrow$ (λn. *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*> $a) --> ((\%n.\ (X\ n)\ \hat{}\ m)$ ----*NS*> $a\ \hat{}\ m)$
⟨*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*$);\ \forall n.\ X\ n \leq a\ |] ==> r \leq a$
⟨*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*⟩

41.1.1 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). $\forall n.\ n \leq f\ n$ ==> *finite* $\{n.\ f\ n \leq u\}$
⟨*proof*⟩

41.1.2 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*: $((\lambda n.\ norm\ (X\ n)) \text{----}NS{>}\ 0) = (X \text{----}NS{>}\ 0)$
⟨*proof*⟩

lemma *NSLIMSEQ-rabs-zero*: $((\%n.\ |f\ n|) \text{----}NS{>}\ 0) = (f \text{----}NS{>}\ (0::real))$
⟨*proof*⟩

Generalization to other limits

lemma *NSLIMSEQ-imp-rabs*: $f \text{----}NS{>}\ (l::real) ==> (\%n.\ |f\ n|) \text{----}NS{>}\ |l|$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse-zero*:
$\forall y::real.\ \exists N.\ \forall n \geq N.\ y < f(n)$
==> $(\%n.\ inverse(f\ n)) \text{----}NS{>}\ 0$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse-real-of-nat*: $(\%n.\ inverse(real(Suc\ n))) \text{----}NS{>}\ 0$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse-real-of-nat-add*:
$(\%n.\ r + inverse(real(Suc\ n))) \text{----}NS{>}\ r$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse-real-of-nat-add-minus*:
$(\%n.\ r + -inverse(real(Suc\ n))) \text{----}NS{>}\ r$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse-real-of-nat-add-minus-mult*:
$(\%n.\ r*(\ 1 + -inverse(real(Suc\ n)))) \text{----}NS{>}\ r$
⟨*proof*⟩

41.2 Convergence

lemma *nslimI*: $X \text{----}NS{>}\ L ==> nslim\ X = L$
⟨*proof*⟩

lemma *lim-nslim-iff*: $lim\ X = nslim\ X$
⟨*proof*⟩

lemma *NSconvergentD*: $NSconvergent\ X ==> \exists L.\ (X \text{----}NS{>}\ L)$
⟨*proof*⟩

lemma *NSconvergentI*: $(X \text{ ----}NS\text{>} L) \Longrightarrow NSconvergent\ X$
⟨*proof*⟩

lemma *convergent-NSconvergent-iff*: $convergent\ X = NSconvergent\ X$
⟨*proof*⟩

lemma *NSconvergent-NSLIMSEQ-iff*: $NSconvergent\ X = (X \text{ ----}NS\text{>} nslim\ X)$
⟨*proof*⟩

41.3 Bounded Monotonic Sequences

lemma *NSBseqD*: $[|\ NSBseq\ X;\ \ N\text{: }HNatInfinite\ |] \Longrightarrow (\ *f*\ X)\ N : HFinite$
⟨*proof*⟩

lemma *Standard-subset-HFinite*: $Standard \subseteq HFinite$
⟨*proof*⟩

lemma *NSBseqD2*: $NSBseq\ X \Longrightarrow (\ *f*\ X)\ N \in HFinite$
⟨*proof*⟩

lemma *NSBseqI*: $\forall N \in HNatInfinite.\ (\ *f*\ X)\ N : HFinite \Longrightarrow NSBseq\ X$
⟨*proof*⟩

The standard definition implies the nonstandard definition

lemma *Bseq-NSBseq*: $Bseq\ X \Longrightarrow NSBseq\ X$
⟨*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*: $NSconvergent\ X \Longrightarrow NSBseq\ X$
⟨*proof*⟩

Standard Version: easily now proved using equivalence of NS and standard definitions

lemma *convergent-Bseq*: $convergent\ X \Longrightarrow Bseq\ X$
⟨*proof*⟩

41.3.1 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*⟩

41.3.2 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*⟩

41.4 Cauchy Sequences

lemma *NSCauchyI*:
$(\bigwedge M\ N.\ [\![M \in HNatInfinite; N \in HNatInfinite]\!] \Longrightarrow starfun\ X\ M \approx starfun\ X\ N)$
$\Longrightarrow NSCauchy\ X$
⟨*proof*⟩

lemma *NSCauchyD*:
$[\![NSCauchy\ X; M \in HNatInfinite; N \in HNatInfinite]\!]$
$\Longrightarrow starfun\ X\ M \approx starfun\ X\ N$
⟨*proof*⟩

41.4.1 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*⟩

41.4.2 Cauchy Sequences are Bounded

A Cauchy sequence is bounded – nonstandard version

lemma *NSCauchy-NSBseq*: *NSCauchy X ==> NSBseq X*
⟨*proof*⟩

41.4.3 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 ⟹ NSCauchy X*
⟨*proof*⟩

lemma *real-NSCauchy-NSconvergent*:
fixes *X* :: *nat ⇒ real*
shows *NSCauchy X ⟹ NSconvergent X*
⟨*proof*⟩

lemma *NSCauchy-NSconvergent*:
fixes *X* :: *nat ⇒ 'a*::*banach*
shows *NSCauchy X ⟹ NSconvergent X*
⟨*proof*⟩

lemma *NSCauchy-NSconvergent-iff*:
fixes *X* :: *nat ⇒ 'a*::*banach*
shows *NSCauchy X = NSconvergent X*
⟨*proof*⟩

41.5 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*: $|c| < (1::real) ==> (\%n.\ c\ \hat{}\ n) ----NS> 0$
⟨*proof*⟩

end

42 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*:
!!$n\ p$. $n < p \Longrightarrow sumhr(0,n,f) + sumhr(n,p,f) = sumhr(0,p,f)$
⟨*proof*⟩

lemma *sumhr-split-diff*: $n<p \Longrightarrow sumhr(0,p,f) - sumhr(0,n,f) = sumhr(n,p,f)$
⟨*proof*⟩

lemma *sumhr-hrabs*: !!$m\ n$. $abs(sumhr(m,n,f)) \leq sumhr(m,n,\%i.\ abs(f\ i))$
⟨*proof*⟩

other general version also needed

lemma *sumhr-fun-hypnat-eq*:
$(\forall r.\ m \leq r\ \&\ r < n \longrightarrow f\ r = g\ r) \longrightarrow$
$sumhr(hypnat\text{-}of\text{-}nat\ m,\ hypnat\text{-}of\text{-}nat\ n,\ f) =$
$sumhr(hypnat\text{-}of\text{-}nat\ m,\ hypnat\text{-}of\text{-}nat\ n,\ g)$
⟨*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*]: !!$m\ n$. $n < m \Longrightarrow sumhr(m,n,f) = 0$
⟨*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*⟩

42.1 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*:
$(\forall n.\ m \leq Suc\ n\ --> f\ n = r)\ \&\ m \leq na$
$==> sumhr(hypnat\text{-}of\text{-}nat\ m,hypnat\text{-}of\text{-}nat\ na,f) =$
$(hypreal\text{-}of\text{-}nat\ (na - m) * hypreal\text{-}of\text{-}real\ r)$
⟨*proof*⟩

lemma *starfunNat-sumr*: $!!N.\ (\ *f*\ (\%n.\ setsum\ f\ \{0..<n\}))\ N = sumhr(0,N,f)$
⟨*proof*⟩

lemma *sumhr-hrabs-approx* [*simp*]: $sumhr(0,\ M,\ f)\ @=\ sumhr(0,\ N,\ f)$
$==> abs\ (sumhr(M,\ N,\ f))\ @=\ 0$
⟨*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*: $f\ NSsums\ l ==> NSsummable\ f$
⟨*proof*⟩

lemma *NSsummable-NSsums*: $NSsummable\ f ==> f\ NSsums\ (NSsuminf\ f)$
⟨*proof*⟩

lemma *NSsums-unique*: $f\ NSsums\ s ==> (s = NSsuminf\ f)$
⟨*proof*⟩

lemma *NSseries-zero*:
$\forall m.\ n \leq Suc\ m \longrightarrow f(m) = 0 \Longrightarrow f\ NSsums\ (setsum\ f\ \{0..<n\})$
⟨*proof*⟩

lemma *NSsummable-NSCauchy*:
$NSsummable\ f =$
$(\forall M \in HNatInfinite.\ \forall N \in HNatInfinite.\ abs\ (sumhr(M,N,f)) \mathbin{@=} 0)$
⟨*proof*⟩

Terms of a convergent series tend to zero

lemma *NSsummable-NSLIMSEQ-zero*: $NSsummable\ f \Longrightarrow f\ ----NS> 0$
⟨*proof*⟩

Nonstandard comparison test

lemma *NSsummable-comparison-test*:
$[|\ \exists N.\ \forall n.\ N \leq n \longrightarrow abs(f\ n) \leq g\ n;\ NSsummable\ g\ |] \Longrightarrow NSsummable\ f$
⟨*proof*⟩

lemma *NSsummable-rabs-comparison-test*:
$[|\ \exists N.\ \forall n.\ N \leq n \longrightarrow abs(f\ n) \leq g\ n;\ NSsummable\ g\ |]$
$\Longrightarrow NSsummable\ (\%k.\ abs\ (f\ k))$
⟨*proof*⟩

end

43 HLim: Limits and Continuity (Nonstandard)

theory *HLim*
imports *Star Lim*
begin

Nonstandard Definitions

definition
$NSLIM :: ['a::real\text{-}normed\text{-}vector \Rightarrow 'b::real\text{-}normed\text{-}vector,\ 'a,\ 'b] \Rightarrow bool$
$(((-)/\ --\ (-)/\ --NS>\ (-))\ [60,\ 0,\ 60]\ 60)$ **where**
$f\ --\ a\ --NS>\ L =$
$(\forall x.\ (x \neq star\text{-}of\ a\ \&\ x \mathbin{@=} star\text{-}of\ a \longrightarrow (\ *f*\ f)\ x \mathbin{@=} star\text{-}of\ L))$

definition
$isNSCont :: ['a::real\text{-}normed\text{-}vector \Rightarrow 'b::real\text{-}normed\text{-}vector,\ 'a] \Rightarrow bool$ **where**
— NS definition dispenses with limit notions
$isNSCont\ f\ a = (\forall y.\ y \mathbin{@=} star\text{-}of\ a \longrightarrow$
$(\ *f*\ f)\ y \mathbin{@=} star\text{-}of\ (f\ a))$

definition

isNSUCont :: ['a::*real-normed-vector* => 'b::*real-normed-vector*] => *bool* **where**
isNSUCont f = (∀ *x y*. *x* @= *y* −−> (∗*f*∗ *f*) *x* @= (∗*f*∗ *f*) *y*)

43.1 Limits of Functions

lemma *NSLIM-I*:
(⋀*x*. ⟦*x* ≠ *star-of a*; *x* ≈ *star-of a*⟧ ⟹ *starfun f x* ≈ *star-of L*)
⟹ *f* −− *a* −−*NS*> *L*
⟨*proof*⟩

lemma *NSLIM-D*:
⟦*f* −− *a* −−*NS*> *L*; *x* ≠ *star-of a*; *x* ≈ *star-of a*⟧
⟹ *starfun f x* ≈ *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*]:
starfun (λ*x*. *f x* ∗*R* *g x*) = (λ*x*. *scaleHR* (*starfun f x*) (*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*⟧ ⟹ (λ*x*. *f x* − *g x*) −− *x* −−*NS*> (*l* − *m*)
⟨*proof*⟩

lemma *NSLIM-add-minus*: [| *f* −− *x* −−*NS*> *l*; *g* −− *x* −−*NS*> *m* |] ==>

$(\%x.\ f(x) + -g(x)) \mathrel{--} x \mathrel{--NS>} (l + -m)$
⟨*proof*⟩

lemma *NSLIM-inverse*:
fixes $L :: 'a$::*real-normed-div-algebra*
shows $[\!|\ f \mathrel{--} a \mathrel{--NS>} L;\ L \neq 0\ |\!]$
$\Longrightarrow (\%x.\ inverse(f(x))) \mathrel{--} a \mathrel{--NS>} (inverse\ L)$
⟨*proof*⟩

lemma *NSLIM-zero*:
assumes f: $f \mathrel{--} a \mathrel{--NS>} l$ **shows** $(\%x.\ f(x) - l) \mathrel{--} a \mathrel{--NS>} 0$
⟨*proof*⟩

lemma *NSLIM-zero-cancel*: $(\%x.\ f(x) - l) \mathrel{--} x \mathrel{--NS>} 0 \Longrightarrow f \mathrel{--} x \mathrel{--NS>} l$
⟨*proof*⟩

lemma *NSLIM-const-not-eq*:
fixes $a :: 'a$::*real-normed-algebra-1*
shows $k \neq L \Longrightarrow \neg\ (\lambda x.\ k) \mathrel{--} a \mathrel{--NS>} L$
⟨*proof*⟩

lemma *NSLIM-not-zero*:
fixes $a :: 'a$::*real-normed-algebra-1*
shows $k \neq 0 \Longrightarrow \neg\ (\lambda x.\ k) \mathrel{--} a \mathrel{--NS>} 0$
⟨*proof*⟩

lemma *NSLIM-const-eq*:
fixes $a :: 'a$::*real-normed-algebra-1*
shows $(\lambda x.\ k) \mathrel{--} a \mathrel{--NS>} L \Longrightarrow k = L$
⟨*proof*⟩

lemma *NSLIM-unique*:
fixes $a :: 'a$::*real-normed-algebra-1*
shows $[\![f \mathrel{--} a \mathrel{--NS>} L; f \mathrel{--} a \mathrel{--NS>} M]\!] \Longrightarrow L = M$
⟨*proof*⟩

lemma *NSLIM-mult-zero*:
fixes $f\ g :: 'a$::*real-normed-vector* $\Rightarrow 'b$::*real-normed-algebra*
shows $[\!|\ f \mathrel{--} x \mathrel{--NS>} 0;\ g \mathrel{--} x \mathrel{--NS>} 0\ |\!] \Longrightarrow (\%x.\ f(x) * g(x)) \mathrel{--} x \mathrel{--NS>} 0$
⟨*proof*⟩

lemma *NSLIM-self*: $(\%x.\ x) \mathrel{--} a \mathrel{--NS>} a$
⟨*proof*⟩

43.1.1 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*⟩

43.2 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*: $(f \dashrightarrow a \dashrightarrow_{NS} f\ a) = ((\%h.\ f(a + h)) \dashrightarrow 0 \dashrightarrow_{NS} f\ a)$
⟨*proof*⟩

lemma *isNSCont-minus*: *isNSCont f a* ==> *isNSCont* $(\%x.\ -f\ x)\ a$
⟨*proof*⟩

lemma *isNSCont-inverse*:
fixes $f :: 'a$::*real-normed-vector* ⇒ $'b$::*real-normed-div-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*⟩

43.3 Uniform Continuity

lemma *isNSUContD*: [| *isNSUCont f*; $x \approx y$|] ==> $(\ *f*\ f)\ x \approx (\ *f*\ f)\ y$
⟨*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

44 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 = $(\forall h \in Infinitesimal - \{0\}.$
$((\ *f*\ f)(star\text{-}of\ x + h)$
$- star\text{-}of\ (f\ x))/h\ @= star\text{-}of\ D)$

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*))

44.1 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*:
⋀*r*. *hnorm* ((∗*f*∗ *of-real*) *r*::*'a*::*real-normed-div-algebra star*) = |*r*|
⟨*proof*⟩

lemma *Infinitesimal-of-hypreal*:
x ∈ *Infinitesimal* ⟹
((∗*f*∗ *of-real*) *x*::*'a*::*real-normed-div-algebra star*) ∈ *Infinitesimal*
⟨*proof*⟩

lemma *of-hypreal-eq-0-iff*:
⋀*x*. ((∗*f*∗ *of-real*) *x* = (0::*'a*::*real-algebra-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*) =
(∀ *w*.
w ≠ *star-of x* & *w* ≈ *star-of x* −−>
(∗*f*∗ (%*z*. (*f z* − *f x*) / (*z*−*x*))) *w* ≈ *star-of D*)
⟨*proof*⟩

lemma *hypreal-not-eq-minus-iff*:
(*x* ≠ *a*) = (*x* − *a* ≠ (*0*::'*a*::*ab-group-add*))
⟨*proof*⟩

lemma *NSDERIVD5*:
(*NSDERIV f x :> D*) ==>
(∀ *u*. *u* ≈ *hypreal-of-real x* −−>
(∗*f*∗ (%*z*. *f z* − *f x*)) *u* ≈ *hypreal-of-real D* ∗ (*u* − *hypreal-of-real x*))
⟨*proof*⟩

lemma *NSDERIVD4*:
(*NSDERIV f x :> D*) ==>
(∀ *h* ∈ *Infinitesimal*.
((∗*f*∗ *f*)(*hypreal-of-real x* + *h*) −
hypreal-of-real (*f x*))≈ (*hypreal-of-real D*) ∗ *h*)
⟨*proof*⟩

lemma *NSDERIVD3*:
(*NSDERIV f x :> D*) ==>
(∀ *h* ∈ *Infinitesimal* − {*0*}.
((∗*f*∗ *f*)(*hypreal-of-real x* + *h*) −
hypreal-of-real (*f x*))≈ (*hypreal-of-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-ring star*
shows $(a*b) - (c*d) = (b*(a - c)) + (c*(b - d))$
⟨*proof*⟩

lemma *lemma-nsderiv2*:
fixes *x y z* :: *'a::real-normed-field star*
shows [| $(x - y) / z = star\text{-}of\ D + yb$; $z \neq 0$;
$z \in Infinitesimal$; $yb \in Infinitesimal$ |]
==> $x - y \approx 0$
⟨*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*;

$(\ *f*\ g)\ (star\text{-}of\ x + xa) = star\text{-}of\ (g\ x);$
$xa \in Infinitesimal;$
$xa \neq 0$
$|] ==> D = 0$
⟨*proof*⟩

lemma *NSDERIV-approx*:
$[|\ NSDERIV\ f\ x :> D;\ \ h \in Infinitesimal;\ \ h \neq 0\ |]$
$==> (\ *f*\ f)\ (star\text{-}of\ x + h) - star\text{-}of\ (f\ x) \approx 0$
⟨*proof*⟩

lemma *NSDERIVD1*: $[|\ NSDERIV\ f\ (g\ x) :> Da;$
$(\ *f*\ g)\ (star\text{-}of(x) + xa) \neq star\text{-}of\ (g\ x);$
$(\ *f*\ g)\ (star\text{-}of(x) + xa) \approx star\text{-}of\ (g\ x)$
$|] ==> ((\ *f*\ f)\ ((\ *f*\ g)\ (star\text{-}of(x) + xa))$
$- star\text{-}of\ (f\ (g\ x)))$
$/ ((\ *f*\ g)\ (star\text{-}of(x) + xa) - star\text{-}of\ (g\ x))$
$\approx star\text{-}of(Da)$
⟨*proof*⟩

lemma *NSDERIVD2*: $[|\ NSDERIV\ g\ x :> Db;\ xa \in Infinitesimal;\ xa \neq 0\ |]$
$==> ((\ *f*\ g)\ (star\text{-}of(x) + xa) - star\text{-}of(g\ x)) / xa$
$\approx star\text{-}of(Db)$
⟨*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*: $[|\ NSDERIV\ f\ (g\ x) :> Da;\ NSDERIV\ g\ x :> Db\ |]$
$==> NSDERIV\ (f\ o\ g)\ x :> Da * Db$
⟨*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*⟩

44.1.1 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*: $NSDERIV\ (\%x.\ x\ \hat{}\ n)\ x :> real\ n * (x\ \hat{}\ (n - Suc\ 0))$
⟨*proof*⟩

Derivative of inverse

lemma *NSDERIV-inverse-fun*:
fixes $x :: 'a::\{real\text{-}normed\text{-}field, recpower\}$
shows $[|\ NSDERIV\ f\ x :> d;\ f(x) \neq 0\ |]$
$==> NSDERIV\ (\%x.\ inverse(f\ x))\ x :> (-\ (d * inverse(f(x))\ \hat{}\ Suc\ (Suc\ 0))))$
⟨*proof*⟩

Derivative of quotient

lemma *NSDERIV-quotient*:
fixes $x :: 'a::\{real\text{-}normed\text{-}field, recpower\}$
shows $[|\ NSDERIV\ f\ x :> d;\ NSDERIV\ g\ x :> e;\ g(x) \neq 0\ |]$
$==> NSDERIV\ (\%y.\ f(y)\ /\ (g\ y))\ x :> (d*g(x) - (e*f(x)))\ /\ (g(x)\ \hat{}\ Suc\ (Suc\ 0))$
⟨*proof*⟩

lemma *CARAT-NSDERIV*: $NSDERIV\ f\ x :> l ==>$
$\exists g.\ (\forall z.\ f\ z - f\ x = g\ z * (z-x))\ \&\ isNSCont\ g\ x\ \&\ g\ x = l$
⟨*proof*⟩

lemma *hypreal-eq-minus-iff3*: $(x = y + z) = (x + -z = (y::hypreal))$
⟨*proof*⟩

lemma *CARAT-DERIVD*:
assumes *all*: $\forall z.\ f\ z - f\ x = g\ z * (z-x)$
and *nsc*: $isNSCont\ g\ x$
shows $NSDERIV\ f\ x :> g\ x$
⟨*proof*⟩

44.1.2 Differentiability predicate

lemma *NSdifferentiableD*: $f\ NSdifferentiable\ x ==> \exists D.\ NSDERIV\ f\ x :> D$
⟨*proof*⟩

lemma *NSdifferentiableI*: *NSDERIV f x :> D ==> f NSdifferentiable x*
⟨*proof*⟩

44.2 (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

45 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 ^ n)))*

definition
sinhr :: *real* => *hypreal* **where**
sinhr x = *st*(*sumhr* (*0*, *whn*, %*n*. (*if even*(*n*) *then 0 else*
((−*1*) ^ ((*n* − *1*) *div 2*))/(*real* (*fact n*))) ∗ (*x* ^ *n*)))

definition
coshr :: *real* => *hypreal* **where**
coshr x = *st*(*sumhr* (*0*, *whn*, %*n*. (*if even*(*n*) *then*
((−*1*) ^ (*n div 2*))/(*real* (*fact n*)) *else 0*) ∗ *x* ^ *n*))

45.1 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*]:

$0 \leq x \Longrightarrow ((\ *f*\ sqrt)(x)\ @=\ 0) = (x\ @=\ 0)$
⟨*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*: $!!x.\ 0 < x \Longrightarrow 0 < (\ *f*\ sqrt)(x)$
⟨*proof*⟩

lemma *hypreal-sqrt-ge-zero*: $0 \leq x \Longrightarrow 0 \leq (\ *f*\ sqrt)(x)$
⟨*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*:
$[|\ x \in HFinite;\ 0 \leq x\ |] \Longrightarrow st((\ *f*\ sqrt)\ x) = (\ *f*\ sqrt)(st\ x)$
⟨*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*:
$[|\ 0 \leq x;\ x \in HFinite\ |] \Longrightarrow (\ *f*\ sqrt)\ x \in HFinite$
⟨*proof*⟩

lemma *HFinite-hypreal-sqrt-imp-HFinite*:
$[|\ 0 \leq x;\ (\ *f*\ sqrt)\ x \in HFinite\ |] \Longrightarrow x \in HFinite$
⟨*proof*⟩

lemma *HFinite-hypreal-sqrt-iff* [*simp*]:
$0 \leq x \Longrightarrow ((\ *f*\ sqrt)\ x \in HFinite) = (x \in HFinite)$
⟨*proof*⟩

lemma *HFinite-sqrt-sum-squares* [*simp*]:
(($*f*$ *sqrt*)(*x*$*$*x* + *y*$*$*y*) ∈ *HFinite*) = (*x*$*$*x* + *y*$*$*y* ∈ *HFinite*)
⟨*proof*⟩

lemma *Infinitesimal-hypreal-sqrt*:
[| *0* ≤ *x*; *x* ∈ *Infinitesimal* |] ==> ($*f*$ *sqrt*) *x* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *Infinitesimal-hypreal-sqrt-imp-Infinitesimal*:
[| *0* ≤ *x*; ($*f*$ *sqrt*) *x* ∈ *Infinitesimal* |] ==> *x* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *Infinitesimal-hypreal-sqrt-iff* [*simp*]:
0 ≤ *x* ==> (($*f*$ *sqrt*) *x* ∈ *Infinitesimal*) = (*x* ∈ *Infinitesimal*)
⟨*proof*⟩

lemma *Infinitesimal-sqrt-sum-squares* [*simp*]:
(($*f*$ *sqrt*)(*x*$*$*x* + *y*$*$*y*) ∈ *Infinitesimal*) = (*x*$*$*x* + *y*$*$*y* ∈ *Infinitesimal*)
⟨*proof*⟩

lemma *HInfinite-hypreal-sqrt*:
[| *0* ≤ *x*; *x* ∈ *HInfinite* |] ==> ($*f*$ *sqrt*) *x* ∈ *HInfinite*
⟨*proof*⟩

lemma *HInfinite-hypreal-sqrt-imp-HInfinite*:
[| *0* ≤ *x*; ($*f*$ *sqrt*) *x* ∈ *HInfinite* |] ==> *x* ∈ *HInfinite*
⟨*proof*⟩

lemma *HInfinite-hypreal-sqrt-iff* [*simp*]:
0 ≤ *x* ==> (($*f*$ *sqrt*) *x* ∈ *HInfinite*) = (*x* ∈ *HInfinite*)
⟨*proof*⟩

lemma *HInfinite-sqrt-sum-squares* [*simp*]:
(($*f*$ *sqrt*)(*x*$*$*x* + *y*$*$*y*) ∈ *HInfinite*) = (*x*$*$*x* + *y*$*$*y* ∈ *HInfinite*)
⟨*proof*⟩

lemma *HFinite-exp* [*simp*]:
sumhr (*0*, *whn*, %*n*. *inverse* (*real* (*fact n*)) $*$ *x* ^ *n*) ∈ *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*: $x \in Infinitesimal \Longrightarrow (\ *f*\ exp)\ (x::hypreal) \mathbin{@=} 1$
⟨*proof*⟩

lemma *STAR-exp-epsilon* [*simp*]: $(\ *f*\ exp)\ epsilon \mathbin{@=} 1$
⟨*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*: $exphr\ x = hypreal\text{-}of\text{-}real\ (exp\ x)$
⟨*proof*⟩

lemma *starfun-exp-ge-add-one-self* [*simp*]: $!!x::hypreal.\ 0 \leq x \Longrightarrow (1 + x) \leq (\ *f*\ exp)\ x$
⟨*proof*⟩

lemma *starfun-exp-HInfinite*:
$[|\ x \in HInfinite;\ 0 \leq x\ |] \Longrightarrow (\ *f*\ exp)\ (x::hypreal) \in HInfinite$
⟨*proof*⟩

lemma *starfun-exp-minus*: $!!x.\ (\ *f*\ exp)\ (-x) = inverse((\ *f*\ exp)\ x)$
⟨*proof*⟩

lemma *starfun-exp-Infinitesimal*:
$[|\ x \in HInfinite;\ x \leq 0\ |] \Longrightarrow (\ *f*\ exp)\ (x::hypreal) \in Infinitesimal$
⟨*proof*⟩

lemma *starfun-exp-gt-one* [*simp*]: $!!x::hypreal.\ 0 < x \Longrightarrow 1 < (\ *f*\ exp)\ x$
⟨*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*: $!!u\ x.\ (\ *f*\ exp)\ u = x \Longrightarrow (\ *f*\ ln)\ x = u$
⟨*proof*⟩

lemma *starfun-ln-less-self* [*simp*]: $!!x.\ 0 < x \Longrightarrow (\ *f*\ ln)\ x < x$
⟨*proof*⟩

lemma *starfun-ln-ge-zero* [*simp*]: $!!x.\ 1 \leq x \Longrightarrow 0 \leq (\ *f*\ ln)\ x$

⟨*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; $x \neq 1$ |] ==> ($*f*$ *ln*) $x \neq 0$
⟨*proof*⟩

lemma *starfun-ln-HFinite*: [| $x \in$ *HFinite*; 1 ≤ x |] ==> ($*f*$ *ln*) $x \in$ *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*) ∈ *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*) ∈ *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* ∈ *Infinitesimal*; *0* < *x* |] ==> (*f* *ln*) *x* < *0*
⟨*proof*⟩

lemma *starfun-ln-HInfinite-gt-zero*:
[| *x* ∈ *HInfinite*; *0* < *x* |] ==> *0* < (*f* *ln*) *x*
⟨*proof*⟩

lemma *HFinite-sin* [*simp*]:
sumhr (*0*, *whn*, %*n*. (*if even*(*n*) *then 0 else*
(−*1* ^ ((*n* − *1*) *div 2*))/(*real* (*fact n*))) * *x* ^ *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* ^ (*n div 2*))/(*real* (*fact n*)) *else*
0) * *x* ^ *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* ^ *2*
⟨*proof*⟩

lemma *STAR-cos-Infinitesimal-approx2*:
x ∈ *Infinitesimal* ==> (∗*f*∗ *cos*) *x* @= *1* − (*x* ^ *2*)/*2*
⟨*proof*⟩

end

46 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*)

46.1 Closure Laws for SComplex, the Standard Complex Numbers

lemma *SComplex-minus-iff* [*simp*]: $(-x \in SComplex) = (x \in SComplex)$
⟨*proof*⟩

lemma *SComplex-add-cancel*:
$[\!| x + y \in SComplex; y \in SComplex |\!] ==> x \in SComplex$
⟨*proof*⟩

lemma *SReal-hcmod-hcomplex-of-complex* [*simp*]:
$hcmod\ (hcomplex\text{-}of\text{-}complex\ r) \in Reals$
⟨*proof*⟩

lemma *SReal-hcmod-number-of* [*simp*]: $hcmod\ (number\text{-}of\ w :: hcomplex) \in Reals$
⟨*proof*⟩

lemma *SReal-hcmod-SComplex*: $x \in SComplex ==> hcmod\ x \in Reals$
⟨*proof*⟩

lemma *SComplex-divide-number-of*:
$r \in SComplex ==> r/(number\text{-}of\ w :: hcomplex) \in SComplex$
⟨*proof*⟩

lemma *SComplex-UNIV-complex*:
$\{x.\ hcomplex\text{-}of\text{-}complex\ x \in SComplex\} = (UNIV :: complex\ set)$
⟨*proof*⟩

lemma *SComplex-iff*: $(x \in SComplex) = (\exists y.\ x = hcomplex\text{-}of\text{-}complex\ y)$
⟨*proof*⟩

lemma *hcomplex-of-complex-image*:
$hcomplex\text{-}of\text{-}complex\ `(UNIV :: complex\ set) = SComplex$
⟨*proof*⟩

lemma *inv-hcomplex-of-complex-image*: $inv\ hcomplex\text{-}of\text{-}complex\ `SComplex = UNIV$
⟨*proof*⟩

lemma *SComplex-hcomplex-of-complex-image*:
$[\!| \exists x.\ x : P; P \leq SComplex |\!] ==> \exists Q.\ P = hcomplex\text{-}of\text{-}complex\ `\ Q$
⟨*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*⟩

46.2 The Finite Elements form a Subring

lemma *HFinite-hcmod-hcomplex-of-complex* [*simp*]:
$hcmod\ (hcomplex\text{-}of\text{-}complex\ r) \in HFinite$
⟨*proof*⟩

lemma *HFinite-hcmod-iff*: $(x \in HFinite) = (hcmod\ x \in HFinite)$
⟨*proof*⟩

lemma *HFinite-bounded-hcmod*:
$[|x \in HFinite;\ y \leq hcmod\ x;\ 0 \leq y\ |] ==> y \in HFinite$
⟨*proof*⟩

46.3 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*⟩

46.4 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*⟩

46.5 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-of-complex $x \neq 0$
==> *hcomplex-of-complex* $x \in HFinite - Infinitesimal$
⟨*proof*⟩

lemma *number-of-not-Infinitesimal* [*simp*]:
number-of $w \neq (0::hcomplex)$ ==> (*number-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-of* w :: *hcomplex*) $\in$ *Infinitesimal*) =
(*number-of* $w = (0::hcomplex)$)
⟨*proof*⟩

lemma *approx-unique-complex*:
[| $r \in SComplex$; $s \in SComplex$; r @= x; s @= x|] ==> $r = s$
⟨*proof*⟩

46.6 Properties of *hRe*, *hIm* and *HComplex*

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*: $[\![0 < u;\ x < u^2]\!] \Longrightarrow sqrt\ x < u$

⟨*proof*⟩

lemma *hypreal-sqrt-lessI*:
$\bigwedge x\ u.\ [\![0 < u;\ x < u^2]\!] \Longrightarrow (\ *f*\ sqrt)\ x < u$
⟨*proof*⟩

lemma *hypreal-sqrt-ge-zero*: $\bigwedge x.\ 0 \leq x \Longrightarrow 0 \leq (\ *f*\ sqrt)\ x$
⟨*proof*⟩

lemma *Infinitesimal-sqrt*:
$[\![x \in Infinitesimal;\ 0 \leq x]\!] \Longrightarrow (\ *f*\ sqrt)\ x \in Infinitesimal$
⟨*proof*⟩

lemma *Infinitesimal-HComplex*:
$[\![x \in Infinitesimal;\ y \in Infinitesimal]\!] \Longrightarrow HComplex\ x\ y \in Infinitesimal$
⟨*proof*⟩

lemma *hcomplex-Infinitesimal-iff*:
$(x \in Infinitesimal) = (hRe\ x \in Infinitesimal \wedge hIm\ x \in Infinitesimal)$
⟨*proof*⟩

lemma *hRe-diff* [*simp*]: $\bigwedge x\ y.\ hRe\ (x - y) = hRe\ x - hRe\ y$
⟨*proof*⟩

lemma *hIm-diff* [*simp*]: $\bigwedge x\ y.\ hIm\ (x - y) = hIm\ x - hIm\ y$
⟨*proof*⟩

lemma *approx-hRe*: $x \approx y \Longrightarrow hRe\ x \approx hRe\ y$
⟨*proof*⟩

lemma *approx-hIm*: $x \approx y \Longrightarrow hIm\ x \approx hIm\ y$
⟨*proof*⟩

lemma *approx-HComplex*:
$[\![a \approx b;\ c \approx d]\!] \Longrightarrow HComplex\ a\ c \approx HComplex\ b\ d$
⟨*proof*⟩

lemma *hcomplex-approx-iff*:
$(x \approx y) = (hRe\ x \approx hRe\ y \wedge hIm\ x \approx hIm\ y)$
⟨*proof*⟩

lemma *HFinite-hRe*: $x \in HFinite \Longrightarrow hRe\ x \in HFinite$
⟨*proof*⟩

lemma *HFinite-hIm*: $x \in HFinite \Longrightarrow hIm\ x \in HFinite$
⟨*proof*⟩

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*]:
$(hcomplex\text{-}of\text{-}hypreal\ x\ @=\ hcomplex\text{-}of\text{-}hypreal\ z) = (x\ @=\ z)$
⟨*proof*⟩

lemma *Standard-HComplex*:
$[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow HComplex\ x\ y \in Standard$
⟨*proof*⟩

lemma *stc-part-Ex*: $x{:}HFinite \Longrightarrow \exists t \in SComplex.\ x\ @=\ t$
⟨*proof*⟩

lemma *stc-part-Ex1*: $x{:}HFinite \Longrightarrow EX!\ t.\ t \in SComplex\ \&\ \ x\ @=\ t$
⟨*proof*⟩

lemmas *hcomplex-of-complex-approx-inverse* =
hcomplex-of-complex-HFinite-diff-Infinitesimal [*THEN* [*2*] *approx-inverse*]

46.7 Theorems About Monads

lemma *monad-zero-hcmod-iff*: $(x \in monad\ 0) = (hcmod\ x{:}monad\ 0)$
⟨*proof*⟩

46.8 Theorems About Standard Part

lemma *stc-approx-self*: $x \in HFinite \Longrightarrow stc\ x\ @=\ x$
⟨*proof*⟩

lemma *stc-SComplex*: $x \in HFinite \Longrightarrow stc\ x \in SComplex$
⟨*proof*⟩

lemma *stc-HFinite*: $x \in HFinite \Longrightarrow stc\ x \in HFinite$
⟨*proof*⟩

lemma *stc-unique*: $[\![y \in SComplex;\ y \approx x]\!] \Longrightarrow stc\ x = y$
⟨*proof*⟩

lemma *stc-SComplex-eq* [*simp*]: $x \in SComplex \Longrightarrow stc\ x = x$

⟨*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

47 CStar: Star-transforms in NSA, Extending Sets of Complex Numbers and Complex Functions

theory *CStar*
imports *NSCA*
begin

47.1 Properties of the *-Transform Applied to Sets of Reals

lemma *STARC-hcomplex-of-complex-Int*:
$*s* X\ Int\ SComplex = hcomplex\text{-}of\text{-}complex\ `\ X$
⟨*proof*⟩

lemma *lemma-not-hcomplexA*:
$x \notin hcomplex\text{-}of\text{-}complex\ `\ A \Longrightarrow \forall y \in A.\ x \neq hcomplex\text{-}of\text{-}complex\ y$
⟨*proof*⟩

47.2 Theorems about Nonstandard Extensions of Functions

lemma *starfunC-hcpow*: $!!Z.\ (\ *f* (\%z.\ z\ \hat{}\ n))\ Z = Z\ pow\ hypnat\text{-}of\text{-}nat\ n$
⟨*proof*⟩

lemma *starfunCR-cmod*: $*f*\ cmod = hcmod$
⟨*proof*⟩

47.3 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

48 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*⟩

48.1 Limit of Complex to Complex Function

lemma *NSLIM-Re*: $f\ -{-}\ a\ -{-}NS{>}\ L \Longrightarrow (\%x.\ Re(f\ x))\ -{-}\ a\ -{-}NS{>}\ Re(L)$
⟨*proof*⟩

lemma *NSLIM-Im*: $f\ -{-}\ a\ -{-}NS{>}\ L \Longrightarrow (\%x.\ Im(f\ x))\ -{-}\ a\ -{-}NS{>}\ Im(L)$
⟨*proof*⟩

lemma *LIM-Re*: $f\ -{-}\ a\ -{-}{>}\ L \Longrightarrow (\%x.\ Re(f\ x))\ -{-}\ a\ -{-}{>}\ Re(L)$
⟨*proof*⟩

lemma *LIM-Im*: $f\ -{-}\ a\ -{-}{>}\ L \Longrightarrow (\%x.\ Im(f\ x))\ -{-}\ a\ -{-}{>}\ Im(L)$
⟨*proof*⟩

lemma *LIM-cnj*: $f\ -{-}\ a\ -{-}{>}\ L \Longrightarrow (\%x.\ cnj\ (f\ x))\ -{-}\ a\ -{-}{>}\ cnj\ L$
⟨*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*⟩

48.2 Continuity

lemma *NSLIM-isContc-iff*:
(*f* −− *a* −−*NS*> *f a*) = ((%*h*. *f*(*a* + *h*)) −− *0* −−*NS*> *f a*)
⟨*proof*⟩

48.3 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*⟩

48.4 Differentiation of Natural Number Powers

lemma *CDERIV-pow* [*simp*]:
DERIV (%*x*. *x* ^ *n*) *x* :> (*complex-of-real* (*real n*)) * (*x* ^ (*n* − *Suc 0*))
⟨*proof*⟩

Nonstandard version

lemma *NSCDERIV-pow*:
NSDERIV (%*x*. *x* ^ *n*) *x* :> *complex-of-real* (*real n*) * (*x* ^ (*n* − *1*))
⟨*proof*⟩

Can't relax the premise $x \neq (0::'a)$: it isn't continuous at zero

lemma *NSCDERIV-inverse*:
(*x*::*complex*) ≠ *0* ==> *NSDERIV* (%*x*. *inverse*(*x*)) *x* :> (− (*inverse x* ^ *2*))
⟨*proof*⟩

lemma *CDERIV-inverse*:
(*x*::*complex*) ≠ *0* ==> *DERIV* (%*x*. *inverse*(*x*)) *x* :> (−(*inverse x* ^ *2*))
⟨*proof*⟩

48.5 Derivative of Reciprocals (Function *inverse*)

lemma *CDERIV-inverse-fun*:
[| *DERIV f x* :> *d*; *f*(*x*) ≠ (*0*::*complex*) |]
==> *DERIV* (%*x*. *inverse*(*f x*)) *x* :> (− (*d* * *inverse*(*f*(*x*) ^ *2*)))
⟨*proof*⟩

lemma *NSCDERIV-inverse-fun*:
[| *NSDERIV f x* :> *d*; *f*(*x*) ≠ (*0*::*complex*) |]
==> *NSDERIV* (%*x*. *inverse*(*f x*)) *x* :> (− (*d* * *inverse*(*f*(*x*) ^ *2*)))
⟨*proof*⟩

48.6 Derivative of Quotient

lemma *CDERIV-quotient*:
[| *DERIV f x* :> *d*; *DERIV g x* :> *e*; *g*(*x*) ≠ (*0*::*complex*) |]
==> *DERIV* (%*y*. *f*(*y*) / (*g y*)) *x* :> (*d*∗*g*(*x*) − (*e*∗*f*(*x*))) / (*g*(*x*) ^ *2*)
⟨*proof*⟩

lemma *NSCDERIV-quotient*:
[| *NSDERIV f x* :> *d*; *NSDERIV g x* :> *e*; *g*(*x*) ≠ (*0*::*complex*) |]
==> *NSDERIV* (%*y*. *f*(*y*) / (*g y*)) *x* :> (*d*∗*g*(*x*) − (*e*∗*f*(*x*))) / (*g*(*x*) ^ *2*)
⟨*proof*⟩

48.7 Caratheodory Formulation of Derivative at a Point: Standard Proof

lemma *CARAT-CDERIVD*:
(∀ *z*. *f z* − *f x* = *g z* * (*z* − *x*)) & *isNSCont g x* & *g x* = *l*
==> *NSDERIV f x* :> *l*

⟨*proof*⟩

end

49 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*: $-1 < x \Longrightarrow ln(1 + x) <= x$
⟨*proof*⟩

lemma *abs-ln-one-plus-x-minus-x-bound-nonneg*:
$0 <= x \Longrightarrow x <= 1 \Longrightarrow abs(ln\ (1 + x) - x) <= x\hat{}2$
⟨*proof*⟩

lemma *abs-ln-one-plus-x-minus-x-bound-nonpos*:
$-(1\ /\ 2) <= x \Longrightarrow x <= 0 \Longrightarrow abs(ln\ (1 + x) - x) <= 2 * x\hat{}2$
⟨*proof*⟩

lemma *abs-ln-one-plus-x-minus-x-bound*:
$abs\ x <= 1\ /\ 2 \Longrightarrow abs(ln\ (1 + x) - x) <= 2 * x\hat{}2$
⟨*proof*⟩

lemma *DERIV-ln*: $0 < x \Longrightarrow DERIV\ ln\ x :> 1\ /\ x$
⟨*proof*⟩

lemma *ln-x-over-x-mono*: $exp\ 1 <= x \Longrightarrow x <= y \Longrightarrow (ln\ y\ /\ y) <= (ln\ x\ /\ x)$
⟨*proof*⟩

end

50 MacLaurin: MacLaurin Series

theory *MacLaurin*
imports *Dense-Linear-Order Transcendental*
begin

50.1 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*:
$0 < h \Longrightarrow$
$\exists B.\ f\ h = (\sum m{=}0..{<}n.\ (j\ m\ /\ real\ (fact\ m)) * (h\hat{}m)) +$
$(B * ((h\hat{}n)\ /\ real(fact\ n)))$
⟨*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*:
[| ∀ *m t*. *m* < *n* ∧ 0≤*t* ∧ *t*≤*h* ⟶ *DERIV* (*diff m*) *t* :> *diff* (*Suc m*) *t*;
n = *Suc k*;
difg =
(λ*m t*. *diff m t* −
((∑ *p* = 0..<*n* − *m*. *diff* (*m* + *p*) 0 / *real* (*fact p*) ∗ *t* ^ *p*) +
B ∗ (*t* ^ (*n* − *m*) / *real* (*fact* (*n* − *m*))))|] ==>
∀ *m t*. *m* < *n* & 0 ≤ *t* & *t* ≤ *h* −−>
DERIV (*difg m*) *t* :> *difg* (*Suc m*) *t*
⟨*proof*⟩

lemma *Maclaurin-lemma3*:
fixes *difg* :: *nat* => *real* => *real* **shows**
[|∀ *k t*. *k* < *Suc m* ∧ 0≤*t* & *t*≤*h* ⟶ *DERIV* (*difg k*) *t* :> *difg* (*Suc k*) *t*;
∀ *k*<*Suc m*. *difg k* 0 = 0; *DERIV* (*difg n*) *t* :> 0; *n* < *m*; 0 < *t*;
t < *h*|]
==> ∃ *ta*. 0 < *ta* & *ta* < *t* & *DERIV* (*difg* (*Suc n*)) *ta* :> 0
⟨*proof*⟩

lemma *Maclaurin*:
[| 0 < *h*; *n* > 0; *diff* 0 = *f*;
∀ *m t*. *m* < *n* & 0 ≤ *t* & *t* ≤ *h* −−> *DERIV* (*diff m*) *t* :> *diff* (*Suc m*) *t* |]
==> ∃ *t*. 0 < *t* &
t < *h* &
f h =
setsum (%*m*. (*diff m* 0 / *real* (*fact m*)) ∗ *h* ^ *m*) {0..<*n*} +
(*diff n t* / *real* (*fact n*)) ∗ *h* ^ *n*
⟨*proof*⟩

lemma *Maclaurin-objl*:
0 < *h* & *n*>0 & *diff* 0 = *f* &
(∀ *m t*. *m* < *n* & 0 ≤ *t* & *t* ≤ *h* −−> *DERIV* (*diff m*) *t* :> *diff* (*Suc m*) *t*)
−−> (∃ *t*. 0 < *t* & *t* < *h* &
f h = (∑ *m*=0..<*n*. *diff m* 0 / *real* (*fact m*) ∗ *h* ^ *m*) +
diff n t / *real* (*fact n*) ∗ *h* ^ *n*)
⟨*proof*⟩

lemma *Maclaurin2*:

$[\!|\ 0 < h;\ diff\ 0 = f;$
$\forall m\ t.$
$m < n\ \&\ 0 \leq t\ \&\ t \leq h \longrightarrow DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t\ |\!]$
$\Longrightarrow \exists t.\ 0 < t\ \&$
$t \leq h\ \&$
$f\ h =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n$

⟨*proof*⟩

lemma *Maclaurin2-objl*:
$0 < h\ \&\ diff\ 0 = f\ \&$
$(\forall m\ t.$
$m < n\ \&\ 0 \leq t\ \&\ t \leq h \longrightarrow DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t)$
$\longrightarrow (\exists t.\ 0 < t\ \&$
$t \leq h\ \&$
$f\ h =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n)$

⟨*proof*⟩

lemma *Maclaurin-minus*:
$[\!|\ h < 0;\ n > 0;\ diff\ 0 = f;$
$\forall m\ t.\ m < n\ \&\ h \leq t\ \&\ t \leq 0 \longrightarrow DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t\ |\!]$
$\Longrightarrow \exists t.\ h < t\ \&$
$t < 0\ \&$
$f\ h =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n$

⟨*proof*⟩

lemma *Maclaurin-minus-objl*:
$(h < 0\ \&\ n > 0\ \&\ diff\ 0 = f\ \&$
$(\forall m\ t.$
$m < n\ \&\ h \leq t\ \&\ t \leq 0 \longrightarrow DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t))$
$\longrightarrow (\exists t.\ h < t\ \&$
$t < 0\ \&$
$f\ h =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n)$

⟨*proof*⟩

50.2 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*;
∀ *m t*. *m < n & abs t ≤ abs x −−> DERIV (diff m) t :> diff (Suc m) t* |]
==> ∃ *t*. *abs t ≤ abs x &*
f x =
*(∑ m=0..<n. diff m 0 / real (fact m) * x ^ m) +*
*diff n t / real (fact n) * x ^ n*
⟨*proof*⟩

lemma *Maclaurin-all-lt*:
[| *diff 0 = f*;
∀ *m x*. *DERIV (diff m) x :> diff(Suc m) x*;
x ~= 0; *n > 0*
|] ==> ∃ *t*. *0 < abs t & abs t < abs x &*
*f x = (∑ m=0..<n. (diff m 0 / real (fact m)) * x ^ m) +*
*(diff n t / real (fact n)) * x ^ n*
⟨*proof*⟩

lemma *Maclaurin-all-lt-objl*:
diff 0 = f &
(∀ *m x*. *DERIV (diff m) x :> diff(Suc m) x*) &
x ~= 0 & n > 0
−−> (∃ *t*. *0 < abs t & abs t < abs x &*
*f x = (∑ m=0..<n. (diff m 0 / real (fact m)) * x ^ m) +*
*(diff n t / real (fact n)) * x ^ n*)
⟨*proof*⟩

lemma *Maclaurin-zero* [*rule-format*]:
x = (0::real)
==> *n ≠ 0* −−>
*(∑ m=0..<n. (diff m (0::real) / real (fact m)) * x ^ m) =*
diff 0 0
⟨*proof*⟩

lemma *Maclaurin-all-le*: [| *diff 0 = f*;
∀ *m x*. *DERIV (diff m) x :> diff (Suc m) x*
|] ==> ∃ *t*. *abs t ≤ abs x &*
*f x = (∑ m=0..<n. (diff m 0 / real (fact m)) * x ^ m) +*
*(diff n t / real (fact n)) * x ^ n*
⟨*proof*⟩

lemma *Maclaurin-all-le-objl*: *diff 0 = f &*
(∀ *m x*. *DERIV (diff m) x :> diff (Suc m) x*)
−−> (∃ *t*. *abs t ≤ abs x &*
*f x = (∑ m=0..<n. (diff m 0 / real (fact m)) * x ^ m) +*
*(diff n t / real (fact n)) * x ^ n*)
⟨*proof*⟩

50.3 Version for Exponential Function

lemma *Maclaurin-exp-lt*: [| $x \sim= 0$; $n > 0$ |]
$\Longrightarrow$ ($\exists t.\ 0 < abs\ t$ &
$abs\ t < abs\ x$ &
$exp\ x = (\sum m{=}0..{<}n.\ (x \hat{}\ m) / real\ (fact\ m)) +$
$(exp\ t / real\ (fact\ n)) * x \hat{}\ n$)
⟨*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*⟩

50.4 Version for Sine Function

lemma *MVT2*:
[| $a < b$; $\forall x.\ a \leq x$ & $x \leq b$ −−> $DERIV\ f\ x :> f'(x)$ |]
$\Longrightarrow \exists z{::}real.\ a < z$ & $z < b$ & $(f\ b - f\ a = (b - a) * f'(z))$
⟨*proof*⟩

lemma *mod-exhaust-less-4*:
$m\ mod\ 4 = 0 \mid m\ mod\ 4 = 1 \mid m\ mod\ 4 = 2 \mid m\ mod\ 4 = (3{::}nat)$
⟨*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*:
$\exists t.\ abs\ t \leq abs\ x$ &
$sin\ x =$
$(\sum m{=}0..{<}n.\ (if\ even\ m\ then\ 0$
$else\ (-1 \hat{}\ ((m - Suc\ 0)\ div\ 2)) / real\ (fact\ m)) *$
$x \hat{}\ m)$
$+ ((sin(t + 1/2 * real\ (n) * pi) / real\ (fact\ n)) * x \hat{}\ n)$
⟨*proof*⟩

lemma *Maclaurin-sin-expansion*:

$\exists t.\ sin\ x =$
$(\sum m{=}0..{<}n.\ (if\ even\ m\ then\ 0$
$else\ (-1\ \hat{}\ ((m - Suc\ 0)\ div\ 2))\ /\ real\ (fact\ m))\ *$
$x\ \hat{}\ m)$
$+\ ((sin(t + 1/2 * real\ (n)\ *pi)\ /\ real\ (fact\ n))\ *\ x\ \hat{}\ n)$
⟨*proof*⟩

lemma *Maclaurin-sin-expansion3*:
$[|\ n > 0;\ 0 < x\ |] ==>$
$\exists t.\ 0 < t\ \&\ t < x\ \&$
$sin\ x =$
$(\sum m{=}0..{<}n.\ (if\ even\ m\ then\ 0$
$else\ (-1\ \hat{}\ ((m - Suc\ 0)\ div\ 2))\ /\ real\ (fact\ m))\ *$
$x\ \hat{}\ m)$
$+\ ((sin(t + 1/2 * real(n)\ *pi)\ /\ real\ (fact\ n))\ *\ x\ \hat{}\ n)$
⟨*proof*⟩

lemma *Maclaurin-sin-expansion4*:
$0 < x ==>$
$\exists t.\ 0 < t\ \&\ t \leq x\ \&$
$sin\ x =$
$(\sum m{=}0..{<}n.\ (if\ even\ m\ then\ 0$
$else\ (-1\ \hat{}\ ((m - Suc\ 0)\ div\ 2))\ /\ real\ (fact\ m))\ *$
$x\ \hat{}\ m)$
$+\ ((sin(t + 1/2 * real\ (n)\ *pi)\ /\ real\ (fact\ n))\ *\ x\ \hat{}\ n)$
⟨*proof*⟩

50.5 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*:
$[|\ 0 < x;\ n > 0\ |] ==>$
$\exists t.\ 0 < t\ \&\ t < x\ \&$
$cos\ x =$

$$(\sum m{=}0..{<}n.\ (\mathit{if}\ \mathit{even}\ m\ \mathit{then}\ -1\ \hat{}\ (m\ \mathit{div}\ 2)/(\mathit{real}\ (\mathit{fact}\ m))\ \mathit{else}\ 0)\ *\ x\ \hat{}\ m) + ((\mathit{cos}(t + 1/2 * \mathit{real}\ (n)\ *\mathit{pi})\ /\ \mathit{real}\ (\mathit{fact}\ n)) * x\ \hat{}\ n)$$

⟨*proof*⟩

lemma *Maclaurin-minus-cos-expansion*:

$$[|\ x < 0;\ n > 0\ |] ==> \exists t.\ x < t\ \&\ t < 0\ \&\ \mathit{cos}\ x = (\sum m{=}0..{<}n.\ (\mathit{if}\ \mathit{even}\ m\ \mathit{then}\ -1\ \hat{}\ (m\ \mathit{div}\ 2)/(\mathit{real}\ (\mathit{fact}\ m))\ \mathit{else}\ 0)\ *\ x\ \hat{}\ m) + ((\mathit{cos}(t + 1/2 * \mathit{real}\ (n)\ *\mathit{pi})\ /\ \mathit{real}\ (\mathit{fact}\ n)) * x\ \hat{}\ n)$$

⟨*proof*⟩

lemma *sin-bound-lemma*:

$$[|x = y;\ \mathit{abs}\ u \leq (v{::}\mathit{real})\ |] ==> |(x + u) - y| \leq v$$

⟨*proof*⟩

lemma *Maclaurin-sin-bound*:

$$\mathit{abs}(\mathit{sin}\ x - (\sum m{=}0..{<}n.\ (\mathit{if}\ \mathit{even}\ m\ \mathit{then}\ 0\ \mathit{else}\ (-1\ \hat{}\ ((m - \mathit{Suc}\ 0)\ \mathit{div}\ 2))\ /\ \mathit{real}\ (\mathit{fact}\ m))\ *\ x\ \hat{}\ m)) \leq \mathit{inverse}(\mathit{real}\ (\mathit{fact}\ n)) * |x|\ \hat{}\ n$$

⟨*proof*⟩

end

51 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*: $n{>}0$ $\mathit{diff}\ 0 = f$
and *DERIV*: $(\forall\ m\ t.\ m < n\ \&\ a \leq t\ \&\ t \leq b \longrightarrow \mathit{DERIV}\ (\mathit{diff}\ m)\ t :> (\mathit{diff}\ (\mathit{Suc}\ m)\ t))$
and *INTERV*: $a \leq c\ c < b$
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*: $(\forall\ m\ t.\ m < n\ \&\ a \leq t\ \&\ t \leq b \longrightarrow DERIV\ (diff\ m)\ t :> (diff\ (Suc\ m)\ t))$
and *INTERV*: $a < c\ c \leq b$
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*: $(\forall\ m\ t.\ m < n\ \&\ a \leq t\ \&\ t \leq b \longrightarrow DERIV\ (diff\ m)\ t :> (diff\ (Suc\ m)\ t))$
and *INTERV*: $a \leq c\ \ c \leq b\ a \leq x\ x \leq b\ x \neq c$
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

52 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 = $(\%(a,b)\ D.\ D\ 0 = a\ \&$
$(\exists N.\ (\forall n < N.\ D(n) < D(Suc\ n))\ \&$
$(\forall n \geq N.\ D(n) = b)))$

definition
psize :: (*nat* => *real*) => *nat* **where**
psize D = $(SOME\ N.\ (\forall n < N.\ D(n) < D(Suc\ n))\ \&$
$(\forall n \geq N.\ D(n) = D(N)))$

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* &
(∀ *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 ≤ *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*) &
(∀ *n* < *psize D*. *D n* < *D*(*Suc n*)) &
(∀ *n* ≥ *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* ≤ *n* |] ==> (*D n* = *b*)
⟨*proof*⟩

lemma *lemma-partition-lt-gen* [*rule-format*]:
partition(*a*,*b*) *D* & *m* + *Suc d* ≤ *n* & *n* ≤ (*psize D*) −−> *D*(*m*) < *D*(*m* + *Suc d*)
⟨*proof*⟩

lemma *less-eq-add-Suc*: *m* < *n* ==> ∃ *d*. *n* = *m* + *Suc d*
⟨*proof*⟩

lemma *partition-lt-gen*:
[|*partition*(*a*,*b*) *D*; *m* < *n*; *n* ≤ (*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* ≤ *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* ≤ *D*(*r*)
⟨*proof*⟩

lemma *partition-lb-lt*: [| *partition*(*a*,*b*) *D*; *psize D* ~= *0* |] ==> *a* < *D*(*Suc n*)
⟨*proof*⟩

lemma *partition-ub*: *partition*(*a*,*b*) *D* ==> *D*(*r*) ≤ *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* |]
==> (∀ *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*))) &
(∀ *n* ≥ *psize D1* + *psize D2*.

$(\mathit{if}\ n < \mathit{psize}\ D1\ \mathit{then}\ D1\ n\ \mathit{else}\ D2\ (n - \mathit{psize}\ D1)) =$
$(\mathit{if}\ \mathit{psize}\ D1 + \mathit{psize}\ D2 < \mathit{psize}\ D1\ \mathit{then}\ D1\ (\mathit{psize}\ D1 + \mathit{psize}\ D2)$
$\mathit{else}\ D2\ (\mathit{psize}\ D1 + \mathit{psize}\ D2 - \mathit{psize}\ D1)))$
⟨*proof*⟩

lemma *lemma-psize1*:
$[|\ \mathit{partition}\ (a,\ b)\ D1;\ \mathit{partition}\ (b,\ c)\ D2;\ N < \mathit{psize}\ D1\ |]$
$\Longrightarrow D1(N) < D2\ (\mathit{psize}\ D2)$
⟨*proof*⟩

lemma *lemma-partition-append2*:
$[|\ \mathit{partition}\ (a,\ b)\ D1;\ \mathit{partition}\ (b,\ c)\ D2\ |]$
$\Longrightarrow \mathit{psize}\ (\%n.\ \mathit{if}\ n < \mathit{psize}\ D1\ \mathit{then}\ D1\ n\ \mathit{else}\ D2\ (n - \mathit{psize}\ D1)) =$
$\mathit{psize}\ D1 + \mathit{psize}\ D2$
⟨*proof*⟩

lemma *tpart-eq-lhs-rhs*: $[|\mathit{psize}\ D = 0;\ \mathit{tpart}(a,b)\ (D,p)|] \Longrightarrow a = b$
⟨*proof*⟩

lemma *tpart-partition*: $\mathit{tpart}(a,b)\ (D,p) \Longrightarrow \mathit{partition}(a,b)\ D$
⟨*proof*⟩

lemma *partition-append*:
$[|\ \mathit{tpart}(a,b)\ (D1,p1);\ \mathit{fine}(g)\ (D1,p1);$
$\mathit{tpart}(b,c)\ (D2,p2);\ \mathit{fine}(g)\ (D2,p2)\ |]$
$\Longrightarrow \exists D\ p.\ \mathit{tpart}(a,c)\ (D,p)\ \&\ \mathit{fine}(g)\ (D,p)$
⟨*proof*⟩

We can always find a division that is fine wrt any gauge

lemma *partition-exists*:
$[|\ a \le b;\ \mathit{gauge}(\%x.\ a \le x\ \&\ x \le b)\ g\ |]$
$\Longrightarrow \exists D\ p.\ \mathit{tpart}(a,b)\ (D,p)\ \&\ \mathit{fine}\ g\ (D,p)$
⟨*proof*⟩

Lemmas about combining gauges

lemma *gauge-min*:
$[|\ \mathit{gauge}(E)\ g1;\ \mathit{gauge}(E)\ g2\ |]$
$\Longrightarrow \mathit{gauge}(E)\ (\%x.\ \mathit{if}\ g1(x) < g2(x)\ \mathit{then}\ g1(x)\ \mathit{else}\ g2(x))$
⟨*proof*⟩

lemma *fine-min*:
$\mathit{fine}\ (\%x.\ \mathit{if}\ g1(x) < g2(x)\ \mathit{then}\ g1(x)\ \mathit{else}\ g2(x))\ (D,p)$
$\Longrightarrow \mathit{fine}(g1)\ (D,p)\ \&\ \mathit{fine}(g2)\ (D,p)$
⟨*proof*⟩

The integral is unique if it exists

lemma *Integral-unique*:
$[|\ a \le b;\ \mathit{Integral}(a,b)\ f\ k1;\ \mathit{Integral}(a,b)\ f\ k2\ |] \Longrightarrow k1 = k2$
⟨*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*: $a \leq b$ ==> *Integral*(*a*,*b*) (%*x*. *1*) (*b* − *a*)
⟨*proof*⟩

lemma *Integral-mult-const*: $a \leq b$ ==> *Integral*(*a*,*b*) (%*x*. *c*) (*c*∗(*b* − *a*))
⟨*proof*⟩

lemma *Integral-mult*:
[| $a \leq b$; *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*: ∀ *x*. *P*(*x*) −−> (∃ *y*. *Q x y*) ==> ∃*f*. (∀ *x*. *P*(*x*) −−> *Q x* (*f x*))
⟨*proof*⟩

lemma *strad1*:
⟦∀ *xa*::*real*. *xa* ≠ *x* ∧ |*xa* − *x*| < *s* ⟶
|(*f xa* − *f x*) / (*xa* − *x*) − *f*′ *x*| ∗ *2* < *e*;
0 < *e*; *a* ≤ *x*; *x* ≤ *b*; *0* < *s*⟧
⟹ ∀ *z*. |*z* − *x*| < *s* −−>|*f z* − *f x* − *f*′ *x* ∗ (*z* − *x*)| ∗ *2* ≤ *e* ∗ |*z* − *x*|
⟨*proof*⟩

lemma *lemma-straddle*:
[| ∀ *x*. *a* ≤ *x* & *x* ≤ *b* −−> *DERIV f x* :> *f*′(*x*); *0* < *e* |]
==> ∃ *g*. *gauge*(%*x*. *a* ≤ *x* & *x* ≤ *b*) *g* &
(∀ *x u v*. *a* ≤ *u* & *u* ≤ *x* & *x* ≤ *v* & *v* ≤ *b* & (*v* − *u*) < *g*(*x*)
−−> |(*f*(*v*) − *f*(*u*)) − (*f*′(*x*) ∗ (*v* − *u*))| ≤ *e* ∗ (*v* − *u*))
⟨*proof*⟩

lemma *FTC1*: [|*a* ≤ *b*; ∀ *x*. *a* ≤ *x* & *x* ≤ *b* −−> *DERIV f x* :> *f*′(*x*) |]
==> *Integral*(*a*,*b*) *f*′ (*f*(*b*) − *f*(*a*))
⟨*proof*⟩

lemma *Integral-subst*: [| *Integral*(*a*,*b*) *f k1*; *k2*=*k1* |] ==> *Integral*(*a*,*b*) *f k2*
⟨*proof*⟩

lemma *Integral-add*:
[| $a \leq b$; $b \leq c$; *Integral*(a,b) f' *k1*; *Integral*(b,c) f' *k2*;
$\forall x.\ a \leq x\ \&\ x \leq c$ −−> *DERIV* $f\ x$:> $f'\ x$ |]
==> *Integral*(a,c) f' $(k1 + k2)$
⟨*proof*⟩

lemma *partition-psize-Least*:
partition(a,b) D ==> *psize* $D = (LEAST\ n.\ D(n) = b)$
⟨*proof*⟩

lemma *lemma-partition-bounded*: *partition* $(a,\ c)$ D ==> ~ $(\exists\, n.\ c < D(n))$
⟨*proof*⟩

lemma *lemma-partition-eq*:
partition $(a,\ c)$ D ==> $D = (\%n.$ *if* $D\ n < c$ *then* $D\ n$ *else* $c)$
⟨*proof*⟩

lemma *lemma-partition-eq2*:
partition $(a,\ c)$ D ==> $D = (\%n.$ *if* $D\ n \leq c$ *then* $D\ n$ *else* $c)$
⟨*proof*⟩

lemma *partition-lt-Suc*:
[| *partition*(a,b) D; $n <$ *psize* D |] ==> $D\ n < D\ (Suc\ n)$
⟨*proof*⟩

lemma *tpart-tag-eq*: *tpart*(a,c) (D,p) ==> $p = (\%n.$ *if* $D\ n < c$ *then* $p\ n$ *else* $c)$
⟨*proof*⟩

52.1 Lemmas for Additivity Theorem of Gauge Integral

lemma *lemma-additivity1*:
[| $a \leq D\ n$; $D\ n < b$; *partition*(a,b) D |] ==> $n <$ *psize* D
⟨*proof*⟩

lemma *lemma-additivity2*: [| $a \leq D\ n$; *partition*$(a,D\ n)$ D |] ==> *psize* $D \leq n$
⟨*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* $D < m$ |] ==> $D(r) \leq D(m)$
⟨*proof*⟩

lemma *tag-point-eq-partition-point*:
[| *tpart*(a,b) (D,p); *psize* $D \leq m$ |] ==> $p(m) = D(m)$
⟨*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* ≤ *D n*; *tpart* (*a*, *b*) (*D*, *p*) |]
==> *tpart*(*D n*, *b*) (%*x*. *D*(*x* + *n*),%*x*. *p*(*x* + *n*))
⟨*proof*⟩

lemma *fine-right1*:
[| *a* ≤ *D n*; *tpart* (*a*, *b*) (*D*, *p*); *gauge* (%*x*. *D n* ≤ *x* & *x* ≤ *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*:

$[|\ a \leq b;\ Integral(a,b)\ f\ k1;\ Integral(a,b)\ g\ k2\ |]$
$==>\ Integral(a,b)\ (\%x.\ f\ x + g\ x)\ (k1 + k2)$
⟨*proof*⟩

lemma *partition-lt-gen2*:
$[|\ partition(a,b)\ D;\ r < psize\ D\ |] ==>\ 0 < D\ (Suc\ r) - D\ r$
⟨*proof*⟩

lemma *lemma-Integral-le*:
$[|\ \forall x.\ a \leq x\ \&\ x \leq b\ -->\ f\ x \leq g\ x;$
$tpart(a,b)\ (D,p)$
$|] ==>\ \forall n \leq psize\ D.\ f\ (p\ n) \leq g\ (p\ n)$
⟨*proof*⟩

lemma *lemma-Integral-rsum-le*:
$[|\ \forall x.\ a \leq x\ \&\ x \leq b\ -->\ f\ x \leq g\ x;$
$tpart(a,b)\ (D,p)$
$|] ==>\ rsum(D,p)\ f \leq rsum(D,p)\ g$
⟨*proof*⟩

lemma *Integral-le*:
$[|\ a \leq b;$
$\forall x.\ a \leq x\ \&\ x \leq b\ -->\ f(x) \leq g(x);$
$Integral(a,b)\ f\ k1;\ Integral(a,b)\ g\ k2$
$|] ==>\ k1 \leq k2$
⟨*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

53 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*:

$[|\ 0 < a;\ a \neq 1;\ 0 < b;\ b \neq 1;\ 0 < x\ |]$
$\Longrightarrow log\ a\ x = (ln\ b / ln\ a) * log\ b\ x$
⟨*proof*⟩

Base 10 logarithms

lemma *log-base-10-eq1*: $0 < x \Longrightarrow log\ 10\ x = (ln\ (exp\ 1)\ /\ ln\ 10) * ln\ x$
⟨*proof*⟩

lemma *log-base-10-eq2*: $0 < x \Longrightarrow log\ 10\ x = (log\ 10\ (exp\ 1)) * ln\ x$
⟨*proof*⟩

lemma *log-one* [*simp*]: $log\ a\ 1 = 0$
⟨*proof*⟩

lemma *log-eq-one* [*simp*]: $[|\ 0 < a;\ a \neq 1\ |] \Longrightarrow log\ a\ a = 1$
⟨*proof*⟩

lemma *log-inverse*:
$[|\ 0 < a;\ a \neq 1;\ 0 < x\ |] \Longrightarrow log\ a\ (inverse\ x) = -\ log\ a\ x$
⟨*proof*⟩

lemma *log-divide*:
$[|0 < a;\ a \neq 1;\ 0 < x;\ 0 < y|] \Longrightarrow log\ a\ (x/y) = log\ a\ x - log\ a\ y$
⟨*proof*⟩

lemma *log-less-cancel-iff* [*simp*]:
$[|\ 1 < a;\ 0 < x;\ 0 < y\ |] \Longrightarrow (log\ a\ x < log\ a\ y) = (x < y)$
⟨*proof*⟩

lemma *log-le-cancel-iff* [*simp*]:
$[|\ 1 < a;\ 0 < x;\ 0 < y\ |] \Longrightarrow (log\ a\ x \leq log\ a\ y) = (x \leq y)$
⟨*proof*⟩

lemma *powr-realpow*: $0 < x \Longrightarrow x\ powr\ (real\ n) = x\hat{}n$
⟨*proof*⟩

lemma *powr-realpow2*: $0 <= x \Longrightarrow 0 < n \Longrightarrow x\hat{}n = (if\ (x = 0)\ then\ 0$
else x powr (real n))
⟨*proof*⟩

lemma *ln-pwr*: $0 < x \Longrightarrow 0 < y \Longrightarrow ln(x\ powr\ y) = y * ln\ x$
⟨*proof*⟩

lemma *ln-bound*: $1 <= x \Longrightarrow ln\ x <= x$
⟨*proof*⟩

lemma *powr-mono*: $a <= b \Longrightarrow 1 <= x \Longrightarrow x\ powr\ a <= x\ powr\ b$
⟨*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

54 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*]:
  $1 < x \Longrightarrow (x \text{ powhr } a \leq x \text{ powhr } b) = (a \leq b)$
⟨*proof*⟩

lemma *hlog*:
  $hlog\ (star\text{-}n\ X)\ (star\text{-}n\ Y) =$
   $star\text{-}n\ (\%n.\ log\ (X\ n)\ (Y\ n))$
⟨*proof*⟩

lemma *hlog-starfun-ln*: $!!x.\ (*f*\ ln)\ x = hlog\ ((*f*\ exp)\ 1)\ x$
⟨*proof*⟩

lemma *powhr-hlog-cancel* [*simp*]:
  $!!a\ x.\ [|\ 0 < a;\ a \neq 1;\ 0 < x\ |] \Longrightarrow a \text{ powhr } (hlog\ a\ x) = x$
⟨*proof*⟩

lemma *hlog-powhr-cancel* [*simp*]:
  $!!a\ y.\ [|\ 0 < a;\ a \neq 1\ |] \Longrightarrow hlog\ a\ (a \text{ powhr } y) = y$
⟨*proof*⟩

lemma *hlog-mult*:
  $!!a\ x\ y.\ [|\ 0 < a;\ a \neq 1;\ 0 < x;\ 0 < y\ |]$
   $\Longrightarrow hlog\ a\ (x * y) = hlog\ a\ x + hlog\ a\ y$
⟨*proof*⟩

lemma *hlog-as-starfun*:
  $!!a\ x.\ [|\ 0 < a;\ a \neq 1\ |] \Longrightarrow hlog\ a\ x = (*f*\ ln)\ x\ /\ (*f*\ ln)\ a$
⟨*proof*⟩

lemma *hlog-eq-div-starfun-ln-mult-hlog*:
  $!!a\ b\ x.\ [|\ 0 < a;\ a \neq 1;\ 0 < b;\ b \neq 1;\ 0 < x\ |]$
   $\Longrightarrow hlog\ a\ x = ((*f*\ ln)\ b / (*f*ln)\ a) * hlog\ b\ x$
⟨*proof*⟩

lemma *powhr-as-starfun*: $!!a\ x.\ x \text{ powhr } a = (*f*\ exp)\ (a * (*f*\ ln)\ x)$
⟨*proof*⟩

lemma *HInfinite-powhr*:
  $[|\ x : HInfinite;\ 0 < x;\ a : HFinite - Infinitesimal;$
    $0 < a\ |] \Longrightarrow x \text{ powhr } a : HInfinite$
⟨*proof*⟩

lemma *hlog-hrabs-HInfinite-Infinitesimal*:
  $[|\ x : HFinite - Infinitesimal;\ a : HInfinite;\ 0 < a\ |]$
   $\Longrightarrow hlog\ a\ (abs\ x) : Infinitesimal$
⟨*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* ≠ *1* |] ==> *hlog a a* = *1*
⟨*proof*⟩

lemma *hlog-inverse*:
[| *0* < *a*; *a* ≠ *1*; *0* < *x* |] ==> *hlog a* (*inverse x*) = − *hlog a x*
⟨*proof*⟩

lemma *hlog-divide*:
[| *0* < *a*; *a* ≠ *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* ≤ *hlog a y*) = (*x* ≤ *y*)
⟨*proof*⟩

end

theory *Hyperreal*
imports *Ln Deriv Taylor Integration HLog*
begin

end

55 Complex-Main: Comprehensive Complex Theory

theory *Complex-Main*
imports *Fundamental-Theorem-Algebra CLim ../Hyperreal/Hyperreal*
begin

end