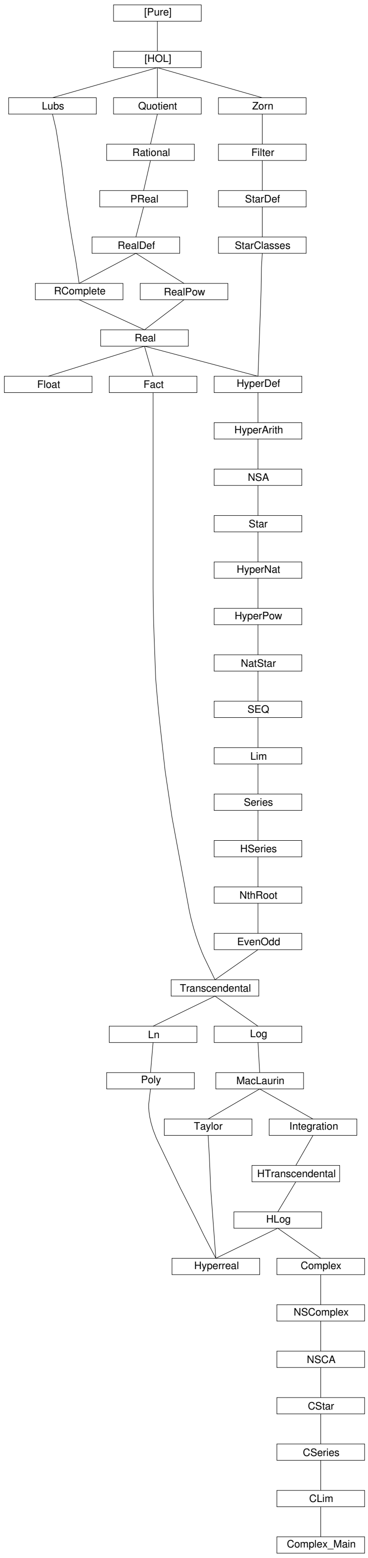
Isabelle/HOL-Complex — Higher-Order Logic with Complex Numbers

October 1, 2005

Contents


[Pure]
[HOL]
Lubs
Quotient
Zorn
Rational
Filter
PReal
StarDef
RealDef
StarClasses
RComplete
RealPow
Real
Float
Fact
HyperDef
HyperArith
NSA
Star
HyperNat
HyperPow
NatStar
SEQ
Lim
Series
HSeries
NthRoot
EvenOdd
Transcendental
Ln
Log
Poly
MacLaurin
Taylor
Integration
HTranscendental
HLog
Hyperreal
Complex
NSComplex
NSCA
CStar
CSeries
CLim
Complex_Main

1 Lubs: Definitions of Upper Bounds and Least Upper Bounds

theory *Lubs*
imports *Main*
begin

Thanks to suggestions by James Margetson

constdefs

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

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

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

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

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

ubs :: [*'a set*, *'a::ord set*] => *'a set*
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*⟩

⟨*ML*⟩

end

2 Quotient: Quotient types

theory *Quotient*
imports *Main*
begin

We introduce the notion of quotient types over equivalence relations via axiomatic type classes.

2.1 Equivalence relations and quotient types

Type class *equiv* models equivalence relations $\sim$ :: *'a => 'a => bool*.

axclass *eqv* ⊆ *type*
consts
eqv :: (*'a*::*eqv*) => *'a* => *bool* (**infixl** $\sim$ *50*)

axclass *equiv* ⊆ *eqv*
equiv-refl [*intro*]: $x \sim x$
equiv-trans [*trans*]: $x \sim y$ ==> $y \sim z$ ==> $x \sim z$
equiv-sym [*sym*]: $x \sim y$ ==> $y \sim x$

lemma *equiv-not-sym* [*sym*]: ¬ ($x \sim y$) ==> ¬ ($y \sim$ (*x*::*'a*::*equiv*))
⟨*proof*⟩

lemma *not-equiv-trans1* [*trans*]: ¬ ($x \sim y$) ==> $y \sim z$ ==> ¬ ($x \sim$ (*z*::*'a*::*equiv*))
⟨*proof*⟩

lemma *not-equiv-trans2* [*trans*]: $x \sim y$ ==> ¬ ($y \sim z$) ==> ¬ ($x \sim$ (*z*::*'a*::*equiv*))
⟨*proof*⟩

The quotient type *'a quot* consists of all *equivalence classes* over elements of the base type *'a*.

typedef *'a quot* = {{*x*. $a \sim x$} | *a*::*'a*::*eqv*. *True*}
⟨*proof*⟩

lemma *quotI* [*intro*]: {*x*. $a \sim x$} ∈ *quot*
⟨*proof*⟩

lemma *quotE* [*elim*]: *R* ∈ *quot* ==> (!!*a*. *R* = {*x*. $a \sim x$} ==> *C*) ==> *C*
⟨*proof*⟩

Abstracted equivalence classes are the canonical representation of elements of a quotient type.

constdefs
class :: *'a*::*equiv* => *'a quot* (⌊-⌋)
⌊*a*⌋ == *Abs-quot* {*x*. $a \sim x$}

theorem *quot-exhaust*: $\exists\, a.\ A = \lfloor a \rfloor$
⟨*proof*⟩

lemma *quot-cases* [*cases type*: *quot*]: $(!!a.\ A = \lfloor a \rfloor \texttt{==>}\ C) \texttt{==>}\ C$
⟨*proof*⟩

2.2 Equality on quotients

Equality of canonical quotient elements coincides with the original relation.

theorem *quot-equality* [*iff?*]: $(\lfloor a \rfloor = \lfloor b \rfloor) = (a \sim b)$
⟨*proof*⟩

2.3 Picking representing elements

constdefs
pick :: *'a::equiv quot => 'a*
$pick\ A == SOME\ a.\ A = \lfloor a \rfloor$

theorem *pick-equiv* [*intro*]: $pick\ \lfloor a \rfloor \sim a$
⟨*proof*⟩

theorem *pick-inverse* [*intro*]: $\lfloor pick\ A \rfloor = A$
⟨*proof*⟩

The following rules support canonical function definitions on quotient types (with up to two arguments). Note that the stripped-down version without additional conditions is sufficient most of the time.

theorem *quot-cond-function*:
$(!!X\ Y.\ P\ X\ Y \texttt{==>} f\ X\ Y == g\ (pick\ X)\ (pick\ Y)) \texttt{==>}$
$(!!x\ x'\ y\ y'.\ \lfloor x \rfloor = \lfloor x' \rfloor \texttt{==>} \lfloor y \rfloor = \lfloor y' \rfloor$
$\texttt{==>} P\ \lfloor x \rfloor\ \lfloor y \rfloor \texttt{==>} P\ \lfloor x' \rfloor\ \lfloor y' \rfloor \texttt{==>} g\ x\ y = g\ x'\ y') \texttt{==>}$
$P\ \lfloor a \rfloor\ \lfloor b \rfloor \texttt{==>} f\ \lfloor a \rfloor\ \lfloor b \rfloor = g\ a\ b$
(**is** *PROP ?eq* ==> *PROP ?cong* ==> - ==> -)
⟨*proof*⟩

theorem *quot-function*:
$(!!X\ Y.\ f\ X\ Y == g\ (pick\ X)\ (pick\ Y)) \texttt{==>}$
$(!!x\ x'\ y\ y'.\ \lfloor x \rfloor = \lfloor x' \rfloor \texttt{==>} \lfloor y \rfloor = \lfloor y' \rfloor \texttt{==>} g\ x\ y = g\ x'\ y') \texttt{==>}$
$f\ \lfloor a \rfloor\ \lfloor b \rfloor = g\ a\ b$
⟨*proof*⟩

theorem *quot-function'*:
$(!!X\ Y.\ f\ X\ Y == g\ (pick\ X)\ (pick\ Y)) \texttt{==>}$
$(!!x\ x'\ y\ y'.\ x \sim x' \texttt{==>} y \sim y' \texttt{==>} g\ x\ y = g\ x'\ y') \texttt{==>}$
$f\ \lfloor a \rfloor\ \lfloor b \rfloor = g\ a\ b$
⟨*proof*⟩

end

3 Rational: Rational numbers

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

3.1 Fractions

3.1.1 The type of fractions

typedef $\mathit{fraction} = \{(a, b) :: \mathit{int} \times \mathit{int} \mid a\ b.\ b \neq 0\}$
⟨*proof*⟩

constdefs
$\mathit{fract} :: \mathit{int} => \mathit{int} => \mathit{fraction}$
$\mathit{fract}\ a\ b == \mathit{Abs\text{-}fraction}\ (a, b)$
$\mathit{num} :: \mathit{fraction} => \mathit{int}$
$\mathit{num}\ Q == \mathit{fst}\ (\mathit{Rep\text{-}fraction}\ Q)$
$\mathit{den} :: \mathit{fraction} => \mathit{int}$
$\mathit{den}\ Q == \mathit{snd}\ (\mathit{Rep\text{-}fraction}\ Q)$

lemma *fract-num* [*simp*]: $b \neq 0 ==> \mathit{num}\ (\mathit{fract}\ a\ b) = a$
⟨*proof*⟩

lemma *fract-den* [*simp*]: $b \neq 0 ==> \mathit{den}\ (\mathit{fract}\ a\ b) = b$
⟨*proof*⟩

lemma *fraction-cases* [*case-names fract*, *cases type*: *fraction*]:
$(!!a\ b.\ Q = \mathit{fract}\ a\ b ==> b \neq 0 ==> C) ==> C$
⟨*proof*⟩

lemma *fraction-induct* [*case-names fract*, *induct type*: *fraction*]:
$(!!a\ b.\ b \neq 0 ==> P\ (\mathit{fract}\ a\ b)) ==> P\ Q$
⟨*proof*⟩

3.1.2 Equivalence of fractions

instance *fraction* :: *eqv* ⟨*proof*⟩

defs (**overloaded**)
equiv-fraction-def: $Q \sim R == \mathit{num}\ Q * \mathit{den}\ R = \mathit{num}\ R * \mathit{den}\ Q$

lemma *equiv-fraction-iff* [*iff*]:
$b \neq 0 ==> b' \neq 0 ==> (\mathit{fract}\ a\ b \sim \mathit{fract}\ a'\ b') = (a * b' = a' * b)$
⟨*proof*⟩

instance *fraction* :: *equiv*
⟨*proof*⟩

lemma *eq-fraction-iff* [*iff*]:
$b \neq 0 \implies b' \neq 0 \implies (\lfloor fract\ a\ b \rfloor = \lfloor fract\ a'\ b' \rfloor) = (a * b' = a' * b)$
⟨*proof*⟩

3.1.3 Operations on fractions

We define the basic arithmetic operations on fractions and demonstrate their "well-definedness", i.e. congruence with respect to equivalence of fractions.

instance *fraction* :: {*zero*, *one*, *plus*, *minus*, *times*, *inverse*, *ord*} ⟨*proof*⟩

defs (**overloaded**)
zero-fraction-def: $0 == fract\ 0\ 1$
one-fraction-def: $1 == fract\ 1\ 1$
add-fraction-def: $Q + R ==$
$fract\ (num\ Q * den\ R + num\ R * den\ Q)\ (den\ Q * den\ R)$
minus-fraction-def: $-Q == fract\ (-(num\ Q))\ (den\ Q)$
mult-fraction-def: $Q * R == fract\ (num\ Q * num\ R)\ (den\ Q * den\ R)$
inverse-fraction-def: $inverse\ Q == fract\ (den\ Q)\ (num\ Q)$
le-fraction-def: $Q \leq R ==$
$(num\ Q * den\ R) * (den\ Q * den\ R) \leq (num\ R * den\ Q) * (den\ Q * den\ R)$

lemma *is-zero-fraction-iff*: $b \neq 0 \implies (\lfloor fract\ a\ b \rfloor = \lfloor 0 \rfloor) = (a = 0)$
⟨*proof*⟩

theorem *add-fraction-cong*:
$\lfloor fract\ a\ b \rfloor = \lfloor fract\ a'\ b' \rfloor \implies \lfloor fract\ c\ d \rfloor = \lfloor fract\ c'\ d' \rfloor$
$\implies b \neq 0 \implies b' \neq 0 \implies d \neq 0 \implies d' \neq 0$
$\implies \lfloor fract\ a\ b + fract\ c\ d \rfloor = \lfloor fract\ a'\ b' + fract\ c'\ d' \rfloor$
⟨*proof*⟩

theorem *minus-fraction-cong*:
$\lfloor fract\ a\ b \rfloor = \lfloor fract\ a'\ b' \rfloor \implies b \neq 0 \implies b' \neq 0$
$\implies \lfloor -(fract\ a\ b) \rfloor = \lfloor -(fract\ a'\ b') \rfloor$
⟨*proof*⟩

theorem *mult-fraction-cong*:
$\lfloor fract\ a\ b \rfloor = \lfloor fract\ a'\ b' \rfloor \implies \lfloor fract\ c\ d \rfloor = \lfloor fract\ c'\ d' \rfloor$
$\implies b \neq 0 \implies b' \neq 0 \implies d \neq 0 \implies d' \neq 0$
$\implies \lfloor fract\ a\ b * fract\ c\ d \rfloor = \lfloor fract\ a'\ b' * fract\ c'\ d' \rfloor$
⟨*proof*⟩

theorem *inverse-fraction-cong*:
$\lfloor fract\ a\ b \rfloor = \lfloor fract\ a'\ b' \rfloor \implies \lfloor fract\ a\ b \rfloor \neq \lfloor 0 \rfloor \implies \lfloor fract\ a'\ b' \rfloor \neq \lfloor 0 \rfloor$
$\implies b \neq 0 \implies b' \neq 0$
$\implies \lfloor inverse\ (fract\ a\ b) \rfloor = \lfloor inverse\ (fract\ a'\ b') \rfloor$

⟨*proof*⟩

theorem *le-fraction-cong*:
  $\lfloor fract\ a\ b\rfloor = \lfloor fract\ a'\ b'\rfloor$ ==> $\lfloor fract\ c\ d\rfloor = \lfloor fract\ c'\ d'\rfloor$
    ==> $b \neq 0$ ==> $b' \neq 0$ ==> $d \neq 0$ ==> $d' \neq 0$
    ==> $(fract\ a\ b \leq fract\ c\ d) = (fract\ a'\ b' \leq fract\ c'\ d')$
⟨*proof*⟩

3.2 Rational numbers

3.2.1 The type of rational numbers

typedef (*Rat*)
  rat = *UNIV* :: *fraction quot set* ⟨*proof*⟩

lemma *RatI* [*intro*, *simp*]: $Q \in Rat$
  ⟨*proof*⟩

constdefs
  fraction-of :: *rat* => *fraction*
  fraction-of q == *pick* (*Rep-Rat q*)
  rat-of :: *fraction* => *rat*
  rat-of Q == *Abs-Rat* $\lfloor Q\rfloor$

theorem *rat-of-equality* [*iff?*]: (*rat-of Q* = *rat-of Q'*) = ($\lfloor Q\rfloor = \lfloor Q'\rfloor$)
  ⟨*proof*⟩

lemma *rat-of*: $\lfloor Q\rfloor = \lfloor Q'\rfloor$ ==> *rat-of Q* = *rat-of Q'* ⟨*proof*⟩

constdefs
  Fract :: *int* => *int* => *rat*
  Fract a b == *rat-of* (*fract a b*)

theorem *Fract-inverse*: $\lfloor$*fraction-of* (*Fract a b*)$\rfloor$ = $\lfloor$*fract a b*$\rfloor$
  ⟨*proof*⟩

theorem *Fract-equality* [*iff?*]:
    (*Fract a b* = *Fract c d*) = ($\lfloor$*fract a b*$\rfloor$ = $\lfloor$*fract c d*$\rfloor$)
  ⟨*proof*⟩

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

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

theorem *Rat-induct* [*case-names Fract*, *induct type*: *rat*]:
    (!!*a b*. $b \neq 0$ ==> *P* (*Fract a b*)) ==> *P q*

$\langle proof \rangle$

3.2.2 Canonical function definitions

Note that the unconditional version below is much easier to read.

theorem *rat-cond-function*:
$(!!q\ r.\ P\ \lfloor fraction\text{-}of\ q\rfloor\ \lfloor fraction\text{-}of\ r\rfloor ==>$
$f\ q\ r == g\ (fraction\text{-}of\ q)\ (fraction\text{-}of\ r)) ==>$
$(!!a\ b\ a'\ b'\ c\ d\ c'\ d'.$
$\lfloor fract\ a\ b\rfloor = \lfloor fract\ a'\ b'\rfloor ==> \lfloor fract\ c\ d\rfloor = \lfloor fract\ c'\ d'\rfloor ==>$
$P\ \lfloor fract\ a\ b\rfloor\ \lfloor fract\ c\ d\rfloor ==> P\ \lfloor fract\ a'\ b'\rfloor\ \lfloor fract\ c'\ d'\rfloor ==>$
$b \neq 0 ==> b' \neq 0 ==> d \neq 0 ==> d' \neq 0 ==>$
$g\ (fract\ a\ b)\ (fract\ c\ d) = g\ (fract\ a'\ b')\ (fract\ c'\ d')) ==>$
$P\ \lfloor fract\ a\ b\rfloor\ \lfloor fract\ c\ d\rfloor ==>$
$f\ (Fract\ a\ b)\ (Fract\ c\ d) = g\ (fract\ a\ b)\ (fract\ c\ d)$
(**is** *PROP ?eq ==> PROP ?cong ==> ?P ==> -*)
$\langle proof \rangle$

theorem *rat-function*:
$(!!q\ r.\ f\ q\ r == g\ (fraction\text{-}of\ q)\ (fraction\text{-}of\ r)) ==>$
$(!!a\ b\ a'\ b'\ c\ d\ c'\ d'.$
$\lfloor fract\ a\ b\rfloor = \lfloor fract\ a'\ b'\rfloor ==> \lfloor fract\ c\ d\rfloor = \lfloor fract\ c'\ d'\rfloor ==>$
$b \neq 0 ==> b' \neq 0 ==> d \neq 0 ==> d' \neq 0 ==>$
$g\ (fract\ a\ b)\ (fract\ c\ d) = g\ (fract\ a'\ b')\ (fract\ c'\ d')) ==>$
$f\ (Fract\ a\ b)\ (Fract\ c\ d) = g\ (fract\ a\ b)\ (fract\ c\ d)$
$\langle proof \rangle$

3.2.3 Standard operations on rational numbers

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

defs (overloaded)
zero-rat-def: $0 == rat\text{-}of\ 0$
one-rat-def: $1 == rat\text{-}of\ 1$
add-rat-def: $q + r == rat\text{-}of\ (fraction\text{-}of\ q + fraction\text{-}of\ r)$
minus-rat-def: $-q == rat\text{-}of\ (-(fraction\text{-}of\ q))$
diff-rat-def: $q - r == q + (-(r::rat))$
mult-rat-def: $q * r == rat\text{-}of\ (fraction\text{-}of\ q * fraction\text{-}of\ r)$
inverse-rat-def: $inverse\ q ==$
$if\ q{=}0\ then\ 0\ else\ rat\text{-}of\ (inverse\ (fraction\text{-}of\ q))$
divide-rat-def: $q\ /\ r == q * inverse\ (r::rat)$
le-rat-def: $q \leq r == fraction\text{-}of\ q \leq fraction\text{-}of\ r$
less-rat-def: $q < r == q \leq r \wedge q \neq (r::rat)$
abs-rat-def: $|q| == if\ q < 0\ then\ -q\ else\ (q::rat)$

theorem *zero-rat*: $0 = Fract\ 0\ 1$
$\langle proof \rangle$

theorem *one-rat*: $1 = Fract\ 1\ 1$

$\langle proof \rangle$

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

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

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

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

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

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

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

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

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

3.2.4 The ordered field of rational numbers

lemma *rat-add-assoc*: $(q + r) + s = q + (r + (s::rat))$
$\langle proof \rangle$

lemma *rat-add-0*: $0 + q = (q::rat)$
$\langle proof \rangle$

lemma *rat-left-minus*: $(-q) + q = (0::rat)$
$\langle proof \rangle$

instance *rat* :: *field*

⟨*proof*⟩

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

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

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

3.3 Various Other Results

lemma *minus-rat-cancel* [*simp*]: $b \neq 0 ==> Fract\ (-a)\ (-b) = Fract\ a\ b$
⟨*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*: $n \neq 0 ==> Fract\ (m + n)\ n = Fract\ m\ n + 1$
⟨*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*⟩

3.4 Numerals and Arithmetic

instance *rat* :: *number* ⟨*proof*⟩

defs (overloaded)
 rat-number-of-def: (*number-of* w :: *rat*) == *of-int* (*Rep-Bin* w)
  — the type constraint is essential!

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

declare *diff-rat-def* [*symmetric*]

⟨*ML*⟩

end

4 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*⟩

As a special case, the sum of two positives is positive. One of the premises could be weakened to the relation $\leq$.

lemma *pos-add-strict*: $[|0{<}a;\ b{<}c|] \mathrel{{=}{=}{>}} b < a + (c{::}'a{::}ordered\text{-}semidom)$
⟨*proof*⟩

lemma *interval-empty-iff*:
$(\{y{::}'a{::}ordered\text{-}field.\ x < y\ \&\ y < z\} = \{\}) = ({\sim}(x < z))$
⟨*proof*⟩

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

lemma *cut-of-rat*:
assumes q: $0 < q$ **shows** $cut\ \{r{::}rat.\ 0 < r\ \&\ r < q\}$
⟨*proof*⟩

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

instance *preal* :: {*ord*, *plus*, *minus*, *times*, *inverse*} ⟨*proof*⟩

constdefs
preal-of-rat :: *rat* => *preal*
$preal\text{-}of\text{-}rat\ q == Abs\text{-}preal(\{x{::}rat.\ 0 < x\ \&\ x < q\})$

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

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

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

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

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

defs (**overloaded**)

preal-less-def:
$R <$ (S::*preal*) == *Rep-preal* $R <$ *Rep-preal* S

preal-le-def:
$R \leq$ (S::*preal*) == *Rep-preal* $R \subseteq$ *Rep-preal* S

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

preal-diff-def:
$R - S$ == *Abs-preal* (*diff-set* (*Rep-preal* R) (*Rep-preal* S))

preal-mult-def:
$R * S$ == *Abs-preal*(*mult-set* (*Rep-preal* R) (*Rep-preal* S))

preal-inverse-def:
inverse R == *Abs-preal*(*inverse-set* (*Rep-preal* R))

Reduces equality on abstractions to equality on representatives

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

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

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

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'*:
$[| A \in preal;\ y \in A;\ 0 < z;\ z \le y\ |] ==> z \in A$
⟨*proof*⟩

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

4.1 *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*:
$[|\{u::rat.\ 0 < u\ \&\ u < x\} \subseteq \{u.\ 0 < u\ \&\ u < y\};\ 0<x|] ==> x \le y$
⟨*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*⟩

4.2 Theorems for Ordering

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*⟩

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

4.3 The $\le$ Ordering

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

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

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

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

4.4 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*:
$[|A \in preal; B \in preal|] ==> \exists q.\ 0 < q \ \& \ q \notin add\text{-}set\ A\ B$
⟨*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-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*⟩

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*

4.5 Properties of Multiplication

Proofs essentially same as for addition

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

Multiplication of two positive reals gives a positive real.

Lemmas for proving positive reals multiplication set in *preal*

Part 1 of Dedekind sections definition

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

Part 2 of Dedekind sections definition

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

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

Part 3 of Dedekind sections definition

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

Part 4 of Dedekind sections definition

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

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

lemma *preal-mult-assoc*: $((x::preal) * y) * z = x * (y * z)$
$\langle proof \rangle$

lemma *preal-mult-left-commute*: $x * (y * z) = y * ((x * z)::preal)$
$\langle proof \rangle$

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

lemma *preal-mult-1*: $(preal\text{-}of\text{-}rat\ 1) * z = z$
$\langle proof \rangle$

lemma *preal-mult-1-right*: $z * (preal\text{-}of\text{-}rat\ 1) = z$
$\langle proof \rangle$

4.6 Distribution of Multiplication across Addition

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

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

lemma *distrib-subset1*:

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

lemma *linorder-le-cases* [*case-names le ge*]:
$((x::'a::linorder) <= y ==> P) ==> (y <= x ==> P) ==> P$
⟨*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*⟩

4.7 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*:
$A \in preal ==> \{\} \subset inverse\text{-}set\ A$
⟨*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*:
$[|A \in preal;\ u \in inverse\text{-}set\ A;\ 0 < z;\ z < u|]$
$==> z \in inverse\text{-}set\ A$

⟨*proof*⟩

Part 4 of Dedekind sections definition

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

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

4.8 Gleason's Lemma 9-3.4, page 122

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

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

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

4.9 Gleason's Lemma 9-3.6

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

4.10 Existence of Inverse: Part 2

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

lemma *Rep-preal-of-rat*:
$0 < q ==> \mathit{Rep\text{-}preal}\ (\mathit{preal\text{-}of\text{-}rat}\ q) = \{x.\ 0 < x \wedge x < q\}$

⟨*proof*⟩

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

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

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

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

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

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

Theorems needing *Gleason9-34*

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

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

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

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

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

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

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

4.11 Subtraction for Positive Reals

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

Part 1 of Dedekind sections definition

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

Part 2 of Dedekind sections definition

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

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

Part 3 of Dedekind sections definition

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

Part 4 of Dedekind sections definition

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

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

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

proving that $R + D \le S$

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

and $0 < c$
shows $a + b \in \mathit{Rep\text{-}preal}\ S$
$\langle proof \rangle$

lemma *less-add-left-le1*:
$R < (S::preal) ==> R + (S-R) \le S$
$\langle proof \rangle$

4.12 proving that $S \le R + D$ — trickier

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

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

lemma *less-add-left-le2*: $R < (S::preal) ==> S \le R + (S-R)$
$\langle proof \rangle$

lemma *less-add-left*: $R < (S::preal) ==> R + (S-R) = S$
$\langle proof \rangle$

lemma *less-add-left-Ex*: $R < (S::preal) ==> \exists D.\ R + D = S$
$\langle proof \rangle$

lemma *preal-add-less2-mono1*: $R < (S::preal) ==> R + T < S + T$
$\langle proof \rangle$

lemma *preal-add-less2-mono2*: $R < (S::preal) ==> T + R < T + S$
$\langle proof \rangle$

lemma *preal-add-right-less-cancel*: $R + T < S + T ==> R < (S::preal)$
$\langle proof \rangle$

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

lemma *preal-add-less-cancel-right*: $((R::preal) + T < S + T) = (R < S)$
$\langle proof \rangle$

lemma *preal-add-less-cancel-left*: $(T + (R::preal) < T + S) = (R < S)$
$\langle proof \rangle$

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

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

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

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

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

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

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

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

4.13 Completeness of type *preal*

Prove that supremum is a cut

Part 1 of Dedekind sections definition

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

Part 2 of Dedekind sections definition

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

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

Part 3 of Dedekind sections definition

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

$==> z \in (\bigcup X \in P.\ \mathit{Rep\text{-}preal}(X))$
$\langle \mathit{proof} \rangle$

Part 4 of Dedekind sections definition

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

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

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

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

Supremum property

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

4.14 The Embadding from *rat* into *preal*

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

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

lemma *preal-of-rat-add*:
$[|\ 0 < x;\ 0 < y|]$
$==> \mathit{preal\text{-}of\text{-}rat}\ ((x{::}\mathit{rat}) + y) = \mathit{preal\text{-}of\text{-}rat}\ x + \mathit{preal\text{-}of\text{-}rat}\ y$
$\langle \mathit{proof} \rangle$

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

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*⟩

⟨*ML*⟩

end

5 RealDef: Defining the Reals from the Positive Reals

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

constdefs
realrel :: ((*preal* $*$ *preal*) $*$ (*preal* $*$ *preal*)) *set*
realrel $== \{p.\ \exists\, x1\ y1\ x2\ y2.\ p = ((x1,y1),(x2,y2))\ \&\ x1+y2 = x2+y1\}$

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

instance *real* :: {*ord*, *zero*, *one*, *plus*, *times*, *minus*, *inverse*} ⟨*proof*⟩

constdefs

real-of-preal :: *preal* => *real*
real-of-preal m ==
$\quad$ *Abs-Real*(*realrel*‘‘{(*m* + *preal-of-rat 1*, *preal-of-rat 1*)})

consts

Reals :: '*a set*

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

syntax (*xsymbols*)
Reals :: '*a set* (ℝ)

defs (**overloaded**)

real-zero-def:
0 == *Abs-Real*(*realrel*‘‘{(*preal-of-rat 1*, *preal-of-rat 1*)})

real-one-def:
1 == *Abs-Real*(*realrel*‘‘
$\quad$ {(*preal-of-rat 1* + *preal-of-rat 1*,
$\quad$ *preal-of-rat 1*)})

real-minus-def:
$-$ *r* == *contents* ($\bigcup$ (*x*,*y*) ∈ *Rep-Real*(*r*). { *Abs-Real*(*realrel*‘‘{(*y*,*x*)}) })

real-add-def:
z + *w* ==
$\quad$ *contents* ($\bigcup$ (*x*,*y*) ∈ *Rep-Real*(*z*). $\bigcup$ (*u*,*v*) ∈ *Rep-Real*(*w*).
$\quad$ { *Abs-Real*(*realrel*‘‘{(*x*+*u*, *y*+*v*)}) })

real-diff-def:
r − (*s*::*real*) == *r* + − *s*

real-mult-def:
z ∗ *w* ==
$\quad$ *contents* ($\bigcup$ (*x*,*y*) ∈ *Rep-Real*(*z*). $\bigcup$ (*u*,*v*) ∈ *Rep-Real*(*w*).

$\{\ Abs\text{-}Real(realrel``\{(x{*}u\ +\ y{*}v,\ x{*}v\ +\ y{*}u)\})\ \})$

real-inverse-def:
$inverse\ (R::real) == (SOME\ S.\ (R = 0\ \&\ S = 0) \mid S * R = 1)$

real-divide-def:
$R\ /\ (S::real) == R * inverse\ S$

real-le-def:
$z \leq (w::real) ==$
$\exists\, x\ y\ u\ v.\ x{+}v \leq u{+}y\ \&\ (x,y) \in Rep\text{-}Real\ z\ \&\ (u,v) \in Rep\text{-}Real\ w$

real-less-def: $(x < (y::real)) == (x \leq y\ \&\ x \neq y)$

real-abs-def: $abs\ (r::real) == (if\ 0 \leq r\ then\ r\ else\ -r)$

5.1 Proving that realrel is an equivalence relation

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*⟩

lemma *inj-on-Abs-Real*: *inj-on Abs-Real Real*
⟨*proof*⟩

declare *inj-on-Abs-Real* [*THEN inj-on-iff*, *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*⟩

5.2 Congruence property for addition

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-add-commute*: (z::*real*) + w = w + z
⟨*proof*⟩

lemma *real-add-assoc*: (($z1$::*real*) + $z2$) + $z3$ = $z1$ + ($z2$ + $z3$)
⟨*proof*⟩

lemma *real-add-zero-left*: (0::*real*) + z = z
⟨*proof*⟩

instance *real* :: *comm-monoid-add*
  ⟨*proof*⟩

5.3 Additive Inverse on real

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

lemma *real-add-minus-left*: (−z) + z = (0::*real*)
⟨*proof*⟩

5.4 Congruence property for 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*:
$Abs\text{-}Real((realrel``\{(x1,y1)\})) * Abs\text{-}Real((realrel``\{(x2,y2)\})) =$
$Abs\text{-}Real(realrel `` \{(x1*x2+y1*y2,x1*y2+y1*x2)\})$
⟨*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*: $0 \neq (1::real)$
⟨*proof*⟩

5.5 existence of inverse

lemma *real-zero-iff*: $Abs\text{-}Real\ (realrel `` \{(x, x)\}) = 0$
⟨*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*: $x \neq 0 ==> \exists y.\ y*x = (1::real)$
⟨*proof*⟩

lemma *real-mult-inverse-left*: $x \neq 0 ==> inverse(x)*x = (1::real)$
⟨*proof*⟩

5.6 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*⟩

declare *minus-mult-right* [*symmetric*, *simp*]
minus-mult-left [*symmetric*, *simp*]

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

5.7 The ≤ 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*⟩

lemma for proving $0 < 1$

lemma *real-zero-le-one*: $0 \leq (1::real)$
⟨*proof*⟩

5.8 The Reals Form an Ordered Field

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

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

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

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

⟨*proof*⟩

Gleason prop 9-4.4 p 127

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

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

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

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

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

lemma *real-of-preal-le-iff*:
(*real-of-preal m1* $\leq$ *real-of-preal m2*) = (*m1* $\leq$ *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*⟩

5.9 Theorems About the Ordering

obsolete but used a lot

lemma *real-not-refl2*: $x < y ==> x \neq (y::real)$
⟨*proof*⟩

lemma *real-le-imp-less-or-eq*: !!(x::*real*). $x \leq y ==> x < y \mid x = y$
⟨*proof*⟩

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

lemma *real-gt-preal-preal-Ex*:

real-of-preal z $<$ *x* $==>$ $\exists$ *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*⟩

lemma *real-add-less-le-mono*: [| *w'*<*w*; *z'*$\leq$*z* |] $==>$ *w'* + *z'* $<$ *w* + (*z*::*real*)
⟨*proof*⟩

lemma *real-add-le-less-mono*:
!!*z z'*::*real*. [| *w'*$\leq$*w*; *z'*<*z* |] $==>$ *w'* + *z'* $<$ *w* + *z*
⟨*proof*⟩

lemma *real-le-square* [*simp*]: (*0*::*real*) $\leq$ *x*∗*x*
⟨*proof*⟩

5.10 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*⟩

The precondition could be weakened to (*0*::*'a*) $\leq$ *x*

lemma *real-mult-less-mono*:
[| *u*<*v*; *x*<*y*; (*0*::*real*) < *v*; *0* < *x* |] $==>$ *u*∗*x* < *v*∗ *y*
⟨*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*⟩

Only two uses?

lemma *real-mult-less-mono'*:
[| *x* < *y*; *r1* < *r2*; (*0*::*real*) $\leq$ *r1*; *0* $\leq$ *x*|] $==>$ *r1* ∗ *x* < *r2* ∗ *y*
⟨*proof*⟩

FIXME: delete or at least combine the next two lemmas

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

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

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

lemma *real-le-add-order*: $[| \ 0 \leq x; \ 0 \leq y \ |] ==> (0::real) \leq x + y$
⟨*proof*⟩

lemma *real-inverse-unique*: $x*y = (1::real) ==> y = inverse\ x$
⟨*proof*⟩

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

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

5.11 Embedding the Integers into the Reals

defs (overloaded)
real-of-nat-def: $real\ z == of\text{-}nat\ z$
real-of-int-def: $real\ z == of\text{-}int\ z$

lemma *real-eq-of-nat*: $real = of\text{-}nat$
⟨*proof*⟩

lemma *real-eq-of-int*: $real = of\text{-}int$
⟨*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*) $\leq$ *real* y) = ($x \leq y$)
⟨*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*⟩

5.12 Embedding the Naturals into the Reals

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.13 Numerals and Arithmetic

instance *real* :: *number* ⟨*proof*⟩

defs (**overloaded**)
real-number-of-def: (*number-of w* :: *real*) == *of-int* (*Rep-Bin w*)
— the type constraint is essential!

instance *real* :: *number-ring*
⟨*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*⟩

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

Needed in this non-standard form by Hyperreal/Transcendental

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

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

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

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

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

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

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

5.14.1 Density of the Reals

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

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

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

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

5.15 Absolute Value Function for the Reals

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

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

Used only in Hyperreal/Lim.ML

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

⟨*proof*⟩

⟨*ML*⟩

end

6 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*⟩

6.1 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*⟩

6.2 The Archimedean Property of the Reals

theorem *reals-Archimedean*:
  assumes *x-pos*: $0 < x$
  shows $\exists n.\ inverse\ (real\ (Suc\ n)) < x$
⟨*proof*⟩

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

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

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

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

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

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

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

⟨*ML*⟩

6.3 Floor and Ceiling Functions from the Reals to the Integers

constdefs

  $floor :: real => int$
   $floor\ r == (LEAST\ n::int.\ r < real\ (n+1))$

ceiling :: *real* => *int*
ceiling r == − *floor* (− *r*)

syntax (*xsymbols*)
floor :: *real* => *int* (⌊-⌋)
ceiling :: *real* => *int* (⌈-⌉)

syntax (*HTML* **output**)
floor :: *real* => *int* (⌊-⌋)
ceiling :: *real* => *int* (⌈-⌉)

lemma *number-of-less-real-of-int-iff* [*simp*]:
((*number-of n*) < *real* (*m*::*int*)) = (*number-of n* < *m*)
⟨*proof*⟩

lemma *number-of-less-real-of-int-iff2* [*simp*]:
(*real* (*m*::*int*) < (*number-of n*)) = (*m* < *number-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* $m \leq r$ **and** *a2*: $r <$ *real* $n + 1$
shows $m \leq (n::int)$
⟨*proof*⟩

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

⟨*proof*⟩

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

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

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

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

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

lemma *ceiling-mono2*: $x \le y ==> ceiling\ x \le ceiling\ y$
⟨*proof*⟩

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

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

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

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

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

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

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

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

lemma *real-of-int-ceiling-le-add-one* [*simp*]: $real\ (ceiling\ r) \le r + 1$
⟨*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*]:
$(ceiling\ x <= number\text{-}of\ n) = (x <= number\text{-}of\ n)$
⟨*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-of* n) = (x <= *number-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-of* n <= *ceiling* x) = (*number-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-of* n) =

ceiling x + number-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-of n*) =
ceiling x − *number-of n*
⟨*proof*⟩

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

6.4 Versions for the natural numbers

constdefs
natfloor :: *real* => *nat*
natfloor x == *nat*(*floor x*)
natceiling :: *real* => *nat*
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*]:
~*neg ((number-of n)::int) ==> 0 <= x ==>*
natfloor (x + number-of n) = natfloor x + number-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-of n*) = *number-of n*
⟨*proof*⟩

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

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

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

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

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

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

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

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

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

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

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

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

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

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

⟨*ML*⟩

end

theory *RealPow*
imports *RealDef*
begin

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

instance *real* :: *power* ⟨*proof*⟩

primrec (*realpow*)
realpow-0: *r ^ 0 = 1*
realpow-Suc: *r ^ (Suc n) = (r::real) ∗ (r ^ n)*

instance *real* :: *recpower*
⟨*proof*⟩

lemma *realpow-not-zero*: *r ≠ (0::real) ==> r ^ n ≠ 0*
⟨*proof*⟩

lemma *realpow-zero-zero*: *r ^ n = (0::real) ==> r = 0*
⟨*proof*⟩

lemma *realpow-two*: *(r::real) ^ (Suc (Suc 0)) = r ∗ r*
⟨*proof*⟩

Legacy: weaker version of the theorem *power-strict-mono*, used 6 times in NthRoot and Transcendental

lemma *realpow-less*:
[|(0::real) < x; x < y; 0 < n|] ==> x ^ n < y ^ n
⟨*proof*⟩

lemma *realpow-two-le* [*simp*]: *(0::real) ≤ r ^ Suc (Suc 0)*
⟨*proof*⟩

lemma *abs-realpow-two* [*simp*]: *abs((x::real) ^Suc (Suc 0)) = x ^Suc (Suc 0)*
⟨*proof*⟩

lemma *realpow-two-abs* [*simp*]: *abs(x::real) ^Suc (Suc 0) = x ^Suc (Suc 0)*

$\langle proof \rangle$

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

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

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

Used ONCE in Transcendental

lemma *realpow-Suc-less-one*: $[|\ 0 < r;\ r < (1::real)\ |] ==> r \;\hat{}\; Suc\ n < 1$
$\langle proof \rangle$

Used ONCE in Lim.ML

lemma *realpow-minus-mult* [*rule-format*]:
$0 < n \dashrightarrow (x::real) \;\hat{}\; (n - 1) * x = x \;\hat{}\; n$
$\langle proof \rangle$

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

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

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

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

lemma *realpow-real-of-nat*: $real\ (m::nat) \;\hat{}\; n = real\ (m \;\hat{}\; n)$
$\langle proof \rangle$

lemma *realpow-real-of-nat-two-pos* [*simp*] : $0 < real\ (Suc\ (Suc\ 0) \;\hat{}\; n)$
$\langle proof \rangle$

lemma *realpow-increasing*:
$[|(0::real) \leq x;\ 0 \leq y;\ x \;\hat{}\; Suc\ n \leq y \;\hat{}\; Suc\ n|] ==> x \leq y$
$\langle proof \rangle$

lemma *zero-less-realpow-abs-iff* [*simp*]:
$(0 < (abs\ x)\hat{}n) = (x \neq (0::real) \mid n{=}0)$
$\langle proof \rangle$

lemma *zero-le-realpow-abs* [*simp*]: $(0::real) \leq (abs\ x)\hat{}n$
⟨*proof*⟩

6.5 Literal Arithmetic Involving Powers, Type *real*

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

lemma *power-real-number-of*:
$(number\text{-}of\ v :: real)\ \hat{}\ n = real\ ((number\text{-}of\ v :: int)\ \hat{}\ n)$
⟨*proof*⟩

declare *power-real-number-of* [*of - number-of w*, *standard*, *simp*]

6.6 Various Other Theorems

Used several times in Hyperreal/Transcendental.ML

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

lemma *real-squared-diff-one-factored*: $x*x - (1::real) = (x + 1)*(x - 1)$
⟨*proof*⟩

lemma *real-mult-is-one* [*simp*]: $(x*x = (1::real)) = (x = 1 \mid x = - 1)$
⟨*proof*⟩

lemma *real-le-add-half-cancel*: $(x + y/2 \leq (y::real)) = (x \leq y\ /2)$
⟨*proof*⟩

lemma *real-minus-half-eq* [*simp*]: $(x::real) - x/2 = x/2$
⟨*proof*⟩

lemma *real-mult-inverse-cancel*:
$[|(0::real) < x;\ 0 < x1;\ x1 * y < x * u\ |]$
$==> inverse\ x * y < inverse\ x1 * u$
⟨*proof*⟩

Used once: in Hyperreal/Transcendental.ML

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

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

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

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

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

6.7 Various Other Theorems

lemma *realpow-divide*:
$(x/y) \hat{} n = ((x::real) \hat{} n/ y \hat{} n)$
⟨*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*: $x \sim= 0 ==> (0::real) < x * x + y * y$
⟨*proof*⟩

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

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

lemma *realpow-square-minus-le* [*simp*]: $-(u \hat{} 2) \leq (x::real) \hat{} 2$
⟨*proof*⟩

lemma *realpow-num-eq-if*: $(m::real) \hat{} n = (\mathit{if}\ n=0\ \mathit{then}\ 1\ \mathit{else}\ m * m \hat{} (n - 1))$
⟨*proof*⟩

lemma *real-num-zero-less-two-pow* [*simp*]: $0 < (2::real) \hat{} (4*d)$
⟨*proof*⟩

lemma *lemma-realpow-num-two-mono*:
$x * (4::real) < y ==> x * (2 \hat{} 8) < y * (2 \hat{} 6)$
⟨*proof*⟩

⟨*ML*⟩

end

theory *Real*
imports *RComplete RealPow*
begin
end

theory *Float* **imports** *Real* **begin**

constdefs
$pow2 :: int \Rightarrow real$
$pow2\ a == if\ (0 <= a)\ then\ (2\hat{}(nat\ a))\ else\ (inverse\ (2\hat{}(nat\ (-a))))$
$float :: int * int \Rightarrow real$
$float\ x == (real\ (fst\ x)) * (pow2\ (snd\ x))$

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*⟩

constdefs
$int\text{-}of\text{-}real :: real \Rightarrow int$
$int\text{-}of\text{-}real\ x == SOME\ y.\ real\ y = x$
$real\text{-}is\text{-}int :: real \Rightarrow bool$
$real\text{-}is\text{-}int\ x == ?\ (u::int).\ x = real\ u$

lemma *real-is-int-def2*: $real\text{-}is\text{-}int\ x = (x = real\ (int\text{-}of\text{-}real\ x))$
⟨*proof*⟩

lemma *float-transfer*: $real\text{-}is\text{-}int\ ((real\ a)*(pow2\ c)) \Longrightarrow float\ (a,\ b) = float\ (int\text{-}of\text{-}real\ ((real\ a)*(pow2\ c)),\ b - c)$
⟨*proof*⟩

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

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

lemma *real-is-int-real*[*simp*]: *real-is-int* (*real* (*x*::*int*))
⟨*proof*⟩

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

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

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

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

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

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

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

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

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

lemma *real-is-int-0*[*simp*]: *real-is-int* (*0*::*real*)
⟨*proof*⟩

lemma *real-is-int-1*[*simp*]: *real-is-int* (*1*::*real*)
⟨*proof*⟩

lemma *real-is-int-n1*: *real-is-int* ($-$*1*::*real*)
⟨*proof*⟩

lemma *real-is-int-number-of*[*simp*]: *real-is-int* ((*number-of*::*bin*$\Rightarrow$*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*⟩

⟨*ML*⟩
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*: *terminating-norm-float*)
⟨*ML*⟩

lemma *norm-float*: *float x* = *float* (*norm-float x*)
⟨*proof*⟩

lemma *pow2-int*: *pow2* (*int n*) = *2*ˆ*n*
⟨*proof*⟩

lemma *float-add*:
float (*a1*, *e1*) + *float* (*a2*, *e2*) =
(*if e1*<=*e2 then float* (*a1*+*a2*∗*2*ˆ(*nat*(*e2*−*e1*)), *e1*)
else float (*a1*∗*2*ˆ(*nat* (*e1*−*e2*))+*a2*, *e2*))
⟨*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 *float-nprt*:
$nprt\ (float\ (a,\ b)) = (if\ 0 <= a\ then\ (float\ (0,b))\ else\ (float\ (a,\ b)))$
⟨*proof*⟩

lemma *norm-0-1*: $(0::\text{-}::number\text{-}ring) = Numeral0\ \&\ (1::\text{-}::number\text{-}ring) = Numeral1$
⟨*proof*⟩

lemma *add-left-zero*: $0 + a = (a::'a::comm\text{-}monoid\text{-}add)$
⟨*proof*⟩

lemma *add-right-zero*: $a + 0 = (a::'a::comm\text{-}monoid\text{-}add)$
⟨*proof*⟩

lemma *mult-left-one*: $1 * a = (a::'a::semiring\text{-}1)$
⟨*proof*⟩

lemma *mult-right-one*: $a * 1 = (a::'a::semiring\text{-}1)$

⟨*proof*⟩

lemma *int-pow-0*: $(a::int)\,\hat{}\,(Numeral0) = 1$
⟨*proof*⟩

lemma *int-pow-1*: $(a::int)\,\hat{}\,(Numeral1) = a$
⟨*proof*⟩

lemma *zero-eq-Numeral0-nring*: $(0::'a::number\text{-}ring) = Numeral0$
⟨*proof*⟩

lemma *one-eq-Numeral1-nring*: $(1::'a::number\text{-}ring) = Numeral1$
⟨*proof*⟩

lemma *zero-eq-Numeral0-nat*: $(0::nat) = Numeral0$
⟨*proof*⟩

lemma *one-eq-Numeral1-nat*: $(1::nat) = Numeral1$
⟨*proof*⟩

lemma *zpower-Pls*: $(z::int)\,\hat{}\,Numeral0 = Numeral1$
⟨*proof*⟩

lemma *zpower-Min*: $(z::int)\,\hat{}\,((-1)::nat) = Numeral1$
⟨*proof*⟩

lemma *fst-cong*: $a{=}a' \Longrightarrow fst\ (a,b) = fst\ (a',b)$
⟨*proof*⟩

lemma *snd-cong*: $b{=}b' \Longrightarrow snd\ (a,b) = snd\ (a,b')$
⟨*proof*⟩

lemma *lift-bool*: $x \Longrightarrow x{=}True$
⟨*proof*⟩

lemma *nlift-bool*: $\sim x \Longrightarrow x{=}False$
⟨*proof*⟩

lemma *not-false-eq-true*: $(\sim\ False) = True$ ⟨*proof*⟩

lemma *not-true-eq-false*: $(\sim\ True) = False$ ⟨*proof*⟩

lemmas *binarith* =
Pls-0-eq Min-1-eq
bin-pred-Pls bin-pred-Min bin-pred-1 bin-pred-0
bin-succ-Pls bin-succ-Min bin-succ-1 bin-succ-0
bin-add-Pls bin-add-Min bin-add-BIT-0 bin-add-BIT-10
bin-add-BIT-11 bin-minus-Pls bin-minus-Min bin-minus-1

bin-minus-0 bin-mult-Pls bin-mult-Min bin-mult-1 bin-mult-0 bin-add-Pls-right bin-add-Min-right

lemma *int-eq-number-of-eq*: $(((\mathit{number\text{-}of}\ v)::\mathit{int})=(\mathit{number\text{-}of}\ w)) = \mathit{iszero}\ ((\mathit{number\text{-}of}\ (\mathit{bin\text{-}add}\ v\ (\mathit{bin\text{-}minus}\ w)))::\mathit{int})$
⟨*proof*⟩

lemma *int-iszero-number-of-Pls*: $\mathit{iszero}\ (\mathit{Numeral0}::\mathit{int})$
⟨*proof*⟩

lemma *int-nonzero-number-of-Min*: $\sim(\mathit{iszero}\ ((-1)::\mathit{int}))$
⟨*proof*⟩

lemma *int-iszero-number-of-0*: $\mathit{iszero}\ ((\mathit{number\text{-}of}\ (w\ \mathit{BIT}\ \mathit{bit.B0}))::\mathit{int}) = \mathit{iszero}\ ((\mathit{number\text{-}of}\ w)::\mathit{int})$
⟨*proof*⟩

lemma *int-iszero-number-of-1*: $\neg\ \mathit{iszero}\ ((\mathit{number\text{-}of}\ (w\ \mathit{BIT}\ \mathit{bit.B1}))::\mathit{int})$
⟨*proof*⟩

lemma *int-less-number-of-eq-neg*: $(((\mathit{number\text{-}of}\ x)::\mathit{int}) < \mathit{number\text{-}of}\ y) = \mathit{neg}\ ((\mathit{number\text{-}of}\ (\mathit{bin\text{-}add}\ x\ (\mathit{bin\text{-}minus}\ y)))::\mathit{int})$
⟨*proof*⟩

lemma *int-not-neg-number-of-Pls*: $\neg\ (\mathit{neg}\ (\mathit{Numeral0}::\mathit{int}))$
⟨*proof*⟩

lemma *int-neg-number-of-Min*: $\mathit{neg}\ (-1::\mathit{int})$
⟨*proof*⟩

lemma *int-neg-number-of-BIT*: $\mathit{neg}\ ((\mathit{number\text{-}of}\ (w\ \mathit{BIT}\ x))::\mathit{int}) = \mathit{neg}\ ((\mathit{number\text{-}of}\ w)::\mathit{int})$
⟨*proof*⟩

lemma *int-le-number-of-eq*: $(((\mathit{number\text{-}of}\ x)::\mathit{int}) \le \mathit{number\text{-}of}\ y) = (\neg\ \mathit{neg}\ ((\mathit{number\text{-}of}\ (\mathit{bin\text{-}add}\ y\ (\mathit{bin\text{-}minus}\ x)))::\mathit{int}))$
⟨*proof*⟩

lemmas *intarithrel* =
int-eq-number-of-eq
lift-bool[*OF int-iszero-number-of-Pls*] *nlift-bool*[*OF int-nonzero-number-of-Min*]
int-iszero-number-of-0
lift-bool[*OF int-iszero-number-of-1*] *int-less-number-of-eq-neg nlift-bool*[*OF int-not-neg-number-of-Pls*]
lift-bool[*OF int-neg-number-of-Min*]
int-neg-number-of-BIT int-le-number-of-eq

lemma *int-number-of-add-sym*: $((\mathit{number\text{-}of}\ v)::\mathit{int}) + \mathit{number\text{-}of}\ w = \mathit{number\text{-}of}\ (\mathit{bin\text{-}add}\ v\ w)$
⟨*proof*⟩

lemma *int-number-of-diff-sym*: $((number\text{-}of\ v)::int) - number\text{-}of\ w = number\text{-}of$ $(bin\text{-}add\ v\ (bin\text{-}minus\ w))$
⟨*proof*⟩

lemma *int-number-of-mult-sym*: $((number\text{-}of\ v)::int) * number\text{-}of\ w = number\text{-}of$ $(bin\text{-}mult\ v\ w)$
⟨*proof*⟩

lemma *int-number-of-minus-sym*: $-\ ((number\text{-}of\ v)::int) = number\text{-}of\ (bin\text{-}minus$ $v)$
⟨*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-mult float-minus float-abs zero-le-float float-pprt float-nprt*

lemmas *arith* = *binarith intarith intarithrel natarith powerarith floatarith not-false-eq-true not-true-eq-false*

end

7 Zorn: Zorn's Lemma

theory *Zorn*
imports *Main*
begin

The lemma and section numbers refer to an unpublished article [**?**].

constdefs
$chain\ ::\ 'a\ set\ set => 'a\ set\ set\ set$
$chain\ S\ == \{F.\ F \subseteq S\ \&\ (\forall x \in F.\ \forall y \in F.\ x \subseteq y \mid y \subseteq x)\}$

$super\ ::\ ['a\ set\ set, 'a\ set\ set] => 'a\ set\ set\ set$
$super\ S\ c == \{d.\ d \in chain\ S\ \&\ c \subset d\}$

```
  maxchain :: 'a set set => 'a set set set
  maxchain S == {c. c ∈ chain S & super S c = {}}

  succ     :: ['a set set,'a set set] => 'a set set
  succ S c ==
    if c ∉ chain S | c ∈ maxchain S
    then c else SOME c'. c' ∈ super S c
```

consts
```
  TFin :: 'a set set => 'a set set set
```

inductive *TFin S*
```
  intros
    succI:        x ∈ TFin S ==> succ S x ∈ TFin S
    Pow-UnionI:   Y ∈ Pow(TFin S) ==> Union(Y) ∈ TFin S
  monos           Pow-mono
```

7.1 Mathematical Preamble

lemma *Union-lemma0*:
```
    (∀x ∈ C. x ⊆ A | B ⊆ x) ==> Union(C)<=A | B ⊆ Union(C)
  ⟨proof⟩
```

This is theorem *increasingD2* of ZF/Zorn.thy

lemma *Abrial-axiom1*: *x ⊆ succ S x*
```
  ⟨proof⟩
```

lemmas *TFin-UnionI = TFin.Pow-UnionI [OF PowI]*

lemma *TFin-induct*:
```
          [| n ∈ TFin S;
             !!x. [| x ∈ TFin S; P(x) |] ==> P(succ S x);
             !!Y. [| Y ⊆ TFin S; Ball Y P |] ==> P(Union Y) |]
          ==> P(n)
  ⟨proof⟩
```

lemma *succ-trans*: *x ⊆ y ==> x ⊆ succ S y*
```
  ⟨proof⟩
```

Lemma 1 of section 3.1

lemma *TFin-linear-lemma1*:
```
     [| n ∈ TFin S;  m ∈ TFin S;
         ∀x ∈ TFin S. x ⊆ m --> x = m | succ S x ⊆ m
      |] ==> n ⊆ m | succ S m ⊆ n
  ⟨proof⟩
```

Lemma 2 of section 3.2

lemma *TFin-linear-lemma2*:
```
     m ∈ TFin S ==> ∀n ∈ TFin S. n ⊆ m --> n=m | succ S n ⊆ 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*⟩

7.2 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*: $c \in chain\ S - maxchain\ S ==> ?\ d.\ d \in super\ S\ c$
⟨*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*: $c \in TFin\ S ==> (c :: 'a\ set\ set): chain\ S$
⟨*proof*⟩

theorem *Hausdorff*: $\exists\, c.\ (c :: 'a\ set\ set): maxchain\ S$
⟨*proof*⟩

7.3 Zorn's Lemma: If All Chains Have Upper Bounds Then There Is a Maximal Element

lemma *chain-extend*:
$[|\ c \in chain\ S;\ z \in S;$
$\forall\, x \in c.\ x<=(z::\ 'a\ set)\ |] ==> \{z\}\ Un\ c \in chain\ S$
⟨*proof*⟩

lemma *chain-Union-upper*: $[|\ c \in chain\ S;\ x \in c\ |] ==> x \subseteq Union(c)$
⟨*proof*⟩

lemma *chain-ball-Union-upper*: $c \in chain\ S ==> \forall\, x \in c.\ x \subseteq Union(c)$
⟨*proof*⟩

lemma *maxchain-Zorn*:
$[|\ c \in maxchain\ S;\ u \in S;\ Union(c) \subseteq u\ |] ==> Union(c) = u$
⟨*proof*⟩

theorem *Zorn-Lemma*:
$\forall\, c \in chain\ S.\ Union(c): S ==> \exists\, y \in S.\ \forall\, z \in S.\ y \subseteq z\ --> y = z$
⟨*proof*⟩

7.4 Alternative version of Zorn's Lemma

lemma *Zorn-Lemma2*:
$\forall\, c \in chain\ S.\ \exists\, y \in S.\ \forall\, x \in c.\ x \subseteq y$
$==> \exists\, y \in S.\ \forall\, x \in S.\ (y :: 'a\ set) \subseteq x\ --> y = x$
⟨*proof*⟩

Various other lemmas

lemma *chainD*: $[|\ c \in chain\ S;\ x \in c;\ y \in c\ |] ==> x \subseteq y\ |\ y \subseteq x$
⟨*proof*⟩

lemma *chainD2*: $!!(c :: 'a\ set\ set).\ c \in chain\ S ==> c \subseteq S$
⟨*proof*⟩

end

8 Filter: Filters and Ultrafilters

theory *Filter*
imports *Zorn*
begin

8.1 Definitions and basic properties

8.1.1 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*⟩

8.1.2 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*⟩

8.1.3 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*) *filter*: *filter* F ⟨*proof*⟩

lemma (**in** *freeultrafilter*) *ultrafilter*: *ultrafilter* F
⟨*proof*⟩

8.2 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*⟩

8.3 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*⟩

8.4 Ultrafilter Theorem

A locale makes proof of ultrafilter Theorem more modular

locale (**open**) *UFT* =
 fixes *frechet* :: *'a set set*
 and *superfrechet* :: *'a set set set*

 assumes *infinite-UNIV*: *infinite* (*UNIV* :: *'a set*)

 defines *frechet-def*: $frechet \equiv \{A.\ finite\ (-\ A)\}$
 and *superfrechet-def*: $superfrechet \equiv \{G.\ filter\ G \wedge frechet \subseteq G\}$

lemma (**in** *UFT*) *superfrechetI*:
 $[\![filter\ G;\ frechet \subseteq G]\!] \Longrightarrow G \in superfrechet$
⟨*proof*⟩

lemma (**in** *UFT*) *superfrechetD1*:
 $G \in superfrechet \Longrightarrow filter\ G$
⟨*proof*⟩

lemma (**in** *UFT*) *superfrechetD2*:
 $G \in superfrechet \Longrightarrow frechet \subseteq G$
⟨*proof*⟩

A few properties of free filters

lemma *filter-cofinite*:
assumes *inf*: *infinite* (*UNIV* :: *'a set*)
shows *filter* $\{A{::}\ 'a\ set.\ finite\ (-\ A)\}$ (**is** *filter ?F*)
⟨*proof*⟩

We prove: 1. Existence of maximal filter i.e. ultrafilter; 2. Freeness property i.e ultrafilter is free. Use a locale to prove various lemmas and then export main result: The ultrafilter Theorem

lemma (**in** *UFT*) *filter-frechet*: *filter frechet*
⟨*proof*⟩

lemma (**in** *UFT*) *frechet-in-superfrechet*: $frechet \in superfrechet$
⟨*proof*⟩

lemma (**in** *UFT*) *lemma-mem-chain-filter*:
$[\![c \in \mathit{chain\ superfrechet};\ x \in c]\!] \Longrightarrow \mathit{filter}\ x$
⟨*proof*⟩

8.4.1 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 $c \in \mathit{chain\ superfrechet}$ **and** $c \neq \{\}$
shows $\mathit{filter}\ (\bigcup c)$
⟨*proof*⟩

lemma (**in** *UFT*) *lemma-mem-chain-frechet-subset*:
$[\![c \in \mathit{chain\ superfrechet};\ x \in c]\!] \Longrightarrow \mathit{frechet} \subseteq x$
⟨*proof*⟩

lemma (**in** *UFT*) *Union-chain-superfrechet*:
$[\![c \neq \{\};\ c \in \mathit{chain\ superfrechet}]\!] \Longrightarrow \bigcup c \in \mathit{superfrechet}$
⟨*proof*⟩

8.4.2 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*:
$[\![U \in \mathit{superfrechet};\ A \in U]\!] \Longrightarrow \mathit{infinite}\ A$

$\langle proof \rangle$

There exists a free ultrafilter on any infinite set

lemma (**in** UFT) *freeultrafilter-ex*:
$\exists\, U{::}'a\ set\ set.\ freeultrafilter\ U$
$\langle proof \rangle$

lemmas *freeultrafilter-Ex* = *UFT.freeultrafilter-ex*

end

9 StarDef: Construction of Star Types Using Ultrafilters

theory *StarDef*
imports *Filter*
uses (*transfer.ML*)
begin

9.1 A Free Ultrafilter over the Naturals

constdefs
FreeUltrafilterNat :: *nat set set* ($\mathcal{U}$)
$\mathcal{U} \equiv SOME\ U.\ freeultrafilter\ U$

lemma *freeultrafilter-FUFNat*: *freeultrafilter* $\mathcal{U}$
$\langle proof \rangle$

interpretation *FUFNat*: *freeultrafilter* [*FreeUltrafilterNat*]
$\langle proof \rangle$

This rule takes the place of the old ultra tactic

lemma *ultra*:
$[\![\{n.\ P\ n\} \in \mathcal{U};\ \{n.\ P\ n \longrightarrow Q\ n\} \in \mathcal{U}]\!] \Longrightarrow \{n.\ Q\ n\} \in \mathcal{U}$
$\langle proof \rangle$

9.2 Definition of *star* type constructor

constdefs
$starrel :: ((nat \Rightarrow 'a) \times (nat \Rightarrow 'a))\ set$
$starrel \equiv \{(X,Y).\ \{n.\ X\ n = Y\ n\} \in \mathcal{U}\}$

typedef $'a\ star = (UNIV :: (nat \Rightarrow 'a)\ set)\ //\ starrel$
$\langle proof \rangle$

constdefs
$star\text{-}n :: (nat \Rightarrow 'a) \Rightarrow 'a\ star$

$star\text{-}n\ X \equiv Abs\text{-}star\ (starrel\ ``\ \{X\})$

theorem *star-cases* [*case-names star-n*, *cases type*: *star*]:
$(\bigwedge X.\ x = star\text{-}n\ X \Longrightarrow P) \Longrightarrow P$
⟨*proof*⟩

lemma *all-star-eq*: $(\forall x.\ P\ x) = (\forall X.\ P\ (star\text{-}n\ X))$
⟨*proof*⟩

lemma *ex-star-eq*: $(\exists x.\ P\ x) = (\exists X.\ P\ (star\text{-}n\ X))$
⟨*proof*⟩

Proving that *starrel* is an equivalence relation

lemma *starrel-iff* [*iff*]: $((X,Y) \in starrel) = (\{n.\ X\ n = Y\ n\} \in \mathcal{U})$
⟨*proof*⟩

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

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

lemma *starrel-in-star*: $starrel``\{x\} \in star$
⟨*proof*⟩

lemma *star-n-eq-iff*: $(star\text{-}n\ X = star\text{-}n\ Y) = (\{n.\ X\ n = Y\ n\} \in \mathcal{U})$
⟨*proof*⟩

9.3 Transfer principle

This introduction rule starts each transfer proof.

lemma *transfer-start*:
$P \equiv \{n.\ Q\} \in \mathcal{U} \Longrightarrow Trueprop\ P \equiv Trueprop\ Q$
⟨*proof*⟩

Initialize transfer tactic.

⟨*ML*⟩

Transfer introduction rules.

lemma *transfer-ex* [*transfer-intro*]:
$[\![\bigwedge X.\ p\ (star\text{-}n\ X) \equiv \{n.\ P\ n\ (X\ n)\} \in \mathcal{U}]\!]$
$\Longrightarrow \exists x{::}'a\ star.\ p\ x \equiv \{n.\ \exists x.\ P\ n\ x\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-all* [*transfer-intro*]:
$[\![\bigwedge X.\ p\ (star\text{-}n\ X) \equiv \{n.\ P\ n\ (X\ n)\} \in \mathcal{U}]\!]$
$\Longrightarrow \forall x{::}'a\ star.\ p\ x \equiv \{n.\ \forall x.\ P\ n\ x\} \in \mathcal{U}$
⟨*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*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ x \equiv \{n.\ X\ n\} \in \mathcal{U};\ y \equiv \{n.\ Y\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow (\textit{if}\ p\ \textit{then}\ x\ \textit{else}\ y) \equiv \{n.\ \textit{if}\ P\ n\ \textit{then}\ X\ n\ \textit{else}\ Y\ n\} \in \mathcal{U}$
⟨*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*⟩

9.4 Standard elements

constdefs
  $star\text{-}of :: {}'a \Rightarrow {}'a\ star$
  $star\text{-}of\ x \equiv star\text{-}n\ (\lambda n.\ x)$

Transfer tactic should remove occurrences of *star-of*

⟨*ML*⟩
declare *star-of-def* [*transfer-intro*]

lemma *star-of-inject*: $(star\text{-}of\ x = star\text{-}of\ y) = (x = y)$
⟨*proof*⟩

9.5 Internal functions

constdefs
  $Ifun :: ({}'a \Rightarrow {}'b)\ star \Rightarrow {}'a\ star \Rightarrow {}'b\ star\ (\text{-} \star \text{-}\ [300,301]\ 300)$
  $Ifun\ f \equiv \lambda x.\ Abs\text{-}star$
    $(\bigcup F{\in}Rep\text{-}star\ f.\ \bigcup X{\in}Rep\text{-}star\ x.\ starrel``\{\lambda n.\ F\ n\ (X\ n)\})$

lemma *Ifun-congruent2*:
  $(\lambda F\ X.\ starrel``\{\lambda n.\ F\ n\ (X\ n)\})\ respects2\ starrel$
⟨*proof*⟩

lemma *Ifun-star-n*: $star\text{-}n\ F \star star\text{-}n\ X = star\text{-}n\ (\lambda n.\ F\ n\ (X\ n))$
⟨*proof*⟩

Transfer tactic should remove occurrences of *Ifun*

⟨*ML*⟩

lemma *transfer-Ifun* [*transfer-intro*]:
  $[\![f \equiv star\text{-}n\ F;\ x \equiv star\text{-}n\ X]\!] \Longrightarrow f \star x \equiv star\text{-}n\ (\lambda n.\ F\ n\ (X\ n))$
⟨*proof*⟩

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

Nonstandard extensions of functions

constdefs
  $starfun :: ({}'a \Rightarrow {}'b) \Rightarrow ({}'a\ star \Rightarrow {}'b\ star)$
    $({*}f{*}\ \text{-}\ [80]\ 80)$
  $starfun\ f \equiv \lambda x.\ star\text{-}of\ f \star x$

  $starfun2 :: ({}'a \Rightarrow {}'b \Rightarrow {}'c) \Rightarrow ({}'a\ star \Rightarrow {}'b\ star \Rightarrow {}'c\ star)$
    $({*}f2{*}\ \text{-}\ [80]\ 80)$
  $starfun2\ f \equiv \lambda x\ y.\ star\text{-}of\ f \star x \star y$

declare *starfun-def* [*transfer-unfold*]
declare *starfun2-def* [*transfer-unfold*]

lemma *starfun-star-n*: $(*f* \ f) \ (star\text{-}n \ X) = star\text{-}n \ (\lambda n. \ f \ (X \ n))$
⟨*proof*⟩

lemma *starfun2-star-n*:
$(*f2* \ f) \ (star\text{-}n \ X) \ (star\text{-}n \ Y) = star\text{-}n \ (\lambda n. \ f \ (X \ n) \ (Y \ n))$
⟨*proof*⟩

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

lemma *starfun2-star-of* [*simp*]: $(*f2* \ f) \ (star\text{-}of \ x) = *f* \ f \ x$
⟨*proof*⟩

9.6 Internal predicates

constdefs
$unstar :: bool \ star \Rightarrow bool$
$unstar \ b \equiv b = star\text{-}of \ True$

lemma *unstar-star-n*: $unstar \ (star\text{-}n \ P) = (\{n. \ P \ n\} \in \mathcal{U})$
⟨*proof*⟩

lemma *unstar-star-of* [*simp*]: $unstar \ (star\text{-}of \ p) = p$
⟨*proof*⟩

Transfer tactic should remove occurrences of *unstar*

⟨*ML*⟩

lemma *transfer-unstar* [*transfer-intro*]:
$p \equiv star\text{-}n \ P \Longrightarrow unstar \ p \equiv \{n. \ P \ n\} \in \mathcal{U}$
⟨*proof*⟩

constdefs
$starP :: ('a \Rightarrow bool) \Rightarrow 'a \ star \Rightarrow bool$
$(*p* \text{ - } [80] \ 80)$
$*p* \ P \equiv \lambda x. \ unstar \ (star\text{-}of \ P \star x)$

$starP2 :: ('a \Rightarrow 'b \Rightarrow bool) \Rightarrow 'a \ star \Rightarrow 'b \ star \Rightarrow bool$
$(*p2* \text{ - } [80] \ 80)$
$*p2* \ P \equiv \lambda x \ y. \ unstar \ (star\text{-}of \ P \star x \star y)$

declare *starP-def* [*transfer-unfold*]
declare *starP2-def* [*transfer-unfold*]

lemma *starP-star-n*: $(*p* \ P) \ (star\text{-}n \ X) = (\{n. \ P \ (X \ n)\} \in \mathcal{U})$
⟨*proof*⟩

lemma *starP2-star-n*:
$(*p2* \ P) \ (star\text{-}n \ X) \ (star\text{-}n \ Y) = (\{n. \ P \ (X \ n) \ (Y \ n)\} \in \mathcal{U})$
⟨*proof*⟩

lemma *starP-star-of* [*simp*]: $(*p* \ P) \ (star\text{-}of \ x) = P \ x$
⟨*proof*⟩

lemma *starP2-star-of* [*simp*]: $(*p2* \ P) \ (star\text{-}of \ x) = *p* \ P \ x$
⟨*proof*⟩

9.7 Internal sets

constdefs
$Iset :: 'a \ set \ star \Rightarrow 'a \ star \ set$
$Iset \ A \equiv \{x. \ (*p2* \ op \in) \ x \ A\}$

lemma *Iset-star-n*:
$(star\text{-}n \ X \in Iset \ (star\text{-}n \ A)) = (\{n. \ X \ n \in A \ n\} \in \mathcal{U})$
⟨*proof*⟩

Transfer tactic should remove occurrences of *Iset*

⟨*ML*⟩

lemma *transfer-mem* [*transfer-intro*]:
$[\![x \equiv star\text{-}n \ X; \ a \equiv Iset \ (star\text{-}n \ A)]\!]$
$\Longrightarrow x \in a \equiv \{n. \ X \ n \in A \ n\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-Collect* [*transfer-intro*]:
$[\![\bigwedge X. \ p \ (star\text{-}n \ X) \equiv \{n. \ P \ n \ (X \ n)\} \in \mathcal{U}]\!]$
$\Longrightarrow Collect \ p \equiv Iset \ (star\text{-}n \ (\lambda n. \ Collect \ (P \ n)))$
⟨*proof*⟩

lemma *transfer-set-eq* [*transfer-intro*]:
$[\![a \equiv Iset \ (star\text{-}n \ A); \ b \equiv Iset \ (star\text{-}n \ B)]\!]$
$\Longrightarrow a = b \equiv \{n. \ A \ n = B \ n\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-ball* [*transfer-intro*]:
$[\![a \equiv Iset \ (star\text{-}n \ A); \ \bigwedge X. \ p \ (star\text{-}n \ X) \equiv \{n. \ P \ n \ (X \ n)\} \in \mathcal{U}]\!]$
$\Longrightarrow \forall x \in a. \ p \ x \equiv \{n. \ \forall x \in A \ n. \ P \ n \ x\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-bex* [*transfer-intro*]:
$[\![a \equiv Iset \ (star\text{-}n \ A); \ \bigwedge X. \ p \ (star\text{-}n \ X) \equiv \{n. \ P \ n \ (X \ n)\} \in \mathcal{U}]\!]$
$\Longrightarrow \exists x \in a. \ p \ x \equiv \{n. \ \exists x \in A \ n. \ P \ n \ x\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-Iset* [*transfer-intro*]:
  $[\![a \equiv \textit{star-n } A]\!] \Longrightarrow \textit{Iset } a \equiv \textit{Iset } (\textit{star-n } (\lambda n.\ A\ n))$
$\langle \textit{proof} \rangle$

Nonstandard extensions of sets.

constdefs
  $\textit{starset} :: {}'a\ \textit{set} \Rightarrow {}'a\ \textit{star set}\ (*s*\ \text{-}\ [80]\ 80)$
  $\textit{starset } A \equiv \textit{Iset } (\textit{star-of } A)$

declare *starset-def* [*transfer-unfold*]

lemma *starset-mem*: $(\textit{star-of } x \in *s*\ A) = (x \in A)$
$\langle \textit{proof} \rangle$

lemma *starset-UNIV*: $*s*\ (\textit{UNIV}::'a\ \textit{set}) = (\textit{UNIV}::'a\ \textit{star set})$
$\langle \textit{proof} \rangle$

lemma *starset-empty*: $*s*\ \{\} = \{\}$
$\langle \textit{proof} \rangle$

lemma *starset-insert*: $*s*\ (\textit{insert } x\ A) = \textit{insert } (\textit{star-of } x)\ (\ *s*\ A)$
$\langle \textit{proof} \rangle$

lemma *starset-Un*: $*s*\ (A \cup B) = *s*\ A \cup *s*\ B$
$\langle \textit{proof} \rangle$

lemma *starset-Int*: $*s*\ (A \cap B) = *s*\ A \cap *s*\ B$
$\langle \textit{proof} \rangle$

lemma *starset-Compl*: $*s*\ -A = -(\ *s*\ A)$
$\langle \textit{proof} \rangle$

lemma *starset-diff*: $*s*\ (A - B) = *s*\ A - *s*\ B$
$\langle \textit{proof} \rangle$

lemma *starset-image*: $*s*\ (f\ `\ A) = (\ *f*\ f)\ `\ (\ *s*\ A)$
$\langle \textit{proof} \rangle$

lemma *starset-vimage*: $*s*\ (f\ -`\ A) = (\ *f*\ f)\ -`\ (\ *s*\ A)$
$\langle \textit{proof} \rangle$

lemma *starset-subset*: $(\ *s*\ A \subseteq *s*\ B) = (A \subseteq B)$
$\langle \textit{proof} \rangle$

lemma *starset-eq*: $(\ *s*\ A = *s*\ B) = (A = B)$
$\langle \textit{proof} \rangle$

lemmas *starset-simps* [*simp*] =
  starset-mem   *starset-UNIV*

starset-empty *starset-insert*
starset-Un *starset-Int*
starset-Compl *starset-diff*
starset-image *starset-vimage*
starset-subset *starset-eq*

end

10 StarClasses: Class Instances

theory *StarClasses*
imports *StarDef*
begin

10.1 Syntactic classes

instance *star* :: (*ord*) *ord* ⟨*proof*⟩
instance *star* :: (*zero*) *zero* ⟨*proof*⟩
instance *star* :: (*one*) *one* ⟨*proof*⟩
instance *star* :: (*plus*) *plus* ⟨*proof*⟩
instance *star* :: (*times*) *times* ⟨*proof*⟩
instance *star* :: (*minus*) *minus* ⟨*proof*⟩
instance *star* :: (*inverse*) *inverse* ⟨*proof*⟩
instance *star* :: (*number*) *number* ⟨*proof*⟩
instance *star* :: (*Divides.div*) *Divides.div* ⟨*proof*⟩
instance *star* :: (*power*) *power* ⟨*proof*⟩

defs (**overloaded**)
star-zero-def: *0* ≡ *star-of 0*
star-one-def: *1* ≡ *star-of 1*
star-number-def: *number-of b* ≡ *star-of* (*number-of b*)
star-add-def: (*op* +) ≡ ∗*f2*∗ (*op* +)
star-diff-def: (*op* −) ≡ ∗*f2*∗ (*op* −)
star-minus-def: *uminus* ≡ ∗*f*∗ *uminus*
star-mult-def: (*op* ∗) ≡ ∗*f2*∗ (*op* ∗)
star-divide-def: (*op* /) ≡ ∗*f2*∗ (*op* /)
star-inverse-def: *inverse* ≡ ∗*f*∗ *inverse*
star-le-def: (*op* ≤) ≡ ∗*p2*∗ (*op* ≤)
star-less-def: (*op* <) ≡ ∗*p2*∗ (*op* <)
star-abs-def: *abs* ≡ ∗*f*∗ *abs*
star-div-def: (*op div*) ≡ ∗*f2*∗ (*op div*)
star-mod-def: (*op mod*) ≡ ∗*f2*∗ (*op mod*)
star-power-def: (*op* ˆ) ≡ λ*x n*. (∗*f*∗ (λ*x*. *x* ˆ *n*)) *x*

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-div-def *star-mod-def* *star-power-def*

star-of preserves class operations

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

lemma *star-of-diff*: *star-of* $(x - y)$ = *star-of* x − *star-of* y
⟨*proof*⟩

lemma *star-of-minus*: *star-of* $(-x)$ = − *star-of* x
⟨*proof*⟩

lemma *star-of-mult*: *star-of* $(x * y)$ = *star-of* x ∗ *star-of* y
⟨*proof*⟩

lemma *star-of-divide*: *star-of* (x / y) = *star-of* x / *star-of* y
⟨*proof*⟩

lemma *star-of-inverse*: *star-of* (*inverse* x) = *inverse* (*star-of* x)
⟨*proof*⟩

lemma *star-of-div*: *star-of* (x *div* y) = *star-of* x *div* *star-of* y
⟨*proof*⟩

lemma *star-of-mod*: *star-of* (x *mod* y) = *star-of* x *mod* *star-of* y
⟨*proof*⟩

lemma *star-of-power*: *star-of* $(x$ ˆ $n)$ = *star-of* x ˆ n
⟨*proof*⟩

lemma *star-of-abs*: *star-of* (*abs* x) = *abs* (*star-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-of* x) = *number-of* x
⟨*proof*⟩

star-of preserves orderings

lemma *star-of-less*: (*star-of* x < *star-of* y) = $(x < y)$
⟨*proof*⟩

lemma *star-of-le*: (*star-of* x ≤ *star-of* y) = $(x \leq y)$

⟨*proof*⟩

lemma *star-of-eq*: (*star-of* x = *star-of* y) = (x = y)
⟨*proof*⟩

As above, for 0

lemmas *star-of-0-less* = *star-of-less* [*of 0*, *simplified star-of-zero*]
lemmas *star-of-0-le* = *star-of-le* [*of 0*, *simplified star-of-zero*]
lemmas *star-of-0-eq* = *star-of-eq* [*of 0*, *simplified star-of-zero*]

lemmas *star-of-less-0* = *star-of-less* [*of - 0*, *simplified star-of-zero*]
lemmas *star-of-le-0* = *star-of-le* [*of - 0*, *simplified star-of-zero*]
lemmas *star-of-eq-0* = *star-of-eq* [*of - 0*, *simplified star-of-zero*]

As above, for 1

lemmas *star-of-1-less* = *star-of-less* [*of 1*, *simplified star-of-one*]
lemmas *star-of-1-le* = *star-of-le* [*of 1*, *simplified star-of-one*]
lemmas *star-of-1-eq* = *star-of-eq* [*of 1*, *simplified star-of-one*]

lemmas *star-of-less-1* = *star-of-less* [*of - 1*, *simplified star-of-one*]
lemmas *star-of-le-1* = *star-of-le* [*of - 1*, *simplified star-of-one*]
lemmas *star-of-eq-1* = *star-of-eq* [*of - 1*, *simplified star-of-one*]

As above, for numerals

lemmas *star-of-number-less* =
star-of-less [*of number-of* w, *standard*, *simplified star-of-number-of*]
lemmas *star-of-number-le* =
star-of-le [*of number-of* w, *standard*, *simplified star-of-number-of*]
lemmas *star-of-number-eq* =
star-of-eq [*of number-of* w, *standard*, *simplified star-of-number-of*]

lemmas *star-of-less-number* =
star-of-less [*of - number-of* w, *standard*, *simplified star-of-number-of*]
lemmas *star-of-le-number* =
star-of-le [*of - number-of* w, *standard*, *simplified star-of-number-of*]
lemmas *star-of-eq-number* =
star-of-eq [*of - number-of* w, *standard*, *simplified star-of-number-of*]

lemmas *star-of-simps* [*simp*] =
star-of-add *star-of-diff* *star-of-minus*
star-of-mult *star-of-divide* *star-of-inverse*
star-of-div *star-of-mod*
star-of-power *star-of-abs*
star-of-zero *star-of-one* *star-of-number-of*
star-of-less *star-of-le* *star-of-eq*
star-of-0-less *star-of-0-le* *star-of-0-eq*
star-of-less-0 *star-of-le-0* *star-of-eq-0*
star-of-1-less *star-of-1-le* *star-of-1-eq*
star-of-less-1 *star-of-le-1* *star-of-eq-1*

star-of-number-less star-of-number-le star-of-number-eq
star-of-less-number star-of-le-number star-of-eq-number

10.2 Ordering classes

instance *star* :: (*order*) *order*
⟨*proof*⟩

instance *star* :: (*linorder*) *linorder*
⟨*proof*⟩

10.3 Lattice ordering classes

Some extra trouble is necessary because the class axioms for *meet* and *join* use quantification over function spaces.

lemma *ex-star-fun*:
$\exists f{::}('a \Rightarrow 'b)\ star.\ P\ (\lambda x.\ f \star x)$
$\Longrightarrow \exists f{::}'a\ star \Rightarrow 'b\ star.\ P\ f$
⟨*proof*⟩

lemma *ex-star-fun2*:
$\exists f{::}('a \Rightarrow 'b \Rightarrow 'c)\ star.\ P\ (\lambda x\ y.\ f \star x \star y)$
$\Longrightarrow \exists f{::}'a\ star \Rightarrow 'b\ star \Rightarrow 'c\ star.\ P\ f$
⟨*proof*⟩

instance *star* :: (*join-semilorder*) *join-semilorder*
⟨*proof*⟩

instance *star* :: (*meet-semilorder*) *meet-semilorder*
⟨*proof*⟩

instance *star* :: (*lorder*) *lorder* ⟨*proof*⟩

lemma *star-join-def* [*transfer-unfold*]: *join* ≡ ∗*f2*∗ *join*
⟨*proof*⟩

lemma *star-meet-def* [*transfer-unfold*]: *meet* ≡ ∗*f2*∗ *meet*
⟨*proof*⟩

10.4 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-ab-group-add*) *pordered-ab-group-add* ⟨*proof*⟩
instance *star* :: (*ordered-cancel-ab-semigroup-add*) *ordered-cancel-ab-semigroup-add*
⟨*proof*⟩
instance *star* :: (*lordered-ab-group-meet*) *lordered-ab-group-meet* ⟨*proof*⟩
instance *star* :: (*lordered-ab-group-meet*) *lordered-ab-group-meet* ⟨*proof*⟩
instance *star* :: (*lordered-ab-group*) *lordered-ab-group* ⟨*proof*⟩

instance *star* :: (*lordered-ab-group-abs*) *lordered-ab-group-abs*
⟨*proof*⟩

10.5 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* :: (*axclass-0-neq-1*) *axclass-0-neq-1*
⟨*proof*⟩

instance *star* :: (*semiring-1*) *semiring-1* ⟨*proof*⟩
instance *star* :: (*comm-semiring-1*) *comm-semiring-1* ⟨*proof*⟩

instance *star* :: (*axclass-no-zero-divisors*) *axclass-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* :: (*idom*) *idom* ⟨*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* :: (*lordered-ring*) *lordered-ring* ⟨*proof*⟩

instance *star* :: (*axclass-abs-if*) *axclass-abs-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*⟩

10.6 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*⟩

10.7 Number classes

lemma *star-of-nat-def* [*transfer-unfold*]: *of-nat n* ≡ *star-of* (*of-nat n*)
⟨*proof*⟩

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

lemma *int-diff-cases*:
assumes *prem*: $\bigwedge m\ n.\ z = int\ m - int\ n \Longrightarrow P$ **shows** *P*
⟨*proof*⟩

lemma *star-of-int-def* [*transfer-unfold*]: *of-int z* ≡ *star-of* (*of-int z*)
⟨*proof*⟩

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

instance *star* :: (*number-ring*) *number-ring*
⟨*proof*⟩

10.8 Finite class

lemma *starset-finite*: *finite A* ⟹ **s* A* = *star-of ‘ A*
⟨*proof*⟩

instance *star* :: (*finite*) *finite*
⟨*proof*⟩

end

11 HyperDef: Construction of Hyperreals Using Ultrafilters

theory *HyperDef*
imports *StarClasses ../Real/Real*
uses (*fuf.ML*)
begin

types *hypreal* = *real star*

syntax *hypreal-of-real* :: *real* => *real star*
translations *hypreal-of-real* => *star-of* :: *real* => *real star*

constdefs

omega :: *hypreal* — an infinite number = [<*1*,*2*,*3*,...>]
omega == *star-n* (%*n*. *real* (*Suc n*))

epsilon :: *hypreal* — an infinitesimal number = [<*1*,*1*/*2*,*1*/*3*,...>]
epsilon == *star-n* (%*n*. *inverse* (*real* (*Suc n*)))

syntax (*xsymbols*)
omega :: *hypreal* (ω)
epsilon :: *hypreal* (ε)

syntax (*HTML* **output**)
omega :: *hypreal* (ω)
epsilon :: *hypreal* (ε)

11.1 Existence of Free Ultrafilter over the Naturals

Also, proof of various properties of $\mathcal{U}$: an arbitrary free ultrafilter

lemma *FreeUltrafilterNat-Ex*: $\exists$ *U*::*nat set set*. *freeultrafilter U*
⟨*proof*⟩

lemma *FreeUltrafilterNat-mem*: *freeultrafilter FreeUltrafilterNat*
⟨*proof*⟩

lemma *UltrafilterNat-mem*: *ultrafilter FreeUltrafilterNat*
⟨*proof*⟩

lemma *FilterNat-mem*: *filter FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-finite*: *finite* x ==> $x \notin$ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-not-finite*: $x \in$ *FreeUltrafilterNat* ==> ~ *finite* x
⟨*proof*⟩

lemma *FreeUltrafilterNat-empty* [*simp*]: $\{\} \notin$ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-Int*:
[| $X \in$ *FreeUltrafilterNat*; $Y \in$ *FreeUltrafilterNat* |]
==> X *Int* $Y \in$ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-subset*:
[| $X \in$ *FreeUltrafilterNat*; $X \subseteq Y$ |]
==> $Y \in$ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-Compl*:
$X \in$ *FreeUltrafilterNat* ==> $-X \notin$ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-Compl-mem*:
$X \notin$ *FreeUltrafilterNat* ==> $-X \in$ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-Compl-iff1*:
($X \notin$ *FreeUltrafilterNat*) = ($-X \in$ *FreeUltrafilterNat*)
⟨*proof*⟩

lemma *FreeUltrafilterNat-Compl-iff2*:
($X \in$ *FreeUltrafilterNat*) = ($-X \notin$ *FreeUltrafilterNat*)
⟨*proof*⟩

lemma *cofinite-mem-FreeUltrafilterNat*: *finite* $(-X)$ ==> $X \in$ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-UNIV* [*iff*]: *UNIV* $\in$ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-Nat-set-refl* [*intro*]:
$\{n.\ P(n) = P(n)\} \in$ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-P*: $\{n{::}nat.\ P\} \in$ *FreeUltrafilterNat* ==> P
⟨*proof*⟩

lemma *FreeUltrafilterNat-Ex-P*: $\{n.\ P(n)\} \in$ *FreeUltrafilterNat* ==> $\exists\, n.\ P(n)$

⟨*proof*⟩

lemma *FreeUltrafilterNat-all*: $\forall\, n.\ P(n) ==> \{n.\ P(n)\} \in$ *FreeUltrafilterNat*
⟨*proof*⟩

Define and use Ultrafilter tactics

⟨*ML*⟩

One further property of our free ultrafilter

lemma *FreeUltrafilterNat-Un*:
  X Un Y $\in$ *FreeUltrafilterNat*
  $==>$ *X* $\in$ *FreeUltrafilterNat* | *Y* $\in$ *FreeUltrafilterNat*
⟨*proof*⟩

11.2 Properties of *starrel*

Proving that *starrel* is an equivalence relation

lemma *starrel-iff*: $((X,Y) \in$ *starrel*$) = (\{n.\ X\ n = Y\ n\} \in$ *FreeUltrafilterNat*$)$
⟨*proof*⟩

lemma *starrel-refl*: $(x,x) \in$ *starrel*
⟨*proof*⟩

lemma *starrel-sym* [*rule-format* (*no-asm*)]: $(x,y) \in$ *starrel* $-->$ $(y,x) \in$ *starrel*
⟨*proof*⟩

lemma *starrel-trans*:
  $[|(x,y) \in$ *starrel*; $(y,z) \in$ *starrel*$|] ==> (x,z) \in$ *starrel*
⟨*proof*⟩

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

lemmas *equiv-starrel-iff* =
  eq-equiv-class-iff [*OF equiv-starrel UNIV-I UNIV-I*, *simp*]

lemma *starrel-in-hypreal* [*simp*]: *starrel*``$\{x\}$:*star*
⟨*proof*⟩

declare *Abs-star-inject* [*simp*] *Abs-star-inverse* [*simp*]
declare *equiv-starrel* [*THEN eq-equiv-class-iff*, *simp*]

lemmas *eq-starrelD* = *eq-equiv-class* [*OF - equiv-starrel*]

lemma *lemma-starrel-refl* [*simp*]: $x \in$ *starrel* `` $\{x\}$
⟨*proof*⟩

lemma *hypreal-empty-not-mem* [*simp*]: $\{\} \notin$ *star*

⟨*proof*⟩

lemma *Rep-hypreal-nonempty* [*simp*]: *Rep-star* $x \neq \{\}$
⟨*proof*⟩

11.3 *star-of*: the Injection from *real* to *hypreal*

lemma *inj-hypreal-of-real*: *inj*(*hypreal-of-real*)
⟨*proof*⟩

lemma *Rep-star-star-n-iff* [*simp*]:
$(X \in \textit{Rep-star}\ (\textit{star-n}\ Y)) = (\{n.\ Y\ n = X\ n\} \in \mathcal{U})$
⟨*proof*⟩

lemma *Rep-star-star-n*: $X \in \textit{Rep-star}\ (\textit{star-n}\ X)$
⟨*proof*⟩

11.4 Properties of *star-n*

lemma *star-n-add*:
$\textit{star-n}\ X + \textit{star-n}\ Y = \textit{star-n}\ (\%n.\ X\ n + Y\ n)$
⟨*proof*⟩

lemma *star-n-minus*:
$- \textit{star-n}\ X = \textit{star-n}\ (\%n.\ -(X\ n))$
⟨*proof*⟩

lemma *star-n-diff*:
$\textit{star-n}\ X - \textit{star-n}\ Y = \textit{star-n}\ (\%n.\ X\ n - Y\ n)$
⟨*proof*⟩

lemma *star-n-mult*:
$\textit{star-n}\ X * \textit{star-n}\ Y = \textit{star-n}\ (\%n.\ X\ n * Y\ n)$
⟨*proof*⟩

lemma *star-n-inverse*:
$\textit{inverse}\ (\textit{star-n}\ X) = \textit{star-n}\ (\%n.\ \textit{inverse}(X\ n))$
⟨*proof*⟩

lemma *star-n-le*:
$\textit{star-n}\ X \leq \textit{star-n}\ Y =$
$(\{n.\ X\ n \leq Y\ n\} \in \textit{FreeUltrafilterNat})$
⟨*proof*⟩

lemma *star-n-less*:
$\textit{star-n}\ X < \textit{star-n}\ Y = (\{n.\ X\ n < Y\ n\} \in \textit{FreeUltrafilterNat})$
⟨*proof*⟩

lemma *star-n-zero-num*: $0 = \textit{star-n}\ (\%n.\ 0)$
⟨*proof*⟩

lemma *star-n-one-num*: $1 = star\text{-}n\ (\%n.\ 1)$
⟨*proof*⟩

lemma *star-n-abs*:
$abs\ (star\text{-}n\ X) = star\text{-}n\ (\%n.\ abs\ (X\ n))$
⟨*proof*⟩

11.5 Misc Others

lemma *hypreal-not-refl2*: $!!(x::hypreal).\ x < y ==> x \neq y$
⟨*proof*⟩

lemma *hypreal-eq-minus-iff*: $((x::hypreal) = y) = (x + - y = 0)$
⟨*proof*⟩

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

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

lemma *hypreal-omega-gt-zero* [*simp*]: $0 < omega$
⟨*proof*⟩

11.6 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*: ~ $(\exists\, x.\ hypreal\text{-}of\text{-}real\ x = epsilon)$
⟨*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*⟩

⟨*ML*⟩

end

12 HyperArith: Binary arithmetic and Simplification for the Hyperreals

theory *HyperArith*
imports *HyperDef*
uses (*hypreal-arith.ML*)
begin

12.1 Numerals and Arithmetic

⟨*ML*⟩

12.2 Absolute Value Function for the Hyperreals

lemma *hrabs-add-less*:
$[|\ abs\ x < r;\ abs\ y < s\ |] ==> abs(x+y) < r + (s::hypreal)$
⟨*proof*⟩

used once in NSA

lemma *hrabs-less-gt-zero*: $abs\ x < r ==> (0::hypreal) < r$
⟨*proof*⟩

lemma *hrabs-disj*: $abs\ x = (x::hypreal) \mid abs\ x = -x$
⟨*proof*⟩

lemma *hrabs-add-lemma-disj*: $(y::hypreal) + - x + (y + - z) = abs\ (x + - z)$ $==> y = z \mid x = y$
⟨*proof*⟩

lemma *hypreal-of-real-hrabs*:
$abs\ (hypreal\text{-}of\text{-}real\ r) = hypreal\text{-}of\text{-}real\ (abs\ r)$
⟨*proof*⟩

12.3 Embedding the Naturals into the Hyperreals

constdefs
hypreal-of-nat :: *nat* => *hypreal*
hypreal-of-nat m == *of-nat* m

lemma *SNat-eq*: $Nats = \{n.\ \exists N.\ n = hypreal\text{-}of\text{-}nat\ N\}$
⟨*proof*⟩

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

lemma *hypreal-of-nat-mult*: $hypreal\text{-}of\text{-}nat\ (m * n) = hypreal\text{-}of\text{-}nat\ m * hypreal\text{-}of\text{-}nat\ n$
⟨*proof*⟩
declare *hypreal-of-nat-mult* [*simp*]

lemma *hypreal-of-nat-less-iff*:
$(n < m) = (hypreal\text{-}of\text{-}nat\ n < hypreal\text{-}of\text{-}nat\ m)$
⟨*proof*⟩
declare *hypreal-of-nat-less-iff* [*symmetric*, *simp*]

lemma *hypreal-of-nat-eq*:
$hypreal\text{-}of\text{-}nat\ (n::nat) = hypreal\text{-}of\text{-}real\ (real\ n)$
⟨*proof*⟩

lemma *hypreal-of-nat*:
$hypreal\text{-}of\text{-}nat\ m = star\text{-}n\ (\%n.\ real\ m)$
⟨*proof*⟩

lemma *hypreal-of-nat-Suc*:
$hypreal\text{-}of\text{-}nat\ (Suc\ n) = hypreal\text{-}of\text{-}nat\ n + (1::hypreal)$
⟨*proof*⟩

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

lemma *hypreal-of-nat-zero* [*simp*]: *hypreal-of-nat 0* = *0*
⟨*proof*⟩

lemma *hypreal-of-nat-one* [*simp*]: *hypreal-of-nat 1* = *1*
⟨*proof*⟩

lemma *hypreal-of-nat-le-iff* [*simp*]:
(*hypreal-of-nat n* ≤ *hypreal-of-nat m*) = (*n* ≤ *m*)
⟨*proof*⟩

lemma *hypreal-of-nat-ge-zero* [*simp*]: *0* ≤ *hypreal-of-nat n*
⟨*proof*⟩

⟨*ML*⟩

end

13 NSA: Infinite Numbers, Infinitesimals, Infinitely Close Relation

theory *NSA*
imports *HyperArith ../Real/RComplete*
begin

constdefs

Infinitesimal :: *hypreal set*
Infinitesimal == {*x*. ∀ *r* ∈ *Reals*. *0* < *r* −−> *abs x* < *r*}

HFinite :: *hypreal set*
HFinite == {*x*. ∃ *r* ∈ *Reals*. *abs x* < *r*}

HInfinite :: *hypreal set*
HInfinite == {*x*. ∀ *r* ∈ *Reals*. *r* < *abs x*}

approx :: [*hypreal*, *hypreal*] => *bool* (**infixl** @= *50*)
— the 'infinitely close' relation

x @= y == $(x + -y) \in$ *Infinitesimal*

st :: *hypreal* => *hypreal*
— the standard part of a hyperreal
st == (%x. @r. $x \in$ *HFinite* & $r \in$ *Reals* & r @= x)

monad :: *hypreal* => *hypreal set*
monad x == $\{y.\ x$ @= $y\}$

galaxy :: *hypreal* => *hypreal set*
galaxy x == $\{y.\ (x + -y) \in$ *HFinite*$\}$

defs (**overloaded**)

SReal-def: *Reals* == $\{x.\ \exists\, r.\ x =$ *hypreal-of-real* $r\}$
— the standard real numbers as a subset of the hyperreals

syntax (*xsymbols*)
approx :: [*hypreal*, *hypreal*] => *bool* (**infixl** $\approx$ *50*)

syntax (*HTML* **output**)
approx :: [*hypreal*, *hypreal*] => *bool* (**infixl** $\approx$ *50*)

13.1 Closure Laws for the Standard Reals

lemma *SReal-add* [*simp*]:
[| (x::*hypreal*) $\in$ *Reals*; $y \in$ *Reals* |] ==> $x + y \in$ *Reals*
⟨*proof*⟩

lemma *SReal-mult*: [| (x::*hypreal*) $\in$ *Reals*; $y \in$ *Reals* |] ==> $x * y \in$ *Reals*
⟨*proof*⟩

lemma *SReal-inverse*: (x::*hypreal*) $\in$ *Reals* ==> *inverse* $x \in$ *Reals*
⟨*proof*⟩

lemma *SReal-divide*: [| (x::*hypreal*) $\in$ *Reals*; $y \in$ *Reals* |] ==> $x/y \in$ *Reals*
⟨*proof*⟩

lemma *SReal-minus*: (x::*hypreal*) $\in$ *Reals* ==> $-x \in$ *Reals*
⟨*proof*⟩

lemma *SReal-minus-iff* [*simp*]: $(-x \in$ *Reals*$)$ = ((x::*hypreal*) $\in$ *Reals*)
⟨*proof*⟩

lemma *SReal-add-cancel*:
[| (x::*hypreal*) + $y \in$ *Reals*; $y \in$ *Reals* |] ==> $x \in$ *Reals*
⟨*proof*⟩

lemma *SReal-hrabs*: $(x::hypreal) \in Reals ==> abs\ x \in Reals$
⟨*proof*⟩

lemma *SReal-hypreal-of-real* [*simp*]: $hypreal\text{-}of\text{-}real\ x \in Reals$
⟨*proof*⟩

lemma *SReal-number-of* [*simp*]: $(number\text{-}of\ w\ ::hypreal) \in Reals$
⟨*proof*⟩

lemma *Reals-0* [*simp*]: $(0::hypreal) \in Reals$
⟨*proof*⟩

lemma *Reals-1* [*simp*]: $(1::hypreal) \in Reals$
⟨*proof*⟩

lemma *SReal-divide-number-of*: $r \in Reals ==> r/(number\text{-}of\ w::hypreal) \in Reals$
⟨*proof*⟩

epsilon is not in Reals because it is an infinitesimal

lemma *SReal-epsilon-not-mem*: $epsilon \notin Reals$
⟨*proof*⟩

lemma *SReal-omega-not-mem*: $omega \notin Reals$
⟨*proof*⟩

lemma *SReal-UNIV-real*: $\{x.\ hypreal\text{-}of\text{-}real\ x \in Reals\} = (UNIV::real\ set)$
⟨*proof*⟩

lemma *SReal-iff*: $(x \in Reals) = (\exists\, y.\ x = hypreal\text{-}of\text{-}real\ y)$
⟨*proof*⟩

lemma *hypreal-of-real-image*: $hypreal\text{-}of\text{-}real\ `(UNIV::real\ set) = Reals$
⟨*proof*⟩

lemma *inv-hypreal-of-real-image*: $inv\ hypreal\text{-}of\text{-}real\ `\ Reals = UNIV$
⟨*proof*⟩

lemma *SReal-hypreal-of-real-image*:
  $[|\ \exists\, x.\ x: P;\ P \subseteq Reals\ |] ==> \exists\, Q.\ P = hypreal\text{-}of\text{-}real\ `\ Q$
⟨*proof*⟩

lemma *SReal-dense*:
  $[|\ (x::hypreal) \in Reals;\ y \in Reals;\ \ x<y\ |] ==> \exists\, r \in Reals.\ x<r\ \&\ r<y$
⟨*proof*⟩

Completeness of Reals, but both lemmas are unused.

lemma *SReal-sup-lemma*:

$P \subseteq Reals ==> ((\exists x \in P.\ y < x) =$
$(\exists X.\ hypreal\text{-}of\text{-}real\ X \in P\ \&\ y < hypreal\text{-}of\text{-}real\ X))$
⟨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⟩

13.2 Lifting of the Ub and Lub Properties

lemma *hypreal-of-real-isUb-iff*:
$(isUb\ (Reals)\ (hypreal\text{-}of\text{-}real\ `\ Q)\ (hypreal\text{-}of\text{-}real\ Y)) =$
$(isUb\ (UNIV :: real\ set)\ Q\ Y)$
⟨proof⟩

lemma *hypreal-of-real-isLub1*:
$isLub\ Reals\ (hypreal\text{-}of\text{-}real\ `\ Q)\ (hypreal\text{-}of\text{-}real\ Y)$
$==> isLub\ (UNIV :: real\ set)\ Q\ Y$
⟨proof⟩

lemma *hypreal-of-real-isLub2*:
$isLub\ (UNIV :: real\ set)\ Q\ Y$
$==> isLub\ Reals\ (hypreal\text{-}of\text{-}real\ `\ Q)\ (hypreal\text{-}of\text{-}real\ Y)$
⟨proof⟩

lemma *hypreal-of-real-isLub-iff*:
$(isLub\ Reals\ (hypreal\text{-}of\text{-}real\ `\ Q)\ (hypreal\text{-}of\text{-}real\ Y)) =$
$(isLub\ (UNIV :: real\ set)\ Q\ Y)$
⟨proof⟩

lemma *lemma-isUb-hypreal-of-real*:
$isUb\ Reals\ P\ Y ==> \exists Yo.\ isUb\ Reals\ P\ (hypreal\text{-}of\text{-}real\ Yo)$
⟨proof⟩

lemma *lemma-isLub-hypreal-of-real*:
$isLub\ Reals\ P\ Y ==> \exists Yo.\ isLub\ Reals\ P\ (hypreal\text{-}of\text{-}real\ Yo)$
⟨proof⟩

lemma *lemma-isLub-hypreal-of-real2*:
$\exists Yo.\ isLub\ Reals\ P\ (hypreal\text{-}of\text{-}real\ Yo) ==> \exists Y.\ isLub\ Reals\ P\ Y$
⟨proof⟩

lemma *SReal-complete*:
$[|\ P \subseteq Reals;\ \exists x.\ x \in P;\ \exists Y.\ isUb\ Reals\ P\ Y\ |]$
$==> \exists t::hypreal.\ isLub\ Reals\ P\ t$
⟨proof⟩

13.3 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*: $[|x \in HFinite;\ y \in HFinite|] ==> x*y \in HFinite$
⟨*proof*⟩

lemma *HFinite-minus-iff*: $(-x \in HFinite) = (x \in HFinite)$
⟨*proof*⟩

lemma *SReal-subset-HFinite*: $Reals \subseteq HFinite$
⟨*proof*⟩

lemma *HFinite-hypreal-of-real* [*simp*]: $hypreal\text{-}of\text{-}real\ x \in HFinite$
⟨*proof*⟩

lemma *HFiniteD*: $x \in HFinite ==> \exists\, t \in Reals.\ abs\ x < t$
⟨*proof*⟩

lemma *HFinite-hrabs-iff* [*iff*]: $(abs\ x \in HFinite) = (x \in HFinite)$
⟨*proof*⟩

lemma *HFinite-number-of* [*simp*]: $number\text{-}of\ w \in HFinite$
⟨*proof*⟩

lemma *HFinite-0* [*simp*]: $0 \in HFinite$
⟨*proof*⟩

lemma *HFinite-1* [*simp*]: $1 \in HFinite$
⟨*proof*⟩

lemma *HFinite-bounded*: $[|x \in HFinite;\ y \le x;\ 0 \le y\ |] ==> y \in HFinite$
⟨*proof*⟩

13.4 Set of Infinitesimals is a Subring of the Hyperreals

lemma *InfinitesimalD*:
$x \in Infinitesimal ==> \forall\, r \in Reals.\ 0 < r --> abs\ x < r$
⟨*proof*⟩

lemma *Infinitesimal-zero* [*iff*]: $0 \in Infinitesimal$
⟨*proof*⟩

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

lemma *Infinitesimal-add*:
$[| \ x \in Infinitesimal; \ y \in Infinitesimal \ |] ==> (x+y) \in Infinitesimal$
⟨*proof*⟩

lemma *Infinitesimal-minus-iff* [*simp*]: $(-x{:}Infinitesimal) = (x{:}Infinitesimal)$
⟨*proof*⟩

lemma *Infinitesimal-diff*:
$[| \ x \in Infinitesimal; \ \ y \in Infinitesimal \ |] ==> x-y \in Infinitesimal$
⟨*proof*⟩

lemma *Infinitesimal-mult*:
$[| \ x \in Infinitesimal; \ y \in Infinitesimal \ |] ==> (x * y) \in Infinitesimal$
⟨*proof*⟩

lemma *Infinitesimal-HFinite-mult*:
$[| \ x \in Infinitesimal; \ y \in HFinite \ |] ==> (x * y) \in Infinitesimal$
⟨*proof*⟩

lemma *Infinitesimal-HFinite-mult2*:
$[| \ x \in Infinitesimal; \ y \in HFinite \ |] ==> (y * x) \in Infinitesimal$
⟨*proof*⟩

lemma *HInfinite-inverse-Infinitesimal*:
$x \in HInfinite ==> inverse \ x{:} \ Infinitesimal$
⟨*proof*⟩

lemma *HInfinite-mult*: $[|x \in HInfinite;y \in HInfinite|] ==> (x*y) \in HInfinite$
⟨*proof*⟩

lemma *hypreal-add-zero-less-le-mono*: $[|r < x; \ (0{::}hypreal) \le y|] ==> r < x+y$
⟨*proof*⟩

lemma *HInfinite-add-ge-zero*:
$[|x \in HInfinite; \ 0 \le y; \ 0 \le x|] ==> (x + y){:} \ HInfinite$
⟨*proof*⟩

lemma *HInfinite-add-ge-zero2*:
$[|x \in HInfinite; \ 0 \le y; \ 0 \le x|] ==> (y + x){:} \ HInfinite$
⟨*proof*⟩

lemma *HInfinite-add-gt-zero*:
$[|x \in HInfinite; \ 0 < y; \ 0 < x|] ==> (x + y){:} \ HInfinite$
⟨*proof*⟩

lemma *HInfinite-minus-iff*: $(-x \in HInfinite) = (x \in HInfinite)$
⟨*proof*⟩

lemma *HInfinite-add-le-zero*:
$[|x \in \mathit{HInfinite};\ y \leq 0;\ x \leq 0|]$ ==> $(x + y)$: *HInfinite*
⟨*proof*⟩

lemma *HInfinite-add-lt-zero*:
$[|x \in \mathit{HInfinite};\ y < 0;\ x < 0|]$ ==> $(x + y)$: *HInfinite*
⟨*proof*⟩

lemma *HFinite-sum-squares*:
$[|a$: *HFinite*; b: *HFinite*; c: *HFinite*$|]$
==> $a*a + b*b + c*c \in \mathit{HFinite}$
⟨*proof*⟩

lemma *not-Infinitesimal-not-zero*: $x \notin \mathit{Infinitesimal}$ ==> $x \neq 0$
⟨*proof*⟩

lemma *not-Infinitesimal-not-zero2*: $x \in \mathit{HFinite} - \mathit{Infinitesimal}$ ==> $x \neq 0$
⟨*proof*⟩

lemma *Infinitesimal-hrabs-iff* [*iff*]:
$(\mathit{abs}\ x \in \mathit{Infinitesimal}) = (x \in \mathit{Infinitesimal})$
⟨*proof*⟩

lemma *HFinite-diff-Infinitesimal-hrabs*:
$x \in \mathit{HFinite} - \mathit{Infinitesimal}$ ==> $\mathit{abs}\ x \in \mathit{HFinite} - \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *hrabs-less-Infinitesimal*:
$[|\ e \in \mathit{Infinitesimal};\ \mathit{abs}\ x < e\ |]$ ==> $x \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *hrabs-le-Infinitesimal*:
$[|\ e \in \mathit{Infinitesimal};\ \mathit{abs}\ x \leq e\ |]$ ==> $x \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *Infinitesimal-interval*:
$[|\ e \in \mathit{Infinitesimal};\ e' \in \mathit{Infinitesimal};\ e' < x\ ;\ x < e\ |]$
==> $x \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *Infinitesimal-interval2*:
$[|\ e \in \mathit{Infinitesimal};\ e' \in \mathit{Infinitesimal};$
$e' \leq x\ ;\ x \leq e\ |]$ ==> $x \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *not-Infinitesimal-mult*:
$[|\ x \notin \mathit{Infinitesimal};\ \ y \notin \mathit{Infinitesimal}|]$ ==> $(x*y) \notin \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *Infinitesimal-mult-disj*:
$x{*}y \in Infinitesimal ==> x \in Infinitesimal \mid y \in Infinitesimal$
⟨*proof*⟩

lemma *HFinite-Infinitesimal-not-zero*: $x \in HFinite-Infinitesimal ==> x \neq 0$
⟨*proof*⟩

lemma *HFinite-Infinitesimal-diff-mult*:
$[| \ x \in HFinite - Infinitesimal;$
$y \in HFinite - Infinitesimal$
$|] ==> (x{*}y) \in HFinite - Infinitesimal$
⟨*proof*⟩

lemma *Infinitesimal-subset-HFinite*:
$Infinitesimal \subseteq HFinite$
⟨*proof*⟩

lemma *Infinitesimal-hypreal-of-real-mult*:
$x \in Infinitesimal ==> x * hypreal\text{-}of\text{-}real\ r \in Infinitesimal$
⟨*proof*⟩

lemma *Infinitesimal-hypreal-of-real-mult2*:
$x \in Infinitesimal ==> hypreal\text{-}of\text{-}real\ r * x \in Infinitesimal$
⟨*proof*⟩

13.5 The Infinitely Close Relation

lemma *mem-infmal-iff*: $(x \in Infinitesimal) = (x \ @= \ 0)$
⟨*proof*⟩

lemma *approx-minus-iff*: $(x \ @= \ y) = (x + -y \ @= \ 0)$
⟨*proof*⟩

lemma *approx-minus-iff2*: $(x \ @= \ y) = (-y + x \ @= \ 0)$
⟨*proof*⟩

lemma *approx-refl* [*iff*]: $x \ @= \ x$
⟨*proof*⟩

lemma *hypreal-minus-distrib1*: $-(y + -(x::hypreal)) = x + -y$
⟨*proof*⟩

lemma *approx-sym*: $x \ @= \ y ==> y \ @= \ x$
⟨*proof*⟩

lemma *approx-trans*: $[| \ x \ @= \ y; \ y \ @= \ z \ |] ==> x \ @= \ z$
⟨*proof*⟩

lemma *approx-trans2*: $[| \ r \ @= \ x; \ s \ @= \ x \ |] ==> r \ @= \ s$

⟨proof⟩

lemma *approx-trans3*: [| x @= r; x @= s|] ==> r @= s
⟨proof⟩

lemma *number-of-approx-reorient*: (*number-of* w @= x) = (x @= *number-of* w)
⟨proof⟩

lemma *zero-approx-reorient*: (*0* @= x) = (x @= *0*)
⟨proof⟩

lemma *one-approx-reorient*: (*1* @= x) = (x @= *1*)
⟨proof⟩

⟨ML⟩

lemma *Infinitesimal-approx-minus*: ($x-y$ ∈ *Infinitesimal*) = (x @= y)
⟨proof⟩

lemma *approx-monad-iff*: (x @= y) = (*monad*(x)=*monad*(y))
⟨proof⟩

lemma *Infinitesimal-approx*:
  [| x ∈ *Infinitesimal*; y ∈ *Infinitesimal* |] ==> x @= y
⟨proof⟩

lemma *approx-add*: [| a @= b; c @= d |] ==> $a+c$ @= $b+d$
⟨proof⟩

lemma *approx-minus*: a @= b ==> $-a$ @= $-b$
⟨proof⟩

lemma *approx-minus2*: $-a$ @= $-b$ ==> a @= b
⟨proof⟩

lemma *approx-minus-cancel* [*simp*]: ($-a$ @= $-b$) = (a @= b)
⟨proof⟩

lemma *approx-add-minus*: [| a @= b; c @= d |] ==> $a + -c$ @= $b + -d$
⟨proof⟩

lemma *approx-mult1*: [| a @= b; c: *HFinite*|] ==> $a*c$ @= $b*c$
⟨proof⟩

lemma *approx-mult2*: [|a @= b; c: *HFinite*|] ==> $c*a$ @= $c*b$
⟨proof⟩

lemma *approx-mult-subst*: [|u @= $v*x$; x @= y; v ∈ *HFinite*|] ==> u @= $v*y$

⟨*proof*⟩

lemma *approx-mult-subst2*: [| u @= $x*v$; x @= y; $v \in$ *HFinite* |] ==> u @= $y*v$
⟨*proof*⟩

lemma *approx-mult-subst-SReal*:
[| u @= $x*$*hypreal-of-real* v; x @= y |] ==> u @= $y*$*hypreal-of-real* v
⟨*proof*⟩

lemma *approx-eq-imp*: $a = b$ ==> a @= b
⟨*proof*⟩

lemma *Infinitesimal-minus-approx*: $x \in$ *Infinitesimal* ==> $-x$ @= x
⟨*proof*⟩

lemma *bex-Infinitesimal-iff*: ($\exists\, y \in$ *Infinitesimal*. $x + -z = y$) = (x @= z)
⟨*proof*⟩

lemma *bex-Infinitesimal-iff2*: ($\exists\, y \in$ *Infinitesimal*. $x = z + y$) = (x @= z)
⟨*proof*⟩

lemma *Infinitesimal-add-approx*: [| $y \in$ *Infinitesimal*; $x + y = z$ |] ==> x @= z
⟨*proof*⟩

lemma *Infinitesimal-add-approx-self*: $y \in$ *Infinitesimal* ==> x @= $x + y$
⟨*proof*⟩

lemma *Infinitesimal-add-approx-self2*: $y \in$ *Infinitesimal* ==> x @= $y + x$
⟨*proof*⟩

lemma *Infinitesimal-add-minus-approx-self*: $y \in$ *Infinitesimal* ==> x @= $x + -y$
⟨*proof*⟩

lemma *Infinitesimal-add-cancel*: [| $y \in$ *Infinitesimal*; $x+y$ @= z|] ==> x @= z
⟨*proof*⟩

lemma *Infinitesimal-add-right-cancel*:
[| $y \in$ *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*: [| $x \in$ *HFinite*; x @= y |] ==> $y \in$ *HFinite*
⟨*proof*⟩

lemma *approx-hypreal-of-real-HFinite*: x @= *hypreal-of-real D* ==> $x \in$ *HFinite*
⟨*proof*⟩

lemma *approx-mult-HFinite*:
[|a @= b; c @= d; b: *HFinite*; d: *HFinite*|] ==> $a*c$ @= $b*d$
⟨*proof*⟩

lemma *approx-mult-hypreal-of-real*:
[|a @= *hypreal-of-real b*; c @= *hypreal-of-real d* |]
==> $a*c$ @= *hypreal-of-real b***hypreal-of-real d*
⟨*proof*⟩

lemma *approx-SReal-mult-cancel-zero*:
[| $a \in$ *Reals*; $a \neq 0$; $a*x$ @= 0 |] ==> x @= 0
⟨*proof*⟩

lemma *approx-mult-SReal1*: [| $a \in$ *Reals*; x @= 0 |] ==> $x*a$ @= 0
⟨*proof*⟩

lemma *approx-mult-SReal2*: [| $a \in$ *Reals*; x @= 0 |] ==> $a*x$ @= 0
⟨*proof*⟩

lemma *approx-mult-SReal-zero-cancel-iff* [*simp*]:
[|$a \in$ *Reals*; $a \neq 0$ |] ==> ($a*x$ @= 0) = (x @= 0)
⟨*proof*⟩

lemma *approx-SReal-mult-cancel*:
[| $a \in$ *Reals*; $a \neq 0$; $a*$ w @= $a*z$ |] ==> w @= z
⟨*proof*⟩

lemma *approx-SReal-mult-cancel-iff1* [*simp*]:
[| $a \in$ *Reals*; $a \neq 0$|] ==> ($a*$ w @= $a*z$) = (w @= z)
⟨*proof*⟩

lemma *approx-le-bound*: [| $z \leq f$; f @= g; $g \leq z$ |] ==> f @= z
⟨*proof*⟩

13.6 Zero is the Only Infinitesimal that is also a Real

lemma *Infinitesimal-less-SReal*:
$[| \; x \in Reals;\; y \in Infinitesimal;\; 0 < x \; |] ==> y < x$
⟨*proof*⟩

lemma *Infinitesimal-less-SReal2*:
$y \in Infinitesimal ==> \forall\, r \in Reals.\; 0 < r --> y < r$
⟨*proof*⟩

lemma *SReal-not-Infinitesimal*:
$[| \; 0 < y;\;\; y \in Reals |] ==> y \notin Infinitesimal$
⟨*proof*⟩

lemma *SReal-minus-not-Infinitesimal*:
$[| \; y < 0;\;\; y \in Reals \; |] ==> y \notin Infinitesimal$
⟨*proof*⟩

lemma *SReal-Int-Infinitesimal-zero*: $Reals\ Int\ Infinitesimal = \{0\}$
⟨*proof*⟩

lemma *SReal-Infinitesimal-zero*: $[| \; x \in Reals;\; x \in Infinitesimal |] ==> x = 0$
⟨*proof*⟩

lemma *SReal-HFinite-diff-Infinitesimal*:
$[| \; x \in Reals;\; x \neq 0 \; |] ==> x \in HFinite - Infinitesimal$
⟨*proof*⟩

lemma *hypreal-of-real-HFinite-diff-Infinitesimal*:
$hypreal\text{-}of\text{-}real\ x \neq 0 ==> hypreal\text{-}of\text{-}real\ x \in HFinite - Infinitesimal$
⟨*proof*⟩

lemma *hypreal-of-real-Infinitesimal-iff-0* [*iff*]:
$(hypreal\text{-}of\text{-}real\ x \in Infinitesimal) = (x{=}0)$
⟨*proof*⟩

lemma *number-of-not-Infinitesimal* [*simp*]:
$number\text{-}of\ w \neq (0{::}hypreal) ==> number\text{-}of\ w \notin Infinitesimal$
⟨*proof*⟩

lemma *one-not-Infinitesimal* [*simp*]: $1 \notin Infinitesimal$
⟨*proof*⟩

lemma *approx-SReal-not-zero*: $[| \; y \in Reals;\; x \;@{=}\; y;\; y{\neq}\ 0 \; |] ==> x \neq 0$
⟨*proof*⟩

lemma *HFinite-diff-Infinitesimal-approx*:
$[| \; x \;@{=}\; y;\; y \in HFinite - Infinitesimal \; |]$
$==> x \in HFinite - Infinitesimal$

⟨*proof*⟩

lemma *Infinitesimal-ratio*:
$[| \ y \neq 0; \ \ y \in \mathit{Infinitesimal}; \ \ x/y \in \mathit{HFinite} \ |] \ ==> x \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *Infinitesimal-SReal-divide*:
$[| \ x \in \mathit{Infinitesimal}; \ y \in \mathit{Reals} \ |] \ ==> x/y \in \mathit{Infinitesimal}$
⟨*proof*⟩

13.7 Uniqueness: Two Infinitely Close Reals are Equal

lemma *SReal-approx-iff*: $[|x \in \mathit{Reals}; \ y \in \mathit{Reals}|] \ ==> (x \ @= \ y) = (x = y)$
⟨*proof*⟩

lemma *number-of-approx-iff* [*simp*]:
$(\mathit{number\text{-}of} \ v \ @= \ \mathit{number\text{-}of} \ w) = (\mathit{number\text{-}of} \ v = (\mathit{number\text{-}of} \ w :: \mathit{hypreal}))$
⟨*proof*⟩

lemma [*simp*]: $(0 \ @= \ \mathit{number\text{-}of} \ w) = ((\mathit{number\text{-}of} \ w :: \mathit{hypreal}) = 0)$
$(\mathit{number\text{-}of} \ w \ @= \ 0) = ((\mathit{number\text{-}of} \ w :: \mathit{hypreal}) = 0)$
$(1 \ @= \ \mathit{number\text{-}of} \ w) = ((\mathit{number\text{-}of} \ w :: \mathit{hypreal}) = 1)$
$(\mathit{number\text{-}of} \ w \ @= \ 1) = ((\mathit{number\text{-}of} \ w :: \mathit{hypreal}) = 1)$
$\sim (0 \ @= \ 1) \sim (1 \ @= \ 0)$
⟨*proof*⟩

lemma *hypreal-of-real-approx-iff* [*simp*]:
$(\mathit{hypreal\text{-}of\text{-}real} \ k \ @= \ \mathit{hypreal\text{-}of\text{-}real} \ m) = (k = m)$
⟨*proof*⟩

lemma *hypreal-of-real-approx-number-of-iff* [*simp*]:
$(\mathit{hypreal\text{-}of\text{-}real} \ k \ @= \ \mathit{number\text{-}of} \ w) = (k = \mathit{number\text{-}of} \ w)$
⟨*proof*⟩

lemma [*simp*]: $(\mathit{hypreal\text{-}of\text{-}real} \ k \ @= \ 0) = (k = 0)$
$(\mathit{hypreal\text{-}of\text{-}real} \ k \ @= \ 1) = (k = 1)$
⟨*proof*⟩

lemma *approx-unique-real*:
$[| \ r \in \mathit{Reals}; \ s \in \mathit{Reals}; \ r \ @= \ x; \ s \ @= \ x|] \ ==> r = s$
⟨*proof*⟩

13.8 Existence of Unique Real Infinitely Close

lemma *hypreal-isLub-unique*:
$[| \ \mathit{isLub} \ R \ S \ x; \ \mathit{isLub} \ R \ S \ y \ |] \ ==> x = (y::\mathit{hypreal})$

⟨*proof*⟩

lemma *lemma-st-part-ub*:
$x \in HFinite ==> \exists u.\ isUb\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ u$
⟨*proof*⟩

lemma *lemma-st-part-nonempty*: $x \in HFinite ==> \exists y.\ y \in \{s.\ s \in Reals\ \&\ s < x\}$
⟨*proof*⟩

lemma *lemma-st-part-subset*: $\{s.\ s \in Reals\ \&\ s < x\} \subseteq Reals$
⟨*proof*⟩

lemma *lemma-st-part-lub*:
$x \in HFinite ==> \exists t.\ isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t$
⟨*proof*⟩

lemma *lemma-hypreal-le-left-cancel*: $((t::hypreal) + r \leq t) = (r \leq 0)$
⟨*proof*⟩

lemma *lemma-st-part-le1*:
$[|\ x \in HFinite;\ \ isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ \ 0 < r\ |] ==> x \leq t + r$
⟨*proof*⟩

lemma *hypreal-setle-less-trans*:
$!!x::hypreal.\ [|\ S *<= x;\ x < y\ |] ==> S *<= y$
⟨*proof*⟩

lemma *hypreal-gt-isUb*:
$!!x::hypreal.\ [|\ isUb\ R\ S\ x;\ x < y;\ y \in R\ |] ==> isUb\ R\ S\ y$
⟨*proof*⟩

lemma *lemma-st-part-gt-ub*:
$[|\ x \in HFinite;\ x < y;\ y \in Reals\ |]$
$==> isUb\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ y$
⟨*proof*⟩

lemma *lemma-minus-le-zero*: $t \leq t + -r ==> r \leq (0::hypreal)$
⟨*proof*⟩

lemma *lemma-st-part-le2*:
$[|\ x \in HFinite;$
$isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ 0 < r\ |]$
$==> t + -r \leq x$
⟨*proof*⟩

lemma *lemma-st-part1a*:

$[| \ x \in HFinite;$
$\quad isLub \ Reals \ \{s. \ s \in Reals \ \& \ s < x\} \ t;$
$\quad r \in Reals; \ 0 < r \ |]$
$==> x + -t \leq r$
⟨*proof*⟩

lemma *lemma-st-part2a*:
$[| \ x \in HFinite;$
$\quad isLub \ Reals \ \{s. \ s \in Reals \ \& \ s < x\} \ t;$
$\quad r \in Reals; \ 0 < r \ |]$
$==> -(x + -t) \leq r$
⟨*proof*⟩

lemma *lemma-SReal-ub*:
$(x::hypreal) \in Reals ==> isUb \ Reals \ \{s. \ s \in Reals \ \& \ s < x\} \ x$
⟨*proof*⟩

lemma *lemma-SReal-lub*:
$(x::hypreal) \in Reals ==> isLub \ Reals \ \{s. \ s \in Reals \ \& \ s < x\} \ x$
⟨*proof*⟩

lemma *lemma-st-part-not-eq1*:
$[| \ x \in HFinite;$
$\quad isLub \ Reals \ \{s. \ s \in Reals \ \& \ s < x\} \ t;$
$\quad r \in Reals; \ 0 < r \ |]$
$==> x + -t \neq r$
⟨*proof*⟩

lemma *lemma-st-part-not-eq2*:
$[| \ x \in HFinite;$
$\quad isLub \ Reals \ \{s. \ s \in Reals \ \& \ s < x\} \ t;$
$\quad r \in Reals; \ 0 < r \ |]$
$==> -(x + -t) \neq r$
⟨*proof*⟩

lemma *lemma-st-part-major*:
$[| \ x \in HFinite;$
$\quad isLub \ Reals \ \{s. \ s \in Reals \ \& \ s < x\} \ t;$
$\quad r \in Reals; \ 0 < r \ |]$
$==> abs \ (x + -t) < r$
⟨*proof*⟩

lemma *lemma-st-part-major2*:
$[| \ x \in HFinite; isLub \ Reals \ \{s. \ s \in Reals \ \& \ s < x\} \ t \ |]$
$==> \forall r \in Reals. \ 0 < r --> abs \ (x + -t) < r$
⟨*proof*⟩

Existence of real and Standard Part Theorem

lemma *lemma-st-part-Ex*:

$x \in HFinite ==> \exists\, t \in Reals.\ \forall\, r \in Reals.\ 0 < r --> abs\ (x + -t) < r$
⟨proof⟩

lemma *st-part-Ex*:
$x \in HFinite ==> \exists\, t \in Reals.\ x\ @= t$
⟨proof⟩

There is a unique real infinitely close

lemma *st-part-Ex1*: $x \in HFinite ==> EX!\ t.\ t \in Reals\ \&\ x\ @= t$
⟨proof⟩

13.9 Finite, Infinite and Infinitesimal

lemma *HFinite-Int-HInfinite-empty* [*simp*]: $HFinite\ Int\ HInfinite = \{\}$
⟨proof⟩

lemma *HFinite-not-HInfinite*:
assumes x: $x \in HFinite$ **shows** $x \notin HInfinite$
⟨proof⟩

lemma *not-HFinite-HInfinite*: $x \notin HFinite ==> x \in HInfinite$
⟨proof⟩

lemma *HInfinite-HFinite-disj*: $x \in HInfinite \mid x \in HFinite$
⟨proof⟩

lemma *HInfinite-HFinite-iff*: $(x \in HInfinite) = (x \notin HFinite)$
⟨proof⟩

lemma *HFinite-HInfinite-iff*: $(x \in HFinite) = (x \notin HInfinite)$
⟨proof⟩

lemma *HInfinite-diff-HFinite-Infinitesimal-disj*:
$x \notin Infinitesimal ==> x \in HInfinite \mid x \in HFinite - Infinitesimal$
⟨proof⟩

lemma *HFinite-inverse*:
$[|\ x \in HFinite;\ x \notin Infinitesimal\ |] ==> inverse\ x \in HFinite$
⟨proof⟩

lemma *HFinite-inverse2*: $x \in HFinite - Infinitesimal ==> inverse\ x \in HFinite$
⟨proof⟩

lemma *Infinitesimal-inverse-HFinite*:
$x \notin Infinitesimal ==> inverse(x) \in HFinite$
⟨proof⟩

lemma *HFinite-not-Infinitesimal-inverse*:
$x \in HFinite - Infinitesimal ==> inverse\ x \in HFinite - Infinitesimal$
⟨*proof*⟩

lemma *approx-inverse*:
$[|\ x\ @=\ y;\ y \in\ HFinite - Infinitesimal\ |]$
$==> inverse\ x\ @=\ inverse\ y$
⟨*proof*⟩

lemmas *hypreal-of-real-approx-inverse* = *hypreal-of-real-HFinite-diff-Infinitesimal* [*THEN* [*2*] *approx-inverse*]

lemma *inverse-add-Infinitesimal-approx*:
$[|\ x \in HFinite - Infinitesimal;$
$h \in Infinitesimal\ |] ==> inverse(x + h)\ @=\ inverse\ x$
⟨*proof*⟩

lemma *inverse-add-Infinitesimal-approx2*:
$[|\ x \in HFinite - Infinitesimal;$
$h \in Infinitesimal\ |] ==> inverse(h + x)\ @=\ inverse\ x$
⟨*proof*⟩

lemma *inverse-add-Infinitesimal-approx-Infinitesimal*:
$[|\ x \in HFinite - Infinitesimal;$
$h \in Infinitesimal\ |] ==> inverse(x + h) + -inverse\ x\ @=\ h$
⟨*proof*⟩

lemma *Infinitesimal-square-iff*: $(x \in Infinitesimal) = (x*x \in Infinitesimal)$
⟨*proof*⟩
declare *Infinitesimal-square-iff* [*symmetric*, *simp*]

lemma *HFinite-square-iff* [*simp*]: $(x*x \in HFinite) = (x \in HFinite)$
⟨*proof*⟩

lemma *HInfinite-square-iff* [*simp*]: $(x*x \in HInfinite) = (x \in HInfinite)$
⟨*proof*⟩

lemma *approx-HFinite-mult-cancel*:
$[|\ a\!: HFinite{-}Infinitesimal;\ a*\ w\ @=\ a*z\ |] ==> w\ @=\ z$
⟨*proof*⟩

lemma *approx-HFinite-mult-cancel-iff1*:
$a\!: HFinite{-}Infinitesimal ==> (a * w\ @=\ a * z) = (w\ @=\ z)$
⟨*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*:
$[| \ x \in HInfinite; \ x \leq y; \ 0 \leq x \ |] ==> y \in HInfinite$
⟨*proof*⟩

lemma *Infinitesimal-inverse-HInfinite*:
$[| \ x \in Infinitesimal; \ x \neq 0 \ |] ==> inverse \ x \in HInfinite$
⟨*proof*⟩

lemma *HInfinite-HFinite-not-Infinitesimal-mult*:
$[| \ x \in HInfinite; \ y \in HFinite - Infinitesimal \ |]$
$==> x * y \in HInfinite$
⟨*proof*⟩

lemma *HInfinite-HFinite-not-Infinitesimal-mult2*:
$[| \ x \in HInfinite; \ y \in HFinite - Infinitesimal \ |]$
$==> y * x \in HInfinite$
⟨*proof*⟩

lemma *HInfinite-gt-SReal*: $[| \ x \in HInfinite; \ 0 < x; \ y \in Reals \ |] ==> y < x$
⟨*proof*⟩

lemma *HInfinite-gt-zero-gt-one*: $[| \ x \in HInfinite; \ 0 < x \ |] ==> 1 < x$
⟨*proof*⟩

lemma *not-HInfinite-one* [*simp*]: $1 \notin HInfinite$
⟨*proof*⟩

lemma *approx-hrabs-disj*: $abs \ x \ @= x \ | \ abs \ x \ @= -x$
⟨*proof*⟩

13.10 Theorems about Monads

lemma *monad-hrabs-Un-subset*: $monad \ (abs \ x) \leq monad(x) \ Un \ monad(-x)$
⟨*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*: $(x \in monad\ 0) = (abs\ x \in monad\ 0)$
⟨*proof*⟩

lemma *mem-monad-self* [*simp*]: $x \in monad\ x$
⟨*proof*⟩

13.11 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; $x \in Infinitesimal$ |] ==> $abs\ x$ @= $abs\ y$
⟨*proof*⟩

lemma *less-Infinitesimal-less*:
[| $0 < x$; $x \notin Infinitesimal$; e :*Infinitesimal* |] ==> $e < x$
⟨*proof*⟩

lemma *Ball-mem-monad-gt-zero*:
[| $0 < x$; $x \notin Infinitesimal$; $u \in monad\ x$ |] ==> $0 < u$
⟨*proof*⟩

lemma *Ball-mem-monad-less-zero*:
[| $x < 0$; $x \notin Infinitesimal$; $u \in monad\ x$ |] ==> $u < 0$
⟨*proof*⟩

lemma *lemma-approx-gt-zero*:
[|$0 < x$; $x \notin Infinitesimal$; x @= y|] ==> $0 < y$
⟨*proof*⟩

lemma *lemma-approx-less-zero*:
 $[|x < 0;\ x \notin Infinitesimal;\ x\ @{=}\ y|]\ ==>\ y < 0$
⟨*proof*⟩

theorem *approx-hrabs*: $x\ @{=}\ y\ ==>\ abs\ x\ @{=}\ abs\ y$
⟨*proof*⟩

lemma *approx-hrabs-zero-cancel*: $abs(x)\ @{=}\ 0\ ==>\ x\ @{=}\ 0$
⟨*proof*⟩

lemma *approx-hrabs-add-Infinitesimal*: $e \in Infinitesimal\ ==>\ abs\ x\ @{=}\ abs(x{+}e)$
⟨*proof*⟩

lemma *approx-hrabs-add-minus-Infinitesimal*:
 $e \in Infinitesimal\ ==>\ abs\ x\ @{=}\ abs(x + -e)$
⟨*proof*⟩

lemma *hrabs-add-Infinitesimal-cancel*:
 $[|\ e \in Infinitesimal;\ e' \in Infinitesimal;$
  $abs(x{+}e) = abs(y{+}e')|]\ ==>\ abs\ x\ @{=}\ abs\ y$
⟨*proof*⟩

lemma *hrabs-add-minus-Infinitesimal-cancel*:
 $[|\ e \in Infinitesimal;\ e' \in Infinitesimal;$
  $abs(x + -e) = abs(y + -e')|]\ ==>\ abs\ x\ @{=}\ abs\ y$
⟨*proof*⟩

lemma *Infinitesimal-add-hypreal-of-real-less*:
 $[|\ x < y;\ \ u \in Infinitesimal\ |]$
 $==>\ hypreal\text{-}of\text{-}real\ x + u < hypreal\text{-}of\text{-}real\ y$
⟨*proof*⟩

lemma *Infinitesimal-add-hrabs-hypreal-of-real-less*:
 $[|\ x \in Infinitesimal;\ abs(hypreal\text{-}of\text{-}real\ r) < hypreal\text{-}of\text{-}real\ y\ |]$
 $==>\ abs\ (hypreal\text{-}of\text{-}real\ r + x) < hypreal\text{-}of\text{-}real\ y$
⟨*proof*⟩

lemma *Infinitesimal-add-hrabs-hypreal-of-real-less2*:
 $[|\ x \in Infinitesimal;\ \ abs(hypreal\text{-}of\text{-}real\ r) < hypreal\text{-}of\text{-}real\ y\ |]$
 $==>\ abs\ (x + hypreal\text{-}of\text{-}real\ r) < hypreal\text{-}of\text{-}real\ y$
⟨*proof*⟩

lemma *hypreal-of-real-le-add-Infininitesimal-cancel*:
 $[|\ u \in Infinitesimal;\ v \in Infinitesimal;$
  $hypreal\text{-}of\text{-}real\ x + u \le hypreal\text{-}of\text{-}real\ y + v\ |]$
 $==>\ hypreal\text{-}of\text{-}real\ x \le hypreal\text{-}of\text{-}real\ y$
⟨*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*; $x \neq 0$ |] ==> *hypreal-of-real* x + $h \neq 0$
⟨*proof*⟩

lemma *Infinitesimal-square-cancel* [*simp*]:
$x*x + y*y \in$ *Infinitesimal* ==> $x*x \in$ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-square-cancel* [*simp*]: $x*x + y*y \in$ *HFinite* ==> $x*x \in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-square-cancel2* [*simp*]:
$x*x + y*y \in$ *Infinitesimal* ==> $y*y \in$ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-square-cancel2* [*simp*]: $x*x + y*y \in$ *HFinite* ==> $y*y \in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-sum-square-cancel* [*simp*]:
$x*x + y*y + z*z \in$ *Infinitesimal* ==> $x*x \in$ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-sum-square-cancel* [*simp*]:
$x*x + y*y + z*z \in$ *HFinite* ==> $x*x \in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-sum-square-cancel2* [*simp*]:
$y*y + x*x + z*z \in$ *Infinitesimal* ==> $x*x \in$ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-sum-square-cancel2* [*simp*]:
$y*y + x*x + z*z \in$ *HFinite* ==> $x*x \in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-sum-square-cancel3* [*simp*]:
$z*z + y*y + x*x \in$ *Infinitesimal* ==> $x*x \in$ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-sum-square-cancel3* [*simp*]:
$z*z + y*y + x*x \in HFinite ==> x*x \in HFinite$
⟨*proof*⟩

lemma *monad-hrabs-less*:
$[| \ y \in monad\ x;\ 0 < hypreal\text{-}of\text{-}real\ e \ |]$
$==> abs\ (y + -x) < hypreal\text{-}of\text{-}real\ e$
⟨*proof*⟩

lemma *mem-monad-SReal-HFinite*:
$x \in monad\ (hypreal\text{-}of\text{-}real\ \ a) ==> x \in HFinite$
⟨*proof*⟩

13.12 Theorems about Standard Part

lemma *st-approx-self*: $x \in HFinite ==> st\ x\ @= x$
⟨*proof*⟩

lemma *st-SReal*: $x \in HFinite ==> st\ x \in Reals$
⟨*proof*⟩

lemma *st-HFinite*: $x \in HFinite ==> st\ x \in HFinite$
⟨*proof*⟩

lemma *st-SReal-eq*: $x \in Reals ==> st\ x = x$
⟨*proof*⟩

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

lemma *st-eq-approx*: $[| \ x \in HFinite;\ y \in HFinite;\ st\ x = st\ y \ |] ==> x\ @= y$
⟨*proof*⟩

lemma *approx-st-eq*:
assumes $x \in HFinite$ **and** $y \in HFinite$ **and** $x\ @= y$
shows $st\ x = st\ y$
⟨*proof*⟩

lemma *st-eq-approx-iff*:
$[| \ x \in HFinite;\ y \in HFinite|]$
$==> (x\ @= y) = (st\ x = st\ y)$
⟨*proof*⟩

lemma *st-Infinitesimal-add-SReal*:
$[| \ x \in Reals;\ e \in Infinitesimal \ |] ==> st(x + e) = x$
⟨*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*:
assumes *x*: $x \in HFinite$ **and** *y*: $y \in HFinite$
shows $st \; (x + y) = st(x) + st(y)$
⟨*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*: **assumes** $y \in HFinite$ **shows** $st(-y) = -st(y)$
⟨*proof*⟩

lemma *st-diff*: $[| \; x \in HFinite; \; y \in HFinite \; |] \; ==> \; st \; (x-y) = st(x) - st(y)$
⟨*proof*⟩

lemma *lemma-st-mult*:
$[| \; x \in HFinite; \; y \in HFinite; \; e \in Infinitesimal; \; ea \in Infinitesimal \; |]$
$==> \; e*y + x*ea + e*ea \in Infinitesimal$
⟨*proof*⟩

lemma *st-mult*: $[| \; x \in HFinite; \; y \in HFinite \; |] \; ==> \; st \; (x * y) = st(x) * st(y)$
⟨*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*⟩

13.13 Alternative Definitions for *HFinite* using Free Ultrafilter

lemma *FreeUltrafilterNat-Rep-hypreal*:
  $[| X \in Rep\text{-}star\ x; Y \in Rep\text{-}star\ x |]$
  $==> \{n.\ X\ n = Y\ n\} \in FreeUltrafilterNat$
⟨*proof*⟩

lemma *HFinite-FreeUltrafilterNat*:
  $x \in HFinite$
  $==> \exists X \in Rep\text{-}star\ x.\ \exists u.\ \{n.\ abs\ (X\ n) < u\} \in FreeUltrafilterNat$
⟨*proof*⟩

lemma *FreeUltrafilterNat-HFinite*:
  $\exists X \in Rep\text{-}star\ x.$
    $\exists u.\ \{n.\ abs\ (X\ n) < u\} \in FreeUltrafilterNat$
    $==> x \in HFinite$
⟨*proof*⟩

lemma *HFinite-FreeUltrafilterNat-iff*:
  $(x \in HFinite) = (\exists X \in Rep\text{-}star\ x.$
    $\exists u.\ \{n.\ abs\ (X\ n) < u\} \in FreeUltrafilterNat)$
⟨*proof*⟩

13.14 Alternative Definitions for *HInfinite* using Free Ultrafilter

lemma *lemma-Compl-eq*: $- \{n.\ (u{::}real) < abs\ (xa\ n)\} = \{n.\ abs\ (xa\ n) \leq u\}$
⟨*proof*⟩

lemma *lemma-Compl-eq2*: $- \{n.\ abs\ (xa\ n) < (u{::}real)\} = \{n.\ u \leq abs\ (xa\ n)\}$
⟨*proof*⟩

lemma *lemma-Int-eq1*:
$\{n.\ abs\ (xa\ n) \leq (u{::}real)\}\ Int\ \{n.\ u \leq abs\ (xa\ n)\}$
$= \{n.\ abs(xa\ n) = u\}$
⟨*proof*⟩

lemma *lemma-FreeUltrafilterNat-one*:
$\{n.\ abs\ (xa\ n) = u\} \leq \{n.\ abs\ (xa\ n) < u + (1{::}real)\}$
⟨*proof*⟩

lemma *FreeUltrafilterNat-const-Finite*:
$[|\ xa$: $Rep\text{-}star\ x$;
$\{n.\ abs\ (xa\ n) = u\} \in FreeUltrafilterNat$
$|] ==> x \in HFinite$
⟨*proof*⟩

lemma *HInfinite-FreeUltrafilterNat*:
$x \in HInfinite ==> \exists X \in Rep\text{-}star\ x.$
$\forall u.\ \{n.\ u < abs\ (X\ n)\} \in FreeUltrafilterNat$
⟨*proof*⟩

lemma *lemma-Int-HI*:
$\{n.\ abs\ (Xa\ n) < u\}\ Int\ \{n.\ X\ n = Xa\ n\} \subseteq \{n.\ abs\ (X\ n) < (u{::}real)\}$
⟨*proof*⟩

lemma *lemma-Int-HIa*: $\{n.\ u < abs\ (X\ n)\}\ Int\ \{n.\ abs\ (X\ n) < (u{::}real)\} = \{\}$
⟨*proof*⟩

lemma *FreeUltrafilterNat-HInfinite*:
$\exists X \in Rep\text{-}star\ x.\ \forall u.$
$\{n.\ u < abs\ (X\ n)\} \in FreeUltrafilterNat$
$==>\ x \in HInfinite$
⟨*proof*⟩

lemma *HInfinite-FreeUltrafilterNat-iff*:
$(x \in HInfinite) = (\exists X \in Rep\text{-}star\ x.$
$\forall u.\ \{n.\ u < abs\ (X\ n)\} \in FreeUltrafilterNat)$
⟨*proof*⟩

13.15 Alternative Definitions for *Infinitesimal* using Free Ultrafilter

lemma *Infinitesimal-FreeUltrafilterNat*:
$x \in \mathit{Infinitesimal} ==> \exists X \in \mathit{Rep\text{-}star}\ x.$
$\forall u.\ 0 < u --> \{n.\ \mathit{abs}\ (X\ n) < u\} \in \mathit{FreeUltrafilterNat}$
⟨*proof*⟩

lemma *FreeUltrafilterNat-Infinitesimal*:
$\exists X \in \mathit{Rep\text{-}star}\ x.$
$\forall u.\ 0 < u --> \{n.\ \mathit{abs}\ (X\ n) < u\} \in \mathit{FreeUltrafilterNat}$
$==> x \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *Infinitesimal-FreeUltrafilterNat-iff*:
$(x \in \mathit{Infinitesimal}) = (\exists X \in \mathit{Rep\text{-}star}\ x.$
$\forall u.\ 0 < u --> \{n.\ \mathit{abs}\ (X\ n) < u\} \in \mathit{FreeUltrafilterNat})$
⟨*proof*⟩

lemma *lemma-Infinitesimal*:
$(\forall r.\ 0 < r --> x < r) = (\forall n.\ x < \mathit{inverse}(\mathit{real}\ (\mathit{Suc}\ n)))$
⟨*proof*⟩

lemma *of-nat-in-Reals* [*simp*]: $(\mathit{of\text{-}nat}\ n{::}\mathit{hypreal}) \in \mathbb{R}$
⟨*proof*⟩

lemma *lemma-Infinitesimal2*:
$(\forall r \in \mathit{Reals}.\ 0 < r --> x < r) =$
$(\forall n.\ x < \mathit{inverse}(\mathit{hypreal\text{-}of\text{-}nat}\ (\mathit{Suc}\ n)))$
⟨*proof*⟩

lemma *Infinitesimal-hypreal-of-nat-iff*:
$\mathit{Infinitesimal} = \{x.\ \forall n.\ \mathit{abs}\ x < \mathit{inverse}\ (\mathit{hypreal\text{-}of\text{-}nat}\ (\mathit{Suc}\ n))\}$
⟨*proof*⟩

13.16 Proof that ω is an infinite number

It will follow that epsilon is an infinitesimal number.

lemma *Suc-Un-eq*: $\{n.\ n < \mathit{Suc}\ m\} = \{n.\ n < m\}\ \mathit{Un}\ \{n.\ n = m\}$
⟨*proof*⟩

lemma *finite-nat-segment*: $\mathit{finite}\ \{n{::}\mathit{nat}.\ n < m\}$
⟨*proof*⟩

lemma *finite-real-of-nat-segment*: $\mathit{finite}\ \{n{::}\mathit{nat}.\ \mathit{real}\ n < \mathit{real}\ (m{::}\mathit{nat})\}$

⟨*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*:
$0 < u \implies (u < inverse\ (real(Suc\ n))) = (real(Suc\ n) < inverse\ u)$

⟨*proof*⟩

lemma *finite-inverse-real-of-posnat-gt-real*:
$0 < u ==> \mathit{finite}\ \{n.\ u < \mathit{inverse}(\mathit{real}(\mathit{Suc}\ n))\}$
⟨*proof*⟩

lemma *lemma-real-le-Un-eq2*:
$\{n.\ u \le \mathit{inverse}(\mathit{real}(\mathit{Suc}\ n))\} =$
$\{n.\ u < \mathit{inverse}(\mathit{real}(\mathit{Suc}\ n))\}\ \mathit{Un}\ \{n.\ u = \mathit{inverse}(\mathit{real}(\mathit{Suc}\ n))\}$
⟨*proof*⟩

lemma *real-of-nat-inverse-le-iff*:
$(\mathit{inverse}\ (\mathit{real}(\mathit{Suc}\ n)) \le r) = (1 \le r * \mathit{real}(\mathit{Suc}\ n))$
⟨*proof*⟩

lemma *real-of-nat-inverse-eq-iff*:
$(u = \mathit{inverse}\ (\mathit{real}(\mathit{Suc}\ n))) = (\mathit{real}(\mathit{Suc}\ n) = \mathit{inverse}\ u)$
⟨*proof*⟩

lemma *lemma-finite-omega-set2*: $\mathit{finite}\ \{n{::}\mathit{nat}.\ u = \mathit{inverse}(\mathit{real}(\mathit{Suc}\ n))\}$
⟨*proof*⟩

lemma *finite-inverse-real-of-posnat-ge-real*:
$0 < u ==> \mathit{finite}\ \{n.\ u \le \mathit{inverse}(\mathit{real}(\mathit{Suc}\ n))\}$
⟨*proof*⟩

lemma *inverse-real-of-posnat-ge-real-FreeUltrafilterNat*:
$0 < u ==> \{n.\ u \le \mathit{inverse}(\mathit{real}(\mathit{Suc}\ n))\} \notin \mathit{FreeUltrafilterNat}$
⟨*proof*⟩

lemma *Compl-le-inverse-eq*:
$-\ \{n.\ u \le \mathit{inverse}(\mathit{real}(\mathit{Suc}\ n))\} =$
$\{n.\ \mathit{inverse}(\mathit{real}(\mathit{Suc}\ n)) < u\}$
⟨*proof*⟩

lemma *FreeUltrafilterNat-inverse-real-of-posnat*:
$0 < u ==>$
$\{n.\ \mathit{inverse}(\mathit{real}(\mathit{Suc}\ n)) < u\} \in \mathit{FreeUltrafilterNat}$
⟨*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*:
$\forall n.\ \mathit{abs}(X\ n + -x) < \mathit{inverse}(\mathit{real}(\mathit{Suc}\ n))$
$==> \mathit{star\text{-}n}\ X + -\mathit{hypreal\text{-}of\text{-}real}\ x \in \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *real-seq-to-hypreal-approx*:
$\forall n.\ abs(X\ n + -x) < inverse(real(Suc\ n))$
$==> star\text{-}n\ X\ @= hypreal\text{-}of\text{-}real\ x$
⟨*proof*⟩

lemma *real-seq-to-hypreal-approx2*:
$\forall n.\ abs(x + -X\ n) < inverse(real(Suc\ n))$
$==> star\text{-}n\ X\ @= hypreal\text{-}of\text{-}real\ x$
⟨*proof*⟩

lemma *real-seq-to-hypreal-Infinitesimal2*:
$\forall n.\ abs(X\ n + -Y\ n) < inverse(real(Suc\ n))$
$==> star\text{-}n\ X + -star\text{-}n\ Y \in Infinitesimal$
⟨*proof*⟩

⟨*ML*⟩

end

14 Star: Star-Transforms in Non-Standard Analysis

theory *Star*
imports *NSA*
begin

constdefs

$starset\text{-}n :: (nat => 'a\ set) => 'a\ star\ set \qquad (*sn*\ \text{-}\ [80]\ 80)$
$*sn*\ As == Iset\ (star\text{-}n\ As)$

$InternalSets :: 'a\ star\ set\ set$
$InternalSets == \{X.\ \exists As.\ X = *sn*\ As\}$

$is\text{-}starext :: ['a\ star => 'a\ star, 'a => 'a] => bool$
$is\text{-}starext\ F\ f == (\forall x\ y.\ \exists X \in Rep\text{-}star(x).\ \exists Y \in Rep\text{-}star(y).$
$((y = (F\ x)) = (\{n.\ Y\ n = f(X\ n)\} : FreeUltrafilterNat)))$

$starfun\text{-}n :: (nat => ('a => 'b)) => 'a\ star => 'b\ star$
$(*fn*\ \text{-}\ [80]\ 80)$
$*fn*\ F == Ifun\ (star\text{-}n\ F)$

$InternalFuns :: ('a\ star => 'b\ star)\ set$
$InternalFuns == \{X.\ \exists F.\ X = *fn*\ F\}$

lemma *no-choice*: $\forall x.\ \exists y.\ Q\ x\ y ==> \exists (f :: nat => nat).\ \forall x.\ Q\ x\ (f\ x)$
⟨*proof*⟩

14.1 Properties of the Star-transform Applied to Sets of Reals

lemma *STAR-UNIV-set*: $*s*(UNIV::'a\ set) = (UNIV::'a\ star\ set)$
⟨*proof*⟩

lemma *STAR-empty-set*: $*s*\ \{\} = \{\}$
⟨*proof*⟩

lemma *STAR-Un*: $*s*\ (A\ Un\ B) = *s*\ A\ Un\ *s*\ B$
⟨*proof*⟩

lemma *STAR-Int*: $*s*\ (A\ Int\ B) = *s*\ A\ Int\ *s*\ B$
⟨*proof*⟩

lemma *STAR-Compl*: $*s*\ -A = -(\ *s*\ A)$
⟨*proof*⟩

lemma *STAR-mem-Compl*: $!!x.\ x \notin *s*\ F ==> x : *s*\ (-\ F)$
⟨*proof*⟩

lemma *STAR-diff*: $*s*\ (A - B) = *s*\ A - *s*\ B$
⟨*proof*⟩

lemma *STAR-subset*: $A <= B ==> *s*\ A <= *s*\ B$
⟨*proof*⟩

lemma *STAR-mem*: $a \in A ==> star\text{-}of\ a : *s*\ A$
⟨*proof*⟩

lemma *STAR-mem-iff*: $(star\text{-}of\ x \in *s*\ A) = (x \in A)$
⟨*proof*⟩

lemma *STAR-star-of-image-subset*: $star\text{-}of\ `\ A <= *s*\ A$
⟨*proof*⟩

lemma *STAR-hypreal-of-real-Int*: $*s*\ X\ Int\ Reals = hypreal\text{-}of\text{-}real\ `\ X$
⟨*proof*⟩

lemma *lemma-not-hyprealA*: $x \notin hypreal\text{-}of\text{-}real\ `\ A ==> \forall y \in A.\ x \neq hypreal\text{-}of\text{-}real$

y
⟨*proof*⟩

lemma *lemma-Compl-eq*: $- \{n.\ X\ n = xa\} = \{n.\ X\ n \neq xa\}$
⟨*proof*⟩

lemma *STAR-real-seq-to-hypreal*:
$\forall n.\ (X\ n) \notin M$ ==> *star-n* $X \notin$ ∗*s*∗ M
⟨*proof*⟩

lemma *STAR-singleton*: ∗*s*∗ $\{x\} = \{$*star-of* $x\}$
⟨*proof*⟩

lemma *STAR-not-mem*: $x \notin F$ ==> *star-of* $x \notin$ ∗*s*∗ F
⟨*proof*⟩

lemma *STAR-subset-closed*: [| x : ∗*s*∗ A; A <= B |] ==> x : ∗*s*∗ B
⟨*proof*⟩

Nonstandard extension of a set (defined using a constant sequence) as a special case of an internal set

lemma *starset-n-starset*: $\forall n.\ (As\ n = A)$ ==> ∗*sn*∗ As = ∗*s*∗ A
⟨*proof*⟩

lemma *starfun-n-starfun*: $\forall n.\ (F\ n = f)$ ==> ∗*fn*∗ F = ∗*f*∗ f
⟨*proof*⟩

lemma *hrabs-is-starext-rabs*: *is-starext abs abs*
⟨*proof*⟩

lemma *Rep-star-FreeUltrafilterNat*:
[| $X \in$ *Rep-star* z; $Y \in$ *Rep-star* z |]
==> $\{n.\ X\ n = Y\ n\}$: *FreeUltrafilterNat*
⟨*proof*⟩

Nonstandard extension of functions

lemma *starfun*:
$(*f* \ f) \ (star\text{-}n \ X) = star\text{-}n \ (\%n. \ f \ (X \ n))$
⟨*proof*⟩

lemma *starfun-if-eq*:
$!!w. \ w \neq star\text{-}of \ x$
$==> (*f* \ (\lambda z. \ if \ z = x \ then \ a \ else \ g \ z)) \ w = (*f* \ g) \ w$
⟨*proof*⟩

lemma *starfun-mult*: $!!x. \ (*f* \ f) \ x * (*f* \ g) \ x = (*f* \ (\%x. \ f \ x * g \ x)) \ x$
⟨*proof*⟩
declare *starfun-mult* [*symmetric*, *simp*]

lemma *starfun-add*: $!!x. \ (*f* \ f) \ x + (*f* \ g) \ x = (*f* \ (\%x. \ f \ x + g \ x)) \ x$
⟨*proof*⟩
declare *starfun-add* [*symmetric*, *simp*]

lemma *starfun-minus*: $!!x. \ - (*f* \ f) \ x = (*f* \ (\%x. \ - f \ x)) \ x$
⟨*proof*⟩
declare *starfun-minus* [*symmetric*, *simp*]

lemma *starfun-add-minus*: $!!x. \ (*f* \ f) \ x + -(*f* \ g) \ x = (*f* \ (\%x. \ f \ x + -g \ x)) \ x$
⟨*proof*⟩
declare *starfun-add-minus* [*symmetric*, *simp*]

lemma *starfun-diff*: $!!x. \ (*f* \ f) \ x \ - (*f* \ g) \ x = (*f* \ (\%x. \ f \ x - g \ x)) \ x$
⟨*proof*⟩
declare *starfun-diff* [*symmetric*, *simp*]

lemma *starfun-o2*: $(\%x. \ (*f* \ f) \ ((*f* \ g) \ x)) = *f* \ (\%x. \ f \ (g \ x))$
⟨*proof*⟩

lemma *starfun-o*: $(*f* \ f) \ o \ (*f* \ g) = (*f* \ (f \ o \ g))$
⟨*proof*⟩

NS extension of constant function

lemma *starfun-const-fun* [*simp*]: $!!x. \ (*f* \ (\%x. \ k)) \ x = star\text{-}of \ k$
⟨*proof*⟩

the NS extension of the identity function

lemma *starfun-Id* [*simp*]: $!!x. \ (*f* \ (\%x. \ x)) \ x = x$
⟨*proof*⟩

lemma *starfun-Idfun-approx*:
x @= *hypreal-of-real* a ==> (*f* (%x. x)) x @= *hypreal-of-real* 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* [*simp*]: (*f* f) (*star-of* a) = *star-of* (f a)
⟨*proof*⟩

lemma *starfun-approx*: (*f* f) (*star-of* a) @= *hypreal-of-real* (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 o g)) (*star-of* x + x')
⟨*proof*⟩

lemma *starfun-mult-HFinite-approx*: [| (*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*]: $(*f* \ inverse) \ x = inverse(x)$
$\langle proof \rangle$

lemma *starfun-inverse*: $!!x. \ inverse \ ((*f* \ f) \ x) = (*f* \ (\%x. \ inverse \ (f \ x))) \ x$
$\langle proof \rangle$
declare *starfun-inverse* [*symmetric*, *simp*]

lemma *starfun-divide*: $!!x. \ (*f* \ f) \ x \ / \ (*f* \ g) \ x = (*f* \ (\%x. \ f \ x \ / \ g \ x)) \ x$
$\langle proof \rangle$
declare *starfun-divide* [*symmetric*, *simp*]

lemma *starfun-inverse2*: $!!x. \ inverse \ ((*f* \ f) \ x) = (*f* \ (\%x. \ inverse \ (f \ x))) \ x$
$\langle proof \rangle$

General lemma/theorem needed for proofs in elementary topology of the reals

lemma *starfun-mem-starset*:
$!!x. \ (*f* \ f) \ x : *s* \ A ==> x : *s* \ \{x. \ f \ x \ \in A\}$
$\langle proof \rangle$

Alternative definition for hrabs with rabs function applied entrywise to equivalence class representative. This is easily proved using starfun and ns extension thm

lemma *hypreal-hrabs*:
$abs \ (star\text{-}n \ X) = star\text{-}n \ (\%n. \ abs \ (X \ n))$
$\langle proof \rangle$

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*:
$*s* \ \{x. \ abs \ (x + - \ y) < r\} =$
$\{x. \ abs(x + -hypreal\text{-}of\text{-}real \ y) < hypreal\text{-}of\text{-}real \ r\}$
$\langle proof \rangle$

lemma *STAR-starfun-rabs-add-minus*:
$*s* \ \{x. \ abs \ (f \ x + - \ y) < r\} =$
$\{x. \ abs((*f* \ f) \ x + -hypreal\text{-}of\text{-}real \ y) < hypreal\text{-}of\text{-}real \ r\}$
$\langle proof \rangle$

Another characterization of Infinitesimal and one of @= relation. In this theory since *hypreal-hrabs* proved here. Maybe move both theorems??

lemma *Infinitesimal-FreeUltrafilterNat-iff2*:
$(x \in Infinitesimal) =$
$(\exists X \in Rep\text{-}star(x).$
$\forall m. \ \{n. \ abs(X \ n) < inverse(real(Suc \ m))\}$

$\in$ *FreeUltrafilterNat*)
⟨*proof*⟩

lemma *approx-FreeUltrafilterNat-iff*: *star-n X @= star-n Y* =
($\forall$ *m*. {*n*. *abs* (*X n* + − *Y n*) <
inverse(*real*(*Suc m*))} : *FreeUltrafilterNat*)
⟨*proof*⟩

lemma *inj-starfun*: *inj starfun*
⟨*proof*⟩

⟨*ML*⟩

end

15 HyperNat: Hypernatural numbers

theory *HyperNat*
imports *Star*
begin

types *hypnat* = *nat star*

syntax *hypnat-of-nat* :: *nat* => *nat star*
translations *hypnat-of-nat* => *star-of* :: *nat* => *nat star*

15.1 Properties Transferred from Naturals

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*]: $!!n.\ (n \le (0::hypnat)) = (n = 0)$
⟨*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*]: $!!m\ n.\ ((m::hypnat) - n = 0) = (m \le n)$
⟨*proof*⟩

lemma *hypnat-not-less0* [*iff*]: $!!n.\ \sim n < (0::hypnat)$
⟨*proof*⟩

lemma *hypnat-less-one* [*iff*]:
$!!n.\ (n < (1::hypnat)) = (n{=}0)$
⟨*proof*⟩

lemma *hypnat-add-diff-inverse*: $!!m\ n.\ \sim m{<}n \implies n{+}(m{-}n) = (m::hypnat)$
⟨*proof*⟩

lemma *hypnat-le-add-diff-inverse* [*simp*]: $!!m\ n.\ n \le m \implies n{+}(m{-}n) = (m::hypnat)$
⟨*proof*⟩

lemma *hypnat-le-add-diff-inverse2* [*simp*]: $!!m\ n.\ n{\le}m \implies (m{-}n){+}n = (m::hypnat)$
⟨*proof*⟩

declare *hypnat-le-add-diff-inverse2* [*OF order-less-imp-le*]

lemma *hypnat-le0* [*iff*]: $!!n.\ (0::hypnat) \le n$
⟨*proof*⟩

lemma *hypnat-add-self-le* [*simp*]: $!!x\ n.\ (x::hypnat) \le n + x$

⟨*proof*⟩

lemma *hypnat-add-one-self-less* [*simp*]: $(x::hypnat) < x + (1::hypnat)$
⟨*proof*⟩

lemma *hypnat-neq0-conv* [*iff*]: $!!n.\ (n \neq 0) = (0 < (n::hypnat))$
⟨*proof*⟩

lemma *hypnat-gt-zero-iff*: $((0::hypnat) < n) = ((1::hypnat) \leq n)$
⟨*proof*⟩

lemma *hypnat-gt-zero-iff2*: $(0 < n) = (\exists\, m.\ n = m + (1::hypnat))$
⟨*proof*⟩

lemma *hypnat-add-self-not-less*: $\tilde{}\ (x + y < (x::hypnat))$
⟨*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*⟩

15.2 Properties of the set of embedded natural numbers

lemma *hypnat-of-nat-def*: *hypnat-of-nat m == of-nat m*
⟨*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*]:
$\forall\, d::hypnat.$ *of-nat* m = *of-nat* $n + d \ {-}{-}{>}\ d \in$ *range of-nat*
⟨*proof*⟩

lemma *Nats-diff* [*simp*]: $[|a \in Nats;\ b \in Nats|] \ {=}{=}{>}\ (a{-}b :: hypnat) \in Nats$
⟨*proof*⟩

15.3 Existence of an infinite hypernatural number

consts *whn* :: *hypnat*

defs

hypnat-omega-def: *whn == star-n* $(\%n::nat.\ n)$

Existence of infinite number not corresponding to any natural number follows because member $\mathcal{U}$ is not finite. See *HyperDef.thy* for similar argument.

lemma *lemma-unbounded-set* [*simp*]: $\{n::nat.\ m < n\} \in FreeUltrafilterNat$
⟨*proof*⟩

lemma *Compl-Collect-le*: $- \{n::nat.\ N \le n\} = \{n.\ n < N\}$
⟨*proof*⟩

lemma *hypnat-of-nat-eq*:
$hypnat\text{-}of\text{-}nat\ m\ = star\text{-}n\ (\%n::nat.\ m)$
⟨*proof*⟩

lemma *SHNat-eq*: $Nats = \{n.\ \exists N.\ n = hypnat\text{-}of\text{-}nat\ N\}$
⟨*proof*⟩

lemma *hypnat-omega-gt-SHNat*:
$n \in Nats ==> n < whn$
⟨*proof*⟩

lemma *SHNAT-omega-not-mem* [*simp*]: $whn \notin Nats$
⟨*proof*⟩

lemma *hypnat-of-nat-less-whn* [*simp*]: $hypnat\text{-}of\text{-}nat\ n < whn$
⟨*proof*⟩

lemma *hypnat-of-nat-le-whn* [*simp*]: $hypnat\text{-}of\text{-}nat\ n \le whn$
⟨*proof*⟩

lemma *hypnat-zero-less-hypnat-omega* [*simp*]: $0 < whn$
⟨*proof*⟩

lemma *hypnat-one-less-hypnat-omega* [*simp*]: $(1::hypnat) < whn$
⟨*proof*⟩

15.4 Infinite Hypernatural Numbers – *HNatInfinite*

constdefs

$HNatInfinite :: hypnat\ set$
$HNatInfinite == \{n.\ n \notin Nats\}$

lemma *HNatInfinite-whn* [*simp*]: $whn \in HNatInfinite$
⟨*proof*⟩

lemma *Nats-not-HNatInfinite-iff*: $(x \in Nats) = (x \notin HNatInfinite)$
⟨*proof*⟩

lemma *HNatInfinite-not-Nats-iff*: $(x \in HNatInfinite) = (x \notin Nats)$

⟨*proof*⟩

15.4.1 Alternative characterization of the set of infinite hypernaturals

HNatInfinite $= \{N.\ \forall\, n{\in}\mathbb{N}.\ n < N\}$

lemma *HNatInfinite-FreeUltrafilterNat-lemma*:
$\forall\, N{::}nat.\ \{n.\ f\ n \neq N\} \in$ *FreeUltrafilterNat*
$==> \{n.\ N < f\ n\} \in$ *FreeUltrafilterNat*
⟨*proof*⟩

lemma *HNatInfinite-iff*: *HNatInfinite* $= \{N.\ \forall\, n \in Nats.\ n < N\}$
⟨*proof*⟩

15.4.2 Alternative Characterization of *HNatInfinite* using Free Ultrafilter

lemma *HNatInfinite-FreeUltrafilterNat*:
$x \in$ *HNatInfinite*
$==> \exists\, X \in$ *Rep-star* $x.\ \forall\, u.\ \{n.\ u < X\ n\}$: *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-HNatInfinite*:
$\exists\, X \in$ *Rep-star* $x.\ \forall\, u.\ \{n.\ u < X\ n\}$: *FreeUltrafilterNat*
$==> x \in$ *HNatInfinite*
⟨*proof*⟩

lemma *HNatInfinite-FreeUltrafilterNat-iff*:
$(x \in$ *HNatInfinite*$) =$
$(\exists\, X \in$ *Rep-star* $x.\ \forall\, u.\ \{n.\ u < X\ n\}$: *FreeUltrafilterNat*$)$
⟨*proof*⟩

lemma *HNatInfinite-gt-one* [*simp*]: $x \in$ *HNatInfinite* $==> (1{::}hypnat) < x$
⟨*proof*⟩

lemma *zero-not-mem-HNatInfinite* [*simp*]: $0 \notin$ *HNatInfinite*
⟨*proof*⟩

lemma *HNatInfinite-not-eq-zero*: $x \in$ *HNatInfinite* $==> 0 < x$
⟨*proof*⟩

lemma *HNatInfinite-ge-one* [*simp*]: $x \in$ *HNatInfinite* $==> (1{::}hypnat) \leq x$
⟨*proof*⟩

15.4.3 Closure Rules

lemma *HNatInfinite-add*:
$[|\ x \in$ *HNatInfinite*; $y \in$ *HNatInfinite* $|] ==> x + y \in$ *HNatInfinite*
⟨*proof*⟩

lemma *HNatInfinite-SHNat-add*:
[| $x \in$ *HNatInfinite*; $y \in$ *Nats* |] ==> $x + y \in$ *HNatInfinite*
⟨*proof*⟩

lemma *HNatInfinite-Nats-imp-less*: [| $x \in$ *HNatInfinite*; $y \in$ *Nats* |] ==> $y < x$
⟨*proof*⟩

lemma *HNatInfinite-SHNat-diff*:
assumes x: $x \in$ *HNatInfinite* **and** y: $y \in$ *Nats*
shows $x - y \in$ *HNatInfinite*
⟨*proof*⟩

lemma *HNatInfinite-add-one*:
$x \in$ *HNatInfinite* ==> x + (*1*::*hypnat*) $\in$ *HNatInfinite*
⟨*proof*⟩

lemma *HNatInfinite-is-Suc*: $x \in$ *HNatInfinite* ==> $\exists\, y.\ x = y$ + (*1*::*hypnat*)
⟨*proof*⟩

15.5 Embedding of the Hypernaturals into the Hyperreals

Obtained using the nonstandard extension of the naturals

constdefs
hypreal-of-hypnat :: *hypnat* => *hypreal*
hypreal-of-hypnat == *f* *real*

declare *hypreal-of-hypnat-def* [*transfer-unfold*]

lemma *HNat-hypreal-of-nat* [*simp*]: *hypreal-of-nat* $N \in$ *Nats*
⟨*proof*⟩

lemma *hypreal-of-hypnat*:
hypreal-of-hypnat (*star-n* X) = *star-n* (%n. *real* (X n))
⟨*proof*⟩

lemma *hypreal-of-hypnat-inject* [*simp*]:
!!m n. (*hypreal-of-hypnat* m = *hypreal-of-hypnat* n) = (m=n)
⟨*proof*⟩

lemma *hypreal-of-hypnat-zero*: *hypreal-of-hypnat* *0* = *0*
⟨*proof*⟩

lemma *hypreal-of-hypnat-one*: *hypreal-of-hypnat* (*1*::*hypnat*) = *1*
⟨*proof*⟩

lemma *hypreal-of-hypnat-add* [*simp*]:
!!m n. *hypreal-of-hypnat* (m + n) = *hypreal-of-hypnat* m + *hypreal-of-hypnat* n

⟨*proof*⟩

lemma *hypreal-of-hypnat-mult* [*simp*]:
$!!m\ n.\ hypreal\text{-}of\text{-}hypnat\ (m * n) = hypreal\text{-}of\text{-}hypnat\ m * hypreal\text{-}of\text{-}hypnat\ n$
⟨*proof*⟩

lemma *hypreal-of-hypnat-less-iff* [*simp*]:
$!!m\ n.\ (hypreal\text{-}of\text{-}hypnat\ n < hypreal\text{-}of\text{-}hypnat\ m) = (n < m)$
⟨*proof*⟩

lemma *hypreal-of-hypnat-eq-zero-iff*: $(hypreal\text{-}of\text{-}hypnat\ N = 0) = (N = 0)$
⟨*proof*⟩
declare *hypreal-of-hypnat-eq-zero-iff* [*simp*]

lemma *hypreal-of-hypnat-ge-zero* [*simp*]: $!!n.\ 0 \leq hypreal\text{-}of\text{-}hypnat\ n$
⟨*proof*⟩

lemma *HNatInfinite-inverse-Infinitesimal* [*simp*]:
$n \in HNatInfinite ==> inverse\ (hypreal\text{-}of\text{-}hypnat\ n) \in Infinitesimal$
⟨*proof*⟩

lemma *HNatInfinite-hypreal-of-hypnat-gt-zero*:
$N \in HNatInfinite ==> 0 < hypreal\text{-}of\text{-}hypnat\ N$
⟨*proof*⟩

⟨*ML*⟩

end

16 HyperPow: Exponentials on the Hyperreals

theory *HyperPow*
imports *HyperArith HyperNat*
begin

lemma *hpowr-0* [*simp*]: $r\ \hat{}\ 0 = (1::hypreal)$
⟨*proof*⟩

lemma *hpowr-Suc* [*simp*]: $r\ \hat{}\ (Suc\ n) = (r::hypreal) * (r\ \hat{}\ n)$
⟨*proof*⟩

consts
pow :: [*hypreal*,*hypnat*] => *hypreal* (**infixr** *pow 80*)

defs

hyperpow-def [*transfer-unfold*]:
(R::*hypreal*) *pow* (N::*hypnat*) == (**f2* op* ^) R N

lemma *hrealpow-two*: (r::*hypreal*) ^ *Suc* (*Suc* 0) = $r * r$
⟨*proof*⟩

lemma *hrealpow-two-le* [*simp*]: (0::*hypreal*) ≤ r ^ *Suc* (*Suc* 0)
⟨*proof*⟩

lemma *hrealpow-two-le-add-order* [*simp*]:
(0::*hypreal*) ≤ u ^ *Suc* (*Suc* 0) + v ^ *Suc* (*Suc* 0)
⟨*proof*⟩

lemma *hrealpow-two-le-add-order2* [*simp*]:
(0::*hypreal*) ≤ u ^ *Suc* (*Suc* 0) + v ^ *Suc* (*Suc* 0) + w ^ *Suc* (*Suc* 0)
⟨*proof*⟩

lemma *hypreal-add-nonneg-eq-0-iff*:
[| 0 ≤ x; 0 ≤ y |] ==> (x+y = 0) = (x = 0 & y = (0::*hypreal*))
⟨*proof*⟩

FIXME: DELETE THESE

lemma *hypreal-three-squares-add-zero-iff*:
($x*x$ + $y*y$ + $z*z$ = 0) = (x = 0 & y = 0 & z = (0::*hypreal*))
⟨*proof*⟩

lemma *hrealpow-three-squares-add-zero-iff* [*simp*]:
(x ^ *Suc* (*Suc* 0) + y ^ *Suc* (*Suc* 0) + z ^ *Suc* (*Suc* 0) = (0::*hypreal*)) =
(x = 0 & y = 0 & z = 0)
⟨*proof*⟩

lemma *hrabs-hrealpow-two* [*simp*]:
abs(x ^ *Suc* (*Suc* 0)) = (x::*hypreal*) ^ *Suc* (*Suc* 0)
⟨*proof*⟩

lemma *two-hrealpow-ge-one* [*simp*]: (1::*hypreal*) ≤ 2 ^ n
⟨*proof*⟩

lemma *two-hrealpow-gt* [*simp*]: *hypreal-of-nat* n < 2 ^ n
⟨*proof*⟩

lemma *hrealpow*:
star-n X ^ m = *star-n* (%n. (X n::*real*) ^ m)
⟨*proof*⟩

lemma *hrealpow-sum-square-expand*:

$(x + (y::hypreal))$ ^ $Suc\ (Suc\ 0) =$
x ^ $Suc\ (Suc\ 0) + y$ ^ $Suc\ (Suc\ 0) + (hypreal\text{-}of\text{-}nat\ (Suc\ (Suc\ 0))){*}x{*}y$
⟨*proof*⟩

16.1 Literal Arithmetic Involving Powers and Type *hypreal*

lemma *power-hypreal-of-real-number-of*:
$(number\text{-}of\ v :: hypreal)$ ^ $n = hypreal\text{-}of\text{-}real\ ((number\text{-}of\ v)$ ^ $n)$
⟨*proof*⟩

declare *power-hypreal-of-real-number-of* [*of - number-of w*, *standard*, *simp*]

lemma *hrealpow-HFinite*: $x \in HFinite ==> x$ ^ $n \in HFinite$
⟨*proof*⟩

16.2 Powers with Hypernatural Exponents

lemma *hyperpow*: $star\text{-}n\ X\ pow\ star\text{-}n\ Y = star\text{-}n\ (\%n.\ X\ n$ ^ $Y\ n)$
⟨*proof*⟩

lemma *hyperpow-zero* [*simp*]: $!!n.\ (0::hypreal)\ pow\ (n + (1::hypnat)) = 0$
⟨*proof*⟩

lemma *hyperpow-not-zero*: $!!r\ n.\ r \neq (0::hypreal) ==> r\ pow\ n \neq 0$
⟨*proof*⟩

lemma *hyperpow-inverse*:
$!!r\ n.\ r \neq (0::hypreal) ==> inverse(r\ pow\ n) = (inverse\ r)\ pow\ n$
⟨*proof*⟩

lemma *hyperpow-hrabs*: $!!r\ n.\ abs\ r\ pow\ n = abs\ (r\ pow\ n)$
⟨*proof*⟩

lemma *hyperpow-add*: $!!r\ n\ m.\ r\ pow\ (n + m) = (r\ pow\ n) * (r\ pow\ m)$
⟨*proof*⟩

lemma *hyperpow-one* [*simp*]: $!!r.\ r\ pow\ (1::hypnat) = r$
⟨*proof*⟩

lemma *hyperpow-two*:
$!!r.\ r\ pow\ ((1::hypnat) + (1::hypnat)) = r * r$
⟨*proof*⟩

lemma *hyperpow-gt-zero*: $!!r\ n.\ (0::hypreal) < r ==> 0 < r\ pow\ n$
⟨*proof*⟩

lemma *hyperpow-ge-zero*: $!!r\ n.\ (0::hypreal) \leq r ==> 0 \leq r\ pow\ n$
⟨*proof*⟩

lemma *hyperpow-le*:

$!!x\ y\ n.\ [|(0::hypreal) < x;\ x \leq y|] ==> x\ pow\ n \leq y\ pow\ n$
⟨proof⟩

lemma *hyperpow-eq-one* [*simp*]: $!!n.\ 1\ pow\ n = (1::hypreal)$
⟨proof⟩

lemma *hrabs-hyperpow-minus-one* [*simp*]: $!!n.\ abs(-1\ pow\ n) = (1::hypreal)$
⟨proof⟩

lemma *hyperpow-mult*: $!!r\ s\ n.\ (r * s)\ pow\ n = (r\ pow\ n) * (s\ pow\ n)$
⟨proof⟩

lemma *hyperpow-two-le* [*simp*]: $0 \leq r\ pow\ (1 + 1)$
⟨proof⟩

lemma *hrabs-hyperpow-two* [*simp*]: $abs(x\ pow\ (1 + 1)) = x\ pow\ (1 + 1)$
⟨proof⟩

lemma *hyperpow-two-hrabs* [*simp*]: $abs(x)\ pow\ (1 + 1) = x\ pow\ (1 + 1)$
⟨proof⟩

The precondition could be weakened to $(0::'a) \leq x$

lemma *hypreal-mult-less-mono*:
$[|\ u<v;\ x<y;\ (0::hypreal) < v;\ 0 < x\ |] ==> u*x < v*\ y$
⟨proof⟩

lemma *hyperpow-two-gt-one*: $1 < r ==> 1 < r\ pow\ (1 + 1)$
⟨proof⟩

lemma *hyperpow-two-ge-one*:
$1 \leq r ==> 1 \leq r\ pow\ (1 + 1)$
⟨proof⟩

lemma *two-hyperpow-ge-one* [*simp*]: $(1::hypreal) \leq 2\ pow\ n$
⟨proof⟩

lemma *hyperpow-minus-one2* [*simp*]:
$!!n.\ -1\ pow\ ((1 + 1)*n) = (1::hypreal)$
⟨proof⟩

lemma *hyperpow-less-le*:
$!!r\ n\ N.\ [|(0::hypreal) \leq r;\ r \leq 1;\ n < N|] ==> r\ pow\ N \leq r\ pow\ n$
⟨proof⟩

lemma *hyperpow-SHNat-le*:
$[|\ 0 \leq r;\ r \leq (1::hypreal);\ N \in HNatInfinite\ |]$
$==> ALL\ n: Nats.\ r\ pow\ N \leq r\ pow\ n$
⟨proof⟩

lemma *hyperpow-realpow*:
$(hypreal\text{-}of\text{-}real\ r)\ pow\ (hypnat\text{-}of\text{-}nat\ n) = hypreal\text{-}of\text{-}real\ (r\ \hat{}\ n)$
⟨*proof*⟩

lemma *hyperpow-SReal* [*simp*]:
$(hypreal\text{-}of\text{-}real\ r)\ pow\ (hypnat\text{-}of\text{-}nat\ n) \in Reals$
⟨*proof*⟩

lemma *hyperpow-zero-HNatInfinite* [*simp*]:
$N \in HNatInfinite ==> (0::hypreal)\ pow\ N = 0$
⟨*proof*⟩

lemma *hyperpow-le-le*:
$[|\ (0::hypreal) \leq r;\ r \leq 1;\ n \leq N\ |] ==> r\ pow\ N \leq r\ pow\ n$
⟨*proof*⟩

lemma *hyperpow-Suc-le-self2*:
$[|\ (0::hypreal) \leq r;\ r < 1\ |] ==> r\ pow\ (n + (1::hypnat)) \leq r$
⟨*proof*⟩

lemma *lemma-Infinitesimal-hyperpow*:
$[|\ x \in Infinitesimal;\ 0 < N\ |] ==> abs\ (x\ pow\ N) \leq abs\ x$
⟨*proof*⟩

lemma *Infinitesimal-hyperpow*:
$[|\ x \in Infinitesimal;\ 0 < N\ |] ==> x\ pow\ N \in Infinitesimal$
⟨*proof*⟩

lemma *hrealpow-hyperpow-Infinitesimal-iff*:
$(x\ \hat{}\ n \in Infinitesimal) = (x\ pow\ (hypnat\text{-}of\text{-}nat\ n) \in Infinitesimal)$
⟨*proof*⟩

lemma *Infinitesimal-hrealpow*:
$[|\ x \in Infinitesimal;\ 0 < n\ |] ==> x\ \hat{}\ n \in Infinitesimal$
⟨*proof*⟩

⟨*ML*⟩

end

17 NatStar: Star-transforms for the Hypernaturals

theory *NatStar*
imports *HyperPow*
begin

lemma *star-n-eq-starfun-whn*: *star-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*⟩

17.1 Nonstandard Extensions of Functions

Example of transfer of a property from reals to hyperreals — used for limit comparison of sequences

lemma *starfun-le-mono*:
$\forall n.\ N \le n --> f\ n \le g\ n$
$==> \forall n.\ hypnat\text{-}of\text{-}nat\ N \le n --> (\ *f*\ f)\ n \le (\ *f*\ g)\ n$
⟨*proof*⟩

lemma *starfun-less-mono*:
$\forall n.\ N \le n --> f\ n < g\ n$
$==> \forall n.\ hypnat\text{-}of\text{-}nat\ N \le n --> (\ *f*\ f)\ n < (\ *f*\ g)\ n$
⟨*proof*⟩

Nonstandard extension when we increment the argument by one

lemma *starfun-shift-one*:
$!!N.\ (\ *f*\ (\%n.\ f\ (Suc\ n)))\ N = (\ *f*\ f)\ (N + (1::hypnat))$
⟨*proof*⟩

Nonstandard extension with absolute value

lemma *starfun-abs*: $!!N.\ (\ *f*\ (\%n.\ abs\ (f\ n)))\ N = abs((\ *f*\ f)\ N)$
⟨*proof*⟩

The hyperpow function as a nonstandard extension of realpow

lemma *starfun-pow*: $!!N.\ (\ *f*\ (\%n.\ r\ \hat{}\ n))\ N = (hypreal\text{-}of\text{-}real\ r)\ pow\ N$
⟨*proof*⟩

lemma *starfun-pow2*:
$!!N.\ (\ *f*\ (\%n.\ (X\ n)\ \hat{}\ m))\ N = (\ *f*\ X)\ N\ pow\ hypnat\text{-}of\text{-}nat\ m$
⟨*proof*⟩

lemma *starfun-pow3*: $!!R.\ (\ *f*\ (\%r.\ r\ \hat{}\ n))\ R = (R)\ pow\ hypnat\text{-}of\text{-}nat\ n$
⟨*proof*⟩

The *hypreal-of-hypnat* function as a nonstandard extension of *real-of-nat*

lemma *starfunNat-real-of-nat*: $(\ *f*\ real) = hypreal\text{-}of\text{-}hypnat$
⟨*proof*⟩

lemma *starfun-inverse-real-of-nat-eq*:
$N \in HNatInfinite$
$==> (\ *f*\ (\%x::nat.\ inverse(real\ x)))\ N = inverse(hypreal\text{-}of\text{-}hypnat\ N)$
⟨*proof*⟩

Internal functions - some redundancy with *f* now

lemma *starfun-n*: $(*fn* \ f) \ (star\text{-}n \ X) = star\text{-}n \ (\%n. \ f \ n \ (X \ n))$
⟨*proof*⟩

Multiplication: $(*fn) \ x \ (*gn) = *(fn \ x \ gn)$

lemma *starfun-n-mult*:
$(*fn* \ f) \ z * (*fn* \ g) \ z = (*fn* \ (\% \ i \ x. \ f \ i \ x * g \ i \ x)) \ z$
⟨*proof*⟩

Addition: $(*fn) + (*gn) = *(fn + gn)$

lemma *starfun-n-add*:
$(*fn* \ f) \ z + (*fn* \ g) \ z = (*fn* \ (\%i \ x. \ f \ i \ x + g \ i \ x)) \ z$
⟨*proof*⟩

Subtraction: $(*fn) - (*gn) = *(fn + - \ gn)$

lemma *starfun-n-add-minus*:
$(*fn* \ f) \ z + -(*fn* \ g) \ z = (*fn* \ (\%i \ x. \ f \ i \ x + -g \ i \ x)) \ z$
⟨*proof*⟩

Composition: $(*fn) \ o \ (*gn) = *(fn \ o \ gn)$

lemma *starfun-n-const-fun* [*simp*]:
$(*fn* \ (\%i \ x. \ k)) \ z = star\text{-}of \ k$
⟨*proof*⟩

lemma *starfun-n-minus*: $- \ (*fn* \ f) \ x = (*fn* \ (\%i \ x. - \ (f \ i) \ x)) \ x$
⟨*proof*⟩

lemma *starfun-n-eq* [*simp*]:
$(*fn* \ f) \ (star\text{-}of \ n) = star\text{-}n \ (\%i. \ f \ i \ n)$
⟨*proof*⟩

lemma *starfun-eq-iff*: $((*f* \ f) = (*f* \ g)) = (f = g)$
⟨*proof*⟩

lemma *starfunNat-inverse-real-of-nat-Infinitesimal* [*simp*]:
$N \in HNatInfinite ==> (*f* \ (\%x. \ inverse \ (real \ x))) \ N \in Infinitesimal$
⟨*proof*⟩

⟨*ML*⟩

17.2 Nonstandard Characterization of Induction

constdefs
hSuc :: *hypnat* => *hypnat*
hSuc $n == n + 1$

lemma *starP*: $((*p* \ P) \ (star\text{-}n \ X)) = (\{n. \ P \ (X \ n)\} \in FreeUltrafilterNat)$
⟨*proof*⟩

lemma *hypnat-induct-obj*:

$!!n. ((*p* P) (0::hypnat) \&$
$(\forall n. (*p* P)(n) --> (*p* P)(n + 1)))$
$--> (*p* P)(n)$
$\langle proof \rangle$

lemma *hypnat-induct*:
$!!n. [| (*p* P) (0::hypnat);$
$!!n. (*p* P)(n) ==> (*p* P)(n + 1)|]$
$==> (*p* P)(n)$
$\langle proof \rangle$

lemma *starP2*:
$((*p2* P) (star\text{-}n\ X) (star\text{-}n\ Y)) =$
$(\{n. P\ (X\ n)\ (Y\ n)\} \in FreeUltrafilterNat)$
$\langle proof \rangle$

lemma *starP2-eq-iff*: $(*p2* (op =)) = (op =)$
$\langle proof \rangle$

lemma *starP2-eq-iff2*: $(*p2* (\%x\ y.\ x = y))\ X\ Y = (X = Y)$
$\langle proof \rangle$

lemma *hSuc-not-zero* [*iff*]: $hSuc\ m \neq 0$
$\langle proof \rangle$

lemmas *zero-not-hSuc* = *hSuc-not-zero* [*THEN not-sym*, *standard*, *iff*]

lemma *hSuc-hSuc-eq* [*iff*]: $(hSuc\ m = hSuc\ n) = (m = n)$
$\langle proof \rangle$

lemma *nonempty-nat-set-Least-mem*: $c \in (S :: nat\ set) ==> (LEAST\ n.\ n \in S) \in S$
$\langle proof \rangle$

lemma *nonempty-set-star-has-least*:
$!!S::nat\ set\ star.\ Iset\ S \neq \{\} ==> \exists n \in Iset\ S.\ \forall m \in Iset\ S.\ n \leq m$
$\langle proof \rangle$

lemma *nonempty-InternalNatSet-has-least*:
$[| (S::hypnat\ set) \in InternalSets;\ S \neq \{\} |] ==> \exists n \in S.\ \forall m \in S.\ n \leq m$
$\langle proof \rangle$

Goldblatt page 129 Thm 11.3.2

lemma *internal-induct-lemma*:
$!!X::nat\ set\ star.\ [| (0::hypnat) \in Iset\ X;\ \forall n.\ n \in Iset\ X --> n + 1 \in Iset\ X |]$
$==> Iset\ X = (UNIV:: hypnat\ set)$
$\langle proof \rangle$

lemma *internal-induct*:
  [| $X \in$ *InternalSets*; (*0*::*hypnat*) $\in X$; $\forall\, n.\ n \in X \mathrel{-{-}>} n + 1 \in X$ |]
  $==>$ $X = $ (*UNIV*:: *hypnat set*)
⟨*proof*⟩

end

18 SEQ: Sequences and Series

theory *SEQ*
imports *NatStar*
begin

constdefs

LIMSEQ :: [*nat=>real,real*] => *bool*   (((-)/ ––––> (-)) [*60*, *60*] *60*)
 — Standard definition of convergence of sequence
X ––––> L == ($\forall\, r.\ 0 < r$ ––> ($\exists\, no.\ \forall\, n.\ no \le n$ ––> $|X\ n + -L| < r$))

NSLIMSEQ :: [*nat=>real,real*] => *bool*   (((-)/ ––––*NS*> (-)) [*60*, *60*] *60*)
 — Nonstandard definition of convergence of sequence
X ––––*NS*> L == ($\forall\, N \in$ *HNatInfinite*. ($*f*$ X) $N \approx$ *hypreal-of-real* L)

lim :: (*nat* => *real*) => *real*
 — Standard definition of limit using choice operator
lim X == (@L. (X ––––> L))

nslim :: (*nat* => *real*) => *real*
 — Nonstandard definition of limit using choice operator
nslim X == (@L. (X ––––*NS*> L))

convergent :: (*nat* => *real*) => *bool*
 — Standard definition of convergence
convergent X == ($\exists\, L$. (X ––––> L))

NSconvergent :: (*nat* => *real*) => *bool*
 — Nonstandard definition of convergence
NSconvergent X == ($\exists\, L$. (X ––––*NS*> L))

Bseq :: (*nat* => *real*) => *bool*
 — Standard definition for bounded sequence
Bseq X == $\exists\, K{>}0.\forall\, n.\ |X\ n| \le K$

NSBseq :: (*nat=>real*) => *bool*
 — Nonstandard definition for bounded sequence
NSBseq X == ($\forall\, N \in$ *HNatInfinite*. ($*f*$ X) N : *HFinite*)

monoseq :: (*nat*=>*real*)=>*bool*
— Definition for monotonicity
monoseq X == ($\forall$ *m*. $\forall$ *n*$\geq$*m*. *X m* $\leq$ *X n*) | ($\forall$ *m*. $\forall$ *n*$\geq$*m*. *X n* $\leq$ *X m*)

subseq :: (*nat* => *nat*) => *bool*
— Definition of subsequence
subseq f == $\forall$ *m*. $\forall$ *n*>*m*. (*f m*) < (*f n*)

Cauchy :: (*nat* => *real*) => *bool*
— Standard definition of the Cauchy condition
Cauchy X == $\forall$ *e*>*0*. $\exists$ *M*. $\forall$ *m* $\geq$ *M*. $\forall$ *n* $\geq$ *M*. *abs*((*X m*) + −(*X n*)) < *e*

NSCauchy :: (*nat* => *real*) => *bool*
— Nonstandard definition
NSCauchy X == ($\forall$ *M* $\in$ *HNatInfinite*. $\forall$ *N* $\in$ *HNatInfinite*.
(**f* X*) *M* $\approx$ (**f* X*) *N*)

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*⟩

18.1 LIMSEQ and NSLIMSEQ

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

lemma *NSLIMSEQ-iff*:
(*X* −−−−*NS*> *L*) = ($\forall$ *N* $\in$ *HNatInfinite*. (**f* X*) *N* $\approx$ *hypreal-of-real L*)
⟨*proof*⟩

LIMSEQ ==¿ NSLIMSEQ

lemma *LIMSEQ-NSLIMSEQ*:
X −−−−> *L* ==> *X* −−−−*NS*> *L*
⟨*proof*⟩

NSLIMSEQ ==¿ LIMSEQ

lemma *lemma-NSLIMSEQ1*: !!(*f*::*nat*=>*nat*). $\forall$ *n*. *n* $\leq$ *f n*
==> {*n*. *f n* = *0*} = {*0*} | {*n*. *f n* = *0*} = {}
⟨*proof*⟩

lemma *lemma-NSLIMSEQ2*: {*n*. *f n* $\leq$ *Suc u*} = {*n*. *f n* $\leq$ *u*} *Un* {*n*. *f n* = *Suc u*}
⟨*proof*⟩

lemma *lemma-NSLIMSEQ3*:

  !!(*f*::*nat*=>*nat*). ∀ *n*. *n* ≤ *f n* ==> {*n*. *f n* = *Suc u*} ≤ {*n*. *n* ≤ *Suc u*}
⟨*proof*⟩

the following sequence *f n* defines a hypernatural

lemma *NSLIMSEQ-finite-set*:
  !!(*f*::*nat*=>*nat*). ∀ *n*. *n* ≤ *f n* ==> *finite* {*n*. *f n* ≤ *u*}
⟨*proof*⟩

lemma *Compl-less-set*: − {*n*. *u* < (*f*::*nat*=>*nat*) *n*} = {*n*. *f n* ≤ *u*}
⟨*proof*⟩

the index set is in the free ultrafilter

lemma *FreeUltrafilterNat-NSLIMSEQ*:
  !!(*f*::*nat*=>*nat*). ∀ *n*. *n* ≤ *f n* ==> {*n*. *u* < *f n*} : *FreeUltrafilterNat*
⟨*proof*⟩

thus, the sequence defines an infinite hypernatural!

lemma *HNatInfinite-NSLIMSEQ*: ∀ *n*. *n* ≤ *f n*
    ==> *star-n f* : *HNatInfinite*
⟨*proof*⟩

lemma *lemmaLIM*:
  {*n*. *X* (*f n*) + − *L* = *Y n*} *Int* {*n*. |*Y n*| < *r*} ≤
   {*n*. |*X* (*f n*) + − *L*| < *r*}
⟨*proof*⟩

lemma *lemmaLIM2*:
 {*n*. |*X* (*f n*) + − *L*| < *r*} *Int* {*n*. *r* ≤ *abs* (*X* (*f n*) + − (*L*::*real*))} = {}
⟨*proof*⟩

lemma *lemmaLIM3*: [| *0* < *r*; ∀ *n*. *r* ≤ |*X* (*f n*) + − *L*|;
    (**f** *X*) (*star-n f*) +
    − *hypreal-of-real* *L* ≈ *0* |] ==> *False*
⟨*proof*⟩

lemma *NSLIMSEQ-LIMSEQ*: *X* −−−−*NS*> *L* ==> *X* −−−−> *L*
⟨*proof*⟩

Now, the all-important result is trivially proved!

theorem *LIMSEQ-NSLIMSEQ-iff*: (*f* −−−−> *L*) = (*f* −−−−*NS*> *L*)
⟨*proof*⟩

18.2 Theorems About Sequences

lemma *NSLIMSEQ-const*: (%*n*. *k*) −−−−*NS*> *k*
⟨*proof*⟩

lemma *LIMSEQ-const*: (%*n*. *k*) −−−−> *k*
⟨*proof*⟩

lemma *NSLIMSEQ-add*:
$[| X ----NS> a; Y ----NS> b |] ==> (\%n. X n + Y n) ----NS> a + b$
⟨*proof*⟩

lemma *LIMSEQ-add*: $[| X ----> a; Y ----> b |] ==> (\%n. X n + Y n) ----> a + b$
⟨*proof*⟩

lemma *LIMSEQ-add-const*: $f ----> a ==> (\%n.(f n + b)) ----> a + b$
⟨*proof*⟩

lemma *NSLIMSEQ-add-const*: $f ----NS> a ==> (\%n.(f n + b)) ----NS> a + b$
⟨*proof*⟩

lemma *NSLIMSEQ-mult*:
$[| X ----NS> a; Y ----NS> b |] ==> (\%n. X n * Y n) ----NS> a * b$
⟨*proof*⟩

lemma *LIMSEQ-mult*: $[| X ----> a; Y ----> b |] ==> (\%n. X n * Y n) ----> a * b$
⟨*proof*⟩

lemma *NSLIMSEQ-minus*: $X ----NS> a ==> (\%n. -(X n)) ----NS> -a$
⟨*proof*⟩

lemma *LIMSEQ-minus*: $X ----> a ==> (\%n. -(X n)) ----> -a$
⟨*proof*⟩

lemma *LIMSEQ-minus-cancel*: $(\%n. -(X n)) ----> -a ==> X ----> a$
⟨*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 *LIMSEQ-add-minus*:
$[| X ----> a; Y ----> b |] ==> (\%n. X n + -Y n) ----> a + -b$
⟨*proof*⟩

lemma *LIMSEQ-diff*: $[| X ----> a; Y ----> b |] ==> (\%n. X n - Y n)$

−−−−> *a* − *b*
⟨*proof*⟩

lemma *NSLIMSEQ-diff*:
[| *X* −−−−*NS*> *a*; *Y* −−−−*NS*> *b* |] ==> (%*n*. *X n* − *Y n*) −−−−*NS*> *a* − *b*
⟨*proof*⟩

lemma *LIMSEQ-diff-const*: *f* −−−−> *a* ==> (%*n*.(*f n* − *b*)) −−−−> *a* − *b*
⟨*proof*⟩

lemma *NSLIMSEQ-diff-const*: *f* −−−−*NS*> *a* ==> (%*n*.(*f n* − *b*)) −−−−*NS*> *a* − *b*
⟨*proof*⟩

Proof is like that of *NSLIM-inverse*.

lemma *NSLIMSEQ-inverse*:
[| *X* −−−−*NS*> *a*; *a* ~= *0* |] ==> (%*n*. *inverse*(*X n*)) −−−−*NS*> *inverse*(*a*)
⟨*proof*⟩

Standard version of theorem

lemma *LIMSEQ-inverse*:
[| *X* −−−−> *a*; *a* ~= *0* |] ==> (%*n*. *inverse*(*X n*)) −−−−> *inverse*(*a*)
⟨*proof*⟩

lemma *NSLIMSEQ-mult-inverse*:
[| *X* −−−−*NS*> *a*; *Y* −−−−*NS*> *b*; *b* ~= *0* |] ==> (%*n*. *X n* / *Y n*) −−−−*NS*> *a*/*b*
⟨*proof*⟩

lemma *LIMSEQ-divide*:
[| *X* −−−−> *a*; *Y* −−−−> *b*; *b* ~= *0* |] ==> (%*n*. *X n* / *Y n*) −−−−> *a*/*b*
⟨*proof*⟩

Uniqueness of limit

lemma *NSLIMSEQ-unique*: [| *X* −−−−*NS*> *a*; *X* −−−−*NS*> *b* |] ==> *a* = *b*
⟨*proof*⟩

lemma *LIMSEQ-unique*: [| *X* −−−−> *a*; *X* −−−−> *b* |] ==> *a* = *b*
⟨*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*:
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-ignore-initial-segment*: *f ----> a*
==> (%n. f(n + k)) ----> a
⟨*proof*⟩

lemma *LIMSEQ-offset*: *(%x. f (x + k)) ----> a ==>*
f ----> a
⟨*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*⟩

18.3 Nslim and Lim

lemma *limI*: *X ----> L ==> lim X = L*
⟨*proof*⟩

lemma *nslimI*: *X ----NS> L ==> nslim X = L*
⟨*proof*⟩

lemma *lim-nslim-iff*: *lim X = nslim X*
⟨*proof*⟩

18.4 Convergence

lemma *convergentD*: *convergent X ==> ∃L. (X ----> L)*
⟨*proof*⟩

lemma *convergentI*: *(X ----> L) ==> convergent X*
⟨*proof*⟩

lemma *NSconvergentD*: *NSconvergent X ==> ∃L. (X ----NS> L)*
⟨*proof*⟩

lemma *NSconvergentI*: *(X ----NS> L) ==> NSconvergent X*
⟨*proof*⟩

lemma *convergent-NSconvergent-iff*: *convergent X = NSconvergent X*
⟨*proof*⟩

lemma *NSconvergent-NSLIMSEQ-iff*: $NSconvergent\ X = (X\ {-}{-}{-}{-}NS{>}\ nslim\ X)$
⟨*proof*⟩

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

Subsequence (alternative definition, (e.g. Hoskins)

lemma *subseq-Suc-iff*: $subseq\ f = (\forall n.\ (f\ n) < (f\ (Suc\ n)))$
⟨*proof*⟩

18.5 Monotonicity

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

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

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

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

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

18.6 Bounded Sequence

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

lemma *BseqI*: $[|\ 0 < K; \forall n.\ |X\ n| \leq K\ |] ==> Bseq\ X$
⟨*proof*⟩

lemma *lemma-NBseq-def*:
$$(\exists K > 0.\ \forall n.\ |X\ n| \leq K) = (\exists N.\ \forall n.\ |X\ n| \leq real(Suc\ N))$$
⟨*proof*⟩

alternative definition for Bseq

lemma *Bseq-iff*: $Bseq\ X = (\exists N.\ \forall n.\ |X\ n| \leq real(Suc\ N))$
⟨*proof*⟩

lemma *lemma-NBseq-def2*:
$$(\exists K > 0.\ \forall n.\ |X\ n| \leq K) = (\exists N.\ \forall n.\ |X\ n| < real(Suc\ N))$$

⟨*proof*⟩

lemma *Bseq-iff1a*: $Bseq\ X = (\exists N.\ \forall n.\ |X\ n| < real(Suc\ N))$
⟨*proof*⟩

lemma *NSBseqD*: $[|\ NSBseq\ X;\ \ N: HNatInfinite\ |] ==> (\ *f*\ X)\ N : HFinite$
⟨*proof*⟩

lemma *NSBseqI*: $\forall N \in HNatInfinite.\ (\ *f*\ X)\ N : HFinite ==> NSBseq\ X$
⟨*proof*⟩

The standard definition implies the nonstandard definition

lemma *lemma-Bseq*: $\forall n.\ |X\ n| \leq K ==> \forall n.\ abs(X((f{::}nat{=}{>}nat)\ n)) \leq K$
⟨*proof*⟩

lemma *Bseq-NSBseq*: $Bseq\ X ==> NSBseq\ X$
⟨*proof*⟩

The nonstandard definition implies the standard definition

We need to get rid of the real variable and do so by proving the following, which relies on the Archimedean property of the reals. When we skolemize we then get the required function *f*. Otherwise, we would be stuck with a skolem function f which woulid be useless.

lemma *lemmaNSBseq*:
$\forall K > 0.\ \exists n.\ K < |X\ n|$
$==> \forall N.\ \exists n.\ real(Suc\ N) < |X\ n|$
⟨*proof*⟩

lemma *lemmaNSBseq2*: $\forall K > 0.\ \exists n.\ K < |X\ n|$
$==> \exists f.\ \forall N.\ real(Suc\ N) < |X\ (f\ N)|$
⟨*proof*⟩

lemma *real-seq-to-hypreal-HInfinite*:
$\forall N.\ real(Suc\ N) < |X\ (f\ N)|$
$==>\ \ star\text{-}n\ (X\ o\ f) : HInfinite$
⟨*proof*⟩

Now prove that we can get out an infinite hypernatural as well defined using the skolem function f above

lemma *lemma-finite-NSBseq*:
$\{n.\ f\ n \leq Suc\ u\ \&\ real(Suc\ n) < |X\ (f\ n)|\} \leq$
$\{n.\ f\ n \leq u\ \&\ real(Suc\ n) < |X\ (f\ n)|\}\ Un$
$\{n.\ real(Suc\ n) < |X\ (Suc\ u)|\}$
⟨*proof*⟩

lemma *lemma-finite-NSBseq2*:

finite $\{n.\ f\ n \le (u{::}nat)\ \&\ \ real(Suc\ n) < |X(f\ n)|\}$
⟨*proof*⟩

lemma *HNatInfinite-skolem-f*:
$\forall N.\ real(Suc\ N) < |X\ (f\ N)|$
$==>$ *star-n f* : *HNatInfinite*
⟨*proof*⟩

lemma *NSBseq-Bseq*: *NSBseq X* $==>$ *Bseq X*
⟨*proof*⟩

Equivalence of nonstandard and standard definitions for a bounded sequence

lemma *Bseq-NSBseq-iff*: (*Bseq X*) = (*NSBseq X*)
⟨*proof*⟩

A convergent sequence is bounded: Boundedness as a necessary condition for convergence. The nonstandard version has no existential, as usual

lemma *NSconvergent-NSBseq*: *NSconvergent X* $==>$ *NSBseq X*
⟨*proof*⟩

Standard Version: easily now proved using equivalence of NS and standard definitions

lemma *convergent-Bseq*: *convergent X* $==>$ *Bseq X*
⟨*proof*⟩

18.7 Upper Bounds and Lubs of Bounded Sequences

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

Use completeness of reals (supremum property) to show that any bounded sequence has a least upper bound

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

lemma *NSBseq-isUb*: *NSBseq X* $==> \exists\ U.$ *isUb UNIV* $\{x.\ \exists\ n.\ X\ n = x\}\ U$
⟨*proof*⟩

lemma *NSBseq-isLub*: *NSBseq X* $==> \exists\ U.$ *isLub UNIV* $\{x.\ \exists\ n.\ X\ n = x\}\ U$
⟨*proof*⟩

18.8 A Bounded and Monotonic Sequence Converges

lemma *lemma-converg1*:
!!(X::*nat*=>*real*). [| $\forall m.\ \forall\ \ n \ge m.\ X\ m \le X\ n$;

$isLub\ (UNIV{::}real\ set)\ \{x.\ \exists\, n.\ X\ n = x\}\ (X\ ma)$
$|]\ ==> \forall\, n \geq ma.\ X\ n = X\ ma$
⟨*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*⟩

Now, the same theorem in terms of NS limit

lemma *Bmonoseq-NSLIMSEQ*: $\forall\, n \geq m.\ X\ n = X\ m ==> \exists\, L.\ (X\ ----NS> L)$
⟨*proof*⟩

lemma *lemma-converg2*:
$!!(X{::}nat{=}{>}real).$
$[|\ \forall\, m.\ X\ m\ \tilde{}{=}\ U;\ \ isLub\ UNIV\ \{x.\ \exists\, n.\ X\ n = x\}\ U\ |] ==> \forall\, m.\ X\ m < U$
⟨*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*: $!!(X{::}nat{=}{>}\ real).$
$[|\ \forall\, m.\ X\ m\ \tilde{}{=}\ U;$
$isLub\ UNIV\ \{x.\ \exists\, n.\ X\ n = x\}\ U;$
$0 < T;$
$U + -\ T < U$
$|] ==> \exists\, m.\ U + -T < X\ m\ \&\ X\ m < U$
⟨*proof*⟩

A standard proof of the theorem for monotone increasing sequence

lemma *Bseq-mono-convergent*:
$[|\ Bseq\ X;\ \forall\, m.\ \forall\, n \geq m.\ X\ m \leq X\ n\ |] ==> convergent\ X$
⟨*proof*⟩

Nonstandard version of the theorem

lemma *NSBseq-mono-NSconvergent*:
$[|\ NSBseq\ X;\ \forall\, m.\ \forall\, n \geq m.\ X\ m \leq X\ n\ |] ==> NSconvergent\ X$
⟨*proof*⟩

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

lemma *Bseq-minus-iff*: $Bseq\ (\%n.\ -(X\ n)) = Bseq\ X$

⟨*proof*⟩

Main monotonicity theorem

lemma *Bseq-monoseq-convergent*: $[\mid Bseq\ X;\ monoseq\ X\ \mid] ==> convergent\ X$
⟨*proof*⟩

18.9 A Few More Equivalence Theorems for Boundedness

alternative formulation for boundedness

lemma *Bseq-iff2*: $Bseq\ X = (\exists\, k > 0.\ \exists\, x.\ \forall\, n.\ |X(n) + -x| \le k)$
⟨*proof*⟩

alternative formulation for boundedness

lemma *Bseq-iff3*: $Bseq\ X = (\exists\, k > 0.\ \exists\, N.\ \forall\, n.\ abs(X(n) + -X(N)) \le k)$
⟨*proof*⟩

lemma *BseqI2*: $(\forall\, n.\ k \le f\ n\ \&\ f\ n \le K) ==> Bseq\ f$
⟨*proof*⟩

18.10 Equivalence Between NS and Standard Cauchy Sequences

18.10.1 Standard Implies Nonstandard

lemma *lemmaCauchy1*:
$star\text{-}n\ x : HNatInfinite$
$==> \{n.\ M \le x\ n\} : FreeUltrafilterNat$
⟨*proof*⟩

lemma *lemmaCauchy2*:
$\{n.\ \forall\, m\ n.\ M \le m\ \&\ M \le (n::nat) --> |X\ m + -\ X\ n| < u\}\ Int$
$\{n.\ M \le xa\ n\}\ Int\ \{n.\ M \le x\ n\} \le$
$\{n.\ abs\ (X\ (xa\ n) + -\ X\ (x\ n)) < u\}$
⟨*proof*⟩

lemma *Cauchy-NSCauchy*: $Cauchy\ X ==> NSCauchy\ X$
⟨*proof*⟩

18.10.2 Nonstandard Implies Standard

lemma *NSCauchy-Cauchy*: $NSCauchy\ X ==> Cauchy\ X$
⟨*proof*⟩

theorem *NSCauchy-Cauchy-iff*: $NSCauchy\ X = Cauchy\ X$
⟨*proof*⟩

A Cauchy sequence is bounded – this is the standard proof mechanization rather than the nonstandard proof

lemma *lemmaCauchy*: $\forall\, n \geq M.\ |X\ M + -\ X\ n| < (1::real)$
$\quad ==>\ \ \forall\, n \geq M.\ |X\ n| < 1 + |X\ M|$
⟨*proof*⟩

lemma *less-Suc-cancel-iff*: $(n < Suc\ M) = (n \leq M)$
⟨*proof*⟩

FIXME: Long. Maximal element in subsequence

lemma *SUP-rabs-subseq*:
$\exists\, m \leq M.\ \forall\, n \leq M.\ |(X::nat=>\ real)\ n| \leq |X\ m|$
⟨*proof*⟩

lemma *lemma-Nat-covered*:
$[|\ \forall\, m::nat.\ m \leq M\ -->\ P\ M\ m;$
$\quad \forall\, m \geq M.\ P\ M\ m\ |]$
$==> \forall\, m.\ P\ M\ m$
⟨*proof*⟩

lemma *lemma-trans1*:
$[|\ \forall\, n \leq M.\ |(X::nat=>real)\ n| \leq a;\ \ a < b\ |]$
$==> \forall\, n \leq M.\ |X\ n| \leq b$
⟨*proof*⟩

lemma *lemma-trans2*:
$[|\ \forall\, n \geq M.\ |(X::nat=>real)\ n| < a;\ a < b\ |]$
$==> \forall\, n \geq M.\ |X\ n| \leq b$
⟨*proof*⟩

lemma *lemma-trans3*:
$[|\ \forall\, n \leq M.\ |X\ n| \leq a;\ a = b\ |]$
$==> \forall\, n \leq M.\ |X\ n| \leq b$
⟨*proof*⟩

lemma *lemma-trans4*: $\forall\, n \geq M.\ |(X::nat=>real)\ n| < a$
$\quad ==>\ \ \forall\, n \geq M.\ |X\ n| \leq a$
⟨*proof*⟩

Proof is more involved than outlines sketched by various authors would suggest

lemma *Cauchy-Bseq*: *Cauchy X ==> Bseq X*
⟨*proof*⟩

A Cauchy sequence is bounded – nonstandard version

lemma *NSCauchy-NSBseq*: *NSCauchy X ==> NSBseq X*
⟨*proof*⟩

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 *NSCauchy-NSconvergent-iff*: *NSCauchy X* = *NSconvergent X*
⟨*proof*⟩

Standard proof for free

lemma *Cauchy-convergent-iff*: *Cauchy X* = *convergent X*
⟨*proof*⟩

We can now try and derive a few properties of sequences, starting with the limit comparison property for sequences.

lemma *NSLIMSEQ-le*:
$[| f ----NS> l; g ----NS> m;$
$\exists N. \forall n \geq N. f(n) \leq g(n)$
$|] ==> l \leq m$
⟨*proof*⟩

lemma *LIMSEQ-le*:
$[| f ----> l; g ----> m; \exists N. \forall n \geq N. f(n) \leq g(n) |]$
$==> l \leq m$
⟨*proof*⟩

lemma *LIMSEQ-le-const*: $[| X ----> r; \forall n.\ a \leq X\ n\ |] ==> a \leq r$
⟨*proof*⟩

lemma *NSLIMSEQ-le-const*: $[| X ----NS> r; \forall n.\ a \leq X\ n\ |] ==> a \leq r$
⟨*proof*⟩

lemma *LIMSEQ-le-const2*: $[| X ----> r; \forall n.\ X\ n \leq a\ |] ==> r \leq a$
⟨*proof*⟩

lemma *NSLIMSEQ-le-const2*: $[| X ----NS> r; \forall n.\ X\ n \leq a\ |] ==> r \leq a$
⟨*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*⟩

standard version

lemma *LIMSEQ-Suc*: $f ----> l ==> (\%n.\ f(Suc\ n)) ----> l$

⟨*proof*⟩

lemma *NSLIMSEQ-imp-Suc*: $(\%n.\ f(Suc\ n))\ {-}{-}{-}{-}NS{>}\ l ==> f\ {-}{-}{-}{-}NS{>}\ l$
⟨*proof*⟩

lemma *LIMSEQ-imp-Suc*: $(\%n.\ f(Suc\ n))\ {-}{-}{-}{-}{>}\ l ==> f\ {-}{-}{-}{-}{>}\ l$
⟨*proof*⟩

lemma *LIMSEQ-Suc-iff*: $((\%n.\ f(Suc\ n))\ {-}{-}{-}{-}{>}\ l) = (f\ {-}{-}{-}{-}{>}\ l)$
⟨*proof*⟩

lemma *NSLIMSEQ-Suc-iff*: $((\%n.\ f(Suc\ n))\ {-}{-}{-}{-}NS{>}\ l) = (f\ {-}{-}{-}{-}NS{>}\ l)$
⟨*proof*⟩

A sequence tends to zero iff its abs does

lemma *LIMSEQ-rabs-zero*: $((\%n.\ |f\ n|)\ {-}{-}{-}{-}{>}\ 0) = (f\ {-}{-}{-}{-}{>}\ 0)$
⟨*proof*⟩

We prove the NS version from the standard one, since the NS proof seems more complicated than the standard one above!

lemma *NSLIMSEQ-rabs-zero*: $((\%n.\ |f\ n|)\ {-}{-}{-}{-}NS{>}\ 0) = (f\ {-}{-}{-}{-}NS{>}\ 0)$
⟨*proof*⟩

Generalization to other limits

lemma *NSLIMSEQ-imp-rabs*: $f\ {-}{-}{-}{-}NS{>}\ l ==> (\%n.\ |f\ n|)\ {-}{-}{-}{-}NS{>}\ |l|$
⟨*proof*⟩

standard version

lemma *LIMSEQ-imp-rabs*: $f\ {-}{-}{-}{-}{>}\ l ==> (\%n.\ |f\ n|)\ {-}{-}{-}{-}{>}\ |l|$
⟨*proof*⟩

An unbounded sequence's inverse tends to 0

standard proof seems easier

lemma *LIMSEQ-inverse-zero*:
$\forall y.\ \exists N.\ \forall n \geq N.\ y < f(n) ==> (\%n.\ inverse(f\ n))\ {-}{-}{-}{-}{>}\ 0$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse-zero*:
$\forall y.\ \exists N.\ \forall n \geq N.\ y < f(n)$
$==> (\%n.\ inverse(f\ n))\ {-}{-}{-}{-}NS{>}\ 0$
⟨*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*⟩

lemma *NSLIMSEQ-inverse-real-of-nat*: (%n. inverse(real(Suc n))) ----NS> 0
⟨proof⟩

The sequence r + (1::'a) / n tends to r as n tends to infinity is now easily proved

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

lemma *NSLIMSEQ-inverse-real-of-nat-add*:
(%n. r + inverse(real(Suc n))) ----NS> r
⟨proof⟩

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

lemma *NSLIMSEQ-inverse-real-of-nat-add-minus*:
(%n. r + −inverse(real(Suc n))) ----NS> r
⟨proof⟩

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

lemma *NSLIMSEQ-inverse-real-of-nat-add-minus-mult*:
(%n. r∗(1 + −inverse(real(Suc n)))) ----NS> r
⟨proof⟩

Real Powers

lemma *NSLIMSEQ-pow* [*rule-format*]:
(X ----NS> a) --> ((%n. (X n) ˆ m) ----NS> a ˆ m)
⟨proof⟩

lemma *LIMSEQ-pow*: X ----> a ==> (%n. (X n) ˆ m) ----> a ˆ m
⟨proof⟩

The sequence x ˆ 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 ≤ x; x ≤ 1 |] ==> Bseq (%n. x ˆ n)
⟨proof⟩

lemma *monoseq-realpow*: [| 0 ≤ x; x ≤ 1 |] ==> monoseq (%n. x ˆ n)
⟨proof⟩

lemma *convergent-realpow*: [| 0 ≤ x; x ≤ 1 |] ==> convergent (%n. x ˆ n)
⟨proof⟩

We now use NS criterion to bring proof of theorem through

lemma *NSLIMSEQ-realpow-zero*: [| $0 \leq x$; $x < 1$ |] ==> (%n. x ^ n) ----NS> 0
⟨*proof*⟩

standard version

lemma *LIMSEQ-realpow-zero*: [| $0 \leq x$; $x < 1$ |] ==> (%n. x ^ n) ----> 0
⟨*proof*⟩

lemma *LIMSEQ-divide-realpow-zero*: 1 < x ==> (%n. a / (x ^ n)) ----> 0
⟨*proof*⟩

Limit of c ^ n for $|c| < (1::'a)$

lemma *LIMSEQ-rabs-realpow-zero*: $|c| < 1$ ==> (%n. $|c|$ ^ n) ----> 0
⟨*proof*⟩

lemma *NSLIMSEQ-rabs-realpow-zero*: $|c| < 1$ ==> (%n. $|c|$ ^ n) ----NS> 0
⟨*proof*⟩

lemma *LIMSEQ-rabs-realpow-zero2*: $|c| < 1$ ==> (%n. c ^ n) ----> 0
⟨*proof*⟩

lemma *NSLIMSEQ-rabs-realpow-zero2*: $|c| < 1$ ==> (%n. c ^ n) ----NS> 0
⟨*proof*⟩

18.11 Hyperreals and Sequences

A bounded sequence is a finite hyperreal

lemma *NSBseq-HFinite-hypreal*: *NSBseq X ==> star-n X : HFinite*
⟨*proof*⟩

A sequence converging to zero defines an infinitesimal

lemma *NSLIMSEQ-zero-Infinitesimal-hypreal*:
    X ----NS> 0 ==> star-n X : Infinitesimal
⟨*proof*⟩

⟨*ML*⟩

end

19 Lim: Limits, Continuity and Differentiation

theory *Lim*

imports *SEQ*
begin

Standard and Nonstandard Definitions

constdefs

LIM :: [$real$=>$real$,$real$,$real$] => $bool$
(((-)/ −− (-)/ −−> (-)) [60, 0, 60] 60)
f −− a −−> L ==
$\forall r > 0.\ \exists s > 0.\ \forall x.\ x \neq a$ & $|x + -a| < s$
−−> $|f\, x + -L| < r$

$NSLIM$ :: [$real$=>$real$,$real$,$real$] => $bool$
(((-)/ −− (-)/ −−NS> (-)) [60, 0, 60] 60)
f −− a −−NS> L == ($\forall x.$ ($x \neq$ hypreal-of-real a &
x @= hypreal-of-real a −−>
(∗f∗ f) x @= hypreal-of-real L))

$isCont$ :: [$real$=>$real$,$real$] => $bool$
$isCont\ f\ a$ == (f −− a −−> ($f\ a$))

$isNSCont$ :: [$real$=>$real$,$real$] => $bool$
— NS definition dispenses with limit notions
$isNSCont\ f\ a$ == ($\forall y.$ y @= hypreal-of-real a −−>
(∗f∗ f) y @= hypreal-of-real ($f\ a$))

$deriv$:: [$real$=>$real$,$real$,$real$] => $bool$
— Differentiation: D is derivative of function f at x
(($DERIV$ (-)/ (-)/ :> (-)) [1000, 1000, 60] 60)
$DERIV\ f\ x$:> D == ((%h. ($f(x + h) + -f\, x)/h$) −− 0 −−> D)

$nsderiv$ :: [$real$=>$real$,$real$,$real$] => $bool$
(($NSDERIV$ (-)/ (-)/ :> (-)) [1000, 1000, 60] 60)
$NSDERIV\ f\ x$:> D == ($\forall h \in Infinitesimal - \{0\}$.
((∗f∗ f)(hypreal-of-real $x + h$) +
− hypreal-of-real ($f\ x$))/h @= hypreal-of-real D)

$differentiable$ :: [$real$=>$real$,$real$] => $bool$ (**infixl** $differentiable$ 60)
f $differentiable$ x == ($\exists D.$ $DERIV\ f\ x$:> D)

$NSdifferentiable$ :: [$real$=>$real$,$real$] => $bool$
(**infixl** $NSdifferentiable$ 60)
f $NSdifferentiable$ x == ($\exists D.$ $NSDERIV\ f\ x$:> D)

$increment$ :: [$real$=>$real$,$real$,$hypreal$] => $hypreal$
$increment\ f\ x\ h$ == (@inc. f $NSdifferentiable$ x &
inc = (∗f∗ f)(hypreal-of-real $x + h$) + −hypreal-of-real ($f\ x$))

$isUCont$ :: ($real$=>$real$) => $bool$
$isUCont\ f$ == $\forall r > 0.\ \exists s > 0.\ \forall x\ y.\ |x + -y| < s$ −−> $|f\, x + -f\, y| < r$

isNSUCont :: (*real*=>*real*) => *bool*
isNSUCont f == (∀ *x y*. *x* @= *y* −−> (∗*f*∗ *f*) *x* @= (∗*f*∗ *f*) *y*)

consts
Bolzano-bisect :: [*real*∗*real*=>*bool*, *real*, *real*, *nat*] => (*real*∗*real*)
— Used in the proof of the Bolzano theorem

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 Some Purely Standard Proofs

lemma *LIM-eq*:
f −− *a* −−> *L* =
(∀ *r*>*0*.∃ *s*>*0*.∀ *x*. *x* ≠ *a* & |*x*−*a*| < *s* −−> |*f x* − *L*| < *r*)
⟨*proof*⟩

lemma *LIM-D*:
[| *f* −− *a* −−> *L*; *0*<*r* |]
==> ∃ *s*>*0*.∀ *x*. *x* ≠ *a* & |*x*−*a*| < *s* −−> |*f x* − *L*| < *r*
⟨*proof*⟩

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

lemma *LIM-add*:
assumes *f*: *f* −− *a* −−> *L* **and** *g*: *g* −− *a* −−> *M*
shows (%*x*. *f x* + *g*(*x*)) −− *a* −−> (*L* + *M*)
⟨*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-const-not-eq*: $k \neq L$ ==> ~ ((%x. k) −− a −−> L)
⟨*proof*⟩

lemma *LIM-const-eq*: (%x. k) −− x −−> L ==> $k = L$
⟨*proof*⟩

lemma *LIM-unique*: [| f −− a −−> L; f −− a −−> M |] ==> $L = M$
⟨*proof*⟩

lemma *LIM-mult-zero*:
  assumes f: f −− a −−> 0 **and** g: g −− a −−> 0
  shows (%x. $f(x) * g(x)$) −− a −−> 0
⟨*proof*⟩

lemma *LIM-self*: (%x. x) −− a −−> a
⟨*proof*⟩

Limits are equal for functions equal except at limit point

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

Two uses in Hyperreal/Transcendental.ML

lemma *LIM-trans*:
  [| (%x. $f(x) + -g(x)$) −− a −−> 0; g −− a −−> l |] ==> f −− a −−> l
⟨*proof*⟩

20.1 Relationships Between Standard and Nonstandard Concepts

Standard and NS definitions of Limit

lemma *LIM-NSLIM*:
  f −− x −−> L ==> f −− x −−*NS*> L
⟨*proof*⟩

20.1.1 Limit: The NS definition implies the standard definition.

lemma *lemma-LIM*: $\forall s{>}0.\exists xa.\ \ xa \neq x$ &
    $|xa + -\ x| < s$ & $r \leq |f\,xa + -L|$
  ==> $\forall n$::*nat*. $\exists xa.\ \ xa \neq x$ &
      $|xa + -x| <$ *inverse*(*real*(*Suc* n)) & $r \leq |f\,xa + -L|$
⟨*proof*⟩

lemma *lemma-skolemize-LIM2*:
  $\forall s{>}0.\exists xa.\ \ xa \neq x$ & $|xa + -\ x| < s$ & $r \leq |f\,xa + -L|$
  ==> $\exists X.\ \forall n$::*nat*. $X\ n \neq x$ &
      $|X\ n + -x| <$ *inverse*(*real*(*Suc* n)) & $r \leq$ *abs*(f ($X\ n$) + $-L$)

⟨*proof*⟩

lemma *lemma-simp*: $\forall n.\ X\ n \neq x\ \&$
$|X\ n + -\ x| < inverse\ (real(Suc\ n))\ \&$
$r \leq abs\ (f\ (X\ n) + -\ L) ==>$
$\forall n.\ |X\ n + -\ x| < inverse\ (real(Suc\ n))$
⟨*proof*⟩

NSLIM =¿ LIM

lemma *NSLIM-LIM*: $f\ --\ x\ --NS>\ L ==> f\ --\ x\ -->\ L$
⟨*proof*⟩

theorem *LIM-NSLIM-iff*: $(f\ --\ x\ -->\ L) = (f\ --\ x\ --NS>\ L)$
⟨*proof*⟩

Proving properties of limits using nonstandard definition. The properties hold for standard limits as well!

lemma *NSLIM-mult*:
$[|\ f\ --\ x\ --NS>\ l;\ g\ --\ x\ --NS>\ m\ |]$
$==> (\%x.\ f(x) * g(x))\ --\ x\ --NS>\ (l * m)$
⟨*proof*⟩

lemma *LIM-mult2*:
$[|\ f\ --\ x\ -->\ l;\ g\ --\ x\ -->\ m\ |] ==> (\%x.\ f(x) * g(x))\ --\ x\ -->\ (l$
$*\ m)$
⟨*proof*⟩

lemma *NSLIM-add*:
$[|\ f\ --\ x\ --NS>\ l;\ g\ --\ x\ --NS>\ m\ |]$
$==> (\%x.\ f(x) + g(x))\ --\ x\ --NS>\ (l + m)$
⟨*proof*⟩

lemma *LIM-add2*:
$[|\ f\ --\ x\ -->\ l;\ g\ --\ x\ -->\ m\ |] ==> (\%x.\ f(x) + g(x))\ --\ x\ -->\ (l$
$+\ m)$
⟨*proof*⟩

lemma *NSLIM-const* [*simp*]: $(\%x.\ k)\ --\ x\ --NS>\ k$
⟨*proof*⟩

lemma *LIM-const2*: $(\%x.\ k)\ --\ x\ -->\ k$
⟨*proof*⟩

lemma *NSLIM-minus*: $f\ --\ a\ --NS>\ L ==> (\%x.\ -f(x))\ --\ a\ --NS>\ -L$
⟨*proof*⟩

lemma *LIM-minus2*: $f\ --\ a\ -->\ L ==> (\%x.\ -f(x))\ --\ a\ -->\ -L$

⟨*proof*⟩

lemma *NSLIM-add-minus*: $[| f -- x --NS> l;\ g -- x --NS> m |] ==> (\%x.\ f(x) + -g(x)) -- x --NS> (l + -m)$
⟨*proof*⟩

lemma *LIM-add-minus2*: $[| f -- x --> l;\ g -- x --> m |] ==> (\%x.\ f(x) + -g(x)) -- x --> (l + -m)$
⟨*proof*⟩

lemma *NSLIM-inverse*:
$[| f -- a --NS> L;\ L \neq 0 |]$
$==> (\%x.\ inverse(f(x))) -- a --NS> (inverse\ L)$
⟨*proof*⟩

lemma *LIM-inverse*: $[| f -- a --> L;\ L \neq 0 |] ==> (\%x.\ inverse(f(x))) -- a --> (inverse\ L)$
⟨*proof*⟩

lemma *NSLIM-zero*:
assumes f: $f -- a --NS> l$ **shows** $(\%x.\ f(x) + -l) -- a --NS> 0$
⟨*proof*⟩

lemma *LIM-zero2*: $f -- a --> l ==> (\%x.\ f(x) + -l) -- a --> 0$
⟨*proof*⟩

lemma *NSLIM-zero-cancel*: $(\%x.\ f(x) - l) -- x --NS> 0 ==> f -- x --NS> l$
⟨*proof*⟩

lemma *LIM-zero-cancel*: $(\%x.\ f(x) - l) -- x --> 0 ==> f -- x --> l$
⟨*proof*⟩

lemma *NSLIM-not-zero*: $k \neq 0 ==> \sim ((\%x.\ k) -- x --NS> 0)$
⟨*proof*⟩

lemma *NSLIM-const-not-eq*: $k \neq L ==> \sim ((\%x.\ k) -- x --NS> L)$
⟨*proof*⟩

lemma *NSLIM-const-eq*: $(\%x.\ k) -- x --NS> L ==> k = L$
⟨*proof*⟩

can actually be proved more easily by unfolding the definition!

lemma *NSLIM-unique*: $[| f -- x --NS> L;\ f -- x --NS> M |] ==> L = M$
⟨*proof*⟩

lemma *LIM-unique2*: $[| \; f \; -- \; x \; --> \; L; \; f \; -- \; x \; --> \; M \; |] ==> L = M$
⟨*proof*⟩

lemma *NSLIM-mult-zero*: $[| \; f \; -- \; x \; --NS> \; 0; \; g \; -- \; x \; --NS> \; 0 \; |] ==> (\%x. \; f(x)*g(x)) \; -- \; x \; --NS> \; 0$
⟨*proof*⟩

lemma *LIM-mult-zero2*: $[| \; f \; -- \; x \; --> \; 0; \; g \; -- \; x \; --> \; 0 \; |] ==> (\%x. \; f(x)*g(x)) \; -- \; x \; --> \; 0$
⟨*proof*⟩

lemma *NSLIM-self*: $(\%x. \; x) \; -- \; a \; --NS> \; a$
⟨*proof*⟩

20.2 Derivatives and Continuity: NS and Standard properties

20.2.1 Continuity

lemma *isNSContD*: $[| \; \mathit{isNSCont} \; f \; a; \; y \approx \mathit{hypreal\text{-}of\text{-}real} \; a \; |] ==> (\; *f* \; f) \; y \approx \mathit{hypreal\text{-}of\text{-}real} \; (f \; a)$
⟨*proof*⟩

lemma *isNSCont-NSLIM*: $\mathit{isNSCont} \; f \; a ==> f \; -- \; a \; --NS> \; (f \; a)$
⟨*proof*⟩

lemma *NSLIM-isNSCont*: $f \; -- \; a \; --NS> \; (f \; a) ==> \mathit{isNSCont} \; f \; a$
⟨*proof*⟩

NS continuity can be defined using NS Limit in similar fashion to standard def of continuity

lemma *isNSCont-NSLIM-iff*: $(\mathit{isNSCont} \; f \; a) = (f \; -- \; a \; --NS> \; (f \; a))$
⟨*proof*⟩

Hence, NS continuity can be given in terms of standard limit

lemma *isNSCont-LIM-iff*: $(\mathit{isNSCont} \; f \; a) = (f \; -- \; a \; --> \; (f \; a))$
⟨*proof*⟩

Moreover, it's trivial now that NS continuity is equivalent to standard continuity

lemma *isNSCont-isCont-iff*: $(\mathit{isNSCont} \; f \; a) = (\mathit{isCont} \; f \; a)$
⟨*proof*⟩

Standard continuity ==¿ NS continuity

lemma *isCont-isNSCont*: *isCont f a ==> isNSCont f a*
⟨*proof*⟩

NS continuity ==¿ Standard continuity

lemma *isNSCont-isCont*: *isNSCont f a ==> isCont f a*
⟨*proof*⟩

Alternative definition of continuity

lemma *NSLIM-h-iff*: *(f −− a −−NS> L) = ((%h. f(a + h)) −− 0 −−NS> L)*
⟨*proof*⟩

lemma *NSLIM-isCont-iff*: *(f −− a −−NS> f a) = ((%h. f(a + h)) −− 0 −−NS> f a)*
⟨*proof*⟩

lemma *LIM-isCont-iff*: *(f −− a −−> f a) = ((%h. f(a + h)) −− 0 −−> f(a))*
⟨*proof*⟩

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

Immediate application of nonstandard criterion for continuity can offer very simple proofs of some standard property of continuous functions

sum continuous

lemma *isCont-add*: *[| isCont f a; isCont g a |] ==> isCont (%x. f(x) + g(x)) a*
⟨*proof*⟩

mult continuous

lemma *isCont-mult*: *[| isCont f a; isCont g a |] ==> isCont (%x. f(x) ∗ g(x)) a*
⟨*proof*⟩

composition of continuous functions Note very short straightforard proof!

lemma *isCont-o*: *[| isCont f a; isCont g (f a) |] ==> isCont (g o f) a*
⟨*proof*⟩

lemma *isCont-o2*: *[| isCont f a; isCont g (f a) |] ==> isCont (%x. g (f x)) a*
⟨*proof*⟩

lemma *isNSCont-minus*: *isNSCont f a ==> isNSCont (%x. − f x) a*
⟨*proof*⟩

lemma *isCont-minus*: *isCont f a ==> isCont (%x. − f x) a*
⟨*proof*⟩

lemma *isCont-inverse*:
[| isCont f x; f x ≠ 0 |] ==> isCont (%x. inverse (f x)) x

$\langle proof \rangle$

lemma *isNSCont-inverse*: $[| \; isNSCont\, f\, x; f\, x \neq 0 \; |] ==> isNSCont\ (\%x.\ inverse\ (f\, x))\ x$
$\langle proof \rangle$

lemma *isCont-diff*:
$[| \; isCont\, f\, a; \; isCont\ g\ a \; |] ==> isCont\ (\%x.\ f(x) - g(x))\ a$
$\langle proof \rangle$

lemma *isCont-const* [*simp*]: $isCont\ (\%x.\ k)\ a$
$\langle proof \rangle$

lemma *isNSCont-const* [*simp*]: $isNSCont\ (\%x.\ k)\ a$
$\langle proof \rangle$

lemma *isNSCont-abs* [*simp*]: $isNSCont\ abs\ a$
$\langle proof \rangle$

lemma *isCont-abs* [*simp*]: $isCont\ abs\ a$
$\langle proof \rangle$

Uniform continuity

lemma *isNSUContD*: $[| \; isNSUCont\, f; \; x \approx y |] ==> (\ {*f*}\ f)\ x \approx (\ {*f*}\ f)\ y$
$\langle proof \rangle$

lemma *isUCont-isCont*: $isUCont\, f ==> isCont\, f\, x$
$\langle proof \rangle$

lemma *isUCont-isNSUCont*: $isUCont\, f ==> isNSUCont\, f$
$\langle proof \rangle$

lemma *lemma-LIMu*: $\forall\, s{>}0.\exists\, z\ y.\ |z + -\ y| < s\ \&\ r \leq |f\, z + -f\, y|$
$==> \forall\, n{::}nat.\ \exists\, z\ y.\ |z + -y| < inverse(real(Suc\ n))\ \&\ r \leq |f\, z + -f\, y|$
$\langle proof \rangle$

lemma *lemma-skolemize-LIM2u*:
$\forall\, s{>}0.\exists\, z\ y.\ |z + -\ y| < s\ \ \&\ r \leq |f\, z + -f\, y|$
$==> \exists\, X\ Y.\ \forall\, n{::}nat.$
$abs(X\ n + -(Y\ n)) < inverse(real(Suc\ n))\ \&$
$r \leq abs(f\ (X\ n) + -f\ (Y\ n))$
$\langle proof \rangle$

lemma *lemma-simpu*: $\forall\, n.\ |X\ n + -Y\ n| < inverse\ (real(Suc\ n))\ \&$
$r \leq abs\ (f\ (X\ n) + -\ f(Y\ n)) ==>$
$\forall\, n.\ |X\ n + -\ \ Y\ n| < inverse\ (real(Suc\ n))$
$\langle proof \rangle$

lemma *isNSUCont-isUCont*:

isNSUCont f ==> isUCont f
⟨*proof*⟩

Derivatives

lemma *DERIV-iff*: (*DERIV f x :> D*) = ((%*h*. (*f*(*x* + *h*) + − *f*(*x*))/*h*) −− *0* −−> *D*)
⟨*proof*⟩

lemma *DERIV-NS-iff*:
(*DERIV f x :> D*) = ((%*h*. (*f*(*x* + *h*) + − *f*(*x*))/*h*) −− *0* −−*NS*> *D*)
⟨*proof*⟩

lemma *DERIV-D*: *DERIV f x :> D ==>* (%*h*. (*f*(*x* + *h*) + − *f*(*x*))/*h*) −− *0* −−> *D*
⟨*proof*⟩

lemma *NS-DERIV-D*: *DERIV f x :> D ==>* (%*h*. (*f*(*x* + *h*) + − *f*(*x*))/*h*) −− *0* −−*NS*> *D*
⟨*proof*⟩

20.2.2 Uniqueness

lemma *DERIV-unique*:
[| *DERIV f x :> D*; *DERIV f x :> E* |] *==> D = E*
⟨*proof*⟩

lemma *NSDeriv-unique*:
[| *NSDERIV f x :> D*; *NSDERIV f x :> E* |] *==> D = E*
⟨*proof*⟩

20.2.3 Differentiable

lemma *differentiableD*: *f differentiable x ==>* ∃ *D*. *DERIV f x :> D*
⟨*proof*⟩

lemma *differentiableI*: *DERIV f x :> D ==> f differentiable x*
⟨*proof*⟩

lemma *NSdifferentiableD*: *f NSdifferentiable x ==>* ∃ *D*. *NSDERIV f x :> D*
⟨*proof*⟩

lemma *NSdifferentiableI*: *NSDERIV f x :> D ==> f NSdifferentiable x*
⟨*proof*⟩

20.2.4 Alternative definition for differentiability

lemma *LIM-I*:
(!!*r*. *0*<*r* ==> ∃ *s*>*0*.∀ *x*. *x* ≠ *a* & |*x*−*a*| < *s* −−> |*f x* − *L*| < *r*)
==> *f* −− *a* −−> *L*
⟨*proof*⟩

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*⟩

20.3 Equivalence of NS and standard definitions of differentiation

20.3.1 First NSDERIV in terms of NSLIM

first equivalence

lemma *NSDERIV-NSLIM-iff*:
$(NSDERIV\ f\ x :> D) = ((\%h.\ (f(x + h) + - f(x))/h)\ {-}{-}\ 0\ {-}{-}NS{>}\ D)$
⟨*proof*⟩

second equivalence

lemma *NSDERIV-NSLIM-iff2*:
$(NSDERIV\ f\ x :> D) = ((\%z.\ (f(z) - f(x))\ /\ (z{-}x))\ {-}{-}\ x\ {-}{-}NS{>}\ D)$
⟨*proof*⟩

lemma *NSDERIV-iff2*:
$(NSDERIV\ f\ x :> D) =$
$(\forall\ w.$
$w \neq hypreal\text{-}of\text{-}real\ x\ \&\ w \approx hypreal\text{-}of\text{-}real\ x\ {-}{-}{>}$
$(\ {*}f{*}\ (\%z.\ (f\ z - f\ x)\ /\ (z{-}x)))\ w \approx hypreal\text{-}of\text{-}real\ D)$
⟨*proof*⟩

lemma *hypreal-not-eq-minus-iff*: $(x \neq a) = (x + -a \neq (0::hypreal))$
⟨*proof*⟩

lemma *NSDERIVD5*:
$(NSDERIV\ f\ x :> D) ==>$
$(\forall\ u.\ u \approx hypreal\text{-}of\text{-}real\ x\ {-}{-}{>}$
$(\ {*}f{*}\ (\%z.\ f\ z - f\ x))\ u \approx hypreal\text{-}of\text{-}real\ D * (u - hypreal\text{-}of\text{-}real\ x))$
⟨*proof*⟩

lemma *NSDERIVD4*:
$(NSDERIV\ f\ x :> D) ==>$
$(\forall\ h \in Infinitesimal.$
$((\ {*}f{*}\ f)(hypreal\text{-}of\text{-}real\ x + h) -$
$hypreal\text{-}of\text{-}real\ (f\ x)) \approx (hypreal\text{-}of\text{-}real\ D) * h)$
⟨*proof*⟩

lemma *NSDERIVD3*:
$(NSDERIV\ f\ x :> D) ==>$
$(\forall h \in Infinitesimal - \{0\}.$
$((\ *f*\ f)(hypreal\text{-}of\text{-}real\ x + h) -$
$hypreal\text{-}of\text{-}real\ (f\ x)) \approx (hypreal\text{-}of\text{-}real\ D) * h)$
⟨*proof*⟩

Now equivalence between NSDERIV and DERIV

lemma *NSDERIV-DERIV-iff*: $(NSDERIV\ f\ x :> D) = (DERIV\ f\ x :> D)$
⟨*proof*⟩

Differentiability implies continuity nice and simple "algebraic" proof

lemma *NSDERIV-isNSCont*: $NSDERIV\ f\ x :> D ==> isNSCont\ f\ x$
⟨*proof*⟩

Now Sandard proof

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

lemma *DERIV-const* [*simp*]: $(DERIV\ (\%x.\ k)\ x :> 0)$
⟨*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*⟩

lemma *DERIV-add*: $[|\ DERIV\ f\ x :> Da;\ DERIV\ g\ x :> Db\ |]$
$==> DERIV\ (\%x.\ f\ x + g\ x)\ x :> Da + Db$
⟨*proof*⟩

Product of functions - Proof is trivial but tedious and long due to rearrangement of terms

lemma *lemma-nsderiv1*: $((a{::}hypreal)*b) + -(c*d) = (b*(a + -c)) + (c*(b + -d))$
⟨*proof*⟩

lemma *lemma-nsderiv2*: $[|\ (x + y)\ /\ z = hypreal\text{-}of\text{-}real\ D + yb;\ z \neq 0;$
$z \in Infinitesimal;\ yb \in Infinitesimal\ |]$

$==> x + y \approx 0$
⟨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⟩

lemma *DERIV-mult*:
[| *DERIV f x* :> *Da*; *DERIV g x* :> *Db* |]
$==>$ *DERIV* (%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⟩

lemma *DERIV-cmult*:
DERIV f x :> *D* $==>$ *DERIV* (%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⟩

lemma *DERIV-minus*: *DERIV f x* :> *D* $==>$ *DERIV* (%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 *DERIV-add-minus*: [| *DERIV f x* :> *Da*; *DERIV g x* :> *Db* |] $==>$ *DERIV* (%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⟩

lemma *DERIV-diff*:

[| *DERIV f x* :> *Da*; *DERIV g x* :> *Db* |]
==> *DERIV* (%*x*. *f x* − *g x*) *x* :> *Da*−*Db*
⟨*proof*⟩

(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*⟩

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*) (*hypreal-of-real*(*x*) + *xa*) = *hypreal-of-real*(*g x*);
xa ∈ *Infinitesimal*;
xa ≠ *0*
|] ==> *D* = *0*
⟨*proof*⟩

lemma *NSDERIV-approx*:
[| *NSDERIV f x* :> *D*; *h* ∈ *Infinitesimal*; *h* ≠ *0* |]
==> (**f* f*) (*hypreal-of-real*(*x*) + *h*) + −*hypreal-of-real*(*f x*) ≈ *0*

⟨*proof*⟩

lemma *NSDERIVD1*: [| *NSDERIV f* (*g x*) :> *Da*;
($*f*$ *g*) (*hypreal-of-real*(x) + *xa*) ≠ *hypreal-of-real* (*g x*);
($*f*$ *g*) (*hypreal-of-real*(x) + *xa*) ≈ *hypreal-of-real* (*g x*)
|] ==> (($*f*$ *f*) (($*f*$ *g*) (*hypreal-of-real*(x) + *xa*))
+ − *hypreal-of-real* (*f* (*g x*)))
/ (($*f*$ *g*) (*hypreal-of-real*(x) + *xa*) + − *hypreal-of-real* (*g x*))
≈ *hypreal-of-real*(*Da*)
⟨*proof*⟩

lemma *NSDERIVD2*: [| *NSDERIV g x* :> *Db*; *xa* ∈ *Infinitesimal*; *xa* ≠ *0* |]
==> (($*f*$ *g*) (*hypreal-of-real*(x) + *xa*) + − *hypreal-of-real*(*g x*)) / *xa*
≈ *hypreal-of-real*(*Db*)
⟨*proof*⟩

lemma *lemma-chain*: (*z*::*hypreal*) ≠ *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*⟩

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*⟩

Differentiation of natural number powers

lemma *NSDERIV-Id* [*simp*]: *NSDERIV* (%*x*. *x*) *x* :> *1*
⟨*proof*⟩

lemma *DERIV-Id* [*simp*]: *DERIV* (%*x*. *x*) *x* :> *1*
⟨*proof*⟩

lemmas *isCont-Id* = *DERIV-Id* [*THEN DERIV-isCont*, *standard*]

lemma *DERIV-cmult-Id* [*simp*]: *DERIV* (*op* $*$ *c*) *x* :> *c*
⟨*proof*⟩

lemma *NSDERIV-cmult-Id* [*simp*]: *NSDERIV* (*op* ∗ *c*) *x* :> *c*
⟨*proof*⟩

lemma *DERIV-pow*: *DERIV* (%*x*. *x* ^ *n*) *x* :> *real n* ∗ (*x* ^ (*n* − *Suc 0*))
⟨*proof*⟩

lemma *NSDERIV-pow*: *NSDERIV* (%*x*. *x* ^ *n*) *x* :> *real n* ∗ (*x* ^ (*n* − *Suc 0*))
⟨*proof*⟩

Power of -1

lemma *NSDERIV-inverse*:
x ≠ *0* ==> *NSDERIV* (%*x*. *inverse*(*x*)) *x* :> (− (*inverse x* ^ *Suc* (*Suc 0*)))
⟨*proof*⟩

lemma *DERIV-inverse*: *x* ≠ *0* ==> *DERIV* (%*x*. *inverse*(*x*)) *x* :> (−(*inverse x* ^ *Suc* (*Suc 0*)))
⟨*proof*⟩

Derivative of inverse

lemma *DERIV-inverse-fun*: [| *DERIV f x* :> *d*; *f*(*x*) ≠ *0* |]
==> *DERIV* (%*x*. *inverse*(*f x*)) *x* :> (− (*d* ∗ *inverse*(*f*(*x*) ^ *Suc* (*Suc 0*))))
⟨*proof*⟩

lemma *NSDERIV-inverse-fun*: [| *NSDERIV f x* :> *d*; *f*(*x*) ≠ *0* |]
==> *NSDERIV* (%*x*. *inverse*(*f x*)) *x* :> (− (*d* ∗ *inverse*(*f*(*x*) ^ *Suc* (*Suc 0*))))
⟨*proof*⟩

Derivative of quotient

lemma *DERIV-quotient*: [| *DERIV f x* :> *d*; *DERIV g x* :> *e*; *g*(*x*) ≠ *0* |]
==> *DERIV* (%*y*. *f*(*y*) / (*g y*)) *x* :> (*d*∗*g*(*x*) + −(*e*∗*f*(*x*))) / (*g*(*x*) ^ *Suc* (*Suc 0*))
⟨*proof*⟩

lemma *NSDERIV-quotient*: [| *NSDERIV f x* :> *d*; *DERIV g x* :> *e*; *g*(*x*) ≠ *0* |]
==> *NSDERIV* (%*y*. *f*(*y*) / (*g y*)) *x* :> (*d*∗*g*(*x*)
+ −(*e*∗*f*(*x*))) / (*g*(*x*) ^ *Suc* (*Suc 0*))
⟨*proof*⟩

lemma *CARAT-DERIV*:
(*DERIV f x* :> *l*) =

$(\exists\, g.\ (\forall\, z.\ f\ z\ -\ f\ x\ =\ g\ z\ *\ (z-x))\ \&\ \mathit{isCont}\ g\ x\ \&\ g\ x\ =\ l)$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *CARAT-NSDERIV*: *NSDERIV f x* :> l ==>
$\exists\, g.\ (\forall\, z.\ f\ z\ -\ f\ x\ =\ g\ z\ *\ (z-x))\ \&\ \mathit{isNSCont}\ g\ x\ \&\ g\ x\ =\ l$
⟨*proof*⟩

lemma *hypreal-eq-minus-iff3*: $(x = y + z) = (x + -z = (y{::}\mathit{hypreal}))$
⟨*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*⟩

Lemmas about nested intervals and proof by bisection (cf.Harrison). All considerably tidied by lcp.

lemma *lemma-f-mono-add* [*rule-format* (*no-asm*)]: $(\forall\, n.\ (f{::}\mathit{nat}{=}{>}\mathit{real})\ n\ \leq\ f\ (\mathit{Suc}\ n))$ --> $f\ m \leq f(m + \mathit{no})$
⟨*proof*⟩

lemma *f-inc-g-dec-Beq-f*: [| $\forall\, n.\ f(n) \leq f(\mathit{Suc}\ n)$;
$\forall\, n.\ g(\mathit{Suc}\ n) \leq g(n)$;
$\forall\, n.\ f(n) \leq g(n)$ |]
==> *Bseq f*
⟨*proof*⟩

lemma *f-inc-g-dec-Beq-g*: [| $\forall\, n.\ f(n) \leq f(\mathit{Suc}\ n)$;
$\forall\, n.\ g(\mathit{Suc}\ n) \leq g(n)$;
$\forall\, n.\ f(n) \leq g(n)$ |]
==> *Bseq g*
⟨*proof*⟩

lemma *f-inc-imp-le-lim*: [| $\forall\, n.\ f\ n \leq f\ (\mathit{Suc}\ n)$; *convergent f* |] ==> $f\ n \leq \mathit{lim}\ f$
⟨*proof*⟩

lemma *lim-uminus*: *convergent g* ==> *lim* (%x. − *g x*) = − (*lim g*)
⟨*proof*⟩

lemma *g-dec-imp-lim-le*: [| $\forall\, n.\ g(\mathit{Suc}\ n) \leq g(n)$; *convergent g* |] ==> $\mathit{lim}\ g \leq g\ n$
⟨*proof*⟩

lemma *lemma-nest*: [| $\forall\, n.\ f(n) \leq f(\mathit{Suc}\ n)$;
$\forall\, n.\ g(\mathit{Suc}\ n) \leq g(n)$;
$\forall\, n.\ f(n) \leq g(n)$ |]

$==> \exists\, l\ m.\ l \le m\ \&\ ((\forall\, n.\ f(n) \le l)\ \&\ f\ {---->}\ l)\ \&$
$((\forall\, n.\ m \le g(n))\ \&\ g\ {---->}\ m)$
⟨*proof*⟩

lemma *lemma-nest-unique*: $[|\ \forall\, n.\ f(n) \le f(Suc\ n);$
$\forall\, n.\ g(Suc\ n) \le g(n);$
$\forall\, n.\ f(n) \le g(n);$
$(\%n.\ f(n) - g(n))\ {---->}\ 0\ |]$
$==> \exists\, l.\ ((\forall\, n.\ f(n) \le l)\ \&\ f\ {---->}\ l)\ \&$
$((\forall\, n.\ l \le g(n))\ \&\ g\ {---->}\ l)$
⟨*proof*⟩

The universal quantifiers below are required for the declaration of *Bolzano-nest-unique* below.

lemma *Bolzano-bisect-le*:
$a \le b ==> \forall\, n.\ fst\ (Bolzano\text{-}bisect\ P\ a\ b\ n) \le snd\ (Bolzano\text{-}bisect\ P\ a\ b\ n)$
⟨*proof*⟩

lemma *Bolzano-bisect-fst-le-Suc*: $a \le b ==>$
$\forall\, n.\ fst(Bolzano\text{-}bisect\ P\ a\ b\ n) \le fst\ (Bolzano\text{-}bisect\ P\ a\ b\ (Suc\ n))$
⟨*proof*⟩

lemma *Bolzano-bisect-Suc-le-snd*: $a \le b ==>$
$\forall\, n.\ snd(Bolzano\text{-}bisect\ P\ a\ b\ (Suc\ n)) \le snd\ (Bolzano\text{-}bisect\ P\ a\ b\ n)$
⟨*proof*⟩

lemma *eq-divide-2-times-iff*: $((x::real) = y\ /\ (2 * z)) = (2 * x = y/z)$
⟨*proof*⟩

lemma *Bolzano-bisect-diff*:
$a \le b ==>$
$snd(Bolzano\text{-}bisect\ P\ a\ b\ n) - fst(Bolzano\text{-}bisect\ P\ a\ b\ n) =$
$(b-a)\ /\ (2\ \hat{}\ n)$
⟨*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*: $!!a\ b\ c.\ [|\ P(a,b);\ P(b,c);\ a \le b;\ b \le c|] ==> P(a,c)$
and *notP*: $\sim P(a,b)$
and *le*: $a \le b$
shows $\sim P(fst(Bolzano\text{-}bisect\ P\ a\ b\ n),\ snd(Bolzano\text{-}bisect\ P\ a\ b\ n))$
⟨*proof*⟩

lemma *not-P-Bolzano-bisect'*:

[| $\forall\, a\ b\ c.\ P(a,b)\ \&\ P(b,c)\ \&\ a \leq b\ \&\ b \leq c\ {-}{-}{>}\ P(a,c)$;
$\sim\ P(a,b)$; $a \leq b$ |] $==>$
$\forall\, n. \sim P(fst(Bolzano\text{-}bisect\ P\ a\ b\ n),\ snd(Bolzano\text{-}bisect\ P\ a\ b\ n))$
⟨*proof*⟩

lemma *lemma-BOLZANO*:
[| $\forall\, a\ b\ c.\ P(a,b)\ \&\ P(b,c)\ \&\ a \leq b\ \&\ b \leq c\ {-}{-}{>}\ P(a,c)$;
$\forall\, x.\ \exists\, d{::}real.\ 0 < d\ \&$
$(\forall\, a\ b.\ a \leq x\ \&\ x \leq b\ \&\ (b-a) < d\ {-}{-}{>}\ P(a,b))$;
$a \leq b$ |]
$==>$ $P(a,b)$
⟨*proof*⟩

lemma *lemma-BOLZANO2*: $((\forall\, a\ b\ c.\ (a \leq b\ \&\ b \leq c\ \&\ P(a,b)\ \&\ P(b,c))\ {-}{-}{>}$
$P(a,c))\ \&$
$(\forall\, x.\ \exists\, d{::}real.\ 0 < d\ \&$
$(\forall\, a\ b.\ a \leq x\ \&\ x \leq b\ \&\ (b-a) < d\ {-}{-}{>}\ P(a,b))))$
${-}{-}{>}\ (\forall\, a\ b.\ a \leq b\ {-}{-}{>}\ P(a,b))$
⟨*proof*⟩

20.4 Intermediate Value Theorem: Prove Contrapositive by Bisection

lemma *IVT*: [| $f(a) \leq y$; $y \leq f(b)$;
$a \leq b$;
$(\forall\, x.\ a \leq x\ \&\ x \leq b\ {-}{-}{>}\ isCont\ f\ x)$ |]
$==>\ \exists\, x.\ a \leq x\ \&\ x \leq b\ \&\ f(x) = y$
⟨*proof*⟩

lemma *IVT2*: [| $f(b) \leq y$; $y \leq f(a)$;
$a \leq b$;
$(\forall\, x.\ a \leq x\ \&\ x \leq b\ {-}{-}{>}\ isCont\ f\ x)$
|] $==>\ \exists\, x.\ a \leq x\ \&\ x \leq b\ \&\ f(x) = y$
⟨*proof*⟩

lemma *IVT-objl*: $(f(a) \leq y\ \&\ y \leq f(b)\ \&\ a \leq b\ \&$
$(\forall\, x.\ a \leq x\ \&\ x \leq b\ {-}{-}{>}\ isCont\ f\ x))$
${-}{-}{>}\ (\exists\, x.\ a \leq x\ \&\ x \leq b\ \&\ f(x) = y)$
⟨*proof*⟩

lemma *IVT2-objl*: $(f(b) \leq y\ \&\ y \leq f(a)\ \&\ a \leq b\ \&$
$(\forall\, x.\ a \leq x\ \&\ x \leq b\ {-}{-}{>}\ isCont\ f\ x))$
${-}{-}{>}\ (\exists\, x.\ a \leq x\ \&\ x \leq b\ \&\ f(x) = y)$
⟨*proof*⟩

20.5 By bisection, function continuous on closed interval is bounded above

lemma *isCont-bounded*:
$[| \ a \le b; \forall x. \ a \le x \ \& \ x \le b \ {-}{-}{>} \ isCont \ f \ x \ |]$
$==> \exists M. \forall x. \ a \le x \ \& \ x \le b \ {-}{-}{>} \ f(x) \le M$
⟨*proof*⟩

Refine the above to existence of least upper bound

lemma *lemma-reals-complete*: $((\exists x. \ x \in S) \ \& \ (\exists y. \ isUb \ UNIV \ S \ (y{::}real))) \ {-}{-}{>}$
$(\exists t. \ isLub \ UNIV \ S \ t)$
⟨*proof*⟩

lemma *isCont-has-Ub*: $[| \ a \le b; \forall x. \ a \le x \ \& \ x \le b \ {-}{-}{>} \ isCont \ f \ x \ |]$
$==> \exists M. (\forall x. \ a \le x \ \& \ x \le b \ {-}{-}{>} \ f(x) \le M) \ \&$
$(\forall N. \ N < M \ {-}{-}{>} \ (\exists x. \ a \le x \ \& \ x \le b \ \& \ N < f(x)))$
⟨*proof*⟩

Now show that it attains its upper bound

lemma *isCont-eq-Ub*:
assumes *le*: $a \le b$
and *con*: $\forall x. \ a \le x \ \& \ x \le b \ {-}{-}{>} \ isCont \ f \ x$
shows $\exists M. (\forall x. \ a \le x \ \& \ x \le b \ {-}{-}{>} \ f(x) \le M) \ \&$
$(\exists x. \ a \le x \ \& \ x \le b \ \& \ f(x) = M)$
⟨*proof*⟩

Same theorem for lower bound

lemma *isCont-eq-Lb*: $[| \ a \le b; \forall x. \ a \le x \ \& \ x \le b \ {-}{-}{>} \ isCont \ f \ x \ |]$
$==> \exists M. (\forall x. \ a \le x \ \& \ x \le b \ {-}{-}{>} \ M \le f(x)) \ \&$
$(\exists x. \ a \le x \ \& \ x \le b \ \& \ f(x) = M)$
⟨*proof*⟩

Another version.

lemma *isCont-Lb-Ub*: $[|a \le b; \forall x. \ a \le x \ \& \ x \le b \ {-}{-}{>} \ isCont \ f \ x \ |]$
$==> \exists L \ M. (\forall x. \ a \le x \ \& \ x \le b \ {-}{-}{>} \ L \le f(x) \ \& \ f(x) \le M) \ \&$
$(\forall y. \ L \le y \ \& \ y \le M \ {-}{-}{>} \ (\exists x. \ a \le x \ \& \ x \le b \ \& \ (f(x) = y)))$
⟨*proof*⟩

20.6 If $(0{::}'a) < f' \ x$ then x is Locally Strictly Increasing At The Right

lemma *DERIV-left-inc*:
assumes *der*: $DERIV \ f \ x :> l$
and *l*: $0 < l$
shows $\exists d > 0. \forall h > 0. \ h < d \ {-}{-}{>} \ f(x) < f(x + h)$
⟨*proof*⟩

lemma *DERIV-left-dec*:
assumes *der*: $DERIV \ f \ x :> l$

and *l*: $l < 0$
shows $\exists\, d > 0.\ \forall\, h > 0.\ h < d \dashrightarrow f(x) < f(x-h)$
⟨*proof*⟩

lemma *DERIV-local-max*:
assumes *der*: $DERIV\ f\ x :> l$
and *d*: $0 < d$
and *le*: $\forall\, y.\ |x-y| < d \dashrightarrow f(y) \le f(x)$
shows $l = 0$
⟨*proof*⟩

Similar theorem for a local minimum

lemma *DERIV-local-min*:
$[|\ DERIV\ f\ x :> l;\ 0 < d;\ \forall\, y.\ |x-y| < d \dashrightarrow f(x) \le f(y)\ |] ==> l = 0$
⟨*proof*⟩

In particular, if a function is locally flat

lemma *DERIV-local-const*:
$[|\ DERIV\ f\ x :> l;\ 0 < d;\ \forall\, y.\ |x-y| < d \dashrightarrow f(x) = f(y)\ |] ==> l = 0$
⟨*proof*⟩

Lemma about introducing open ball in open interval

lemma *lemma-interval-lt*:
$[|\ a < x;\ \ x < b\ |]$
$==> \exists\, d{::}real.\ 0 < d\ \&\ (\forall\, y.\ |x-y| < d \dashrightarrow a < y\ \&\ y < b)$
⟨*proof*⟩

lemma *lemma-interval*: $[|\ a < x;\ \ x < b\ |] ==>$
$\exists\, d{::}real.\ 0 < d\ \&\ \ (\forall\, y.\ |x-y| < d \dashrightarrow a \le y\ \&\ y \le b)$
⟨*proof*⟩

Rolle's Theorem. If f is defined and continuous on the closed interval $[a,b]$ and differentiable on the open interval (a,b), and $f\ a = f\ b$, then there exists $x0 \in (a,b)$ such that $f'\ x0 = (0{::}'a)$

theorem *Rolle*:
assumes *lt*: $a < b$
and *eq*: $f(a) = f(b)$
and *con*: $\forall\, x.\ a \le x\ \&\ x \le b \dashrightarrow isCont\ f\ x$
and *dif* [*rule-format*]: $\forall\, x.\ a < x\ \&\ x < b \dashrightarrow f\ differentiable\ x$
shows $\exists\, z.\ a < z\ \&\ z < b\ \&\ DERIV\ f\ z :> 0$
⟨*proof*⟩

20.7 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 $\exists l\ z.\ a < z\ \&\ z < b\ \&\ DERIV\ f\ z :> l\ \&$
$(f(b) - f(a) = (b-a) * l)$
⟨*proof*⟩

A function is constant if its derivative is 0 over an interval.

lemma *DERIV-isconst-end*: $[|\ a < b;$
$\forall x.\ a \leq x\ \&\ x \leq b\ {-}{-}{>}\ isCont\ f\ x;$
$\forall x.\ a < x\ \&\ x < b\ {-}{-}{>}\ DERIV\ f\ x :> 0\ |]$
$=={>}\ f\ b = f\ a$
⟨*proof*⟩

lemma *DERIV-isconst1*: $[|\ a < b;$
$\forall x.\ a \leq x\ \&\ x \leq b\ {-}{-}{>}\ isCont\ f\ x;$
$\forall x.\ a < x\ \&\ x < b\ {-}{-}{>}\ DERIV\ f\ x :> 0\ |]$
$=={>}\ \forall x.\ a \leq x\ \&\ x \leq b\ {-}{-}{>}\ f\ x = f\ a$
⟨*proof*⟩

lemma *DERIV-isconst2*: $[|\ a < b;$
$\forall x.\ a \leq x\ \&\ x \leq b\ {-}{-}{>}\ isCont\ f\ x;$
$\forall x.\ a < x\ \&\ x < b\ {-}{-}{>}\ DERIV\ f\ x :> 0;$
$a \leq x;\ x \leq b\ |]$
$=={>}\ f\ x = f\ a$
⟨*proof*⟩

lemma *DERIV-isconst-all*: $\forall x.\ DERIV\ f\ x :> 0 =={>} f(x) = f(y)$
⟨*proof*⟩

lemma *DERIV-const-ratio-const*:
$[|a \neq b;\ \forall x.\ DERIV\ f\ x :> k\ |] =={>} (f(b) - f(a)) = (b-a) * k$
⟨*proof*⟩

lemma *DERIV-const-ratio-const2*:
$[|a \neq b;\ \forall x.\ DERIV\ f\ x :> k\ |] =={>} (f(b) - f(a))/(b-a) = k$
⟨*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*:
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*:
assumes d: $0 < d$
and *inj* [*rule-format*]: $\forall z.\ |z-x| \leq d \dashrightarrow g(f\ z) = z$
and *cont*: $\forall z.\ |z-x| \leq d \dashrightarrow isCont\ f\ z$
shows $\exists z.\ |z-x| \leq d\ \&\ f\ x < f\ z$
⟨*proof*⟩

Similar version for lower bound.

lemma *lemma-isCont-inj2*:
$[|0 < d; \forall z.\ |z-x| \leq d \dashrightarrow g(f\ z) = z;$
$\forall z.\ |z-x| \leq d \dashrightarrow isCont\ f\ z\ |]$
$\Longrightarrow \exists z.\ |z-x| \leq d\ \&\ f\ z < f\ x$
⟨*proof*⟩

Show there's an interval surrounding $f\ x$ in $f[[x - d, x + d]]$.

lemma *isCont-inj-range*:
assumes d: $0 < d$
and *inj*: $\forall z.\ |z-x| \leq d \dashrightarrow g(f\ z) = z$
and *cont*: $\forall z.\ |z-x| \leq d \dashrightarrow isCont\ f\ z$
shows $\exists e>0.\ \forall y.\ |y - f\ x| \leq e \dashrightarrow (\exists z.\ |z-x| \leq d\ \&\ f\ z = y)$
⟨*proof*⟩

Continuity of inverse function

lemma *isCont-inverse-function*:
assumes d: $0 < d$
and *inj*: $\forall z.\ |z-x| \leq d \dashrightarrow g(f\ z) = z$
and *cont*: $\forall z.\ |z-x| \leq d \dashrightarrow isCont\ f\ z$
shows *isCont g* $(f\ x)$
⟨*proof*⟩

⟨*ML*⟩

end

21 Series: Finite Summation and Infinite Series

theory *Series*
imports *SEQ Lim*
begin

declare *atLeastLessThan-iff* [*iff*]

declare *setsum-op-ivl-Suc*[*simp*]

constdefs
sums :: (*nat* => *real*) => *real* => *bool* (**infixr** *sums 80*)
f sums s == (%*n. setsum f* {*0*..<*n*}) −−−−> *s*

summable :: (*nat*=>*real*) => *bool*
summable f == ($\exists$ *s. f sums s*)

suminf :: (*nat*=>*real*) => *real*
suminf f == *SOME s. f sums s*

syntax
-suminf :: *idt* => *real* => *real* ($\sum$-. - [*0*, *10*] *10*)
translations
$\sum$ *i. b* == *suminf* (%*i. b*)

lemma *sumr-diff-mult-const*:
setsum f {*0*..<*n*} − (*real n***r*) = *setsum* (%*i. f i* − *r*) {*0*..<*n*::*nat*}
⟨*proof*⟩

lemma *real-setsum-nat-ivl-bounded*:
(!!*p. p* < *n* $\Longrightarrow$ *f*(*p*) $\leq$ *K*)
$\Longrightarrow$ *setsum f* {*0*..<*n*::*nat*} $\leq$ *real n* * *K*
⟨*proof*⟩

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

lemma *sumr-one-lb-realpow-zero* [*simp*]:
($\sum$ *n*=*Suc 0*..<*n*. *f*(*n*) * (*0*::*real*) ˆ *n*) = *0*
⟨*proof*⟩

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

lemma *sumr-offset*:
($\sum$ *m*=*0*..<*n*::*nat*. *f*(*m*+*k*)::*real*) = *setsum f* {*0*..<*n*+*k*} − *setsum f* {*0*..<*k*}
⟨*proof*⟩

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

lemma *sumr-offset3*:
  setsum f $\{0::nat..<n+k\} = (\sum m=0..<n.\ f\ (m+k)::real) + setsum\ f\ \{0..<k\}$
⟨*proof*⟩

lemma *sumr-offset4*:
 $\forall n\, f.\ setsum\ f\ \{0::nat..<n+k\} =$
  $(\sum m=0..<n.\ f\ (m+k)::real) + setsum\ f\ \{0..<k\}$
⟨*proof*⟩

21.1 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*:
  $(\forall m.\ n \le m\ {-}{-}{>}\ f(m) = 0)$ *==> f sums (setsum f* $\{0..<n\}$*)*
⟨*proof*⟩

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

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

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

lemma *sums-mult*: *f sums a* ==> (%*n*. *c* ∗ *f n*) *sums* (*c* ∗ *a*)
⟨*proof*⟩

lemma *summable-mult*: *summable f* ==> *summable* (%*n*. *c* ∗ *f n*)
⟨*proof*⟩

lemma *suminf-mult*: *summable f* ==> *suminf* (%*n*. *c* ∗ *f n*) = *c* ∗ *suminf f*
⟨*proof*⟩

lemma *sums-mult2*: *f sums a* ==> (%*n*. *f n* ∗ *c*) *sums* (*a* ∗ *c*)
⟨*proof*⟩

lemma *summable-mult2*: *summable f* ==> *summable* (%*n*. *f n* ∗ *c*)
⟨*proof*⟩

lemma *suminf-mult2*: *summable f* ==> *suminf f* ∗ *c* = ($\sum$ *n*. *f n* ∗ *c*)
⟨*proof*⟩

lemma *sums-divide*: *f sums a* ==> (%*n*. (*f n*)/*c*) *sums* (*a*/*c*)
⟨*proof*⟩

lemma *summable-divide*: *summable f* ==> *summable* (%*n*. (*f n*) / *c*)
⟨*proof*⟩

lemma *suminf-divide*: *summable f* ==> *suminf* (%*n*. (*f n*) / *c*) = (*suminf f*) / *c*
⟨*proof*⟩

lemma *sums-add*: [| *x sums x0*; *y sums y0* |] ==> (%*n*. *x n* + *y n*) *sums* (*x0*+*y0*)
⟨*proof*⟩

lemma *summable-add*: *summable f* ==> *summable g* ==> *summable* (%*x*. *f x* + *g x*)
⟨*proof*⟩

lemma *suminf-add*:
[| *summable f*; *summable g* |]
==> *suminf f* + *suminf g* = ($\sum$ *n*. *f n* + *g n*)
⟨*proof*⟩

lemma *sums-diff*: [| *x sums x0*; *y sums y0* |] ==> (%*n*. *x n* − *y n*) *sums* (*x0*−*y0*)
⟨*proof*⟩

lemma *summable-diff*: *summable f* ==> *summable g* ==> *summable* (%*x*. *f x* −

$g\ x)$
⟨*proof*⟩

lemma *suminf-diff*:
$[|\ summable\ f;\ summable\ g\ |]$
$==>\ suminf\ f\ -\ suminf\ g\ =\ (\sum n.\ f\ n\ -\ g\ n)$
⟨*proof*⟩

lemma *sums-minus*: $f\ sums\ s\ ==>\ (\%x.\ -\ f\ x)\ sums\ (-\ s)$
⟨*proof*⟩

lemma *summable-minus*: $summable\ f\ ==>\ summable\ (\%x.\ -\ f\ x)$
⟨*proof*⟩

lemma *suminf-minus*: $summable\ f\ ==>\ suminf\ (\%x.\ -\ f\ x) = -\ (suminf\ f)$
⟨*proof*⟩

lemma *sums-group*:
$[|summable\ f;\ 0\ <\ k\ |]\ ==>\ (\%n.\ setsum\ f\ \{n*k..<n*k+k\})\ sums\ (suminf\ f)$
⟨*proof*⟩

lemma *sumr-pos-lt-pair-lemma*:
$[|\forall\ d.\ -\ f\ (n\ +\ (d\ +\ d))\ <\ (f\ (Suc\ (n\ +\ (d\ +\ d)))\ ::\ real)\ |]$
$==>\ setsum\ f\ \{0..<n+Suc(Suc\ 0)\}\ \leq\ setsum\ f\ \{0..<Suc(Suc\ 0)\ *\ Suc\ no\ +\ n\}$
⟨*proof*⟩

lemma *sumr-pos-lt-pair*:
$[|summable\ f;$
$\forall\ d.\ 0\ <\ (f(n\ +\ (Suc(Suc\ 0)\ *\ d)))\ +\ f(n\ +\ ((Suc(Suc\ 0)\ *\ d)\ +\ 1))|]$
$==>\ setsum\ f\ \{0..<n\}\ <\ suminf\ f$
⟨*proof*⟩

A summable series of positive terms has limit that is at least as great as any partial sum.

lemma *series-pos-le*:
$[|\ summable\ f;\ \forall\ m\ \geq\ n.\ 0\ \leq\ f(m)\ |]\ ==>\ setsum\ f\ \{0..<n\}\ \leq\ suminf\ f$
⟨*proof*⟩

lemma *series-pos-less*:
$[|\ summable\ f;\ \forall\ m\ \geq\ n.\ 0\ <\ f(m)\ |]\ ==>\ setsum\ f\ \{0..<n\}\ <\ suminf\ f$
⟨*proof*⟩

Sum of a geometric progression.

lemmas *sumr-geometric* = *geometric-sum* [**where** *'a* = *real*]

lemma *geometric-sums*: $abs(x)\ <\ 1\ ==>\ (\%n.\ x\ \hat{}\ n)\ sums\ (1/(1\ -\ x))$
⟨*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-Cauchy*:
summable f =
$(\forall\, e > 0.\ \exists\, N.\ \forall\, m \geq N.\ \forall\, n.\ abs(setsum\ f\ \{m..{<}n\}) < e)$
⟨*proof*⟩

Comparison test

lemma *summable-comparison-test*:
$[|\ \exists\, N.\ \forall\, n \geq N.\ abs(f\ n) \leq g\ n;\ summable\ g\ |] ==> summable\ f$
⟨*proof*⟩

lemma *summable-rabs-comparison-test*:
$[|\ \exists\, N.\ \forall\, n \geq N.\ abs(f\ n) \leq g\ n;\ summable\ g\ |]$
$==> summable\ (\%k.\ abs\ (f\ k))$
⟨*proof*⟩

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

lemma *summable-le*:
$[|\forall\, n.\ f\ n \leq g\ n;\ summable\ f;\ summable\ g\ |] ==> suminf\ f \leq suminf\ g$
⟨*proof*⟩

lemma *summable-le2*:
$[|\forall\, n.\ abs(f\ n) \leq g\ n;\ summable\ g\ |]$
$==> summable\ f\ \&\ suminf\ f \leq suminf\ g$
⟨*proof*⟩

Absolute convergence imples normal convergence

lemma *summable-rabs-cancel*: $summable\ (\%n.\ abs\ (f\ n)) ==> summable\ f$
⟨*proof*⟩

Absolute convergence of series

lemma *summable-rabs*:
$summable\ (\%n.\ abs\ (f\ n)) ==> abs(suminf\ f) \leq (\sum n.\ abs(f\ n))$
⟨*proof*⟩

21.2 The Ratio Test

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*:
$[|\ \forall n \geq N.\ abs(f(Suc\ n)) \leq c*abs(f\ n)\ |]$
$==> 0 < c \mid summable\ f$
⟨*proof*⟩

lemma *ratio-test*:
$[|\ c < 1; \forall n \geq N.\ abs(f(Suc\ n)) \leq c*abs(f\ n)\ |]$
$==> summable\ f$
⟨*proof*⟩

Differentiation of finite sum

lemma *DERIV-sumr* [*rule-format* (*no-asm*)]:
$(\forall r.\ m \leq r\ \&\ r < (m + n) --> DERIV\ (\%x.\ f\ r\ x)\ x :> (f'\ r\ x))$
$--> DERIV\ (\%x.\ \sum n{=}m..{<}n{::}nat.\ f\ n\ x)\ x :> (\sum r{=}m..{<}n.\ f'\ r\ x)$
⟨*proof*⟩

⟨*ML*⟩

end

22 HSeries: Finite Summation and Infinite Series for Hyperreals

theory *HSeries*
imports *Series*
begin

constdefs
sumhr :: (*hypnat* ∗ *hypnat* ∗ (*nat*=>*real*)) => *hypreal*
sumhr ==
%(*M*,*N*,*f*). *starfun2* (%*m n*. *setsum f* {*m*..<*n*}) *M N*

NSsums :: [*nat*=>*real*,*real*] => *bool* (**infixr** *NSsums 80*)
f NSsums s == (%*n*. *setsum f* {*0*..<*n*}) −−−−*NS*> *s*

NSsummable :: (*nat*=>*real*) => *bool*
NSsummable f == (∃ *s*. *f NSsums s*)

NSsuminf :: (*nat*=>*real*) => *real*
NSsuminf f == (@*s*. *f NSsums s*)

lemma *sumhr*:
sumhr(*star-n M*, *star-n N*, *f*) =

star-n (%*n*. *setsum f* {*M n*..<*N n*})
⟨*proof*⟩

Base case in definition of *sumr*

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

Recursive case in definition of *sumr*

lemma *sumhr-if*:
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*]: *sumhr* (*n* + *1*, *n*, *f*) = *0*
⟨*proof*⟩

lemma *sumhr-eq-bounds* [*simp*]: *sumhr* (*n*,*n*,*f*) = *0*
⟨*proof*⟩

lemma *sumhr-Suc* [*simp*]: *sumhr* (*m*,*m* + *1*,*f*) = (**f* f*) *m*
⟨*proof*⟩

lemma *sumhr-add-lbound-zero* [*simp*]: *sumhr*(*m*+*k*,*k*,*f*) = *0*
⟨*proof*⟩

lemma *sumhr-add*: *sumhr* (*m*,*n*,*f*) + *sumhr*(*m*,*n*,*g*) = *sumhr*(*m*,*n*,%*i*. *f i* + *g i*)
⟨*proof*⟩

lemma *sumhr-mult*: *hypreal-of-real r* ∗ *sumhr*(*m*,*n*,*f*) = *sumhr*(*m*,*n*,%*n*. *r* ∗ *f n*)
⟨*proof*⟩

lemma *sumhr-split-add*: *n* < *p* ==> *sumhr*(*0*,*n*,*f*) + *sumhr*(*n*,*p*,*f*) = *sumhr*(*0*,*p*,*f*)
⟨*proof*⟩

lemma *sumhr-split-diff*: *n*<*p* ==> *sumhr*(*0*,*p*,*f*) − *sumhr*(*0*,*n*,*f*) = *sumhr*(*n*,*p*,*f*)
⟨*proof*⟩

lemma *sumhr-hrabs*: *abs*(*sumhr*(*m*,*n*,*f*)) ≤ *sumhr*(*m*,*n*,%*i*. *abs*(*f i*))
⟨*proof*⟩

other general version also needed

lemma *sumhr-fun-hypnat-eq*:
(∀ *r*. *m* ≤ *r* & *r* < *n* −−> *f r* = *g r*) −−>
sumhr(*hypnat-of-nat m*, *hypnat-of-nat n*, *f*) =
sumhr(*hypnat-of-nat m*, *hypnat-of-nat n*, *g*)
⟨*proof*⟩

lemma *sumhr-const*:

$sumhr(0,\ n,\ \%i.\ r) = hypreal\text{-}of\text{-}hypnat\ n * hypreal\text{-}of\text{-}real\ r$
⟨proof⟩

lemma *sumhr-less-bounds-zero* [*simp*]: $n < m ==> sumhr(m,n,f) = 0$
⟨proof⟩

lemma *sumhr-minus*: $sumhr(m,\ n,\ \%i.\ -\ f\ i) = -\ sumhr(m,\ n,\ f)$
⟨proof⟩

lemma *sumhr-shift-bounds*:
$sumhr(m+hypnat\text{-}of\text{-}nat\ k,n+hypnat\text{-}of\text{-}nat\ k,f) = sumhr(m,n,\%i.\ f(i + k))$
⟨proof⟩

22.1 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*]:
$sumhr(0,\ whn + whn,\ \%i.\ (-1)\ \hat{}\ (i+1)) = 0$
⟨proof⟩

lemma *sumhr-interval-const*:
$(\forall n.\ m \le Suc\ n\ --> f\ n = r)\ \&\ m \le na$
$==> sumhr(hypnat\text{-}of\text{-}nat\ m,hypnat\text{-}of\text{-}nat\ na,f) =$
$(hypreal\text{-}of\text{-}nat\ (na - m) * hypreal\text{-}of\text{-}real\ r)$
⟨proof⟩

lemma *starfunNat-sumr*: $(\ {*f*}\ (\%n.\ setsum\ f\ \{0..<n\}))\ N = sumhr(0,N,f)$
⟨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*:
∀ *m*. *n* ≤ *Suc m* --> *f*(*m*) = *0* ==> *f NSsums* (*setsum f* {*0*..<*n*})
⟨proof⟩

lemma *NSsummable-NSCauchy*:
NSsummable f =
(∀ *M* ∈ *HNatInfinite*. ∀ *N* ∈ *HNatInfinite*. *abs* (*sumhr*(*M*,*N*,*f*)) @= *0*)
⟨proof⟩

Terms of a convergent series tend to zero

lemma *NSsummable-NSLIMSEQ-zero*: *NSsummable f* ==> *f* ----NS> *0*
⟨proof⟩

Easy to prove stsandard case now

lemma *summable-LIMSEQ-zero*: *summable f* ==> *f* ----> *0*
⟨proof⟩

Nonstandard comparison test

lemma *NSsummable-comparison-test*:
[| ∃ *N*. ∀ *n*. *N* ≤ *n* --> *abs*(*f n*) ≤ *g n*; *NSsummable g* |] ==> *NSsummable*
f
⟨proof⟩

lemma *NSsummable-rabs-comparison-test*:
[| ∃ *N*. ∀ *n*. *N* ≤ *n* --> *abs*(*f n*) ≤ *g n*; *NSsummable g* |]
==> *NSsummable* (%*k*. *abs* (*f k*))
⟨proof⟩

⟨ML⟩

end

23 NthRoot: Existence of Nth Root

theory *NthRoot*
imports *SEQ HSeries*

begin

Various lemmas needed for this result. We follow the proof given by John Lindsay Orr (`jorr@math.unl.edu`) in his Analysis Webnotes available at http://www.math.unl.edu/~webnotes.

Lemmas about sequences of reals are used to reach the result.

lemma *lemma-nth-realpow-non-empty*:
$[|\ (0::real) < a;\ 0 < n\ |] ==> \exists s.\ s : \{x.\ x\ \hat{}\ n <= a\ \&\ 0 < x\}$
$\langle proof \rangle$

Used only just below

lemma *realpow-ge-self2*: $[|\ (1::real) \leq r;\ 0 < n\ |] ==> r \leq r\ \hat{}\ n$
$\langle proof \rangle$

lemma *lemma-nth-realpow-isUb-ex*:
$[|\ (0::real) < a;\ 0 < n\ |]$
$==> \exists u.\ isUb\ (UNIV::real\ set)\ \{x.\ x\ \hat{}\ n <= a\ \&\ 0 < x\}\ u$
$\langle proof \rangle$

lemma *nth-realpow-isLub-ex*:
$[|\ (0::real) < a;\ 0 < n\ |]$
$==> \exists u.\ isLub\ (UNIV::real\ set)\ \{x.\ x\ \hat{}\ n <= a\ \&\ 0 < x\}\ u$
$\langle proof \rangle$

23.1 First Half – Lemmas First

lemma *lemma-nth-realpow-seq*:
$isLub\ (UNIV::real\ set)\ \{x.\ x\ \hat{}\ n <= a\ \&\ (0::real) < x\}\ u$
$==> u + inverse(real\ (Suc\ k))\ \tilde{}\!:\ \{x.\ x\ \hat{}\ n <= a\ \&\ 0 < x\}$
$\langle proof \rangle$

lemma *lemma-nth-realpow-isLub-gt-zero*:
$[|\ isLub\ (UNIV::real\ set)\ \{x.\ x\ \hat{}\ n <= a\ \&\ (0::real) < x\}\ u;$
$0 < a;\ 0 < n\ |] ==> 0 < u$
$\langle proof \rangle$

lemma *lemma-nth-realpow-isLub-ge*:
$[|\ isLub\ (UNIV::real\ set)\ \{x.\ x\ \hat{}\ n <= a\ \&\ (0::real) < x\}\ u;$
$0 < a;\ 0 < n\ |] ==> ALL\ k.\ a <= (u + inverse(real\ (Suc\ k)))\ \hat{}\ n$
$\langle proof \rangle$

First result we want

lemma *realpow-nth-ge*:
$[|\ (0::real) < a;\ 0 < n;$
$isLub\ (UNIV::real\ set)$
$\{x.\ x\ \hat{}\ n <= a\ \&\ 0 < x\}\ u\ |] ==> a <= u\ \hat{}\ n$
$\langle proof \rangle$

23.2 Second Half

lemma *less-isLub-not-isUb*:
  [| *isLub* (*UNIV*::*real set*) *S u*; $x < u$ |]
    ==> ~ *isUb* (*UNIV*::*real set*) *S x*
⟨*proof*⟩

lemma *not-isUb-less-ex*:
  ~ *isUb* (*UNIV*::*real set*) *S u* ==> $\exists x \in S.\ u < x$
⟨*proof*⟩

lemma *real-mult-less-self*: *0* < *r* ==> *r* ∗ (*1* + −*inverse*(*real* (*Suc n*))) < *r*
⟨*proof*⟩

lemma *real-mult-add-one-minus-ge-zero*:
  0 < *r* ==> *0* <= *r*∗(*1* + −*inverse*(*real* (*Suc n*)))
⟨*proof*⟩

lemma *lemma-nth-realpow-isLub-le*:
  [| *isLub* (*UNIV*::*real set*) {*x*. *x* ˆ *n* <= *a* & (*0*::*real*) < *x*} *u*;
   0 < *a*; *0* < *n* |] ==> *ALL k*. (*u*∗(*1* + −*inverse*(*real* (*Suc k*)))) ˆ *n* <= *a*
⟨*proof*⟩

Second result we want

lemma *realpow-nth-le*:
  [| (*0*::*real*) < *a*; *0* < *n*;
  isLub (*UNIV*::*real set*)
  {*x*. *x* ˆ *n* <= *a* & *0* < *x*} *u* |] ==> *u* ˆ *n* <= *a*
⟨*proof*⟩

The theorem at last!

lemma *realpow-nth*: [| (*0*::*real*) < *a*; *0* < *n* |] ==> ∃ *r*. *r* ˆ *n* = *a*
⟨*proof*⟩

lemma *realpow-pos-nth*: [| (*0*::*real*) < *a*; *0* < *n* |] ==> ∃ *r*. *0* < *r* & *r* ˆ *n* = *a*
⟨*proof*⟩

lemma *realpow-pos-nth2*: (*0*::*real*) < *a* ==> ∃ *r*. *0* < *r* & *r* ˆ *Suc n* = *a*
⟨*proof*⟩

lemma *realpow-pos-nth-unique*:
  [| (*0*::*real*) < *a*; *0* < *n* |] ==> *EX*! *r*. *0* < *r* & *r* ˆ *n* = *a*
⟨*proof*⟩

⟨*ML*⟩

end

24 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*]: $0 < \mathit{fact}\ n$
⟨*proof*⟩

lemma *fact-not-eq-zero* [*simp*]: $\mathit{fact}\ n \neq 0$
⟨*proof*⟩

lemma *real-of-nat-fact-not-zero* [*simp*]: $\mathit{real}\ (\mathit{fact}\ n) \neq 0$
⟨*proof*⟩

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

lemma *real-of-nat-fact-ge-zero* [*simp*]: $0 \leq \mathit{real}(\mathit{fact}\ n)$
⟨*proof*⟩

lemma *fact-ge-one* [*simp*]: $1 \leq \mathit{fact}\ n$
⟨*proof*⟩

lemma *fact-mono*: $m \leq n$ ==> $\mathit{fact}\ m \leq \mathit{fact}\ n$
⟨*proof*⟩

Note that *fact 0 = fact 1*

lemma *fact-less-mono*: [| $0 < m$; $m < n$ |] ==> $\mathit{fact}\ m < \mathit{fact}\ n$
⟨*proof*⟩

lemma *inv-real-of-nat-fact-gt-zero* [*simp*]: $0 < \mathit{inverse}\ (\mathit{real}\ (\mathit{fact}\ n))$
⟨*proof*⟩

lemma *inv-real-of-nat-fact-ge-zero* [*simp*]: $0 \leq \mathit{inverse}\ (\mathit{real}\ (\mathit{fact}\ n))$
⟨*proof*⟩

lemma *fact-diff-Suc* [*rule-format*]:
$\forall m.\ n < \mathit{Suc}\ m$ --> $\mathit{fact}\ (\mathit{Suc}\ m - n) = (\mathit{Suc}\ m - n) * \mathit{fact}\ (m - n)$
⟨*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

25 EvenOdd: Even and Odd Numbers: Compatibility file for Parity

theory *EvenOdd*
imports *NthRoot*
begin

25.1 General Lemmas About Division

lemma *Suc-times-mod-eq*: *1<k ==> Suc (k ∗ m) mod k = 1*
⟨*proof*⟩

declare *Suc-times-mod-eq* [*of number-of w*, *standard*, *simp*]

lemma [*simp*]: *n div k ≤ (Suc n) div k*
⟨*proof*⟩

lemma *Suc-n-div-2-gt-zero* [*simp*]: *(0::nat) < n ==> 0 < (n + 1) div 2*
⟨*proof*⟩

lemma *div-2-gt-zero* [*simp*]: *(1::nat) < n ==> 0 < n div 2*
⟨*proof*⟩

lemma *mod-mult-self3* [*simp*]: *(k∗n + m) mod n = m mod (n::nat)*
⟨*proof*⟩

lemma *mod-mult-self4* [*simp*]: *Suc (k∗n + m) mod n = Suc m mod n*
⟨*proof*⟩

lemma *mod-Suc-eq-Suc-mod*: *Suc m mod n = Suc (m mod n) mod n*
⟨*proof*⟩

25.2 More Even/Odd Results

lemma *even-mult-two-ex*: $even(n) = (\exists\, m{::}nat.\ n = 2{*}m)$
⟨*proof*⟩

lemma *odd-Suc-mult-two-ex*: $odd(n) = (\exists\, m.\ n = Suc\ (2{*}m))$
⟨*proof*⟩

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

lemma *odd-add* [*simp*]: $odd(m + n{::}nat) = (odd\ m \neq odd\ n)$
⟨*proof*⟩

lemma *lemma-even-div2* [*simp*]: $even\ (n{::}nat) ==> (n + 1)\ div\ 2 = n\ div\ 2$
⟨*proof*⟩

lemma *lemma-not-even-div2* [*simp*]: $\sim even\ n ==> (n + 1)\ div\ 2 = Suc\ (n\ div\ 2)$
⟨*proof*⟩

lemma *even-num-iff*: $0 < n ==> even\ n = (\sim\ even(n - 1 :: nat))$
⟨*proof*⟩

lemma *even-even-mod-4-iff*: $even\ (n{::}nat) = even\ (n\ mod\ 4)$
⟨*proof*⟩

lemma *lemma-odd-mod-4-div-2*: $n\ mod\ 4 = (3{::}nat) ==> odd((n - 1)\ div\ 2)$
⟨*proof*⟩

lemma *lemma-even-mod-4-div-2*: $n\ mod\ 4 = (1{::}nat) ==> even\ ((n - 1)\ div\ 2)$
⟨*proof*⟩

⟨*ML*⟩

end

26 Transcendental: Power Series, Transcendental Functions etc.

theory *Transcendental*
imports *NthRoot Fact HSeries EvenOdd Lim*
begin

constdefs
 $root :: [nat,real] => real$
 $root\ n\ x == (@u.\ ((0{::}real) < x --> 0 < u)\ \&\ (u\ \hat{}\ n = x))$

sqrt :: *real* => *real*
sqrt x == *root 2 x*

exp :: *real* => *real*
exp x == $\sum$ *n. inverse*(*real* (*fact n*)) $*$ (*x* ^ *n*)

sin :: *real* => *real*
sin x == $\sum$ *n.* (*if even*(*n*) *then 0 else*
((− *1*) ^ ((*n* − *Suc 0*) *div 2*))/(*real* (*fact n*))) $*$ *x* ^ *n*

diffs :: (*nat* => *real*) => *nat* => *real*
diffs c == (%*n. real* (*Suc n*) $*$ *c*(*Suc n*))

cos :: *real* => *real*
cos x == $\sum$ *n.* (*if even*(*n*) *then* ((− *1*) ^ (*n div 2*))/(*real* (*fact n*))
else 0) $*$ *x* ^ *n*

ln :: *real* => *real*
ln x == (@*u. exp u* = *x*)

pi :: *real*
pi == *2* $*$ (@*x. 0* ≤ (*x*::*real*) & *x* ≤ *2* & *cos x* = *0*)

tan :: *real* => *real*
tan x == (*sin x*)/(*cos x*)

arcsin :: *real* => *real*
arcsin y == (@*x.* −(*pi*/*2*) ≤ *x* & *x* ≤ *pi*/*2* & *sin x* = *y*)

arcos :: *real* => *real*
arcos y == (@*x. 0* ≤ *x* & *x* ≤ *pi* & *cos x* = *y*)

arctan :: *real* => *real*
arctan y == (@*x.* −(*pi*/*2*) < *x* & *x* < *pi*/*2* & *tan x* = *y*)

lemma *real-root-zero* [*simp*]: *root* (*Suc n*) *0* = *0*
⟨*proof*⟩

lemma *real-root-pow-pos*:
0 < *x* ==> (*root*(*Suc n*) *x*) ^ (*Suc n*) = *x*
⟨*proof*⟩

lemma *real-root-pow-pos2*: *0* ≤ *x* ==> (*root*(*Suc n*) *x*) ^ (*Suc n*) = *x*
⟨*proof*⟩

lemma *real-root-pos*:
0 < *x* ==> *root*(*Suc n*) (*x* ^ (*Suc n*)) = *x*

⟨*proof*⟩

lemma *real-root-pos2*: $0 \le x \implies root(Suc\ n)\ (x\ \hat{}\ (Suc\ n)) = x$
⟨*proof*⟩

lemma *real-root-pos-pos*:
$0 < x \implies 0 \le root(Suc\ n)\ x$
⟨*proof*⟩

lemma *real-root-pos-pos-le*: $0 \le x \implies 0 \le root(Suc\ n)\ x$
⟨*proof*⟩

lemma *real-root-one* [*simp*]: $root\ (Suc\ n)\ 1 = 1$
⟨*proof*⟩

26.1 Square Root

needed because 2 is a binary numeral!

lemma *root-2-eq* [*simp*]: $root\ 2 = root\ (Suc\ (Suc\ 0))$
⟨*proof*⟩

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

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

lemma *real-sqrt-pow2-iff* [*iff*]: $((sqrt\ x)^2 = x) = (0 \le x)$
⟨*proof*⟩

lemma [*simp*]: $(sqrt(u2^2 + v2^2))^2 = u2^2 + v2^2$
⟨*proof*⟩

lemma *real-sqrt-pow2* [*simp*]: $0 \le x \implies (sqrt\ x)^2 = x$
⟨*proof*⟩

lemma *real-sqrt-abs-abs* [*simp*]: $sqrt|x|\ \hat{}\ 2 = |x|$
⟨*proof*⟩

lemma *real-pow-sqrt-eq-sqrt-pow*:
$0 \le x \implies (sqrt\ x)^2 = sqrt(x^2)$
⟨*proof*⟩

lemma *real-pow-sqrt-eq-sqrt-abs-pow2*:
$0 \le x \implies (sqrt\ x)^2 = sqrt(|x|\ \hat{}\ 2)$
⟨*proof*⟩

lemma *real-sqrt-pow-abs*: $0 \le x \implies (sqrt\ x)^2 = |x|$
⟨*proof*⟩

lemma *not-real-square-gt-zero* [*simp*]: $(\tilde{}\ (0::real) < x*x) = (x = 0)$
⟨*proof*⟩

lemma *real-sqrt-gt-zero*: $0 < x ==> 0 < sqrt(x)$
⟨*proof*⟩

lemma *real-sqrt-ge-zero*: $0 \le x ==> 0 \le sqrt(x)$
⟨*proof*⟩

lemma *real-sqrt-mult-self-sum-ge-zero* [*simp*]: $0 \le sqrt(x*x + y*y)$
⟨*proof*⟩

lemma *sqrt-eqI*: $[|r^2 = a;\ 0 \le r|] ==> sqrt\ a = r$
⟨*proof*⟩

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

lemma *real-sqrt-mult-distrib2*:
$[|0 \le x;\ 0 \le y\ |] ==> sqrt(x*y) = sqrt(x) * sqrt(y)$
⟨*proof*⟩

lemma *real-sqrt-sum-squares-ge-zero* [*simp*]: $0 \le sqrt\ (x^2 + y^2)$
⟨*proof*⟩

lemma *real-sqrt-sum-squares-mult-ge-zero* [*simp*]:
$0 \le sqrt\ ((x^2 + y^2)*(xa^2 + ya^2))$
⟨*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-abs* [*simp*]: $sqrt(x^2) = |x|$
⟨*proof*⟩

lemma *real-sqrt-abs2* [*simp*]: $sqrt(x*x) = |x|$
⟨*proof*⟩

lemma *real-sqrt-pow2-gt-zero*: $0 < x ==> 0 < (sqrt\ x)^2$
⟨*proof*⟩

lemma *real-sqrt-not-eq-zero*: $0 < x ==> sqrt\ x \neq 0$
⟨*proof*⟩

lemma *real-inv-sqrt-pow2*: $0 < x ==> inverse\ (sqrt(x))\ \hat{}\ 2 = inverse\ x$
⟨*proof*⟩

lemma *real-sqrt-eq-zero-cancel*: $[|\ 0 \leq x;\ sqrt(x) = 0|] ==> x = 0$
⟨*proof*⟩

lemma *real-sqrt-eq-zero-cancel-iff* [*simp*]: $0 \leq x ==> ((sqrt\ x = 0) = (x{=}0))$
⟨*proof*⟩

lemma *real-sqrt-sum-squares-ge1* [*simp*]: $x \leq sqrt(x^2 + y^2)$
⟨*proof*⟩

lemma *real-sqrt-sum-squares-ge2* [*simp*]: $y \leq sqrt(z^2 + y^2)$
⟨*proof*⟩

lemma *real-sqrt-ge-one*: $1 \leq x ==> 1 \leq sqrt\ x$
⟨*proof*⟩

26.2 Exponential Function

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* [*simp*]:
$(if\ even\ n\ then\ 0$
$else\ (-\ 1)\ \hat{}\ ((n - Suc\ 0)\ div\ 2)/(real\ (fact\ n))) * 0\ \hat{}\ n = 0$
⟨*proof*⟩

lemma *lemma-STAR-cos* [*simp*]:
$0 < n -->$
$(-\ 1)\ \hat{}\ (n\ div\ 2)/(real\ (fact\ n)) * 0\ \hat{}\ n = 0$
⟨*proof*⟩

lemma *lemma-STAR-cos1* [*simp*]:
$0 < n -->$

(-1) ^ $(n\ div\ 2)/(real\ (fact\ n)) * 0$ ^ $n = 0$
⟨proof⟩

lemma *lemma-STAR-cos2* [*simp*]:
$(\sum n{=}1..{<}n.$ *if even n then* $(-\ 1)$ ^ $(n\ div\ 2)/(real\ (fact\ n)) *\ \ 0$ ^ n
else $0) = 0$
⟨proof⟩

lemma *exp-converges*: $(\%n.\ inverse\ (real\ (fact\ n)) * x$ ^ $n)$ *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⟩

lemma *lemma-realpow-diff* [*rule-format* (*no-asm*)]:
$p \leq n$ --> y ^ $(Suc\ n - p) = ((y{::}real)$ ^ $(n - p)) * y$
⟨proof⟩

26.3 Properties of Power Series

lemma *lemma-realpow-diff-sumr*:
$(\sum p{=}0..{<}Suc\ n.\ (x$ ^ $p) * y$ ^ $((Suc\ n) - p)) =$
$y * (\sum p{=}0..{<}Suc\ n.\ (x$ ^ $p) * (y$ ^ $(n - p)){::}real)$
⟨proof⟩

lemma *lemma-realpow-diff-sumr2*:
x ^ $(Suc\ n) - y$ ^ $(Suc\ n) =$
$(x - y) * (\sum p{=}0..{<}Suc\ n.\ (x$ ^ $p) * (y$ ^$(n - p)){::}real)$
⟨proof⟩

lemma *lemma-realpow-rev-sumr*:
$(\sum p{=}0..{<}Suc\ n.\ (x$ ^ $p) * (y$ ^ $(n - p))) =$
$(\sum p{=}0..{<}Suc\ n.\ (x$ ^ $(n - p)) * (y$ ^ $p){::}real)$
⟨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*:
[| *summable* $(\%n.\ f(n) * (x$ ^ $n))$; $|z| < |x|$ |]
==> *summable* $(\%n.\ |f(n)| * (z$ ^ $n))$

⟨*proof*⟩

lemma *powser-inside*:
$[| \ summable \ (\%n. \ f(n) * (x \ \hat{} \ n)); \ |z| < |x| \ |]$
$==> summable \ (\%n. \ f(n) * (z \ \hat{} \ n))$
⟨*proof*⟩

26.4 Differentiation of Power Series

Lemma about distributing negation over it

lemma *diffs-minus*: $diffs \ (\%n. - c \ n) = (\%n. - diffs \ c \ n)$
⟨*proof*⟩

Show that we can shift the terms down one

lemma *lemma-diffs*:
$(\sum n=0..<n. \ (diffs \ c)(n) * (x \ \hat{} \ n)) =$
$(\sum n=0..<n. \ real \ n * c(n) * (x \ \hat{} \ (n - Suc \ 0))) +$
$(real \ n * c(n) * x \ \hat{} \ (n - Suc \ 0))$
⟨*proof*⟩

lemma *lemma-diffs2*:
$(\sum n=0..<n. \ real \ n * c(n) * (x \ \hat{} \ (n - Suc \ 0))) =$
$(\sum n=0..<n. \ (diffs \ c)(n) * (x \ \hat{} \ n)) -$
$(real \ n * c(n) * x \ \hat{} \ (n - Suc \ 0))$
⟨*proof*⟩

lemma *diffs-equiv*:
$summable \ (\%n. \ (diffs \ c)(n) * (x \ \hat{} \ n)) ==>$
$(\%n. \ real \ n * c(n) * (x \ \hat{} \ (n - Suc \ 0))) \ sums$
$(\sum n. \ (diffs \ c)(n) * (x \ \hat{} \ n))$
⟨*proof*⟩

26.5 Term-by-Term Differentiability of Power Series

lemma *lemma-termdiff1*:
$(\sum p=0..<m. \ (((z + h) \ \hat{} \ (m - p)) * (z \ \hat{} \ p)) - (z \ \hat{} \ m)) =$
$(\sum p=0..<m. \ (z \ \hat{} \ p) * (((z + h) \ \hat{} \ (m - p)) - (z \ \hat{} \ (m - p)))::real)$
⟨*proof*⟩

lemma *less-add-one*: $m < n ==> (\exists \, d. \ n = m + d + Suc \ 0)$
⟨*proof*⟩

lemma *sumdiff*: $a + b - (c + d) = a - c + b - (d::real)$
⟨*proof*⟩

lemma *lemma-termdiff2*:
$h \neq 0 ==>$

$(((z + h) \ \hat{}\ n) - (z \ \hat{}\ n)) * \mathit{inverse}\ h - \mathit{real}\ n * (z \ \hat{}\ (n - \mathit{Suc}\ 0)) =$
$h * (\sum p{=}0..{<}\ n - \mathit{Suc}\ 0.\ (z \ \hat{}\ p) *$
$(\sum q{=}0..{<}\ (n - \mathit{Suc}\ 0) - p.\ ((z + h) \ \hat{}\ q) * (z \ \hat{}\ (((n - 2) - p) - q))))$
⟨*proof*⟩

lemma *lemma-termdiff3*:
$[|\ h \neq 0;\ |z| \leq K;\ |z + h| \leq K\ |]$
$==> \mathit{abs}\ (((z + h) \ \hat{}\ n - z \ \hat{}\ n) * \mathit{inverse}\ h - \mathit{real}\ n * z \ \hat{}\ (n - \mathit{Suc}\ 0))$
$\leq \mathit{real}\ n * \mathit{real}\ (n - \mathit{Suc}\ 0) * K \ \hat{}\ (n - 2) * |h|$
⟨*proof*⟩

lemma *lemma-termdiff4*:
$[|\ 0 < k;$
$(\forall h.\ 0 < |h|\ \&\ |h| < k \dashrightarrow |f\ h| \leq K * |h|)\ |]$
$==> f\ {-}{-}\ 0\ {-}{-}{>}\ 0$
⟨*proof*⟩

lemma *lemma-termdiff5*:
$[|\ 0 < k;$
$\mathit{summable}\ f;$
$\forall h.\ 0 < |h|\ \&\ |h| < k \dashrightarrow$
$(\forall n.\ \mathit{abs}(g(h)\ (n{::}\mathit{nat})) \leq (f(n) * |h|))\ |]$
$==> (\%h.\ \mathit{suminf}(g\ h))\ {-}{-}\ 0\ {-}{-}{>}\ 0$
⟨*proof*⟩

FIXME: Long proofs

lemma *termdiffs-aux*:
$[|\mathit{summable}\ (\lambda n.\ \mathit{diffs}\ (\mathit{diffs}\ c)\ n * K \ \hat{}\ n);\ |x| < |K|\ |]$
$==> (\lambda h.\ \sum n.\ c\ n *$
$(((x + h) \ \hat{}\ n - x \ \hat{}\ n) * \mathit{inverse}\ h -$
$\mathit{real}\ n * x \ \hat{}\ (n - \mathit{Suc}\ 0)))$
${-}{-}\ 0\ {-}{-}{>}\ 0$
⟨*proof*⟩

lemma *termdiffs*:
$[|\ \mathit{summable}(\%n.\ c(n) * (K \ \hat{}\ n));$
$\mathit{summable}(\%n.\ (\mathit{diffs}\ c)(n) * (K \ \hat{}\ n));$
$\mathit{summable}(\%n.\ (\mathit{diffs}(\mathit{diffs}\ c))(n) * (K \ \hat{}\ n));$
$|x| < |K|\ |]$
$==> \mathit{DERIV}\ (\%x.\ \sum n.\ c(n) * (x \ \hat{}\ n))\ \ x :>$
$(\sum n.\ (\mathit{diffs}\ c)(n) * (x \ \hat{}\ n))$
⟨*proof*⟩

26.6 Formal Derivatives of Exp, Sin, and Cos Series

lemma *exp-fdiffs*:
$\mathit{diffs}\ (\%n.\ \mathit{inverse}(\mathit{real}\ (\mathit{fact}\ n))) = (\%n.\ \mathit{inverse}(\mathit{real}\ (\mathit{fact}\ n)))$
⟨*proof*⟩

lemma *sin-fdiffs*:
$$diffs(\%n.\ if\ even\ n\ then\ 0\ else\ (-\ 1)\ \hat{}\ ((n - Suc\ 0)\ div\ 2)/(real\ (fact\ n))) = (\%n.\ if\ even\ n\ then\ (-\ 1)\ \hat{}\ (n\ div\ 2)/(real\ (fact\ n))\ else\ 0)$$
⟨*proof*⟩

lemma *sin-fdiffs2*:
$$diffs(\%n.\ if\ even\ n\ then\ 0\ else\ (-\ 1)\ \hat{}\ ((n - Suc\ 0)\ div\ 2)/(real\ (fact\ n)))\ n = (if\ even\ n\ then\ (-\ 1)\ \hat{}\ (n\ div\ 2)/(real\ (fact\ n))\ else\ 0)$$
⟨*proof*⟩

lemma *cos-fdiffs*:
$$diffs(\%n.\ if\ even\ n\ then\ (-\ 1)\ \hat{}\ (n\ div\ 2)/(real\ (fact\ n))\ else\ 0) = (\%n.\ -\ (if\ even\ n\ then\ 0\ else\ (-\ 1)\ \hat{}\ ((n - Suc\ 0) div\ 2)/(real\ (fact\ n))))$$
⟨*proof*⟩

lemma *cos-fdiffs2*:
$$diffs(\%n.\ if\ even\ n\ then\ (-\ 1)\ \hat{}\ (n\ div\ 2)/(real\ (fact\ n))\ else\ 0)\ n = -\ (if\ even\ n\ then\ 0\ else\ (-\ 1)\ \hat{}\ ((n - Suc\ 0) div\ 2)/(real\ (fact\ n)))$$
⟨*proof*⟩

Now at last we can get the derivatives of exp, sin and cos

lemma *lemma-sin-minus*:
$$-\ sin\ x = (\sum n.\ -\ ((if\ even\ n\ then\ 0\ else\ (-\ 1)\ \hat{}\ ((n - Suc\ 0)\ div\ 2)/(real\ (fact\ n))) * x\ \hat{}\ n))$$
⟨*proof*⟩

lemma *lemma-exp-ext*: $exp = (\%x.\ \sum n.\ inverse\ (real\ (fact\ n)) * x\ \hat{}\ n)$
⟨*proof*⟩

lemma *DERIV-exp* [*simp*]: $DERIV\ exp\ x :> exp(x)$
⟨*proof*⟩

lemma *lemma-sin-ext*:
$$sin = (\%x.\ \sum n.\ (if\ even\ n\ then\ 0\ else\ (-\ 1)\ \hat{}\ ((n - Suc\ 0)\ div\ 2)/(real\ (fact\ n))) * x\ \hat{}\ n)$$
⟨*proof*⟩

lemma *lemma-cos-ext*:
$cos = (\%x. \sum n.$
$(if\ even\ n\ then\ (-\ 1)\ \hat{}\ (n\ div\ 2)/(real\ (fact\ n))\ else\ 0) *$
$x\ \hat{}\ n)$
⟨*proof*⟩

lemma *DERIV-sin* [*simp*]: $DERIV\ sin\ x :> cos(x)$
⟨*proof*⟩

lemma *DERIV-cos* [*simp*]: $DERIV\ cos\ x :> -sin(x)$
⟨*proof*⟩

26.7 Properties of the Exponential Function

lemma *exp-zero* [*simp*]: $exp\ 0 = 1$
⟨*proof*⟩

lemma *exp-ge-add-one-self-aux*: $0 \leq x ==> (1 + x) \leq exp(x)$
⟨*proof*⟩

lemma *exp-gt-one* [*simp*]: $0 < x ==> 1 < exp\ x$
⟨*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*]: $exp(x + y)*exp(-x) = exp(y)$
⟨*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*⟩

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

Proof: because every exponential can be seen as a square.

lemma *exp-ge-zero* [*simp*]: $0 \le exp\ x$
⟨*proof*⟩

lemma *exp-not-eq-zero* [*simp*]: $exp\ x \ne 0$
⟨*proof*⟩

lemma *exp-gt-zero* [*simp*]: $0 < exp\ x$
⟨*proof*⟩

lemma *inv-exp-gt-zero* [*simp*]: $0 < inverse(exp\ x)$
⟨*proof*⟩

lemma *abs-exp-cancel* [*simp*]: $|exp\ x| = exp\ x$
⟨*proof*⟩

lemma *exp-real-of-nat-mult*: $exp(real\ n * x) = exp(x)\ \hat{}\ n$
⟨*proof*⟩

lemma *exp-diff*: $exp(x - y) = exp(x)/(exp\ y)$
⟨*proof*⟩

lemma *exp-less-mono*:
assumes *xy*: $x < y$ **shows** $exp\ x < exp\ y$
⟨*proof*⟩

lemma *exp-less-cancel*: $exp\ x < exp\ y ==> x < y$
⟨*proof*⟩

lemma *exp-less-cancel-iff* [*iff*]: $(exp(x) < exp(y)) = (x < y)$
⟨*proof*⟩

lemma *exp-le-cancel-iff* [*iff*]: $(exp(x) \le exp(y)) = (x \le y)$
⟨*proof*⟩

lemma *exp-inj-iff* [*iff*]: $(exp\ x = exp\ y) = (x = y)$
⟨*proof*⟩

lemma *lemma-exp-total*: $1 \le y ==> \exists\, x.\ 0 \le x\ \&\ x \le y - 1\ \&\ exp(x) = y$
⟨*proof*⟩

lemma *exp-total*: $0 < y ==> \exists\, x.\ exp\ x = y$
⟨*proof*⟩

26.8 Properties of the Logarithmic Function

lemma *ln-exp*[*simp*]: $ln(exp\ x) = x$
⟨*proof*⟩

lemma *exp-ln-iff* [*simp*]: $(exp(ln\ x) = x) = (0 < x)$
⟨*proof*⟩

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

lemma *ln-inj-iff* [*simp*]: $[|\ 0 < x;\ 0 < y\ |] ==> (ln\ x = ln\ y) = (x = y)$
⟨*proof*⟩

lemma *ln-one*[*simp*]: $ln\ 1 = 0$
⟨*proof*⟩

lemma *ln-inverse*: $0 < x ==> ln(inverse\ x) = -\ ln\ x$
⟨*proof*⟩

lemma *ln-div*:
  $[|0 < x;\ 0 < y|] ==> ln(x/y) = ln\ x - ln\ y$
⟨*proof*⟩

lemma *ln-less-cancel-iff* [*simp*]: $[|\ 0 < x;\ 0 < y|] ==> (ln\ x < ln\ y) = (x < y)$
⟨*proof*⟩

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

lemma *ln-realpow*: $0 < x ==> ln(x\ \hat{}\ n) = real\ n * ln(x)$
⟨*proof*⟩

lemma *ln-add-one-self-le-self* [*simp*]: $0 \leq x ==> ln(1 + x) \leq x$
⟨*proof*⟩

lemma *ln-less-self* [*simp*]: $0 < x ==> ln\ x < x$
⟨*proof*⟩

lemma *ln-ge-zero* [*simp*]:
 assumes *x*: $1 \leq x$ **shows** $0 \leq ln\ x$
⟨*proof*⟩

lemma *ln-ge-zero-imp-ge-one*:
 assumes *ln*: $0 \leq ln\ x$
  and *x*: $0 < x$
 shows $1 \leq x$
⟨*proof*⟩

lemma *ln-ge-zero-iff* [*simp*]: $0 < x ==> (0 \leq ln\ x) = (1 \leq x)$
⟨*proof*⟩

lemma *ln-less-zero-iff* [*simp*]: $0 < x ==> (ln\ x < 0) = (x < 1)$

⟨*proof*⟩

lemma *ln-gt-zero*:
 assumes *x*: *1 < x* **shows** *0 < ln x*
⟨*proof*⟩

lemma *ln-gt-zero-imp-gt-one*:
 assumes *ln*: *0 < ln x*
 and *x*: *0 < x*
 shows *1 < x*
⟨*proof*⟩

lemma *ln-gt-zero-iff* [*simp*]: *0 < x ==> (0 < ln x) = (1 < x)*
⟨*proof*⟩

lemma *ln-eq-zero-iff* [*simp*]: *0 < x ==> (ln x = 0) = (x = 1)*
⟨*proof*⟩

lemma *ln-less-zero*: [| *0 < x*; *x < 1* |] *==> ln x < 0*
⟨*proof*⟩

lemma *exp-ln-eq*: *exp u = x ==> ln x = u*
⟨*proof*⟩

26.9 Basic Properties of the Trigonometric Functions

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

lemma *lemma-series-zero2*:
(∀ *m. n* ≤ *m --> f m = 0*) *--> f sums setsum f* {*0..<n*}
⟨*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*: $(\mathit{sin}\ x)^2 = \mathit{1} - (\mathit{cos}\ x)^2$
⟨*proof*⟩

lemma *cos-squared-eq*: $(\mathit{cos}\ x)^2 = \mathit{1} - (\mathit{sin}\ x)^2$
⟨*proof*⟩

lemma *real-gt-one-ge-zero-add-less*: $[|\ \mathit{1} < x;\ \mathit{0} \leq y\ |] ==> \mathit{1} < x + (y{::}\mathit{real})$
⟨*proof*⟩

lemma *abs-sin-le-one* [*simp*]: $|\mathit{sin}\ x| \leq \mathit{1}$
⟨*proof*⟩

lemma *sin-ge-minus-one* [*simp*]: $-\mathit{1} \leq \mathit{sin}\ x$
⟨*proof*⟩

lemma *sin-le-one* [*simp*]: $\mathit{sin}\ x \leq \mathit{1}$
⟨*proof*⟩

lemma *abs-cos-le-one* [*simp*]: $|\mathit{cos}\ x| \leq \mathit{1}$
⟨*proof*⟩

lemma *cos-ge-minus-one* [*simp*]: $-\mathit{1} \leq \mathit{cos}\ x$
⟨*proof*⟩

lemma *cos-le-one* [*simp*]: $\mathit{cos}\ x \leq \mathit{1}$
⟨*proof*⟩

lemma *DERIV-fun-pow*: $\mathit{DERIV}\ g\ x :> m ==>$
$\mathit{DERIV}\ (\%x.\ (g\ x)\ \hat{}\ n)\ x :> \mathit{real}\ n * (g\ x)\ \hat{}\ (n - \mathit{1}) * m$
⟨*proof*⟩

lemma *DERIV-fun-exp*:
$\mathit{DERIV}\ g\ x :> m ==> \mathit{DERIV}\ (\%x.\ \mathit{exp}(g\ x))\ x :> \mathit{exp}(g\ x) * m$
⟨*proof*⟩

lemma *DERIV-fun-sin*:
$\mathit{DERIV}\ g\ x :> m ==> \mathit{DERIV}\ (\%x.\ \mathit{sin}(g\ x))\ x :> \mathit{cos}(g\ x) * m$
⟨*proof*⟩

lemma *DERIV-fun-cos*:
$\mathit{DERIV}\ g\ x :> m ==> \mathit{DERIV}\ (\%x.\ \mathit{cos}(g\ x))\ x :> -\mathit{sin}(g\ x) * m$
⟨*proof*⟩

lemmas *DERIV-intros* = *DERIV-Id 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*⟩

26.10 The Constant Pi

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*:
$(\%n.\ (-\ 1)\ \hat{}\ n\ /(real\ (fact\ (2 * n))) * x\ \hat{}\ (2 * n))\ sums\ cos\ x$
⟨*proof*⟩

declare *zero-less-power* [*simp*]

lemma *fact-lemma*: $real\ (n::nat) * 4 = real\ (4 * n)$
⟨*proof*⟩

lemma *cos-two-less-zero*: $cos\ (2) < 0$
⟨*proof*⟩
declare *cos-two-less-zero* [*simp*]
declare *cos-two-less-zero* [*THEN real-not-refl2*, *simp*]
declare *cos-two-less-zero* [*THEN order-less-imp-le*, *simp*]

lemma *cos-is-zero*: $EX!\ x.\ 0 \le x\ \&\ x \le 2\ \&\ cos\ x = 0$
⟨*proof*⟩

lemma *pi-half*: $pi/2 = (@x.\ 0 \le x\ \&\ x \le 2\ \&\ cos\ x = 0)$
⟨*proof*⟩

lemma *cos-pi-half* [*simp*]: $cos\ (pi\ /\ 2) = 0$
⟨*proof*⟩

lemma *pi-half-gt-zero*: $0 < pi\ /\ 2$
⟨*proof*⟩
declare *pi-half-gt-zero* [*simp*]
declare *pi-half-gt-zero* [*THEN real-not-refl2*, *THEN not-sym*, *simp*]
declare *pi-half-gt-zero* [*THEN order-less-imp-le*, *simp*]

lemma *pi-half-less-two*: $pi \ / \ 2 < 2$
⟨*proof*⟩
declare *pi-half-less-two* [*simp*]
declare *pi-half-less-two* [*THEN real-not-refl2*, *simp*]
declare *pi-half-less-two* [*THEN order-less-imp-le*, *simp*]

lemma *pi-gt-zero* [*simp*]: $0 < pi$
⟨*proof*⟩

lemma *pi-neq-zero* [*simp*]: $pi \neq 0$
⟨*proof*⟩

lemma *pi-not-less-zero* [*simp*]: $\sim (pi < 0)$
⟨*proof*⟩

lemma *pi-ge-zero* [*simp*]: $0 \leq pi$
⟨*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*⟩

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 *sin-cos-eq* [*symmetric*, *simp*] *cos-sin-eq* [*symmetric*, *simp*]

lemma *sin-periodic-pi* [*simp*]: $sin\ (x + pi) = - sin\ x$
⟨*proof*⟩

lemma *sin-periodic-pi2* [*simp*]: $sin\ (pi + x) = - sin\ x$
⟨*proof*⟩

lemma *cos-periodic-pi* [*simp*]: $cos\ (x + pi) = - cos\ x$
⟨*proof*⟩

lemma *sin-periodic* [*simp*]: $sin\ (x + 2{*}pi) = sin\ x$
⟨*proof*⟩

lemma *cos-periodic* [*simp*]: $cos\ (x + 2{*}pi) = cos\ x$
⟨*proof*⟩

lemma *cos-npi* [*simp*]: $cos\ (real\ n * pi) = -1\ \hat{}\ n$
⟨*proof*⟩

lemma *cos-npi2* [*simp*]: $cos\ (pi * real\ n) = -1\ \hat{}\ n$
⟨*proof*⟩

lemma *sin-npi* [*simp*]: $sin\ (real\ (n{::}nat) * pi) = 0$
⟨*proof*⟩

lemma *sin-npi2* [*simp*]: $sin\ (pi * real\ (n{::}nat)) = 0$
⟨*proof*⟩

lemma *cos-two-pi* [*simp*]: $cos\ (2 * pi) = 1$
⟨*proof*⟩

lemma *sin-two-pi* [*simp*]: $sin\ (2 * pi) = 0$
⟨*proof*⟩

lemma *sin-gt-zero2*: $[|\ 0 < x;\ x < pi/2\ |] ==> 0 < sin\ x$
⟨*proof*⟩

lemma *sin-less-zero*:
 assumes *lb*: $-\ pi/2 < x$ **and** $x < 0$ **shows** $sin\ x < 0$
⟨*proof*⟩

lemma *pi-less-4*: $pi < 4$
⟨*proof*⟩

lemma *cos-gt-zero*: $[|\ 0 < x;\ x < pi/2\ |] ==> 0 < cos\ x$
⟨*proof*⟩

lemma *cos-gt-zero-pi*: $[|\ -(pi/2) < x;\ x < pi/2\ |] ==> 0 < cos\ x$
⟨*proof*⟩

lemma *cos-ge-zero*: $[|\ -(pi/2) \le x;\ x \le pi/2\ |] ==> 0 \le cos\ x$
⟨*proof*⟩

lemma *sin-gt-zero-pi*: $[|\ 0 < x;\ x < pi\ \ |] ==> 0 < sin\ x$
⟨*proof*⟩

lemma *sin-ge-zero*: $[|\ 0 \le x;\ x \le pi\ |] ==> 0 \le sin\ x$
⟨*proof*⟩

lemma *cos-total*: $[| -1 \leq y; y \leq 1 |] ==> EX! x. 0 \leq x \& x \leq pi \& (cos\ x = y)$
⟨*proof*⟩

lemma *sin-total*:
$[| -1 \leq y; y \leq 1 |] ==> EX! x. -(pi/2) \leq x \& x \leq pi/2 \& (sin\ x = y)$
⟨*proof*⟩

lemma *reals-Archimedean4*:
$[| 0 < y; 0 \leq x |] ==> \exists n.\ real\ n * y \leq x \& x < real\ (Suc\ n) * y$
⟨*proof*⟩

lemma *cos-zero-lemma*:
$[| 0 \leq x; cos\ x = 0 |] ==>$
$\exists n::nat.\ \sim even\ n \& x = real\ n * (pi/2)$
⟨*proof*⟩

lemma *sin-zero-lemma*:
$[| 0 \leq x; sin\ x = 0 |] ==>$
$\exists n::nat.\ even\ n \& x = real\ n * (pi/2)$
⟨*proof*⟩

lemma *cos-zero-iff*:
$(cos\ x = 0) =$
$((\exists n::nat.\ \sim even\ n \& (x = real\ n * (pi/2))) |$
$(\exists n::nat.\ \sim even\ n \& (x = -(real\ n * (pi/2)))))$
⟨*proof*⟩

lemma *sin-zero-iff*:
$(sin\ x = 0) =$
$((\exists n::nat.\ even\ n \& (x = real\ n * (pi/2))) |$
$(\exists n::nat.\ even\ n \& (x = -(real\ n * (pi/2)))))$
⟨*proof*⟩

26.11 Tangent

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) ^ $2))$
⟨*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 *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*⟩

lemma *arcsin-pi*:
$[|\ -1 \leq y;\ y \leq 1\ |]$
$==> -(pi/2) \leq arcsin\ y\ \&\ arcsin\ y \leq pi\ \&\ sin(arcsin\ y) = y$
⟨*proof*⟩

lemma *arcsin*:
$[|\ -1 \leq y;\ y \leq 1\ |]$
$==> -(pi/2) \leq arcsin\ y\ \&$
$arcsin\ y \leq pi/2\ \&\ sin(arcsin\ y) = y$
⟨*proof*⟩

lemma *sin-arcsin* [*simp*]: $[|\ -1 \leq y;\ y \leq 1\ |] ==> sin(arcsin\ y) = y$
⟨*proof*⟩

lemma *arcsin-bounded*:
$[|\ -1 \leq y;\ y \leq 1\ |] ==> -(pi/2) \leq arcsin\ y\ \&\ arcsin\ y \leq pi/2$
⟨*proof*⟩

lemma *arcsin-lbound*: $[|\ -1 \leq y;\ y \leq 1\ |] ==> -(pi/2) \leq arcsin\ y$
⟨*proof*⟩

lemma *arcsin-ubound*: $[|\ -1 \leq y;\ y \leq 1\ |] ==> arcsin\ y \leq pi/2$
⟨*proof*⟩

lemma *arcsin-lt-bounded*:
$[|\ -1 < y;\ y < 1\ |] ==> -(pi/2) < arcsin\ y\ \&\ arcsin\ y < pi/2$
⟨*proof*⟩

lemma *arcsin-sin*: $[|-(pi/2) \leq x;\ x \leq pi/2\ |] ==> arcsin(sin\ x) = x$
⟨*proof*⟩

lemma *arcos*:
$[|\ -1 \leq y;\ y \leq 1\ |]$
$==> 0 \leq arcos\ y\ \&\ arcos\ y \leq pi\ \&\ cos(arcos\ y) = y$
⟨*proof*⟩

lemma *cos-arcos* [*simp*]: $[|\ -1 \leq y;\ y \leq 1\ |] ==> cos(arcos\ y) = y$
⟨*proof*⟩

lemma *arcos-bounded*: $[|\ -1 \leq y;\ y \leq 1\ |] ==> 0 \leq arcos\ y\ \&\ arcos\ y \leq pi$
⟨*proof*⟩

lemma *arcos-lbound*: $[|\ -1 \leq y;\ y \leq 1\ |] ==> 0 \leq arcos\ y$

⟨*proof*⟩

lemma *arcos-ubound*: $[| -1 \le y;\ y \le 1\ |] ==> arcos\ y \le pi$
⟨*proof*⟩

lemma *arcos-lt-bounded*:
$[| -1 < y;\ y < 1\ |]$
$==> 0 < arcos\ y\ \&\ arcos\ y < pi$
⟨*proof*⟩

lemma *arcos-cos*: $[|0 \le x;\ x \le pi\ |] ==> arcos(cos\ x) = x$
⟨*proof*⟩

lemma *arcos-cos2*: $[|x \le 0;\ -pi \le x\ |] ==> arcos(cos\ x) = -x$
⟨*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*]: $cos(arctan\ x) \ne 0$
⟨*proof*⟩

lemma *tan-sec*: $cos\ x \ne 0 ==> 1 + tan(x)\ \hat{}\ 2 = inverse(cos\ x)\ \hat{}\ 2$
⟨*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*]: $sin\ (real\ (Suc\ (2 * n)) * pi\ /\ 2) = (-1)\ \hat{}\ n$
⟨*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 *isCont-cos* [*simp*]: *isCont cos x*
⟨*proof*⟩

lemma *isCont-sin* [*simp*]: *isCont sin x*
⟨*proof*⟩

lemma *isCont-exp* [*simp*]: *isCont exp x*
⟨*proof*⟩

lemma *sin-zero-abs-cos-one*: $sin\ x = 0 ==> |cos\ x| = 1$
⟨*proof*⟩

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

lemma *cos-one-sin-zero*: $cos\ x = 1 ==> sin\ x = 0$
⟨*proof*⟩

lemma *real-root-less-mono*:
$[|\ 0 \leq x;\ x < y\ |] ==> root(Suc\ n)\ x < root(Suc\ n)\ y$
⟨*proof*⟩

lemma *real-root-le-mono*:
$[|\ 0 \leq x;\ x \leq y\ |] ==> root(Suc\ n)\ x \leq root(Suc\ n)\ y$
⟨*proof*⟩

lemma *real-root-less-iff* [*simp*]:
$[|\ 0 \leq x;\ 0 \leq y\ |] ==> (root(Suc\ n)\ x < root(Suc\ n)\ y) = (x < y)$
⟨*proof*⟩

lemma *real-root-le-iff* [*simp*]:
$[|\ 0 \leq x;\ 0 \leq y\ |] ==> (root(Suc\ n)\ x \leq root(Suc\ n)\ y) = (x \leq y)$
⟨*proof*⟩

lemma *real-root-eq-iff* [*simp*]:
$[|\ 0 \leq x;\ 0 \leq y\ |] ==> (root(Suc\ n)\ x = root(Suc\ n)\ y) = (x = y)$
⟨*proof*⟩

lemma *real-root-pos-unique*:
$[|\ 0 \leq x;\ 0 \leq y;\ y\ \hat{}\ (Suc\ n) = x\ |] ==> root\ (Suc\ n)\ x = y$
⟨*proof*⟩

lemma *real-root-mult*:
$[|\ 0 \leq x;\ 0 \leq y\ |]$
$==> root(Suc\ n)\ (x * y) = root(Suc\ n)\ x * root(Suc\ n)\ y$
⟨*proof*⟩

lemma *real-root-inverse*:
$0 \leq x ==> (root(Suc\ n)\ (inverse\ x) = inverse(root(Suc\ n)\ x))$
⟨*proof*⟩

lemma *real-root-divide*:
$[|\ 0 \leq x;\ 0 \leq y\ |]$
$==> (root(Suc\ n)\ (x\ /\ y) = root(Suc\ n)\ x\ /\ root(Suc\ n)\ y)$
⟨*proof*⟩

lemma *real-sqrt-less-mono*: $[| \ 0 \leq x; \ x < y \ |] \Longrightarrow sqrt(x) < sqrt(y)$
⟨*proof*⟩

lemma *real-sqrt-le-mono*: $[| \ 0 \leq x; \ x \leq y \ |] \Longrightarrow sqrt(x) \leq sqrt(y)$
⟨*proof*⟩

lemma *real-sqrt-less-iff* [*simp*]:
$[| \ 0 \leq x; \ 0 \leq y \ |] \Longrightarrow (sqrt(x) < sqrt(y)) = (x < y)$
⟨*proof*⟩

lemma *real-sqrt-le-iff* [*simp*]:
$[| \ 0 \leq x; \ 0 \leq y \ |] \Longrightarrow (sqrt(x) \leq sqrt(y)) = (x \leq y)$
⟨*proof*⟩

lemma *real-sqrt-eq-iff* [*simp*]:
$[| \ 0 \leq x; \ 0 \leq y \ |] \Longrightarrow (sqrt(x) = sqrt(y)) = (x = y)$
⟨*proof*⟩

lemma *real-sqrt-sos-less-one-iff* [*simp*]: $(sqrt(x^2 + y^2) < 1) = (x^2 + y^2 < 1)$
⟨*proof*⟩

lemma *real-sqrt-sos-eq-one-iff* [*simp*]: $(sqrt(x^2 + y^2) = 1) = (x^2 + y^2 = 1)$
⟨*proof*⟩

lemma *real-divide-square-eq* [*simp*]: $(((r::real) * a) \ / \ (r * r)) = a \ / \ r$
⟨*proof*⟩

26.12 Theorems About Sqrt, Transcendental Functions for Complex

lemma *le-real-sqrt-sumsq* [*simp*]: $x \leq sqrt \ (x * x + y * y)$
⟨*proof*⟩

lemma *minus-le-real-sqrt-sumsq* [*simp*]: $-x \leq sqrt \ (x * x + y * y)$
⟨*proof*⟩

lemma *lemma-real-divide-sqrt-ge-minus-one*:
$0 < x \Longrightarrow -1 \leq x/(sqrt \ (x * x + y * y))$
⟨*proof*⟩

lemma *real-sqrt-sum-squares-gt-zero1*: $x < 0 \Longrightarrow 0 < sqrt \ (x * x + y * y)$
⟨*proof*⟩

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

lemma *real-sqrt-sum-squares-gt-zero3*: $x \neq 0 \Longrightarrow 0 < sqrt(x^2 + y^2)$
⟨*proof*⟩

lemma *real-sqrt-sum-squares-gt-zero3a*: $y \neq 0 ==> 0 < sqrt(x^2 + y^2)$
⟨*proof*⟩

lemma *real-sqrt-sum-squares-eq-cancel*: $sqrt(x^2 + y^2) = x ==> y = 0$
⟨*proof*⟩

lemma *real-sqrt-sum-squares-eq-cancel2*: $sqrt(x^2 + y^2) = y ==> x = 0$
⟨*proof*⟩

lemma *lemma-real-divide-sqrt-le-one*: $x < 0 ==> x/(sqrt\ (x * x + y * y)) \leq 1$
⟨*proof*⟩

lemma *lemma-real-divide-sqrt-ge-minus-one2*:
$x < 0 ==> -1 \leq x/(sqrt\ (x * x + y * y))$
⟨*proof*⟩

lemma *lemma-real-divide-sqrt-le-one2*: $0 < x ==> x/(sqrt\ (x * x + y * y)) \leq 1$
⟨*proof*⟩

lemma *minus-sqrt-le*: $-\ sqrt\ (x * x + y * y) \leq x$
⟨*proof*⟩

lemma *minus-sqrt-le2*: $-\ sqrt\ (x * x + y * y) \leq y$
⟨*proof*⟩

lemma *not-neg-sqrt-sumsq*: $\tilde{}\ sqrt\ (x * x + y * y) < 0$
⟨*proof*⟩

lemma *cos-x-y-ge-minus-one*: $-1 \leq x\ /\ sqrt\ (x * x + y * y)$
⟨*proof*⟩

lemma *cos-x-y-ge-minus-one1a* [*simp*]: $-1 \leq y\ /\ sqrt\ (x * x + y * y)$
⟨*proof*⟩

lemma *cos-x-y-le-one* [*simp*]: $x\ /\ sqrt\ (x * x + y * y) \leq 1$
⟨*proof*⟩

lemma *cos-x-y-le-one2* [*simp*]: $y\ /\ sqrt\ (x * x + y * y) \leq 1$
⟨*proof*⟩

declare *cos-arcos* [*OF cos-x-y-ge-minus-one cos-x-y-le-one*, *simp*]
declare *arcos-bounded* [*OF cos-x-y-ge-minus-one cos-x-y-le-one*, *simp*]

declare *cos-arcos* [*OF cos-x-y-ge-minus-one1a cos-x-y-le-one2*, *simp*]
declare *arcos-bounded* [*OF cos-x-y-ge-minus-one1a cos-x-y-le-one2*, *simp*]

lemma *cos-abs-x-y-ge-minus-one* [*simp*]:
$-1 \leq |x|\ /\ sqrt\ (x * x + y * y)$
⟨*proof*⟩

lemma *cos-abs-x-y-le-one* [*simp*]: $|x|$ / *sqrt* $(x * x + y * y) \leq 1$
$\langle proof \rangle$

declare *cos-arcos* [*OF cos-abs-x-y-ge-minus-one cos-abs-x-y-le-one*, *simp*]
declare *arcos-bounded* [*OF cos-abs-x-y-ge-minus-one cos-abs-x-y-le-one*, *simp*]

lemma *minus-pi-less-zero*: $-pi < 0$
$\langle proof \rangle$

declare *minus-pi-less-zero* [*simp*]
declare *minus-pi-less-zero* [*THEN order-less-imp-le*, *simp*]

lemma *arcos-ge-minus-pi*: [| $-1 \leq y$; $y \leq 1$ |] ==> $-pi \leq$ *arcos* y
$\langle proof \rangle$

declare *arcos-ge-minus-pi* [*OF cos-x-y-ge-minus-one cos-x-y-le-one*, *simp*]

lemma *lemma-divide-rearrange*:
  [| $x + (y{::}real) \neq 0$; $1 - z = x/(x + y)$ |] ==> $z = y/(x + y)$
$\langle proof \rangle$

lemma *lemma-cos-sin-eq*:
  [| $0 < x * x + y * y$;
    $1 - (sin\ xa)^2 = (x$ / *sqrt* $(x * x + y * y))$ ^ *2* |]
  ==> $(sin\ xa)^2 = (y$ / *sqrt* $(x * x + y * y))$ ^ *2*
$\langle proof \rangle$

lemma *lemma-sin-cos-eq*:
  [| $0 < x * x + y * y$;
    $1 - (cos\ xa)^2 = (y$ / *sqrt* $(x * x + y * y))$ ^ *2* |]
  ==> $(cos\ xa)^2 = (x$ / *sqrt* $(x * x + y * y))$ ^ *2*
$\langle proof \rangle$

lemma *sin-x-y-disj*:
  [| $x \neq 0$;
    cos $xa = x$ / *sqrt* $(x * x + y * y)$
  |] ==> *sin* $xa = y$ / *sqrt* $(x * x + y * y)$ |
      sin $xa = -\ y$ / *sqrt* $(x * x + y * y)$
$\langle proof \rangle$

lemma *lemma-cos-not-eq-zero*: $x \neq 0$ ==> x / *sqrt* $(x * x + y * y) \neq 0$
$\langle proof \rangle$

lemma *cos-x-y-disj*:
  [| $x \neq 0$;
    sin $xa = y$ / *sqrt* $(x * x + y * y)$

|] ==> $cos\ xa = x\ /\ sqrt\ (x * x + y * y)$ |
$cos\ xa = -\ x\ /\ sqrt\ (x * x + y * y)$
⟨proof⟩

lemma *real-sqrt-divide-less-zero*: $0 < y$ ==> $-\ y\ /\ sqrt\ (x * x + y * y) < 0$
⟨proof⟩

lemma *polar-ex1*:
[| $x \neq 0$; $0 < y$ |] ==> $\exists\, r\ a.\ x = r * cos\ a\ \&\ y = r * sin\ a$
⟨proof⟩

lemma *real-sum-squares-cancel2a*: $x * x = -(y * y)$ ==> $y = (0::real)$
⟨proof⟩

lemma *polar-ex2*:
[| $x \neq 0$; $y < 0$ |] ==> $\exists\, r\ a.\ x = r * cos\ a\ \&\ y = r * sin\ a$
⟨proof⟩

lemma *polar-Ex*: $\exists\, r\ a.\ x = r * cos\ a\ \&\ y = r * sin\ a$
⟨proof⟩

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⟩
declare *real-sqrt-ge-abs1* [*simp*] *real-sqrt-ge-abs2* [*simp*]

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 *lemma-real-divide-sqrt-less*: $0 < u$ ==> $u\ /\ sqrt\ 2 < u$
⟨proof⟩

lemma *four-x-squared*:
fixes $x::real$
shows $4 * x^2 = (2 * x)^2$
⟨proof⟩

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*⟩

declare *real-sqrt-sum-squares-ge-zero* [*THEN abs-of-nonneg*, *simp*]

26.13 A Few Theorems Involving Ln, Derivatives, etc.

lemma *lemma-DERIV-ln*:
$DERIV\ ln\ z :> l ==> DERIV\ (\%x.\ exp\ (ln\ x))\ z :> exp\ (ln\ z) * l$
⟨*proof*⟩

lemma *STAR-exp-ln*: $0 < z ==> (*f* (\%x.\ exp\ (ln\ x)))\ z = z$
⟨*proof*⟩

lemma *hypreal-add-Infinitesimal-gt-zero*:
$[|e : Infinitesimal;\ 0 < x\ |] ==> 0 < hypreal\text{-}of\text{-}real\ x + e$
⟨*proof*⟩

lemma *NSDERIV-exp-ln-one*: $0 < z ==> NSDERIV\ (\%x.\ exp\ (ln\ x))\ z :> 1$
⟨*proof*⟩

lemma *DERIV-exp-ln-one*: $0 < z ==> DERIV\ (\%x.\ exp\ (ln\ x))\ z :> 1$
⟨*proof*⟩

lemma *lemma-DERIV-ln2*:
$[|\ 0 < z;\ DERIV\ ln\ z :> l\ |] ==>\ exp\ (ln\ z) * l = 1$
⟨*proof*⟩

lemma *lemma-DERIV-ln3*:
$[|\ 0 < z;\ DERIV\ ln\ z :> l\ |] ==>\ l = 1/(exp\ (ln\ z))$
⟨*proof*⟩

lemma *lemma-DERIV-ln4*: $[|\ 0 < z;\ DERIV\ ln\ z :> l\ |] ==>\ l = 1/z$
⟨*proof*⟩

lemma *isCont-inv-fun*:
$[|\ 0 < d;\ \forall z.\ |z - x| \le d\ --> g(f(z)) = z;$
$\forall z.\ |z - x| \le d\ --> isCont\ f\ z\ |]$
$==> isCont\ g\ (f\ x)$
⟨*proof*⟩

lemma *isCont-inv-fun-inv*:
$[|\ 0 < d;$
$\forall z.\ |z - x| \le d\ --> g(f(z)) = z;$
$\forall z.\ |z - x| \le d\ --> isCont\ f\ z\ |]$
$==> \exists e.\ 0 < e\ \&$
$(\forall y.\ 0 < |y - f(x)|\ \&\ |y - f(x)| < e\ --> f(g(y)) = y)$
⟨*proof*⟩

Bartle/Sherbert: Introduction to Real Analysis, Theorem 4.2.9, p. 110

lemma *LIM-fun-gt-zero*:
[| f -- c --> l; $0 < l$ |]
==> $\exists r.\ 0 < r$ & ($\forall x.\ x \neq c$ & $|c - x| < r$ --> $0 < f\,x$)
⟨*proof*⟩

lemma *LIM-fun-less-zero*:
[| f -- c --> l; $l < 0$ |]
==> $\exists r.\ 0 < r$ & ($\forall x.\ x \neq c$ & $|c - x| < r$ --> $f\,x < 0$)
⟨*proof*⟩

lemma *LIM-fun-not-zero*:
[| f -- c --> l; $l \neq 0$ |]
==> $\exists r.\ 0 < r$ & ($\forall x.\ x \neq c$ & $|c - x| < r$ --> $f\,x \neq 0$)
⟨*proof*⟩

⟨*ML*⟩

end

27 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 ^ 2 / 2 * (1 / 2) ^ n) sums x^2*
⟨*proof*⟩

lemma *exp-bound*: *0 <= x ==> x <= 1 ==> exp x <= 1 + x + x^2*
⟨*proof*⟩

lemma *aux3*: *(0::real) <= x ==> (1 + x + x^2)/(1 + x^2) <= 1 + x*

⟨*proof*⟩

lemma *aux4*: *0 <= x ==> 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 <= exp x*
⟨*proof*⟩

lemma *ln-add-one-self-le-self2*: *−1 < x ==> ln(1 + x) <= x*
⟨*proof*⟩

lemma *abs-ln-one-plus-x-minus-x-bound-nonneg*:
0 <= x ==> x <= 1 ==> abs(ln (1 + x) − x) <= x^2
⟨*proof*⟩

lemma *abs-ln-one-plus-x-minus-x-bound-nonpos*:
−(1 / 2) <= x ==> x <= 0 ==> abs(ln (1 + x) − x) <= 2 ∗ x^2
⟨*proof*⟩

lemma *abs-ln-one-plus-x-minus-x-bound*:
abs x <= 1 / 2 ==> abs(ln (1 + x) − x) <= 2 ∗ x^2
⟨*proof*⟩

lemma *DERIV-ln*: *0 < x ==> DERIV ln x :> 1 / x*
⟨*proof*⟩

lemma *ln-x-over-x-mono*: *exp 1 <= x ==> x <= y ==> (ln y / y) <= (ln x / x)*
⟨*proof*⟩

end

28 Poly: Univariate Real Polynomials

theory *Poly*
imports *Ln*
begin

Application of polynomial as a real function.

consts *poly* :: *real list* => *real* => *real*
primrec
poly-Nil: *poly* [] *x* = *0*
poly-Cons: *poly* (*h*#*t*) *x* = *h* + *x* * *poly t x*

28.1 Arithmetic Operations on Polynomials

addition

consts +++ :: [*real list*, *real list*] => *real list* (**infixl** *65*)
primrec
padd-Nil: [] +++ *l2* = *l2*
padd-Cons: (*h*#*t*) +++ *l2* = (*if l2* = [] *then h*#*t*
else (*h* + *hd l2*)#(*t* +++ *tl l2*))

Multiplication by a constant

consts %* :: [*real*, *real list*] => *real list* (**infixl** *70*)
primrec
cmult-Nil: *c* %* [] = []
cmult-Cons: *c* %* (*h*#*t*) = (*c* * *h*)#(*c* %* *t*)

Multiplication by a polynomial

consts *** :: [*real list*, *real list*] => *real list* (**infixl** *70*)
primrec
pmult-Nil: [] *** *l2* = []
pmult-Cons: (*h*#*t*) *** *l2* = (*if t* = [] *then h* %* *l2*
else (*h* %* *l2*) +++ ((*0*) # (*t* *** *l2*)))

Repeated multiplication by a polynomial

consts *mulexp* :: [*nat*, *real list*, *real list*] => *real list*
primrec
mulexp-zero: *mulexp 0 p q* = *q*
mulexp-Suc: *mulexp* (*Suc n*) *p q* = *p* *** *mulexp n p q*

Exponential

consts %^ :: [*real list*, *nat*] => *real list* (**infixl** *80*)
primrec
pexp-0: *p* %^ *0* = [*1*]
pexp-Suc: *p* %^ (*Suc n*) = *p* *** (*p* %^ *n*)

Quotient related value of dividing a polynomial by x + a

consts *pquot* :: [*real list*, *real*] => *real list*
primrec
  pquot-Nil: *pquot* [] *a*= []
  pquot-Cons: *pquot* (*h*#*t*) *a* = (*if t* = [] *then* [*h*]
    else (*inverse*(*a*) ∗ (*h* − *hd*(*pquot t a*)))#(*pquot t a*))

Differentiation of polynomials (needs an auxiliary function).

consts *pderiv-aux* :: *nat* => *real list* => *real list*
primrec
  pderiv-aux-Nil: *pderiv-aux n* [] = []
  pderiv-aux-Cons: *pderiv-aux n* (*h*#*t*) =
    (*real n* ∗ *h*)#(*pderiv-aux* (*Suc n*) *t*)

normalization of polynomials (remove extra 0 coeff)

consts *pnormalize* :: *real list* => *real list*
primrec
  pnormalize-Nil: *pnormalize* [] = []
  pnormalize-Cons: *pnormalize* (*h*#*p*) = (*if* ((*pnormalize p*) = [])
    then (*if* (*h* = *0*) *then* [] *else* [*h*])
    else (*h*#(*pnormalize p*)))

Other definitions

constdefs
  poly-minus :: *real list* => *real list*   (−− - [*80*] *80*)
  −− *p* == (− *1*) %∗ *p*

  pderiv :: *real list* => *real list*
  pderiv p == *if p* = [] *then* [] *else pderiv-aux 1* (*tl p*)

  divides :: [*real list*,*real list*] => *bool* (**infixl** *divides 70*)
  p1 divides p2 == ∃ *q*. *poly p2* = *poly*(*p1* ∗∗∗ *q*)

  order :: *real* => *real list* => *nat*
   — order of a polynomial
  order a p == (@*n*. ([−*a*, *1*] %ˆ *n*) *divides p* &
    ~ (([−*a*, *1*] %ˆ (*Suc n*)) *divides p*))

  degree :: *real list* => *nat*
   — degree of a polynomial
  degree p == *length* (*pnormalize p*)

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

lemma *padd-Nil2*: *p* +++ [] = *p*

⟨*proof*⟩
declare *padd-Nil2* [*simp*]

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

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

lemma *pmult-singleton*: $[h1]\ {*}{*}{*}\ p1 = h1\ \%{*}\ p1$
⟨*proof*⟩

lemma *poly-ident-mult*: $1\ \%{*}\ t = t$
⟨*proof*⟩
declare *poly-ident-mult* [*simp*]

lemma *poly-simple-add-Cons*: $[a]\ {+}{+}{+}\ ((0)\#t) = (a\#t)$
⟨*proof*⟩
declare *poly-simple-add-Cons* [*simp*]

Handy general properties

lemma *padd-commut*: $b\ {+}{+}{+}\ a = a\ {+}{+}{+}\ b$
⟨*proof*⟩

lemma *padd-assoc* [*rule-format*]: $\forall\, b\ c.\ (a\ {+}{+}{+}\ b)\ {+}{+}{+}\ c = a\ {+}{+}{+}\ (b\ {+}{+}{+}\ c)$
⟨*proof*⟩

lemma *poly-cmult-distr* [*rule-format*]:
$\forall\, q.\ a\ \%{*}\ (\ p\ {+}{+}{+}\ q) = (a\ \%{*}\ p\ {+}{+}{+}\ a\ \%{*}\ q)$
⟨*proof*⟩

lemma *pmult-by-x*: $[0,\ 1]\ {*}{*}{*}\ t = ((0)\#t)$
⟨*proof*⟩
declare *pmult-by-x* [*simp*]

properties of evaluation of polynomials.

lemma *poly-add*: $poly\ (p1\ {+}{+}{+}\ p2)\ x = poly\ p1\ x + poly\ p2\ x$
⟨*proof*⟩

lemma *poly-cmult*: $poly\ (c\ \%{*}\ p)\ x = c * poly\ p\ x$
⟨*proof*⟩

lemma *poly-minus*: $poly\ (--\ p)\ x = -\ (poly\ p\ x)$
⟨*proof*⟩

lemma *poly-mult*: $poly\ (p1\ {*}{*}{*}\ p2)\ x = poly\ p1\ x * poly\ p2\ x$
⟨*proof*⟩

lemma *poly-exp*: $poly\ (p\ \%\hat{}\ n)\ x = (poly\ p\ x)\ \hat{}\ n$
⟨*proof*⟩

More Polynomial Evaluation Lemmas

lemma *poly-add-rzero*: $poly\ (a\ {+}{+}{+}\ [])\ x = poly\ a\ x$
⟨*proof*⟩
declare *poly-add-rzero* [*simp*]

lemma *poly-mult-assoc*: $poly\ ((a *** b) *** c)\ x = poly\ (a *** (b *** c))\ x$
⟨*proof*⟩

lemma *poly-mult-Nil2*: $poly\ (p *** [])\ x = 0$
⟨*proof*⟩
declare *poly-mult-Nil2* [*simp*]

lemma *poly-exp-add*: $poly\ (p\ \%\hat{}\ (n + d))\ x = poly(\ p\ \%\hat{}\ n *** p\ \%\hat{}\ d)\ x$
⟨*proof*⟩

The derivative

lemma *pderiv-Nil*: $pderiv\ [] = []$

⟨*proof*⟩
declare *pderiv-Nil* [*simp*]

lemma *pderiv-singleton*: $pderiv\ [c] = []$
⟨*proof*⟩
declare *pderiv-singleton* [*simp*]

lemma *pderiv-Cons*: $pderiv\ (h\#t) = pderiv\text{-}aux\ 1\ t$
⟨*proof*⟩

lemma *DERIV-cmult2*: $DERIV\ f\ x :> D ==> DERIV\ (\%x.\ (f\ x) * c)\ x :> D * c$
⟨*proof*⟩

lemma *DERIV-pow2*: $DERIV\ (\%x.\ x\ \hat{}\ Suc\ n)\ x :> real\ (Suc\ n) * (x\ \hat{}\ n)$
⟨*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*: $DERIV\ f\ x :> D ==>\ DERIV\ (\%x.\ a + f\ x)\ x :> D$

⟨*proof*⟩

lemma *poly-DERIV*: *DERIV* (%*x*. *poly p x*) *x* :> *poly* (*pderiv p*) *x*
⟨*proof*⟩
declare *poly-DERIV* [*simp*]

Consequences of the derivative theorem above

lemma *poly-differentiable*: (%*x*. *poly p x*) *differentiable x*

⟨*proof*⟩
declare *poly-differentiable* [*simp*]

lemma *poly-isCont*: *isCont* (%*x*. *poly p x*) *x*
⟨*proof*⟩
declare *poly-isCont* [*simp*]

lemma *poly-IVT-pos*: [| $a < b$; *poly p a* < 0; $0 <$ *poly p b* |]
==> $\exists x$. $a < x$ & $x < b$ & (*poly p x* $= 0$)
⟨*proof*⟩

lemma *poly-IVT-neg*: [| $a < b$; $0 <$ *poly p a*; *poly p b* < 0 |]
==> $\exists x$. $a < x$ & $x < b$ & (*poly p x* $= 0$)
⟨*proof*⟩

lemma *poly-MVT*: $a < b$ ==>
$\exists x$. $a < x$ & $x < b$ & (*poly p b* $-$ *poly p a* $= (b - a) *$ *poly* (*pderiv p*) *x*)
⟨*proof*⟩

Lemmas for Derivatives

lemma *lemma-poly-pderiv-aux-add*: $\forall$ *p2 n*. *poly* (*pderiv-aux n* (*p1* +++ *p2*)) *x* =
poly (*pderiv-aux n p1* +++ *pderiv-aux n p2*) *x*
⟨*proof*⟩

lemma *poly-pderiv-aux-add*: *poly* (*pderiv-aux n* (*p1* +++ *p2*)) *x* =
poly (*pderiv-aux n p1* +++ *pderiv-aux n p2*) *x*
⟨*proof*⟩

lemma *lemma-poly-pderiv-aux-cmult*: $\forall$ *n*. *poly* (*pderiv-aux n* (*c* %* *p*)) *x* = *poly* (*c* %* *pderiv-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*: $\forall n.$ *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*: $\forall q.$ *poly* (*pderiv* (p +++ q)) x = *poly* (*pderiv* p +++ *pderiv* q) x
⟨*proof*⟩

lemma *poly-pderiv-add*: *poly* (*pderiv* (p +++ q)) x = *poly* (*pderiv* p +++ *pderiv* q) x
⟨*proof*⟩

lemma *poly-pderiv-cmult*: *poly* (*pderiv* (c %* p)) x = *poly* (c %* (*pderiv* p)) x
⟨*proof*⟩

lemma *poly-pderiv-minus*: *poly* (*pderiv* (−−p)) x = *poly* (−−(*pderiv* p)) x
⟨*proof*⟩

lemma *lemma-poly-mult-pderiv*:
poly (*pderiv* (h#t)) x = *poly* ((0 # (*pderiv* t)) +++ t) x
⟨*proof*⟩

lemma *poly-pderiv-mult*: $\forall q.$ *poly* (*pderiv* (p *** q)) x =
poly (p *** (*pderiv* q) +++ q *** (*pderiv* p)) x
⟨*proof*⟩

lemma *poly-pderiv-exp*: *poly* (*pderiv* (p %^ (*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*⟩

28.2 Key Property: if $f\ a = (0{::}'a)$ then $x - a$ divides $p\ x$

lemma *lemma-poly-linear-rem*: $\forall h.\ \exists q\ r.$ h#t = [r] +++ [$-a$, 1] *** q
⟨*proof*⟩

lemma *poly-linear-rem*: $\exists q\ r.$ h#t = [r] +++ [$-a$, 1] *** q
⟨*proof*⟩

lemma *poly-linear-divides*: (*poly* p a = 0) = ((p = []) | ($\exists q.$ p = [$-a$, 1] *** q))
⟨*proof*⟩

lemma *lemma-poly-length-mult*: $\forall h\ k\ a.\ length\ (k\ \%{*}\ p\ +{+}+\ \ (h\ \#\ (a\ \%{*}\ p))) = Suc\ (length\ p)$
⟨*proof*⟩
declare *lemma-poly-length-mult* [*simp*]

lemma *lemma-poly-length-mult2*: $\forall h\ k.\ length\ (k\ \%{*}\ p\ +{+}+\ \ (h\ \#\ p)) = Suc\ (length\ p)$
⟨*proof*⟩
declare *lemma-poly-length-mult2* [*simp*]

lemma *poly-length-mult*: $length([-a,1]\ {*}{*}{*}\ q) = Suc\ (length\ q)$
⟨*proof*⟩
declare *poly-length-mult* [*simp*]

28.3 Polynomial length

lemma *poly-cmult-length*: $length\ (a\ \%{*}\ p) = length\ p$
⟨*proof*⟩
declare *poly-cmult-length* [*simp*]

lemma *poly-add-length* [*rule-format*]:
$\forall p2.\ length\ (p1\ +{+}+\ p2) =$
$(if\ \ (length\ p1 < length\ p2)\ then\ length\ p2\ else\ length\ p1)$
⟨*proof*⟩

lemma *poly-root-mult-length*: $length([a,b]\ {*}{*}{*}\ p) = Suc\ (length\ p)$
⟨*proof*⟩
declare *poly-root-mult-length* [*simp*]

lemma *poly-mult-not-eq-poly-Nil*: $(poly\ (p\ {*}{*}{*}\ q)\ x \neq poly\ []\ x) =$
$(poly\ p\ x \neq poly\ []\ x\ \&\ poly\ q\ x \neq poly\ []\ x)$
⟨*proof*⟩
declare *poly-mult-not-eq-poly-Nil* [*simp*]

lemma *poly-mult-eq-zero-disj*: $(poly\ (p\ {*}{*}{*}\ q)\ x = 0) = (poly\ p\ x = 0\ |\ poly\ q\ x = 0)$
⟨*proof*⟩

Normalisation Properties

lemma *poly-normalized-nil*: $(pnormalize\ p = [])\ {-}{-}{>}\ (poly\ p\ x = 0)$
⟨*proof*⟩

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

lemma *poly-roots-index-lemma* [*rule-format*]:
$\forall p\ x.\ poly\ p\ x \neq poly\ []\ x\ \&\ length\ p = n$
$\ {-}{-}{>}\ (\exists i.\ \forall x.\ (poly\ p\ x = (0{::}real))\ {-}{-}{>}\ (\exists m.\ (m \leq n\ \&\ x = i\ m)))$
⟨*proof*⟩

lemmas *poly-roots-index-lemma2* = *conjI* [*THEN poly-roots-index-lemma*, *standard*]

lemma *poly-roots-index-length*: *poly p x* $\neq$ *poly* [] *x* ==>
  $\exists$ *i*. $\forall$ *x*. (*poly p x* = *0*) --> ($\exists$ *n*. *n* $\leq$ *length p* & *x* = *i n*)
⟨*proof*⟩

lemma *poly-roots-finite-lemma*: *poly p x* $\neq$ *poly* [] *x* ==>
  $\exists$ *N i*. $\forall$ *x*. (*poly p x* = *0*) --> ($\exists$ *n*. (*n*::*nat*) < *N* & *x* = *i n*)
⟨*proof*⟩

lemma *real-finite-lemma* [*rule-format* (*no-asm*)]:
  $\forall$ *P*. ($\forall$ *x*. *P x* --> ($\exists$ *n*. *n* < *N* & *x* = (*j*::*nat*=>*real*) *n*))
  --> ($\exists$ *a*. $\forall$ *x*. *P x* --> *x* < *a*)
⟨*proof*⟩

lemma *poly-roots-finite*: (*poly p* $\neq$ *poly* []) =
  ($\exists$ *N j*. $\forall$ *x*. *poly p x* = *0* --> ($\exists$ *n*. (*n*::*nat*) < *N* & *x* = *j n*))
⟨*proof*⟩

Entirety and Cancellation for polynomials

lemma *poly-entire-lemma*: [| *poly p* $\neq$ *poly* [] ; *poly q* $\neq$ *poly* [] |]
  ==> *poly* (*p* *** *q*) $\neq$ *poly* []
⟨*proof*⟩

lemma *poly-entire*: (*poly* (*p* *** *q*) = *poly* []) = ((*poly p* = *poly* []) | (*poly q* = *poly* []))
⟨*proof*⟩

lemma *poly-entire-neg*: (*poly* (*p* *** *q*) $\neq$ *poly* []) = ((*poly p* $\neq$ *poly* []) & (*poly q* $\neq$ *poly* []))
⟨*proof*⟩

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

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

lemma *poly-add-minus-mult-eq*: *poly* (*p* *** *q* +++ --(*p* *** *r*)) = *poly* (*p* *** (*q* +++ -- *r*))
⟨*proof*⟩

lemma *poly-mult-left-cancel*: (*poly* (*p* *** *q*) = *poly* (*p* *** *r*)) = (*poly p* = *poly* [] | *poly q* = *poly r*)
⟨*proof*⟩

lemma *real-mult-zero-disj-iff*: $(x * y = 0) = (x = (0::real) \mid y = 0)$
⟨*proof*⟩

lemma *poly-exp-eq-zero*:
$(poly\ (p\ \%\hat{}\ n) = poly\ []) = (poly\ p = poly\ []\ \&\ n \neq 0)$
⟨*proof*⟩
declare *poly-exp-eq-zero* [*simp*]

lemma *poly-prime-eq-zero*: $poly\ [a,1] \neq poly\ []$
⟨*proof*⟩
declare *poly-prime-eq-zero* [*simp*]

lemma *poly-exp-prime-eq-zero*: $(poly\ ([a,\ 1]\ \%\hat{}\ n) \neq poly\ [])$
⟨*proof*⟩
declare *poly-exp-prime-eq-zero* [*simp*]

A more constructive notion of polynomials being trivial

lemma *poly-zero-lemma*: $poly\ (h\ \#\ t) = poly\ [] \Longrightarrow h = 0\ \&\ poly\ t = poly\ []$
⟨*proof*⟩

lemma *poly-zero*: $(poly\ p = poly\ []) = list\text{-}all\ (\%c.\ c = 0)\ p$
⟨*proof*⟩

declare *real-mult-zero-disj-iff* [*simp*]

lemma *pderiv-aux-iszero* [*rule-format*, *simp*]:
$\forall n.\ list\text{-}all\ (\%c.\ c = 0)\ (pderiv\text{-}aux\ (Suc\ n)\ p) = list\text{-}all\ (\%c.\ c = 0)\ p$
⟨*proof*⟩

lemma *pderiv-aux-iszero-num*: $(number\text{-}of\ n :: nat) \neq 0$
$\Longrightarrow (list\text{-}all\ (\%c.\ c = 0)\ (pderiv\text{-}aux\ (number\text{-}of\ n)\ p) =$
$list\text{-}all\ (\%c.\ c = 0)\ p)$
⟨*proof*⟩

lemma *pderiv-iszero* [*rule-format*]:
$poly\ (pderiv\ p) = poly\ [] \longrightarrow (\exists h.\ poly\ p = poly\ [h])$
⟨*proof*⟩

lemma *pderiv-zero-obj*: $poly\ p = poly\ [] \longrightarrow (poly\ (pderiv\ p) = poly\ [])$
⟨*proof*⟩

lemma *pderiv-zero*: $poly\ p = poly\ [] \Longrightarrow (poly\ (pderiv\ p) = poly\ [])$
⟨*proof*⟩
declare *pderiv-zero* [*simp*]

lemma *poly-pderiv-welldef*: $poly\ p = poly\ q \Longrightarrow (poly\ (pderiv\ p) = poly\ (pderiv\ q))$
⟨*proof*⟩

Basics of divisibility.

lemma *poly-primes*: $([a,\ 1]\ divides\ (p\ {*}{*}{*}\ q)) = ([a,\ 1]\ divides\ p \mid [a,\ 1]\ divides\ q)$
⟨*proof*⟩

lemma *poly-divides-refl*: $p\ divides\ p$
⟨*proof*⟩
declare *poly-divides-refl* [*simp*]

lemma *poly-divides-trans*: $[|\ p\ divides\ q;\ q\ divides\ r\ |] ==> p\ divides\ r$
⟨*proof*⟩

lemma *poly-divides-exp*: $m \leq n ==> (p\ \%\hat{}\ m)\ divides\ (p\ \%\hat{}\ n)$
⟨*proof*⟩

lemma *poly-exp-divides*: $[|\ (p\ \%\hat{}\ n)\ divides\ q;\ \ m {\leq} n\ |] ==> (p\ \%\hat{}\ m)\ divides\ q$
⟨*proof*⟩

lemma *poly-divides-add*:
$[|\ p\ divides\ q;\ p\ divides\ r\ |] ==> p\ divides\ (q\ {+}{+}{+}\ r)$
⟨*proof*⟩

lemma *poly-divides-diff*:
$[|\ p\ divides\ q;\ p\ divides\ (q\ {+}{+}{+}\ r)\ |] ==> p\ divides\ r$
⟨*proof*⟩

lemma *poly-divides-diff2*: $[|\ p\ divides\ r;\ p\ divides\ (q\ {+}{+}{+}\ r)\ |] ==> p\ divides\ q$
⟨*proof*⟩

lemma *poly-divides-zero*: $poly\ p = poly\ [] ==> q\ divides\ p$
⟨*proof*⟩

lemma *poly-divides-zero2*: $q\ divides\ []$
⟨*proof*⟩
declare *poly-divides-zero2* [*simp*]

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

lemma *poly-order-exists-lemma* [*rule-format*]:
$\forall\, p.\ length\ p = d\ {-}{-}{>}\ poly\ p \neq poly\ []$
$\quad {-}{-}{>}\ (\exists\, n\ q.\ p = mulexp\ n\ [-a,\ 1]\ q\ \&\ poly\ q\ a \neq 0)$
⟨*proof*⟩

lemma *poly-order-exists*:
$[|\ length\ p = d;\ poly\ p \neq poly\ []\ |]$
$==> \exists\, n.\ ([-a,\ 1]\ \%\hat{}\ n)\ divides\ p\ \&$
$\quad \sim(([-a,\ 1]\ \%\hat{}\ (Suc\ n))\ divides\ p)$
⟨*proof*⟩

lemma *poly-one-divides*: $[1]$ *divides* p
⟨*proof*⟩
declare *poly-one-divides* [*simp*]

lemma *poly-order*: $poly\ p \neq poly\ []$
  $==> EX!\ n.\ ([-a,\ 1]\ \%\hat{}\ n)\ divides\ p\ \&$
    $\sim(([-a,\ 1]\ \%\hat{}\ (Suc\ n))\ divides\ p)$
⟨*proof*⟩

Order

lemma *some1-equalityD*: $[|\ n = (@n.\ P\ n);\ EX!\ n.\ P\ n\ |] ==> P\ n$
⟨*proof*⟩

lemma *order*:
  $(([-a,\ 1]\ \%\hat{}\ n)\ divides\ p\ \&$
   $\sim(([-a,\ 1]\ \%\hat{}\ (Suc\ n))\ divides\ p)) =$
   $((n = order\ a\ p)\ \&\ \sim(poly\ p = poly\ []))$
⟨*proof*⟩

lemma *order2*: $[|\ poly\ p \neq poly\ []\ |]$
  $==> ([-a,\ 1]\ \%\hat{}\ (order\ a\ p))\ divides\ p\ \&$
    $\sim(([-a,\ 1]\ \%\hat{}\ (Suc(order\ a\ p)))\ divides\ p)$
⟨*proof*⟩

lemma *order-unique*: $[|\ poly\ p \neq poly\ [];\ ([-a,\ 1]\ \%\hat{}\ n)\ divides\ p;$
   $\sim(([-a,\ 1]\ \%\hat{}\ (Suc\ n))\ divides\ p)$
  $|] ==> (n = order\ a\ p)$
⟨*proof*⟩

lemma *order-unique-lemma*: $(poly\ p \neq poly\ []\ \&\ ([-a,\ 1]\ \%\hat{}\ n)\ divides\ p\ \&$
   $\sim(([-a,\ 1]\ \%\hat{}\ (Suc\ n))\ divides\ p))$
  $==> (n = order\ a\ p)$
⟨*proof*⟩

lemma *order-poly*: $poly\ p = poly\ q ==> order\ a\ p = order\ a\ q$
⟨*proof*⟩

lemma *pexp-one*: $p\ \%\hat{}\ (Suc\ 0) = p$
⟨*proof*⟩
declare *pexp-one* [*simp*]

lemma *lemma-order-root* [*rule-format*]:
  $\forall\, p\ a.\ 0 < n\ \&\ [-\ a,\ 1]\ \%\hat{}\ n\ divides\ p\ \&\ \sim\ [-\ a,\ 1]\ \%\hat{}\ (Suc\ n)\ divides\ p$
    $--> poly\ p\ a = 0$
⟨*proof*⟩

lemma *order-root*: $(poly\ p\ a = 0) = ((poly\ p = poly\ [])\ |\ order\ a\ p \neq 0)$
⟨*proof*⟩

lemma *order-divides*: $(([-a,\ 1]\ \%\hat{}\ n)\ divides\ p) = ((poly\ p = poly\ []) \mid n \leq order\ a\ p)$
$\langle proof \rangle$

lemma *order-decomp*:
$poly\ p \neq poly\ []$
$==> \exists\, q.\ (poly\ p = poly\ (([-a,\ 1]\ \%\hat{}\ (order\ a\ p))\ {*}{*}{*}\ q))\ \&$
$\sim([-a,\ 1]\ divides\ q)$
$\langle proof \rangle$

Important composition properties of orders.

lemma *order-mult*: $poly\ (p\ {*}{*}{*}\ q) \neq poly\ []$
$==> order\ a\ (p\ {*}{*}{*}\ q) = order\ a\ p + order\ a\ q$
$\langle proof \rangle$

lemma *lemma-order-pderiv* [*rule-format*]:
$\forall\, p\ q\ a.\ 0 < n\ \&$
$poly\ (pderiv\ p) \neq poly\ []\ \&$
$poly\ p = poly\ ([-\ a,\ 1]\ \%\hat{}\ n\ {*}{*}{*}\ q)\ \&\ \sim\ [-\ a,\ 1]\ divides\ q$
$--> n = Suc\ (order\ a\ (pderiv\ p))$
$\langle proof \rangle$

lemma *order-pderiv*: $[|\ poly\ (pderiv\ p) \neq poly\ [];\ order\ a\ p \neq 0\ |]$
$==> (order\ a\ p = Suc\ (order\ a\ (pderiv\ p)))$
$\langle proof \rangle$

Now justify the standard squarefree decomposition, i.e. f / gcd(f,f'). *) (* 'a la Harrison

lemma *poly-squarefree-decomp-order*: $[|\ poly\ (pderiv\ p) \neq poly\ [];$
$poly\ p = poly\ (q\ {*}{*}{*}\ d);$
$poly\ (pderiv\ p) = poly\ (e\ {*}{*}{*}\ d);$
$poly\ d = poly\ (r\ {*}{*}{*}\ p\ {+}{+}{+}\ s\ {*}{*}{*}\ pderiv\ p)$
$|]\ ==> order\ a\ q = (if\ order\ a\ p = 0\ then\ 0\ else\ 1)$
$\langle proof \rangle$

lemma *poly-squarefree-decomp-order2*: $[|\ poly\ (pderiv\ p) \neq poly\ [];$
$poly\ p = poly\ (q\ {*}{*}{*}\ d);$
$poly\ (pderiv\ p) = poly\ (e\ {*}{*}{*}\ d);$
$poly\ d = poly\ (r\ {*}{*}{*}\ p\ {+}{+}{+}\ s\ {*}{*}{*}\ pderiv\ p)$
$|]\ ==> \forall\, a.\ order\ a\ q = (if\ order\ a\ p = 0\ then\ 0\ else\ 1)$
$\langle proof \rangle$

lemma *order-root2*: $poly\ p \neq poly\ []\ ==> (poly\ p\ a = 0) = (order\ a\ p \neq 0)$
$\langle proof \rangle$

lemma *order-pderiv2*: $[|\ poly\ (pderiv\ p) \neq poly\ [];\ order\ a\ p \neq 0\ |]$
$==> (order\ a\ (pderiv\ p) = n) = (order\ a\ p = Suc\ n)$

⟨*proof*⟩

lemma *rsquarefree-roots*:
  rsquarefree p = (∀ *a*. ~(*poly p a* = *0* & *poly* (*pderiv p*) *a* = *0*))
⟨*proof*⟩

lemma *pmult-one*: [*1*] *** *p* = *p*
⟨*proof*⟩
declare *pmult-one* [*simp*]

lemma *poly-Nil-zero*: *poly* [] = *poly* [*0*]
⟨*proof*⟩

lemma *rsquarefree-decomp*:
    [| *rsquarefree p*; *poly p a* = *0* |]
     ==> ∃ *q*. (*poly p* = *poly* ([−*a*, *1*] *** *q*)) & *poly q 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*⟩

Normalization of a polynomial.

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

The degree of a polynomial.

lemma *lemma-degree-zero* [*rule-format*]:
    list-all (%*c*. *c* = *0*) *p* −−> *pnormalize p* = []
⟨*proof*⟩

lemma *degree-zero*: *poly p* = *poly* [] ==> *degree p* = *0*
⟨*proof*⟩

Tidier versions of finiteness of roots.

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

bound for polynomial.

lemma *poly-mono*: *abs*(*x*) ≤ *k* ==> *abs*(*poly p x*) ≤ *poly* (*map abs p*) *k*
⟨*proof*⟩

⟨*ML*⟩

end

29 Log: Logarithms: Standard Version

theory *Log*
imports *Transcendental*
begin

constdefs

powr :: [*real*,*real*] => *real* (**infixr** *powr 80*)
— exponentation with real exponent
x powr a == *exp*(*a* ∗ *ln x*)

log :: [*real*,*real*] => *real*
— logarithm of x to base a
log a x == *ln x* / *ln a*

lemma *powr-one-eq-one* [*simp*]: *1 powr a* = *1*
⟨*proof*⟩

lemma *powr-zero-eq-one* [*simp*]: *x powr 0* = *1*
⟨*proof*⟩

lemma *powr-one-gt-zero-iff* [*simp*]: (*x powr 1* = *x*) = (*0* < *x*)
⟨*proof*⟩
declare *powr-one-gt-zero-iff* [*THEN iffD2*, *simp*]

lemma *powr-mult*:
[| *0* < *x*; *0* < *y* |] ==> (*x* ∗ *y*) *powr a* = (*x powr a*) ∗ (*y powr a*)
⟨*proof*⟩

lemma *powr-gt-zero* [*simp*]: *0* < *x powr a*
⟨*proof*⟩

lemma *powr-ge-pzero* [*simp*]: *0* <= *x powr y*
⟨*proof*⟩

lemma *powr-not-zero* [*simp*]: *x powr a* ≠ *0*
⟨*proof*⟩

lemma *powr-divide*:
[| *0* < *x*; *0* < *y* |] ==> (*x* / *y*) *powr a* = (*x powr a*)/(*y powr a*)
⟨*proof*⟩

lemma *powr-divide2*: *x powr a* / *x powr b* = *x powr* (*a* − *b*)

⟨*proof*⟩

lemma *powr-add*: $x\ powr\ (a + b) = (x\ powr\ a) * (x\ powr\ b)$
⟨*proof*⟩

lemma *powr-powr*: $(x\ powr\ a)\ powr\ b = x\ powr\ (a * b)$
⟨*proof*⟩

lemma *powr-powr-swap*: $(x\ powr\ a)\ powr\ b = (x\ powr\ b)\ powr\ a$
⟨*proof*⟩

lemma *powr-minus*: $x\ powr\ (-a) = inverse\ (x\ powr\ a)$
⟨*proof*⟩

lemma *powr-minus-divide*: $x\ powr\ (-a) = 1/(x\ powr\ a)$
⟨*proof*⟩

lemma *powr-less-mono*: $[|\ a < b;\ 1 < x\ |] ==> x\ powr\ a < x\ powr\ b$
⟨*proof*⟩

lemma *powr-less-cancel*: $[|\ x\ powr\ a < x\ powr\ b;\ 1 < x\ |] ==> a < b$
⟨*proof*⟩

lemma *powr-less-cancel-iff* [*simp*]: $1 < x ==> (x\ powr\ a < x\ powr\ b) = (a < b)$
⟨*proof*⟩

lemma *powr-le-cancel-iff* [*simp*]: $1 < x ==> (x\ powr\ a \leq x\ powr\ b) = (a \leq b)$
⟨*proof*⟩

lemma *log-ln*: $ln\ x = log\ (exp(1))\ x$
⟨*proof*⟩

lemma *powr-log-cancel* [*simp*]:
$[|\ 0 < a;\ a \neq 1;\ 0 < x\ |] ==> a\ powr\ (log\ a\ x) = x$
⟨*proof*⟩

lemma *log-powr-cancel* [*simp*]: $[|\ 0 < a;\ a \neq 1\ |] ==> log\ a\ (a\ powr\ y) = y$
⟨*proof*⟩

lemma *log-mult*:
$[|\ 0 < a;\ a \neq 1;\ 0 < x;\ 0 < y\ |]$
$==> log\ a\ (x * y) = log\ a\ x + log\ a\ y$
⟨*proof*⟩

lemma *log-eq-div-ln-mult-log*:
$[|\ 0 < a;\ a \neq 1;\ 0 < b;\ b \neq 1;\ 0 < x\ |]$
$==> log\ a\ x = (ln\ b/ln\ a) * log\ b\ x$
⟨*proof*⟩

Base 10 logarithms

lemma *log-base-10-eq1*: $0 < x ==> log\ 10\ x = (ln\ (exp\ 1)\ /\ ln\ 10) * ln\ x$
⟨*proof*⟩

lemma *log-base-10-eq2*: $0 < x ==> log\ 10\ x = (log\ 10\ (exp\ 1)) * ln\ x$
⟨*proof*⟩

lemma *log-one* [*simp*]: $log\ a\ 1 = 0$
⟨*proof*⟩

lemma *log-eq-one* [*simp*]: $[|\ 0 < a;\ a \neq 1\ |] ==> log\ a\ a = 1$
⟨*proof*⟩

lemma *log-inverse*:
$[|\ 0 < a;\ a \neq 1;\ 0 < x\ |] ==> log\ a\ (inverse\ x) = -\ log\ a\ x$
⟨*proof*⟩

lemma *log-divide*:
$[|0 < a;\ a \neq 1;\ 0 < x;\ 0 < y|] ==> log\ a\ (x/y) = log\ a\ x - log\ a\ y$
⟨*proof*⟩

lemma *log-less-cancel-iff* [*simp*]:
$[|\ 1 < a;\ 0 < x;\ 0 < y\ |] ==> (log\ a\ x < log\ a\ y) = (x < y)$
⟨*proof*⟩

lemma *log-le-cancel-iff* [*simp*]:
$[|\ 1 < a;\ 0 < x;\ 0 < y\ |] ==> (log\ a\ x \leq log\ a\ y) = (x \leq y)$
⟨*proof*⟩

lemma *powr-realpow*: $0 < x ==> x\ powr\ (real\ n) = x\hat{}n$
⟨*proof*⟩

lemma *powr-realpow2*: $0 <= x ==> 0 < n ==> x\hat{}n = (if\ (x = 0)\ then\ 0$ $else\ x\ powr\ (real\ n))$
⟨*proof*⟩

lemma *ln-pwr*: $0 < x ==> 0 < y ==> ln(x\ powr\ y) = y * ln\ x$
⟨*proof*⟩

lemma *ln-bound*: $1 <= x ==> ln\ x <= x$
⟨*proof*⟩

lemma *powr-mono*: $a <= b ==> 1 <= x ==> x\ powr\ a <= x\ powr\ b$
⟨*proof*⟩

lemma *ge-one-powr-ge-zero*: $1 <= x ==> 0 <= a ==> 1 <= x\ powr\ a$
⟨*proof*⟩

lemma *powr-less-mono2*: $0 < a ==> 0 < x ==> x < y ==> x\ powr\ a <$

y powr a
⟨*proof*⟩

lemma *powr-less-mono2-neg*: *a < 0 ==> 0 < x ==> x < y ==> y powr a < x powr a*
⟨*proof*⟩

lemma *powr-mono2*: *0 <= a ==> 0 < x ==> x <= y ==> x powr a <= y powr a*
⟨*proof*⟩

lemma *ln-powr-bound*: *1 <= x ==> 0 < a ==> ln x <= (x powr a) / a*
⟨*proof*⟩

lemma *ln-powr-bound2*: *1 < x ==> 0 < a ==> (ln x) powr a <= (a powr a) * x*
⟨*proof*⟩

lemma *LIMSEQ-neg-powr*: *0 < s ==> (%x. (real x) powr − s) −−−−> 0*
⟨*proof*⟩

⟨*ML*⟩

end

30 MacLaurin: MacLaurin Series

theory *MacLaurin*
imports *Log*
begin

30.1 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 ==>
*∃ B. f h = (∑ m=0..<n. (j m / real (fact m)) * (h^m)) +*
*(B * ((h^n) / real(fact n)))*
⟨*proof*⟩

lemma *eq-diff-eq'*: *(x = y − z) = (y = x + (z::real))*
⟨*proof*⟩

A crude tactic to differentiate by proof.

⟨*ML*⟩

lemma *Maclaurin-lemma2*:
$[\![\forall m\ t.\ m < n \wedge 0{\le}t \wedge t{\le}h \longrightarrow DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t;$
$\quad n = Suc\ k;$
$\quad difg =$
$\quad (\lambda m\ t.\ diff\ m\ t -$
$\qquad ((\sum p = 0..<n - m.\ diff\ (m + p)\ 0\ /\ real\ (fact\ p) * t\ \hat{}\ p) +$
$\qquad B * (t\ \hat{}\ (n - m)\ /\ real\ (fact\ (n - m)))))]\!] ==>$
$\quad \forall m\ t.\ m < n\ \&\ 0 \le t\ \&\ t \le h -->$
$\qquad DERIV\ (difg\ m)\ t :> difg\ (Suc\ m)\ t$

⟨*proof*⟩

lemma *Maclaurin-lemma3*:
$[\![\forall k\ t.\ k < Suc\ m \wedge 0{\le}t\ \&\ t{\le}h \longrightarrow DERIV\ (difg\ k)\ t :> difg\ (Suc\ k)\ t;$
$\quad \forall k{<}Suc\ m.\ difg\ k\ 0 = 0;\ DERIV\ (difg\ n)\ t :> 0;\ \ n < m;\ 0 < t;$
$\quad t < h]\!]$
$==> \exists ta.\ 0 < ta\ \&\ ta < t\ \&\ DERIV\ (difg\ (Suc\ n))\ ta :> 0$

⟨*proof*⟩

lemma *Maclaurin*:
$[\![\ 0 < h;\ 0 < n;\ diff\ 0 = f;$
$\quad \forall m\ t.\ m < n\ \&\ 0 \le t\ \&\ t \le h --> DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t\ ]\!]$
$==> \exists t.\ 0 < t\ \&$
$\qquad t < h\ \&$
$\qquad f\ h =$
$\qquad setsum\ (\%m.\ (diff\ m\ 0\ /\ real\ (fact\ m)) * h\ \hat{}\ m)\ \{0..<n\} +$
$\qquad (diff\ n\ t\ /\ real\ (fact\ n)) * h\ \hat{}\ n$

⟨*proof*⟩

lemma *Maclaurin-objl*:
$0 < h\ \&\ 0 < n\ \&\ diff\ 0 = f\ \&$
$\quad (\forall m\ t.\ m < n\ \&\ 0 \le t\ \&\ t \le h --> DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t)$
$--> (\exists t.\ 0 < t\ \&$
$\qquad t < h\ \&$
$\qquad f\ h =$
$\qquad (\sum m{=}0..<n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$\qquad diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n)$

⟨*proof*⟩

lemma *Maclaurin2*:
$[\![\ 0 < h;\ diff\ 0 = f;$
$\quad \forall m\ t.$
$\qquad m < n\ \&\ 0 \le t\ \&\ t \le h --> DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t\ ]\!]$
$==> \exists t.\ 0 < t\ \&$
$\qquad t \le h\ \&$

$f\ h =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n$
⟨*proof*⟩

lemma *Maclaurin2-objl*:
$0 < h\ \&\ diff\ 0 = f\ \&$
$(\forall m\ t.$
$m < n\ \&\ 0 \leq t\ \&\ t \leq h\ {-}{-}{>}\ DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t)$
$-->\ (\exists t.\ 0 < t\ \&$
$t \leq h\ \&$
$f\ h =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n)$
⟨*proof*⟩

lemma *Maclaurin-minus*:
$[|\ h < 0;\ 0 < n;\ diff\ 0 = f;$
$\forall m\ t.\ m < n\ \&\ h \leq t\ \&\ t \leq 0\ {-}{-}{>}\ DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t\ |]$
$==>\ \exists t.\ h < t\ \&$
$t < 0\ \&$
$f\ h =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n$
⟨*proof*⟩

lemma *Maclaurin-minus-objl*:
$(h < 0\ \&\ 0 < n\ \&\ diff\ 0 = f\ \&$
$(\forall m\ t.$
$m < n\ \&\ h \leq t\ \&\ t \leq 0\ {-}{-}{>}\ DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t))$
$-->\ (\exists t.\ h < t\ \&$
$t < 0\ \&$
$f\ h =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * h\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * h\ \hat{}\ n)$
⟨*proof*⟩

30.2 More Convenient "Bidirectional" Version.

lemma *Maclaurin-bi-le-lemma* [*rule-format*]:
$0 < n \longrightarrow$
$diff\ 0\ 0 =$
$(\sum m = 0..{<}n.\ diff\ m\ 0 * 0\ \hat{}\ m\ /\ real\ (fact\ m)) +$
$diff\ n\ 0 * 0\ \hat{}\ n\ /\ real\ (fact\ n)$
⟨*proof*⟩

lemma *Maclaurin-bi-le*:
$[|\ diff\ 0 = f;$
$\forall m\ t.\ m < n\ \&\ abs\ t \leq abs\ x\ {-}{-}{>}\ DERIV\ (diff\ m)\ t :> diff\ (Suc\ m)\ t\ |]$

$==> \exists t.\ abs\ t \leq abs\ x\ \&$
$f\ x =$
$(\sum m{=}0..{<}n.\ diff\ m\ 0\ /\ real\ (fact\ m) * x\ \hat{}\ m) +$
$diff\ n\ t\ /\ real\ (fact\ n) * x\ \hat{}\ n$
$\langle proof \rangle$

lemma *Maclaurin-all-lt*:
$[|\ diff\ 0 = f;$
$\forall m\ x.\ DERIV\ (diff\ m)\ x :> diff(Suc\ m)\ x;$
$x\ \tilde{}{=}\ 0;\ 0 < n$
$|] ==> \exists t.\ 0 < abs\ t\ \&\ abs\ t < abs\ x\ \&$
$f\ x = (\sum m{=}0..{<}n.\ (diff\ m\ 0\ /\ real\ (fact\ m)) * x\ \hat{}\ m) +$
$(diff\ n\ t\ /\ real\ (fact\ n)) * x\ \hat{}\ n$
$\langle proof \rangle$

lemma *Maclaurin-all-lt-objl*:
$diff\ 0 = f\ \&$
$(\forall m\ x.\ DERIV\ (diff\ m)\ x :> diff(Suc\ m)\ x)\ \&$
$x\ \tilde{}{=}\ 0\ \&\ 0 < n$
$--> (\exists t.\ 0 < abs\ t\ \&\ abs\ t < abs\ x\ \&$
$f\ x = (\sum m{=}0..{<}n.\ (diff\ m\ 0\ /\ real\ (fact\ m)) * x\ \hat{}\ m) +$
$(diff\ n\ t\ /\ real\ (fact\ n)) * x\ \hat{}\ n)$
$\langle proof \rangle$

lemma *Maclaurin-zero* [*rule-format*]:
$x = (0::real)$
$==> 0 < n -->$
$(\sum m{=}0..{<}n.\ (diff\ m\ (0::real)\ /\ real\ (fact\ m)) * x\ \hat{}\ m) =$
$diff\ 0\ 0$
$\langle proof \rangle$

lemma *Maclaurin-all-le*: $[|\ diff\ 0 = f;$
$\forall m\ x.\ DERIV\ (diff\ m)\ x :> diff\ (Suc\ m)\ x$
$|] ==> \exists t.\ abs\ t \leq abs\ x\ \&$
$f\ x = (\sum m{=}0..{<}n.\ (diff\ m\ 0\ /\ real\ (fact\ m)) * x\ \hat{}\ m) +$
$(diff\ n\ t\ /\ real\ (fact\ n)) * x\ \hat{}\ n$
$\langle proof \rangle$

lemma *Maclaurin-all-le-objl*: $diff\ 0 = f\ \&$
$(\forall m\ x.\ DERIV\ (diff\ m)\ x :> diff\ (Suc\ m)\ x)$
$--> (\exists t.\ abs\ t \leq abs\ x\ \&$
$f\ x = (\sum m{=}0..{<}n.\ (diff\ m\ 0\ /\ real\ (fact\ m)) * x\ \hat{}\ m) +$
$(diff\ n\ t\ /\ real\ (fact\ n)) * x\ \hat{}\ n)$
$\langle proof \rangle$

30.3 Version for Exponential Function

lemma *Maclaurin-exp-lt*: $[|\ x\ \tilde{}{=}\ 0;\ 0 < n\ |]$
$==> (\exists t.\ 0 < abs\ t\ \&$

$abs\ t < abs\ x\ \&$
$exp\ x = (\sum m{=}0..{<}n.\ (x\ \hat{}\ m)\ /\ real\ (fact\ m)) +$
$(exp\ t\ /\ real\ (fact\ n)) * x\ \hat{}\ n)$
⟨*proof*⟩

lemma *Maclaurin-exp-le*:
$\exists\, t.\ abs\ t \le abs\ x\ \&$
$exp\ x = (\sum m{=}0..{<}n.\ (x\ \hat{}\ m)\ /\ real\ (fact\ m)) +$
$(exp\ t\ /\ real\ (fact\ n)) * x\ \hat{}\ n$
⟨*proof*⟩

30.4 Version for Sine Function

lemma *MVT2*:
$[|\ a < b;\ \forall x.\ a \le x\ \&\ x \le b\ {-}{-}{>}\ DERIV\ f\ x :> f'(x)\ |]$
$={=}{>} \exists\, z.\ a < z\ \&\ z < b\ \&\ (f\ b - f\ a = (b - a) * f'(z))$
⟨*proof*⟩

lemma *mod-exhaust-less-4*:
$m\ mod\ 4 = 0\ |\ m\ mod\ 4 = 1\ |\ m\ mod\ 4 = 2\ |\ m\ mod\ 4 = (3{::}nat)$
⟨*proof*⟩

lemma *Suc-Suc-mult-two-diff-two* [*rule-format*, *simp*]:
$0 < n\ {-}{-}{>}\ Suc\ (Suc\ (2 * n - 2)) = 2{*}n$
⟨*proof*⟩

lemma *lemma-Suc-Suc-4n-diff-2* [*rule-format*, *simp*]:
$0 < n\ {-}{-}{>}\ Suc\ (Suc\ (4{*}n - 2)) = 4{*}n$
⟨*proof*⟩

lemma *Suc-mult-two-diff-one* [*rule-format*, *simp*]:
$0 < n\ {-}{-}{>}\ Suc\ (2 * n - 1) = 2{*}n$
⟨*proof*⟩

It is unclear why so many variant results are needed.

lemma *Maclaurin-sin-expansion2*:
$\exists\, t.\ abs\ t \le abs\ x\ \&$
$sin\ x =$
$(\sum m{=}0..{<}n.\ (if\ even\ m\ then\ 0$
$else\ ((-\ 1)\ \hat{}\ ((m - (Suc\ 0))\ div\ 2))\ /\ real\ (fact\ m)) *$
$x\ \hat{}\ m)$
$+ ((sin(t + 1/2 * real\ (n) * pi)\ /\ real\ (fact\ n)) * x\ \hat{}\ n)$
⟨*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*:
$[| \ 0 < n;\ 0 < x \ |] ==>$
$\exists t.\ 0 < t\ \&\ t < x\ \&$
$sin\ x =$
$(\sum m=0..<n.\ (if\ even\ m\ then\ 0$
$else\ ((-\ 1)\ \hat{}\ ((m - (Suc\ 0))\ div\ 2))\ /\ real\ (fact\ m)) *$
$x \,\hat{}\, m)$
$+ ((sin(t + 1/2 * real(n) *pi) / real\ (fact\ n)) * x \,\hat{}\, n)$
⟨proof⟩

lemma *Maclaurin-sin-expansion4*:
$0 < x ==>$
$\exists t.\ 0 < t\ \&\ t \le x\ \&$
$sin\ x =$
$(\sum m=0..<n.\ (if\ even\ m\ then\ 0$
$else\ ((-\ 1)\ \hat{}\ ((m - (Suc\ 0))\ div\ 2))\ /\ real\ (fact\ m)) *$
$x \,\hat{}\, m)$
$+ ((sin(t + 1/2 * real\ (n) *pi) / real\ (fact\ n)) * x \,\hat{}\, n)$
⟨proof⟩

30.5 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*:
$\exists t.\ abs\ t \le abs\ x\ \&$
$cos\ x =$
$(\sum m=0..<n.\ (if\ even\ m$
$then\ (-\ 1)\ \hat{}\ (m\ div\ 2)/(real\ (fact\ m))$
$else\ 0) *$
$x \,\hat{}\, m)$
$+ ((cos(t + 1/2 * real\ (n) *pi) / real\ (fact\ n)) * x \,\hat{}\, n)$
⟨proof⟩

lemma *Maclaurin-cos-expansion2*:
$[| \ 0 < x;\ 0 < n \ |] ==>$
$\exists t.\ 0 < t\ \&\ t < x\ \&$
$cos\ x =$
$(\sum m=0..<n.\ (if\ even\ m$
$then\ (-\ 1)\ \hat{}\ (m\ div\ 2)/(real\ (fact\ m))$

else 0) $*$
x ^ m)
+ ((*cos*(t + *1/2* $*$ *real* (n) $*$*pi*) / *real* (*fact* n)) $*$ x ^ n)
⟨*proof*⟩

lemma *Maclaurin-minus-cos-expansion*:
[| $x < 0$; $0 < n$ |] ==>
$\exists\, t.\ x < t$ & $t < 0$ &
cos x =
($\sum$ m=*0*..<n. (*if even* m
then ($-$ *1*) ^ (m *div 2*)/(*real* (*fact* m))
else 0) $*$
x ^ m)
+ ((*cos*(t + *1/2* $*$ *real* (n) $*$*pi*) / *real* (*fact* n)) $*$ x ^ n)
⟨*proof*⟩

lemma *sin-bound-lemma*:
[|$x = y$; *abs* $u \leq$ (v::*real*) |] ==> $|(x + u) - y| \leq v$
⟨*proof*⟩

lemma *Maclaurin-sin-bound*:
abs(*sin* x $-$ ($\sum$ m=*0*..<n. (*if even* m *then 0 else* (($-$ *1*) ^ ((m $-$ (*Suc 0*)) *div 2*)) / *real* (*fact* m)) $*$
x ^ m)) $\leq$ *inverse*(*real* (*fact* n)) $*$ $|x|$ ^ n
⟨*proof*⟩

end

31 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*: $0 < n$ *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*: $a \leq c$ $c < b$
shows $\exists$ t. $c < t$ & $t < b$ &
f b = *setsum* (%m. (*diff* m c / *real* (*fact* m)) $*$ (b $-$ c)^m) {*0*..<n} +
(*diff* n t / *real* (*fact* n)) $*$ (b $-$ c)^n

⟨*proof*⟩

lemma *taylor-down*:
assumes *INIT*: $0 < n$ *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*: $0 < n$ *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

32 Integration: Theory of Integration

theory *Integration*
imports *MacLaurin*
begin

We follow John Harrison in formalizing the Gauge integral.

constdefs

— Partitions and tagged partitions etc.

partition :: [(*real*∗*real*),*nat* => *real*] => *bool*
partition == %(*a*,*b*) $D.\ D\ 0 = a\ \&$
$(\exists N.\ (\forall n < N.\ D(n) < D(Suc\ n))\ \&$
$(\forall n \geq N.\ D(n) = b))$

psize :: (*nat* => *real*) => *nat*
psize D == *SOME* $N.\ (\forall n < N.\ D(n) < D(Suc\ n))\ \&$
$(\forall n \geq N.\ D(n) = D(N))$

tpart :: [(*real*∗*real*),((*nat* => *real*)∗(*nat* =>*real*))] => *bool*
tpart == %(*a*,*b*) (D,p). *partition*$(a,b)\ D\ \&$
$(\forall n.\ D(n) \leq p(n)\ \&\ p(n) \leq D(Suc\ n))$

— Gauges and gauge-fine divisions

gauge :: [*real* => *bool*, *real* => *real*] => *bool*
gauge E g == $\forall x$. *E x* −−> *0* < *g*(*x*)

fine :: [*real* => *real*, ((*nat* => *real*)∗(*nat* => *real*))] => *bool*
fine == % *g* (*D*,*p*). $\forall n$. *n* < (*psize D*) −−> *D*(*Suc n*) − *D*(*n*) < *g*(*p n*)

— Riemann sum

rsum :: [((*nat*=>*real*)∗(*nat*=>*real*)),*real*=>*real*] => *real*
rsum == %(*D*,*p*) *f*. $\sum$ *n*=*0*..<*psize*(*D*). *f*(*p n*) ∗ (*D*(*Suc n*) − *D*(*n*))

— Gauge integrability (definite)

Integral :: [(*real*∗*real*),*real*=>*real*,*real*] => *bool*
Integral == %(*a*,*b*) *f k*. $\forall e$ > *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*]:

$partition(a,b)\ D\ \&\ m + Suc\ d \leq n\ \&\ n \leq (psize\ D) --> D(m) < D(m + Suc\ d)$
$\langle proof \rangle$

lemma *less-eq-add-Suc*: $m < n ==> \exists\, d.\ n = m + Suc\ d$
$\langle proof \rangle$

lemma *partition-lt-gen*:
  $[|partition(a,b)\ D;\ m < n;\ n \leq (psize\ D)|] ==> D(m) < D(n)$
$\langle proof \rangle$

lemma *partition-lt*: $partition(a,b)\ D ==> n < (psize\ D) ==> D(0) < D(Suc\ n)$
$\langle proof \rangle$

lemma *partition-le*: $partition(a,b)\ D ==> a \leq b$
$\langle proof \rangle$

lemma *partition-gt*: $[|partition(a,b)\ D;\ n < (psize\ D)|] ==> D(n) < D(psize\ D)$
$\langle proof \rangle$

lemma *partition-eq*: $partition(a,b)\ D ==> ((a = b) = (psize\ D = 0))$
$\langle proof \rangle$

lemma *partition-lb*: $partition(a,b)\ D ==> a \leq D(r)$
$\langle proof \rangle$

lemma *partition-lb-lt*: $[|\ partition(a,b)\ D;\ psize\ D\ \tilde{}{=}\ 0\ |] ==> a < D(Suc\ n)$
$\langle proof \rangle$

lemma *partition-ub*: $partition(a,b)\ D ==> D(r) \leq b$
$\langle proof \rangle$

lemma *partition-ub-lt*: $[|\ partition(a,b)\ D;\ n < psize\ D\ |] ==> D(n) < b$
$\langle proof \rangle$

lemma *lemma-partition-append1*:
  $[|\ partition\ (a,\ b)\ D1;\ partition\ (b,\ c)\ D2\ |]$
    $==> (\forall\, n < psize\ D1 + psize\ D2.$
        $(if\ n < psize\ D1\ then\ D1\ n\ else\ D2\ (n - psize\ D1))$
        $< (if\ Suc\ n < psize\ D1\ then\ D1\ (Suc\ n)$
          $else\ D2\ (Suc\ n - psize\ D1)))\ \&$
      $(\forall\, n \geq psize\ D1 + psize\ D2.$
        $(if\ n < psize\ D1\ then\ D1\ n\ else\ D2\ (n - psize\ D1)) =$
        $(if\ psize\ D1 + psize\ D2 < psize\ D1\ then\ D1\ (psize\ D1 + psize\ D2)$
         $else\ D2\ (psize\ D1 + psize\ D2 - psize\ D1)))$
$\langle proof \rangle$

lemma *lemma-psize1*:
  $[|\ partition\ (a,\ b)\ D1;\ partition\ (b,\ c)\ D2;\ N < psize\ D1\ |]$

==> $D1(N) < D2\ (psize\ D2)$
⟨proof⟩

lemma *lemma-partition-append2*:
[| $partition\ (a,\ b)\ D1$; $partition\ (b,\ c)\ D2$ |]
==> $psize\ (\%n.\ if\ n < psize\ D1\ then\ D1\ n\ else\ D2\ (n - psize\ D1)) =$
$psize\ D1 + psize\ D2$
⟨proof⟩

lemma *tpart-eq-lhs-rhs*: [|$psize\ D = 0$; $tpart(a,b)\ (D,p)$|] ==> $a = b$
⟨proof⟩

lemma *tpart-partition*: $tpart(a,b)\ (D,p)$ ==> $partition(a,b)\ D$
⟨proof⟩

lemma *partition-append*:
[| $tpart(a,b)\ (D1,p1)$; $fine(g)\ (D1,p1)$;
$tpart(b,c)\ (D2,p2)$; $fine(g)\ (D2,p2)$ |]
==> $\exists D\ p.\ tpart(a,c)\ (D,p)\ \&\ fine(g)\ (D,p)$
⟨proof⟩

We can always find a division that is fine wrt any gauge

lemma *partition-exists*:
[| $a \leq b$; $gauge(\%x.\ a \leq x\ \&\ x \leq b)\ g$ |]
==> $\exists D\ p.\ tpart(a,b)\ (D,p)\ \&\ fine\ g\ (D,p)$
⟨proof⟩

Lemmas about combining gauges

lemma *gauge-min*:
[| $gauge(E)\ g1$; $gauge(E)\ g2$ |]
==> $gauge(E)\ (\%x.\ if\ g1(x) < g2(x)\ then\ g1(x)\ else\ g2(x))$
⟨proof⟩

lemma *fine-min*:
$fine\ (\%x.\ if\ g1(x) < g2(x)\ then\ g1(x)\ else\ g2(x))\ (D,p)$
==> $fine(g1)\ (D,p)\ \&\ fine(g2)\ (D,p)$
⟨proof⟩

The integral is unique if it exists

lemma *Integral-unique*:
[| $a \leq b$; $Integral(a,b)\ f\ k1$; $Integral(a,b)\ f\ k2$ |] ==> $k1 = k2$
⟨proof⟩

lemma *Integral-zero* [*simp*]: $Integral(a,a)\ f\ 0$
⟨proof⟩

lemma *sumr-partition-eq-diff-bounds* [*simp*]:
$(\sum n{=}0..{<}m.\ D\ (Suc\ n) - D\ n{::}real) = D(m) - D\ 0$
⟨proof⟩

lemma *Integral-eq-diff-bounds*: $a \le b$ ==> *Integral*(a,b) (%x. *1*) $(b - a)$
⟨*proof*⟩

lemma *Integral-mult-const*: $a \le b$ ==> *Integral*(a,b) (%x. c) $(c*(b - a))$
⟨*proof*⟩

lemma *Integral-mult*:
[| $a \le b$; *Integral*(a,b) f k |] ==> *Integral*(a,b) (%x. $c * f\,x$) $(c * k)$
⟨*proof*⟩

Fundamental theorem of calculus (Part I)

"Straddle Lemma" : Swartz and Thompson: AMM 95(7) 1988

lemma *choiceP*: $\forall x.\ P(x)$ --> $(\exists y.\ Q\ x\ y)$ ==> $\exists f.\ (\forall x.\ P(x)$ --> $Q\ x\ (f\ x))$
⟨*proof*⟩

lemma *strad1*:
$[\![\forall xa{::}real.\ xa \neq x \wedge |xa + - x| < s \longrightarrow$
$|(f\,xa - f\,x) / (xa - x) + - f'\,x| * 2 < e;$
$0 < e;\ a \le x;\ x \le b;\ 0 < s]\!]$
$\Longrightarrow \forall z.\ |z - x| < s$ -->$|f\,z - f\,x - f'\,x * (z - x)| * 2 \le e * |z - x|$
⟨*proof*⟩

lemma *lemma-straddle*:
[| $\forall x.\ a \le x$ & $x \le b$ --> *DERIV* $f\ x$:> $f'(x)$; $0 < e$ |]
==> $\exists g.$ *gauge*(%x. $a \le x$ & $x \le b$) g &
($\forall x\ u\ v.\ a \le u$ & $u \le x$ & $x \le v$ & $v \le b$ & $(v - u) < g(x)$
--> $|(f(v) - f(u)) - (f'(x) * (v - u))| \le e * (v - u)$)
⟨*proof*⟩

lemma *FTC1*: [|$a \le b$; $\forall x.\ a \le x$ & $x \le b$ --> *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*:
[| $a \le b$; $b \le c$; *Integral*(a,b) f' *k1*; *Integral*(b,c) f' *k2*;
$\forall x.\ a \le x$ & $x \le c$ --> *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*: $partition\ (a,\ c)\ D ==> \sim\ (\exists\, n.\ c < D(n))$
⟨*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*⟩

32.1 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*:
$[|\ a \leq D\ n;\ D\ n < b;\ partition\ (a,\ b)\ D\ |]$
$==> psize\ (\%x.\ if\ D\ x < D\ n\ then\ D(x)\ else\ D\ n) = n$
⟨*proof*⟩

lemma *lemma-psize-left-less-psize*:
  partition $(a,\ b)\ D$
  $==>$ *psize* $(\%x.\ if\ D\ x < D\ n\ then\ D(x)\ else\ D\ n) \leq psize\ D$
⟨*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*:
  $[|\ partition(a,b)\ D;\ D\ n < b\ |]\ ==>\ psize\ (\%x.\ D\ (x + n)) \leq psize\ D - n$
⟨*proof*⟩

lemma *psize-le-n*: $partition\ (a,\ D\ n)\ D\ ==>\ psize\ D \leq n$
⟨*proof*⟩

lemma *better-lemma-psize-right-eq1a*:
  $partition(a,D\ n)\ D\ ==>\ psize\ (\%x.\ D\ (x + n)) \leq psize\ D - n$
⟨*proof*⟩

lemma *better-lemma-psize-right-eq*:
  $partition(a,b)\ D\ ==>\ psize\ (\%x.\ D\ (x + n)) \leq psize\ D - n$
⟨*proof*⟩

lemma *lemma-psize-right-eq1*:
  $[|\ partition(a,b)\ D;\ D\ n < b\ |]\ ==>\ psize\ (\%x.\ D\ (x + n)) \leq psize\ D$
⟨*proof*⟩

lemma *lemma-psize-right-eq1a*:
  $partition(a,D\ n)\ D\ ==>\ psize\ (\%x.\ D\ (x + n)) \leq psize\ D$

⟨*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* ≤ *b*; *Integral*(*a*,*b*) *f k1*; *Integral*(*a*,*b*) *g k2* |]
==> *Integral*(*a*,*b*) (%*x*. *f x* + *g x*) (*k1* + *k2*)
⟨*proof*⟩

lemma *partition-lt-gen2*:
[| *partition*(*a*,*b*) *D*; *r* < *psize D* |] ==> *0* < *D* (*Suc r*) − *D r*
⟨*proof*⟩

lemma *lemma-Integral-le*:
$[| \; \forall x.\ a \le x \ \&\ x \le b \ {-}{-}{>}\ f\,x \le g\ x;$
$\quad tpart(a,b)\ (D,p)$
$|] \ {=}{=}{>}\ \forall\, n \le psize\ D.\ f\ (p\ n) \le g\ (p\ n)$
⟨*proof*⟩

lemma *lemma-Integral-rsum-le*:
$[| \; \forall x.\ a \le x \ \&\ x \le b \ {-}{-}{>}\ f\,x \le g\ x;$
$\quad tpart(a,b)\ (D,p)$
$|] \ {=}{=}{>}\ rsum(D,p)\ f \le rsum(D,p)\ g$
⟨*proof*⟩

lemma *Integral-le*:
$[| \; a \le b;$
$\quad \forall x.\ a \le x \ \&\ x \le b \ {-}{-}{>}\ f(x) \le g(x);$
$\quad Integral(a,b)\ f\ k1;\ Integral(a,b)\ g\ k2$
$|] \ {=}{=}{>}\ k1 \le k2$
⟨*proof*⟩

lemma *Integral-imp-Cauchy*:
$(\exists k.\ Integral(a,b)\ f\ k)\ {=}{=}{>}$
$(\forall\, e > 0.\ \exists g.\ gauge\ (\%x.\ a \le x \ \&\ x \le b)\ g\ \&$
$\quad (\forall\, D1\ D2\ p1\ p2.$
$\quad\quad tpart(a,b)\ (D1,\ p1)\ \&\ fine\ g\ (D1,p1)\ \&$
$\quad\quad tpart(a,b)\ (D2,\ p2)\ \&\ fine\ g\ (D2,p2)\ {-}{-}{>}$
$\quad\quad |rsum(D1,p1)\ f - rsum(D2,p2)\ f| < e))$
⟨*proof*⟩

lemma *Cauchy-iff2*:
$Cauchy\ X =$
$(\forall j.\ (\exists M.\ \forall\, m \ge M.\ \forall\, n \ge M.\ |X\ m + -\ X\ n| < inverse(real\ (Suc\ j))))$
⟨*proof*⟩

lemma *partition-exists2*:
$[| \; a \le b;\ \forall n.\ gauge\ (\%x.\ a \le x \ \&\ x \le b)\ (fa\ n)\ |]$
${=}{=}{>}\ \forall n.\ \exists D\ p.\ tpart\ (a,\ b)\ (D,\ p)\ \&\ fine\ (fa\ n)\ (D,\ p)$
⟨*proof*⟩

lemma *monotonic-anti-derivative*:
$[| \; a \le b;\ \forall c.\ a \le c \ \&\ c \le b \ {-}{-}{>}\ f'\ c \le g'\ c;$
$\quad \forall x.\ DERIV\ f\ x :> f'\ x;\ \forall x.\ DERIV\ g\ x :> g'\ x\ |]$
${=}{=}{>}\ f\ b - f\ a \le g\ b - g\ a$
⟨*proof*⟩

end

33 HTranscendental: Nonstandard Extensions of Transcendental Functions

theory *HTranscendental*
imports *Transcendental Integration*
begin

really belongs in Transcendental

lemma *sqrt-divide-self-eq*:
  assumes *nneg*: $0 \leq x$
  shows *sqrt x / x = inverse (sqrt x)*
⟨*proof*⟩

constdefs

exphr :: *real => hypreal*
  — define exponential function using standard part
  exphr x == st(sumhr (0, whn, %n. inverse(real (fact n)) ∗ (x ˆ n)))

sinhr :: *real => hypreal*
  sinhr x == st(sumhr (0, whn, %n. (if even(n) then 0 else
    ((−1) ˆ ((n − 1) div 2))/(real (fact n))) ∗ (x ˆ n)))

coshr :: *real => hypreal*
  coshr x == st(sumhr (0, whn, %n. (if even(n) then
    ((−1) ˆ (n div 2))/(real (fact n)) else 0) ∗ x ˆ n))

33.1 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*) ˆ *2* = *x*) = (*0* ≤ *x*)
⟨*proof*⟩

lemma *hypreal-sqrt-gt-zero-pow2*: !!*x*. *0* < *x* ==> (∗*f*∗ *sqrt*) (*x*) ˆ *2* = *x*
⟨*proof*⟩

lemma *hypreal-sqrt-pow2-gt-zero*: *0* < *x* ==> *0* < (∗*f*∗ *sqrt*) (*x*) ˆ *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*:
$[|0 \leq x;\ 0 \leq y\ |] ==>$
$(*f* sqrt)(x*y) = (*f* sqrt)(x) * (*f* sqrt)(y)$
⟨proof⟩

lemma *hypreal-sqrt-approx-zero* [*simp*]:
$0 < x ==> ((*f* sqrt)(x)\ @= 0) = (x\ @= 0)$
⟨proof⟩

lemma *hypreal-sqrt-approx-zero2* [*simp*]:
$0 \leq x ==> ((*f* sqrt)(x)\ @= 0) = (x\ @= 0)$
⟨proof⟩

lemma *hypreal-sqrt-sum-squares* [*simp*]:
$((*f* sqrt)(x*x + y*y + z*z)\ @= 0) = (x*x + y*y + z*z\ @= 0)$
⟨proof⟩

lemma *hypreal-sqrt-sum-squares2* [*simp*]:
$((*f* sqrt)(x*x + y*y)\ @= 0) = (x*x + y*y\ @= 0)$
⟨proof⟩

lemma *hypreal-sqrt-gt-zero*: $!!x.\ 0 < x ==> 0 < (*f* sqrt)(x)$
⟨proof⟩

lemma *hypreal-sqrt-ge-zero*: $0 \leq x ==> 0 \leq (*f* sqrt)(x)$
⟨proof⟩

lemma *hypreal-sqrt-hrabs* [*simp*]: $!!x.\ (*f* sqrt)(x$ ^ $2) = abs(x)$
⟨proof⟩

lemma *hypreal-sqrt-hrabs2* [*simp*]: $!!x.\ (*f* sqrt)(x*x) = abs(x)$
⟨proof⟩

lemma *hypreal-sqrt-hyperpow-hrabs* [*simp*]:
$!!x.\ (*f* sqrt)(x\ pow\ (hypnat\text{-}of\text{-}nat\ 2)) = abs(x)$
⟨proof⟩

lemma *star-sqrt-HFinite*: $[\![x \in HFinite;\ 0 \leq x]\!] \Longrightarrow (*f* sqrt)\ x \in HFinite$
⟨proof⟩

lemma *st-hypreal-sqrt*:

$[| \ x \in HFinite; \ 0 \leq x \ |] ==> st((*f* \ sqrt) \ x) = (*f* \ sqrt)(st \ x)$
⟨proof⟩

lemma *hypreal-sqrt-sum-squares-ge1* [*simp*]: $!!x \ y. \ x \leq (*f* \ sqrt)(x \ \hat{} \ 2 + y \ \hat{} \ 2)$
⟨proof⟩

lemma *HFinite-hypreal-sqrt*:
$[| \ 0 \leq x; \ x \in HFinite \ |] ==> (*f* \ sqrt) \ x \in HFinite$
⟨proof⟩

lemma *HFinite-hypreal-sqrt-imp-HFinite*:
$[| \ 0 \leq x; \ (*f* \ sqrt) \ x \in HFinite \ |] ==> x \in HFinite$
⟨proof⟩

lemma *HFinite-hypreal-sqrt-iff* [*simp*]:
$0 \leq x ==> ((*f* \ sqrt) \ x \in HFinite) = (x \in HFinite)$
⟨proof⟩

lemma *HFinite-sqrt-sum-squares* [*simp*]:
$((*f* \ sqrt)(x*x + y*y) \in HFinite) = (x*x + y*y \in HFinite)$
⟨proof⟩

lemma *Infinitesimal-hypreal-sqrt*:
$[| \ 0 \leq x; \ x \in Infinitesimal \ |] ==> (*f* \ sqrt) \ x \in Infinitesimal$
⟨proof⟩

lemma *Infinitesimal-hypreal-sqrt-imp-Infinitesimal*:
$[| \ 0 \leq x; \ (*f* \ sqrt) \ x \in Infinitesimal \ |] ==> x \in Infinitesimal$
⟨proof⟩

lemma *Infinitesimal-hypreal-sqrt-iff* [*simp*]:
$0 \leq x ==> ((*f* \ sqrt) \ x \in Infinitesimal) = (x \in Infinitesimal)$
⟨proof⟩

lemma *Infinitesimal-sqrt-sum-squares* [*simp*]:
$((*f* \ sqrt)(x*x + y*y) \in Infinitesimal) = (x*x + y*y \in Infinitesimal)$
⟨proof⟩

lemma *HInfinite-hypreal-sqrt*:
$[| \ 0 \leq x; \ x \in HInfinite \ |] ==> (*f* \ sqrt) \ x \in HInfinite$
⟨proof⟩

lemma *HInfinite-hypreal-sqrt-imp-HInfinite*:
$[| \ 0 \leq x; \ (*f* \ sqrt) \ x \in HInfinite \ |] ==> x \in HInfinite$
⟨proof⟩

lemma *HInfinite-hypreal-sqrt-iff* [*simp*]:
$0 \leq x ==> ((*f* \ sqrt) \ x \in HInfinite) = (x \in HInfinite)$
⟨proof⟩

lemma *HInfinite-sqrt-sum-squares* [*simp*]:
$((*f* \ sqrt)(x*x + y*y) \in HInfinite) = (x*x + y*y \in HInfinite)$
⟨*proof*⟩

lemma *HFinite-exp* [*simp*]:
$sumhr\ (0,\ whn,\ \%n.\ inverse\ (real\ (fact\ n)) * x\ \hat{}\ n) \in HFinite$
⟨*proof*⟩

lemma *exphr-zero* [*simp*]: $exphr\ 0 = 1$
⟨*proof*⟩

lemma *coshr-zero* [*simp*]: $coshr\ 0 = 1$
⟨*proof*⟩

lemma *STAR-exp-zero-approx-one* [*simp*]: $(*f* \ exp)\ 0\ @= 1$
⟨*proof*⟩

lemma *STAR-exp-Infinitesimal*: $x \in Infinitesimal ==> (*f* \ exp)\ x\ @= 1$
⟨*proof*⟩

lemma *STAR-exp-epsilon* [*simp*]: $(*f* \ exp)\ epsilon\ @= 1$
⟨*proof*⟩

lemma *STAR-exp-add*: $!!x\ y.\ (*f* \ exp)(x + y) = (*f* \ exp)\ x * (*f* \ exp)\ y$
⟨*proof*⟩

lemma *exphr-hypreal-of-real-exp-eq*: $exphr\ x = hypreal\text{-}of\text{-}real\ (exp\ x)$
⟨*proof*⟩

lemma *starfun-exp-ge-add-one-self* [*simp*]: $!!x.\ 0 \le x ==> (1 + x) \le (*f* \ exp)$
x
⟨*proof*⟩

lemma *starfun-exp-HInfinite*:
$[|\ x \in HInfinite;\ 0 \le x\ |] ==> (*f* \ exp)\ x \in HInfinite$
⟨*proof*⟩

lemma *starfun-exp-minus*: $!!x.\ (*f* \ exp)\ (-x) = inverse((*f* \ exp)\ x)$
⟨*proof*⟩

lemma *starfun-exp-Infinitesimal*:
$[|\ x \in HInfinite;\ x \le 0\ |] ==> (*f* \ exp)\ x \in Infinitesimal$
⟨*proof*⟩

lemma *starfun-exp-gt-one* [*simp*]: $!!x.\ 0 < x ==> 1 < (*f* \ exp)\ x$
⟨*proof*⟩

lemma *starfun-ln-exp* [*simp*]: !!x. ($*f*$ *ln*) (($*f*$ *exp*) x) = x
⟨*proof*⟩

lemma *starfun-exp-ln-iff* [*simp*]: !!x. (($*f*$ *exp*)(($*f*$ *ln*) x) = x) = ($0 < x$)
⟨*proof*⟩

lemma *starfun-exp-ln-eq*: ($*f*$ *exp*) $u = x$ ==> ($*f*$ *ln*) $x = u$
⟨*proof*⟩

lemma *starfun-ln-less-self* [*simp*]: !!x. $0 < x$ ==> ($*f*$ *ln*) $x < x$
⟨*proof*⟩

lemma *starfun-ln-ge-zero* [*simp*]: !!x. $1 \leq x$ ==> $0 \leq$ ($*f*$ *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*; $1 \leq x$ |] ==> ($*f*$ *ln*) $x \in$ *HFinite*
⟨*proof*⟩

lemma *starfun-ln-inverse*: !!x. $0 < x$ ==> ($*f*$ *ln*) (*inverse* x) = $-$($*f*$ *ln*) x
⟨*proof*⟩

lemma *starfun-exp-HFinite*: $x \in$ *HFinite* ==> ($*f*$ *exp*) $x \in$ *HFinite*
⟨*proof*⟩

lemma *starfun-exp-add-HFinite-Infinitesimal-approx*:
[|$x \in$ *Infinitesimal*; $z \in$ *HFinite* |] ==> ($*f*$ *exp*) ($z + x$) @= ($*f*$ *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*) $x \in$ *Infinitesimal*
⟨*proof*⟩

lemma *starfun-ln-HFinite-not-Infinitesimal*:

$[| \ x \in HFinite - Infinitesimal; \ 0 < x \ |] ==> (*f* \ ln) \ x \in HFinite$
⟨proof⟩

lemma *starfun-ln-Infinitesimal-HInfinite*:
$[| \ x \in Infinitesimal; \ 0 < x \ |] ==> (*f* \ ln) \ x \in HInfinite$
⟨proof⟩

lemma *starfun-ln-less-zero*: $!!x. \ [| \ 0 < x; \ x < 1 \ |] ==> (*f* \ ln) \ x < 0$
⟨proof⟩

lemma *starfun-ln-Infinitesimal-less-zero*:
$[| \ x \in Infinitesimal; \ 0 < x \ |] ==> (*f* \ ln) \ x < 0$
⟨proof⟩

lemma *starfun-ln-HInfinite-gt-zero*:
$[| \ x \in HInfinite; \ 0 < x \ |] ==> 0 < (*f* \ ln) \ x$
⟨proof⟩

lemma *HFinite-sin* [*simp*]:
$sumhr \ (0, \ whn, \ \%n. \ (if \ even(n) \ then \ 0 \ else$
$((- \ 1) \ \hat{} \ ((n - 1) \ div \ 2))/(real \ (fact \ n))) * x \ \hat{} \ n)$
$\in HFinite$
⟨proof⟩

lemma *STAR-sin-zero* [*simp*]: $(*f* \ sin) \ 0 = 0$
⟨proof⟩

lemma *STAR-sin-Infinitesimal* [*simp*]: $x \in Infinitesimal ==> (*f* \ sin) \ x \ @= x$
⟨proof⟩

lemma *HFinite-cos* [*simp*]:
$sumhr \ (0, \ whn, \ \%n. \ (if \ even(n) \ then$
$((- \ 1) \ \hat{} \ (n \ div \ 2))/(real \ (fact \ n)) \ else$
$0) * x \ \hat{} \ n) \in HFinite$
⟨proof⟩

lemma *STAR-cos-zero* [*simp*]: $(*f* \ cos) \ 0 = 1$
⟨proof⟩

lemma *STAR-cos-Infinitesimal* [*simp*]: $x \in Infinitesimal ==> (*f* \ cos) \ x \ @= 1$
⟨proof⟩

lemma *STAR-tan-zero* [*simp*]: $(*f* \ tan) \ 0 = 0$
⟨proof⟩

lemma *STAR-tan-Infinitesimal*: $x \in$ *Infinitesimal* ==> (*f* *tan*) x @= x
⟨*proof*⟩

lemma *STAR-sin-cos-Infinitesimal-mult*:
$x \in$ *Infinitesimal* ==> (*f* *sin*) x * (*f* *cos*) x @= x
⟨*proof*⟩

lemma *HFinite-pi*: *hypreal-of-real pi* $\in$ *HFinite*
⟨*proof*⟩

lemma *lemma-split-hypreal-of-real*:
$N \in$ *HNatInfinite*
==> *hypreal-of-real a* =
hypreal-of-hypnat N * (*inverse*(*hypreal-of-hypnat N*) * *hypreal-of-real a*)
⟨*proof*⟩

lemma *STAR-sin-Infinitesimal-divide*:
[|$x \in$ *Infinitesimal*; $x \neq 0$ |] ==> (*f* *sin*) x/x @= *1*
⟨*proof*⟩

lemma *lemma-sin-pi*:
$n \in$ *HNatInfinite*
==> (*f* *sin*) (*inverse* (*hypreal-of-hypnat n*))/(*inverse* (*hypreal-of-hypnat n*)) @= *1*
⟨*proof*⟩

lemma *STAR-sin-inverse-HNatInfinite*:
$n \in$ *HNatInfinite*
==> (*f* *sin*) (*inverse* (*hypreal-of-hypnat n*)) * *hypreal-of-hypnat n* @= *1*
⟨*proof*⟩

lemma *Infinitesimal-pi-divide-HNatInfinite*:
$N \in$ *HNatInfinite*
==> *hypreal-of-real pi*/(*hypreal-of-hypnat N*) $\in$ *Infinitesimal*
⟨*proof*⟩

lemma *pi-divide-HNatInfinite-not-zero* [*simp*]:
$N \in$ *HNatInfinite* ==> *hypreal-of-real pi*/(*hypreal-of-hypnat N*) $\neq$ *0*
⟨*proof*⟩

lemma *STAR-sin-pi-divide-HNatInfinite-approx-pi*:
$n \in$ *HNatInfinite*
==> (*f* *sin*) (*hypreal-of-real pi*/(*hypreal-of-hypnat n*)) * *hypreal-of-hypnat*

n
    @= *hypreal-of-real pi*
⟨*proof*⟩

lemma *STAR-sin-pi-divide-HNatInfinite-approx-pi2*:
  n ∈ *HNatInfinite*
  ==> *hypreal-of-hypnat n* ∗
    (∗*f*∗ *sin*) (*hypreal-of-real pi*/(*hypreal-of-hypnat n*))
    @= *hypreal-of-real pi*
⟨*proof*⟩

lemma *starfunNat-pi-divide-n-Infinitesimal*:
  N ∈ *HNatInfinite* ==> (∗*f*∗ (%*x*. *pi* / *real x*)) *N* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *STAR-sin-pi-divide-n-approx*:
  N ∈ *HNatInfinite* ==>
  (∗*f*∗ *sin*) ((∗*f*∗ (%*x*. *pi* / *real x*)) *N*) @=
  hypreal-of-real pi/(*hypreal-of-hypnat N*)
⟨*proof*⟩

lemma *NSLIMSEQ-sin-pi*: (%*n*. *real n* ∗ *sin* (*pi* / *real n*)) −−−−*NS*> *pi*
⟨*proof*⟩

lemma *NSLIMSEQ-cos-one*: (%*n*. *cos* (*pi* / *real n*))−−−−*NS*> *1*
⟨*proof*⟩

lemma *NSLIMSEQ-sin-cos-pi*:
  (%*n*. *real n* ∗ *sin* (*pi* / *real n*) ∗ *cos* (*pi* / *real n*)) −−−−*NS*> *pi*
⟨*proof*⟩

A familiar approximation to *cos x* when *x* is small

lemma *STAR-cos-Infinitesimal-approx*:
  x ∈ *Infinitesimal* ==> (∗*f*∗ *cos*) *x* @= *1* − *x* ˆ *2*
⟨*proof*⟩

lemma *STAR-cos-Infinitesimal-approx2*:
  x ∈ *Infinitesimal* ==> (∗*f*∗ *cos*) *x* @= *1* − (*x* ˆ *2*)/*2*
⟨*proof*⟩

⟨*ML*⟩

end

34 HLog: Logarithms: Non-Standard Version

theory *HLog*

imports *Log HTranscendental*
begin

lemma *epsilon-ge-zero* [*simp*]: *0* $\leq$ *epsilon*
⟨*proof*⟩

lemma *hpfinite-witness*: *epsilon* : {*x*. *0* $\leq$ *x* & *x* : *HFinite*}
⟨*proof*⟩

constdefs

powhr :: [*hypreal*,*hypreal*] => *hypreal* (**infixr** *powhr 80*)
x powhr a == *starfun2* (*op powr*) *x a*

hlog :: [*hypreal*,*hypreal*] => *hypreal*
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* $\neq$ *0*
⟨*proof*⟩

lemma *powhr-divide*:
!!*a x y*. [| *0* < *x*; *0* < *y* |] ==> (*x* / *y*) *powhr a* = (*x powhr a*)/(*y powhr a*)
⟨*proof*⟩

lemma *powhr-add*: !!*a b x*. *x powhr* (*a* + *b*) = (*x powhr a*) ∗ (*x powhr b*)
⟨*proof*⟩

lemma *powhr-powhr*: !!*a b x*. (*x powhr a*) *powhr b* = *x powhr* (*a* ∗ *b*)
⟨*proof*⟩

lemma *powhr-powhr-swap*: $!!a\ b\ x.\ (x\ powhr\ a)\ powhr\ b = (x\ powhr\ b)\ powhr\ a$
⟨*proof*⟩

lemma *powhr-minus*: $!!a\ x.\ x\ powhr\ (-a) = inverse\ (x\ powhr\ a)$
⟨*proof*⟩

lemma *powhr-minus-divide*: $x\ powhr\ (-a) = 1/(x\ powhr\ a)$
⟨*proof*⟩

lemma *powhr-less-mono*: $!!a\ b\ x.\ [|\ a < b;\ 1 < x\ |] ==> x\ powhr\ a < x\ powhr\ b$
⟨*proof*⟩

lemma *powhr-less-cancel*: $!!a\ b\ x.\ [|\ x\ powhr\ a < x\ powhr\ b;\ 1 < x\ |] ==> a < b$
⟨*proof*⟩

lemma *powhr-less-cancel-iff* [*simp*]:
$1 < x ==> (x\ powhr\ a < x\ powhr\ b) = (a < b)$
⟨*proof*⟩

lemma *powhr-le-cancel-iff* [*simp*]:
$1 < x ==> (x\ powhr\ a \le x\ powhr\ b) = (a \le b)$
⟨*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*]:
$!!a\ x.\ [|\ 0 < a;\ a \neq 1;\ 0 < x\ |] ==> a\ powhr\ (hlog\ a\ x) = x$
⟨*proof*⟩

lemma *hlog-powhr-cancel* [*simp*]:
$!!a\ y.\ [|\ 0 < a;\ a \neq 1\ |] ==> hlog\ a\ (a\ powhr\ y) = y$
⟨*proof*⟩

lemma *hlog-mult*:
$!!a\ x\ y.\ [|\ 0 < a;\ a \neq 1;\ 0 < x;\ 0 < y\ \ |]$
$==> hlog\ a\ (x * y) = hlog\ a\ x + hlog\ a\ y$
⟨*proof*⟩

lemma *hlog-as-starfun*:
$!!a\ x.\ [|\ 0 < a;\ a \neq 1\ |] ==> hlog\ a\ x = (\ {*f*}\ ln)\ x\ /\ (\ {*f*}\ ln)\ a$
⟨*proof*⟩

lemma *hlog-eq-div-starfun-ln-mult-hlog*:

$!!a\ b\ x.\ [|\ 0 < a;\ a \neq 1;\ 0 < b;\ b \neq 1;\ 0 < x\ |]$
$==> hlog\ a\ x = ((\ *f*\ ln)\ b/(\ *f*ln)\ a) * hlog\ b\ x$
⟨*proof*⟩

lemma *powhr-as-starfun*: $!!a\ x.\ x\ powhr\ a = (\ *f*\ exp)\ (a * (\ *f*\ ln)\ x)$
⟨*proof*⟩

lemma *HInfinite-powhr*:
$[|\ x : HInfinite;\ 0 < x;\ a : HFinite - Infinitesimal;$
$0 < a\ |] ==> x\ powhr\ a : HInfinite$
⟨*proof*⟩

lemma *hlog-hrabs-HInfinite-Infinitesimal*:
$[|\ x : HFinite - Infinitesimal;\ a : HInfinite;\ 0 < a\ |]$
$==> hlog\ a\ (abs\ x) : Infinitesimal$
⟨*proof*⟩

lemma *hlog-HInfinite-as-starfun*:
$[|\ a : HInfinite;\ 0 < a\ |] ==> hlog\ a\ x = (\ *f*\ ln)\ x\ /\ (\ *f*\ ln)\ a$
⟨*proof*⟩

lemma *hlog-one* [*simp*]: $!!a.\ hlog\ a\ 1 = 0$
⟨*proof*⟩

lemma *hlog-eq-one* [*simp*]: $!!a.\ [|\ 0 < a;\ a \neq 1\ |] ==> hlog\ a\ a = 1$
⟨*proof*⟩

lemma *hlog-inverse*:
$[|\ 0 < a;\ a \neq 1;\ 0 < x\ |] ==> hlog\ a\ (inverse\ x) = -\ hlog\ a\ x$
⟨*proof*⟩

lemma *hlog-divide*:
$[|\ 0 < a;\ a \neq 1;\ 0 < x;\ 0 < y|] ==> hlog\ a\ (x/y) = hlog\ a\ x - hlog\ a\ y$
⟨*proof*⟩

lemma *hlog-less-cancel-iff* [*simp*]:
$!!a\ x\ y.\ [|\ 1 < a;\ 0 < x;\ 0 < y\ |] ==> (hlog\ a\ x < hlog\ a\ y) = (x < y)$
⟨*proof*⟩

lemma *hlog-le-cancel-iff* [*simp*]:
$[|\ 1 < a;\ 0 < x;\ 0 < y\ |] ==> (hlog\ a\ x \leq hlog\ a\ y) = (x \leq y)$
⟨*proof*⟩

⟨*ML*⟩

end

theory *Hyperreal*
imports *Poly Taylor HLog*
begin

end

35 Complex: Complex Numbers: Rectangular and Polar Representations

theory *Complex*
imports *../Hyperreal/HLog*
begin

datatype *complex* = *Complex real real*

instance *complex* :: {*zero*, *one*, *plus*, *times*, *minus*, *inverse*, *power*} ⟨*proof*⟩

consts
  ii :: *complex* (i)

consts *Re* :: *complex* => *real*
primrec *Re* (*Complex x y*) = *x*

consts *Im* :: *complex* => *real*
primrec *Im* (*Complex x y*) = *y*

lemma *complex-surj* [*simp*]: *Complex* (*Re z*) (*Im z*) = *z*
  ⟨*proof*⟩

constdefs

  cmod :: *complex* => *real*
  cmod z == *sqrt*(*Re*(*z*) ^ *2* + *Im*(*z*) ^ *2*)

  complex-of-real :: *real* => *complex*
  complex-of-real r == *Complex r 0*

  cnj :: *complex* => *complex*
  cnj z == *Complex* (*Re z*) (−*Im z*)

sgn :: *complex* => *complex*
sgn z == *z* / *complex-of-real*(*cmod z*)

arg :: *complex* => *real*
arg z == @*a*. *Re*(*sgn z*) = *cos a* & *Im*(*sgn z*) = *sin a* & $-$*pi* < *a* & *a* $\leq$ *pi*

defs (**overloaded**)

complex-zero-def:
0 == *Complex 0 0*

complex-one-def:
1 == *Complex 1 0*

i-def: *ii* == *Complex 0 1*

complex-minus-def: $-$ *z* == *Complex* ($-$ *Re z*) ($-$ *Im z*)

complex-inverse-def:
inverse z ==
Complex (*Re z* / ((*Re z*)^2 + (*Im z*)^2)) ($-$ *Im z* / ((*Re z*)^2 + (*Im z*)^2))

complex-add-def:
z + *w* == *Complex* (*Re z* + *Re w*) (*Im z* + *Im w*)

complex-diff-def:
z $-$ *w* == *z* + $-$ (*w*::*complex*)

complex-mult-def:
z * *w* == *Complex* (*Re z* * *Re w* $-$ *Im z* * *Im w*) (*Re z* * *Im w* + *Im z* * *Re w*)

complex-divide-def: *w* / (*z*::*complex*) == *w* * *inverse z*

constdefs

cis :: *real* => *complex*
cis a == *Complex* (*cos a*) (*sin a*)

rcis :: [*real*, *real*] => *complex*
rcis r a == *complex-of-real r* * *cis a*

expi :: *complex* => *complex*
expi z == *complex-of-real*(*exp* (*Re z*)) $*$ *cis* (*Im z*)

lemma *complex-equality* [*intro?*]: *Re z* = *Re w* ==> *Im z* = *Im w* ==> *z* = *w*
⟨*proof*⟩

lemma *Re* [*simp*]: *Re*(*Complex x y*) = *x*
⟨*proof*⟩

lemma *Im* [*simp*]: *Im*(*Complex x y*) = *y*
⟨*proof*⟩

lemma *complex-Re-Im-cancel-iff*: (*w*=*z*) = (*Re*(*w*) = *Re*(*z*) & *Im*(*w*) = *Im*(*z*))
⟨*proof*⟩

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

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

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

lemma *complex-Im-one* [*simp*]: *Im 1* = *0*
⟨*proof*⟩

lemma *complex-Re-i* [*simp*]: *Re*(*ii*) = *0*
⟨*proof*⟩

lemma *complex-Im-i* [*simp*]: *Im*(*ii*) = *1*
⟨*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*⟩

35.1 Unary Minus

lemma *complex-minus* [*simp*]: − (*Complex x y*) = *Complex* (−*x*) (−*y*)
⟨*proof*⟩

lemma *complex-Re-minus* [*simp*]: *Re* (−*z*) = − *Re z*
⟨*proof*⟩

lemma *complex-Im-minus* [*simp*]: *Im* (−*z*) = − *Im z*

⟨*proof*⟩

35.2 Addition

lemma *complex-add* [*simp*]:
$Complex\ x1\ y1 + Complex\ x2\ y2 = Complex\ (x1+x2)\ (y1+y2)$
⟨*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-add-commute*: $(u{::}complex) + v = v + u$
⟨*proof*⟩

lemma *complex-add-assoc*: $((u{::}complex) + v) + w = u + (v + w)$
⟨*proof*⟩

lemma *complex-add-zero-left*: $(0{::}complex) + z = z$
⟨*proof*⟩

lemma *complex-add-zero-right*: $z + (0{::}complex) = z$
⟨*proof*⟩

lemma *complex-add-minus-left*: $-z + z = (0{::}complex)$
⟨*proof*⟩

lemma *complex-diff*:
$Complex\ x1\ y1 - Complex\ x2\ y2 = Complex\ (x1-x2)\ (y1-y2)$
⟨*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*⟩

35.3 Multiplication

lemma *complex-mult* [*simp*]:
$Complex\ x1\ y1 * Complex\ x2\ y2 = Complex\ (x1*x2 - y1*y2)\ (x1*y2 + y1*x2)$
⟨*proof*⟩

lemma *complex-mult-commute*: $(w{::}complex) * z = z * w$
⟨*proof*⟩

lemma *complex-mult-assoc*: $((u{::}complex) * v) * w = u * (v * w)$

⟨*proof*⟩

lemma *complex-mult-one-left*: $(1::complex) * z = z$
⟨*proof*⟩

lemma *complex-mult-one-right*: $z * (1::complex) = z$
⟨*proof*⟩

35.4 Inverse

lemma *complex-inverse* [*simp*]:
$inverse\ (Complex\ x\ y) = Complex\ (x/(x\ \hat{}\ 2 + y\ \hat{}\ 2))\ (-y/(x\ \hat{}\ 2 + y\ \hat{}\ 2))$
⟨*proof*⟩

lemma *complex-mult-inv-left*: $z \neq (0::complex) ==> inverse(z) * z = 1$
⟨*proof*⟩

35.5 The field of complex numbers

instance *complex* :: *field*
⟨*proof*⟩

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

35.6 Embedding Properties for *complex-of-real* Map

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*⟩

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 *complex-of-real-one* [*simp*]: *complex-of-real 1* = *1*

⟨*proof*⟩

lemma *complex-of-real-zero* [*simp*]: *complex-of-real 0 = 0*
⟨*proof*⟩

lemma *complex-of-real-eq-iff* [*iff*]:
(*complex-of-real x = complex-of-real y*) = (*x = y*)
⟨*proof*⟩

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

lemma *complex-of-real-inverse* [*simp*]:
complex-of-real(*inverse x*) = *inverse*(*complex-of-real x*)
⟨*proof*⟩

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

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

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

lemma *complex-of-real-divide* [*simp*]:
complex-of-real(*x*/*y*) = *complex-of-real x* / *complex-of-real y*
⟨*proof*⟩

lemma *complex-mod* [*simp*]: *cmod* (*Complex x y*) = *sqrt*(*x* ˆ *2* + *y* ˆ *2*)
⟨*proof*⟩

lemma *complex-mod-zero* [*simp*]: *cmod*(*0*) = *0*
⟨*proof*⟩

lemma *complex-mod-one* [*simp*]: *cmod*(*1*) = *1*
⟨*proof*⟩

lemma *complex-mod-complex-of-real* [*simp*]: *cmod*(*complex-of-real x*) = *abs x*
⟨*proof*⟩

lemma *complex-of-real-abs*:
complex-of-real (*abs x*) = *complex-of-real*(*cmod*(*complex-of-real x*))
⟨*proof*⟩

35.7 The Functions *Re* and *Im*

lemma *complex-Re-mult-eq*: $Re\ (w * z) = Re\ w * Re\ z - Im\ w * Im\ z$
⟨*proof*⟩

lemma *complex-Im-mult-eq*: $Im\ (w * z) = Re\ w * Im\ z + Im\ w * Re\ z$
⟨*proof*⟩

lemma *Re-i-times* [*simp*]: $Re(ii * z) = -\ Im\ z$
⟨*proof*⟩

lemma *Re-times-i* [*simp*]: $Re(z * ii) = -\ Im\ z$
⟨*proof*⟩

lemma *Im-i-times* [*simp*]: $Im(ii * z) = Re\ z$
⟨*proof*⟩

lemma *Im-times-i* [*simp*]: $Im(z * ii) = Re\ z$
⟨*proof*⟩

lemma *complex-Re-mult*: [| $Im\ w = 0$; $Im\ z = 0$ |] ==> $Re(w * z) = Re(w) * Re(z)$
⟨*proof*⟩

lemma *complex-Re-mult-complex-of-real* [*simp*]:
$Re\ (z * complex\text{-}of\text{-}real\ c) = Re(z) * c$
⟨*proof*⟩

lemma *complex-Im-mult-complex-of-real* [*simp*]:
$Im\ (z * complex\text{-}of\text{-}real\ c) = Im(z) * c$
⟨*proof*⟩

lemma *complex-Re-mult-complex-of-real2* [*simp*]:
$Re\ (complex\text{-}of\text{-}real\ c * z) = c * Re(z)$
⟨*proof*⟩

lemma *complex-Im-mult-complex-of-real2* [*simp*]:
$Im\ (complex\text{-}of\text{-}real\ c * z) = c * Im(z)$
⟨*proof*⟩

35.8 Conjugation is an Automorphism

lemma *complex-cnj*: $cnj\ (Complex\ x\ y) = Complex\ x\ (-y)$
⟨*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-complex-of-real* [*simp*]:
  cnj (*complex-of-real* *x*) = *complex-of-real* *x*
⟨*proof*⟩

lemma *complex-mod-cnj* [*simp*]: *cmod* (*cnj* *z*) = *cmod* *z*
⟨*proof*⟩

lemma *complex-cnj-minus*: *cnj* (−*z*) = − *cnj* *z*
⟨*proof*⟩

lemma *complex-cnj-inverse*: *cnj*(*inverse* *z*) = *inverse*(*cnj* *z*)
⟨*proof*⟩

lemma *complex-cnj-add*: *cnj*(*w* + *z*) = *cnj*(*w*) + *cnj*(*z*)
⟨*proof*⟩

lemma *complex-cnj-diff*: *cnj*(*w* − *z*) = *cnj*(*w*) − *cnj*(*z*)
⟨*proof*⟩

lemma *complex-cnj-mult*: *cnj*(*w* * *z*) = *cnj*(*w*) * *cnj*(*z*)
⟨*proof*⟩

lemma *complex-cnj-divide*: *cnj*(*w* / *z*) = (*cnj* *w*)/(*cnj* *z*)
⟨*proof*⟩

lemma *complex-cnj-one* [*simp*]: *cnj* *1* = *1*
⟨*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-cnj-zero* [*simp*]: *cnj* *0* = *0*
⟨*proof*⟩

lemma *complex-cnj-zero-iff* [*iff*]: (*cnj* *z* = *0*) = (*z* = *0*)
⟨*proof*⟩

lemma *complex-mult-cnj*: *z* * *cnj* *z* = *complex-of-real* (*Re*(*z*) ˆ *2* + *Im*(*z*) ˆ *2*)
⟨*proof*⟩

35.9 Modulus

lemma *complex-mod-eq-zero-cancel* [*simp*]: (*cmod* *x* = *0*) = (*x* = *0*)
⟨*proof*⟩

lemma *complex-mod-complex-of-real-of-nat* [*simp*]:
$cmod\ (complex\text{-}of\text{-}real(real\ (n::nat))) = real\ n$
⟨*proof*⟩

lemma *complex-mod-minus* [*simp*]: $cmod\ (-x) = cmod(x)$
⟨*proof*⟩

lemma *complex-mod-mult-cnj*: $cmod(z * cnj(z)) = cmod(z)\ \hat{}\ 2$
⟨*proof*⟩

lemma *complex-mod-squared*: $cmod(Complex\ x\ y)\ \hat{}\ 2 = x\ \hat{}\ 2 + y\ \hat{}\ 2$
⟨*proof*⟩

lemma *complex-mod-ge-zero* [*simp*]: $0 \leq cmod\ x$
⟨*proof*⟩

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

lemma *complex-mod-mult*: $cmod(x*y) = cmod(x) * cmod(y)$
⟨*proof*⟩

lemma *cmod-unit-one* [*simp*]: $cmod\ (Complex\ (cos\ a)\ (sin\ a)) = 1$
⟨*proof*⟩

lemma *cmod-complex-polar* [*simp*]:
$cmod\ (complex\text{-}of\text{-}real\ r * Complex\ (cos\ a)\ (sin\ a)) = abs\ r$
⟨*proof*⟩

lemma *complex-mod-add-squared-eq*:
$cmod(x + y)\ \hat{}\ 2 = cmod(x)\ \hat{}\ 2 + cmod(y)\ \hat{}\ 2 + 2 * Re(x * cnj\ y)$
⟨*proof*⟩

lemma *complex-Re-mult-cnj-le-cmod* [*simp*]: $Re(x * cnj\ y) \leq cmod(x * cnj\ y)$
⟨*proof*⟩

lemma *complex-Re-mult-cnj-le-cmod2* [*simp*]: $Re(x * cnj\ y) \leq cmod(x * y)$
⟨*proof*⟩

lemma *real-sum-squared-expand*:
$((x::real) + y)\ \hat{}\ 2 = x\ \hat{}\ 2 + y\ \hat{}\ 2 + 2 * x * y$
⟨*proof*⟩

lemma *complex-mod-triangle-squared* [*simp*]:
$cmod\ (x + y)\ \hat{}\ 2 \leq (cmod(x) + cmod(y))\ \hat{}\ 2$
⟨*proof*⟩

lemma *complex-mod-minus-le-complex-mod* [*simp*]: $-\ cmod\ x \leq cmod\ x$
⟨*proof*⟩

lemma *complex-mod-triangle-ineq* [*simp*]: $cmod\ (x + y) \le cmod(x) + cmod(y)$
⟨*proof*⟩

lemma *complex-mod-triangle-ineq2* [*simp*]: $cmod(b + a) - cmod\ b \le cmod\ a$
⟨*proof*⟩

lemma *complex-mod-diff-commute*: $cmod\ (x - y) = cmod\ (y - x)$
⟨*proof*⟩

lemma *complex-mod-add-less*:
$[|\ cmod\ x < r;\ cmod\ y < s\ |] ==> cmod\ (x + y) < r + s$
⟨*proof*⟩

lemma *complex-mod-mult-less*:
$[|\ cmod\ x < r;\ cmod\ y < s\ |] ==> cmod\ (x * y) < r * s$
⟨*proof*⟩

lemma *complex-mod-diff-ineq* [*simp*]: $cmod(a) - cmod(b) \le cmod(a + b)$
⟨*proof*⟩

lemma *complex-Re-le-cmod* [*simp*]: $Re\ z \le cmod\ z$
⟨*proof*⟩

lemma *complex-mod-gt-zero*: $z \neq 0 ==> 0 < cmod\ z$
⟨*proof*⟩

35.10 A Few More Theorems

lemma *complex-mod-inverse*: $cmod(inverse\ x) = inverse(cmod\ x)$
⟨*proof*⟩

lemma *complex-mod-divide*: $cmod(x/y) = cmod(x)/(cmod\ y)$
⟨*proof*⟩

35.11 Exponentiation

primrec
complexpow-0: $z\ \hat{}\ 0 = 1$
complexpow-Suc: $z\ \hat{}\ (Suc\ n) = (z{::}complex) * (z\ \hat{}\ n)$

instance *complex* :: *recpower*
⟨*proof*⟩

lemma *complex-of-real-pow*: $complex\text{-}of\text{-}real\ (x\ \hat{}\ n) = (complex\text{-}of\text{-}real\ x)\ \hat{}\ n$
⟨*proof*⟩

lemma *complex-cnj-pow*: $cnj(z\ \hat{}\ n) = cnj(z)\ \hat{}\ n$

⟨proof⟩

lemma *complex-mod-complexpow*: $cmod(x \hat{}\ n) = cmod(x)\ \hat{}\ n$
⟨proof⟩

lemma *complexpow-i-squared* [*simp*]: $ii\ \hat{}\ 2 = -(1::complex)$
⟨proof⟩

lemma *complex-i-not-zero* [*simp*]: $ii \neq 0$
⟨proof⟩

35.12 The Function *sgn*

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

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

lemma *sgn-minus*: $sgn\ (-z) = -\ sgn(z)$
⟨proof⟩

lemma *sgn-eq*: $sgn\ z = z\ /\ complex\text{-}of\text{-}real\ (cmod\ z)$
⟨proof⟩

lemma *i-mult-eq*: $ii * ii = complex\text{-}of\text{-}real\ (-1)$
⟨proof⟩

lemma *i-mult-eq2* [*simp*]: $ii * ii = -(1::complex)$
⟨proof⟩

lemma *complex-eq-cancel-iff2* [*simp*]:
$(Complex\ x\ y = complex\text{-}of\text{-}real\ xa) = (x = xa\ \&\ y = 0)$
⟨proof⟩

lemma *complex-eq-cancel-iff2a* [*simp*]:
$(Complex\ x\ y = complex\text{-}of\text{-}real\ xa) = (x = xa\ \&\ y = 0)$
⟨proof⟩

lemma *Complex-eq-0* [*simp*]: $(Complex\ x\ y = 0) = (x = 0\ \&\ y = 0)$
⟨proof⟩

lemma *Complex-eq-1* [*simp*]: $(Complex\ x\ y = 1) = (x = 1\ \&\ y = 0)$
⟨proof⟩

lemma *Complex-eq-i* [*simp*]: $(Complex\ x\ y = ii) = (x = 0\ \&\ y = 1)$
⟨proof⟩

lemma *Re-sgn* [*simp*]: $Re(sgn\ z) = Re(z)/cmod\ z$
⟨*proof*⟩

lemma *Im-sgn* [*simp*]: $Im(sgn\ z) = Im(z)/cmod\ z$
⟨*proof*⟩

lemma *complex-inverse-complex-split*:
$inverse(complex\text{-}of\text{-}real\ x + ii * complex\text{-}of\text{-}real\ y) =$
$complex\text{-}of\text{-}real(x/(x\ \hat{}\ 2 + y\ \hat{}\ 2)) -$
$ii * complex\text{-}of\text{-}real(y/(x\ \hat{}\ 2 + y\ \hat{}\ 2))$
⟨*proof*⟩

lemma *complex-of-real-zero-iff* [*simp*]: $(complex\text{-}of\text{-}real\ y = 0) = (y = 0)$
⟨*proof*⟩

lemma *cos-arg-i-mult-zero-pos*:
$0 < y ==> cos\ (arg(Complex\ 0\ y)) = 0$
⟨*proof*⟩

lemma *cos-arg-i-mult-zero-neg*:
$y < 0 ==> cos\ (arg(Complex\ 0\ y)) = 0$
⟨*proof*⟩

lemma *cos-arg-i-mult-zero* [*simp*]:
$y \neq 0 ==> cos\ (arg(Complex\ 0\ y)) = 0$
⟨*proof*⟩

35.13 Finally! Polar Form for Complex Numbers

lemma *complex-split-polar*:
$\exists\, r\ a.\ z = complex\text{-}of\text{-}real\ r * (Complex\ (cos\ a)\ (sin\ a))$
⟨*proof*⟩

lemma *rcis-Ex*: $\exists\, r\ a.\ z = rcis\ r\ a$
⟨*proof*⟩

lemma *Re-rcis* [*simp*]: $Re(rcis\ r\ a) = r * cos\ a$
⟨*proof*⟩

lemma *Im-rcis* [*simp*]: $Im(rcis\ r\ a) = r * sin\ a$
⟨*proof*⟩

lemma *sin-cos-squared-add2-mult*: $(r * cos\ a)^2 + (r * sin\ a)^2 = r^2$
⟨*proof*⟩

lemma *complex-mod-rcis* [*simp*]: *cmod*(*rcis r a*) = *abs r*
⟨*proof*⟩

lemma *complex-mod-sqrt-Re-mult-cnj*: *cmod z* = *sqrt* (*Re* (*z* * *cnj z*))
⟨*proof*⟩

lemma *complex-Re-cnj* [*simp*]: *Re*(*cnj z*) = *Re z*
⟨*proof*⟩

lemma *complex-Im-cnj* [*simp*]: *Im*(*cnj z*) = − *Im z*
⟨*proof*⟩

lemma *complex-In-mult-cnj-zero* [*simp*]: *Im* (*z* * *cnj z*) = *0*
⟨*proof*⟩

lemma *cis-rcis-eq*: *cis a* = *rcis 1 a*
⟨*proof*⟩

lemma *rcis-mult*: *rcis r1 a* * *rcis r2 b* = *rcis* (*r1***r2*) (*a* + *b*)
⟨*proof*⟩

lemma *cis-mult*: *cis a* * *cis b* = *cis* (*a* + *b*)
⟨*proof*⟩

lemma *cis-zero* [*simp*]: *cis 0* = *1*
⟨*proof*⟩

lemma *rcis-zero-mod* [*simp*]: *rcis 0 a* = *0*
⟨*proof*⟩

lemma *rcis-zero-arg* [*simp*]: *rcis r 0* = *complex-of-real r*
⟨*proof*⟩

lemma *complex-of-real-minus-one*:
complex-of-real (−(*1*::*real*)) = −(*1*::*complex*)
⟨*proof*⟩

lemma *complex-i-mult-minus* [*simp*]: *ii* * (*ii* * *x*) = − *x*
⟨*proof*⟩

lemma *cis-real-of-nat-Suc-mult*:
cis (*real* (*Suc* *n*) ∗ *a*) = *cis* *a* ∗ *cis* (*real* *n* ∗ *a*)
⟨*proof*⟩

lemma *DeMoivre*: (*cis* *a*) ˆ *n* = *cis* (*real* *n* ∗ *a*)
⟨*proof*⟩

lemma *DeMoivre2*: (*rcis* *r* *a*) ˆ *n* = *rcis* (*r* ˆ *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* ˆ *n*)
⟨*proof*⟩

lemma *sin-n-Im-cis-pow-n*: *sin* (*real* *n* ∗ *a*) = *Im*(*cis* *a* ˆ *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*⟩

35.14 Numerals and Arithmetic

instance *complex* :: *number* ⟨*proof*⟩

defs (overloaded)
complex-number-of-def: (*number-of* *w* :: *complex*) == *of-int* (*Rep-Bin* *w*)

— the type constraint is essential!

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

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

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

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

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

This theorem is necessary because theorems such as *iszero-number-of-0* only hold for ordered rings. They cannot be generalized to fields in general because they fail for finite fields. They work for type complex because the reals can be embedded in them.

lemma *iszero-complex-number-of* [*simp*]:
iszero (*number-of w* :: *complex*) = *iszero* (*number-of w* :: *real*)
⟨*proof*⟩

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

lemma *complex-number-of-cmod*:
cmod(*number-of v* :: *complex*) = *abs* (*number-of v* :: *real*)
⟨*proof*⟩

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

lemma *complex-number-of-Im* [*simp*]: *Im*(*number-of v* :: *complex*) = *0*
⟨*proof*⟩

lemma *expi-two-pi-i* [*simp*]: *expi*((*2*::*complex*) ∗ *complex-of-real pi* ∗ *ii*) = *1*
⟨*proof*⟩

⟨*ML*⟩

end

36 NSComplex: Nonstandard Complex Numbers

theory *NSComplex*
imports *Complex*
begin

types *hcomplex* = *complex star*

syntax *hcomplex-of-complex* :: *real* => *real star*
translations *hcomplex-of-complex* => *star-of* :: *complex* => *complex star*

constdefs

hRe :: *hcomplex* => *hypreal*
hRe == *f* *Re*

hIm :: *hcomplex* => *hypreal*
hIm == *f* *Im*

hcmod :: *hcomplex* => *hypreal*
hcmod == *f* *cmod*

iii :: *hcomplex*
iii == *star-of ii*

hcnj :: *hcomplex* => *hcomplex*
hcnj == *f* *cnj*

hsgn :: *hcomplex* => *hcomplex*
hsgn == *f* *sgn*

harg :: *hcomplex* => *hypreal*
harg == *f* *arg*

hcis :: *hypreal* => *hcomplex*
hcis == **f* cis*

hcomplex-of-hypreal :: *hypreal* => *hcomplex*
hcomplex-of-hypreal == **f* complex-of-real*

hrcis :: [*hypreal*, *hypreal*] => *hcomplex*
hrcis == **f2* rcis*

hexpi :: *hcomplex* => *hcomplex*
hexpi == **f* expi*

HComplex :: [*hypreal*,*hypreal*] => *hcomplex*
HComplex == **f2* Complex*

hcpow :: [*hcomplex*,*hypnat*] => *hcomplex* (**infixr** *hcpow 80*)
(*z*::*hcomplex*) *hcpow* (*n*::*hypnat*) == (**f2* op ^*) *z n*

lemmas *hcomplex-defs* [*transfer-unfold*] =
hRe-def hIm-def hcmod-def iii-def hcnj-def hsgn-def harg-def hcis-def hcomplex-of-hypreal-def hrcis-def hexpi-def HComplex-def hcpow-def

36.1 Properties of Nonstandard Real and Imaginary Parts

lemma *hRe*: *hRe* (*star-n X*) = *star-n* (%*n*. *Re*(*X n*))
⟨*proof*⟩

lemma *hIm*: *hIm* (*star-n X*) = *star-n* (%*n*. *Im*(*X n*))
⟨*proof*⟩

lemma *hcomplex-hRe-hIm-cancel-iff*:
!!*w z*. (*w*=*z*) = (*hRe*(*w*) = *hRe*(*z*) & *hIm*(*w*) = *hIm*(*z*))
⟨*proof*⟩

lemma *hcomplex-equality* [*intro?*]: *hRe z* = *hRe w* ==> *hIm z* = *hIm w* ==> *z* = *w*
⟨*proof*⟩

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

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

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

lemma *hcomplex-hIm-one* [*simp*]: $hIm\ 1 = 0$
⟨*proof*⟩

36.2 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*⟩

36.3 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*]: $iii * (iii * z) = -z$
⟨*proof*⟩

lemma *hcomplex-i-not-zero* [*simp*]: $iii \neq 0$
⟨*proof*⟩

36.4 More Multiplication Laws

lemma *hcomplex-mult-minus-one* [*simp*]: $-\ 1 * (z::hcomplex) = -z$
⟨*proof*⟩

lemma *hcomplex-mult-minus-one-right* [*simp*]: $(z::hcomplex) * -\ 1 = -z$
⟨*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*⟩

36.5 Subraction and Division

lemma *hcomplex-diff-eq-eq* [*simp*]: $((x::hcomplex) - y = z) = (x = z + y)$
⟨*proof*⟩

lemma *hcomplex-add-divide-distrib*: $(x+y)/(z::hcomplex) = x/z + y/z$
⟨*proof*⟩

36.6 Embedding Properties for *hcomplex-of-hypreal* Map

lemma *hcomplex-of-hypreal*:
hcomplex-of-hypreal (*star-n* X) = *star-n* (%n. *complex-of-real* (X n))
⟨*proof*⟩

lemma *hcomplex-of-hypreal-cancel-iff* [*iff*]:
!!x y. (*hcomplex-of-hypreal* x = *hcomplex-of-hypreal* y) = ($x = y$)
⟨*proof*⟩

lemma *hcomplex-of-hypreal-one* [*simp*]: *hcomplex-of-hypreal 1 = 1*
⟨*proof*⟩

lemma *hcomplex-of-hypreal-zero* [*simp*]: *hcomplex-of-hypreal 0 = 0*
⟨*proof*⟩

lemma *hcomplex-of-hypreal-minus* [*simp*]:
!!x. *hcomplex-of-hypreal*($-x$) = $-$ *hcomplex-of-hypreal* x
⟨*proof*⟩

lemma *hcomplex-of-hypreal-inverse* [*simp*]:
!!x. *hcomplex-of-hypreal*(*inverse* x) = *inverse*(*hcomplex-of-hypreal* x)
⟨*proof*⟩

lemma *hcomplex-of-hypreal-add* [*simp*]:
!!x y. *hcomplex-of-hypreal* ($x + y$) =
hcomplex-of-hypreal x + *hcomplex-of-hypreal* y
⟨*proof*⟩

lemma *hcomplex-of-hypreal-diff* [*simp*]:
!!x y. *hcomplex-of-hypreal* ($x - y$) =
hcomplex-of-hypreal x $-$ *hcomplex-of-hypreal* y
⟨*proof*⟩

lemma *hcomplex-of-hypreal-mult* [*simp*]:
!!x y. *hcomplex-of-hypreal* ($x * y$) =
hcomplex-of-hypreal x * *hcomplex-of-hypreal* y
⟨*proof*⟩

lemma *hcomplex-of-hypreal-divide* [*simp*]:
!!x y. *hcomplex-of-hypreal*(x/y) =
hcomplex-of-hypreal x / *hcomplex-of-hypreal* y

⟨*proof*⟩

lemma *hRe-hcomplex-of-hypreal* [*simp*]: !!z. *hRe*(*hcomplex-of-hypreal* z) = z
⟨*proof*⟩

lemma *hIm-hcomplex-of-hypreal* [*simp*]: !!z. *hIm*(*hcomplex-of-hypreal* z) = *0*
⟨*proof*⟩

lemma *hcomplex-of-hypreal-epsilon-not-zero* [*simp*]:
hcomplex-of-hypreal epsilon ≠ *0*
⟨*proof*⟩

36.7 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*⟩

Relates the two nonstandard constructions

lemma *HComplex-eq-Abs-star-Complex*:
HComplex (*star-n* X) (*star-n* Y) =
star-n (%n::*nat*. *Complex* (X n) (Y n))
⟨*proof*⟩

lemma *hcomplex-surj* [*simp*]: *HComplex* (*hRe* z) (*hIm* z) = z
⟨*proof*⟩

lemma *hcomplex-induct* [*case-names rect*]:
($\bigwedge x$ y. P (*HComplex* x y)) ==> P z
⟨*proof*⟩

36.8 Modulus (Absolute Value) of Nonstandard Complex Number

lemma *hcmod*: *hcmod* (*star-n* X) = *star-n* (%n. *cmod* (X n))
⟨*proof*⟩

lemma *hcmod-zero* [*simp*]: *hcmod*(*0*) = *0*
⟨*proof*⟩

lemma *hcmod-one* [*simp*]: *hcmod*(*1*) = *1*
⟨*proof*⟩

lemma *hcmod-hcomplex-of-hypreal* [*simp*]:
!!x. *hcmod*(*hcomplex-of-hypreal* x) = *abs* x
⟨*proof*⟩

lemma *hcomplex-of-hypreal-abs*:
  $hcomplex\text{-}of\text{-}hypreal\ (abs\ x) =$
  $hcomplex\text{-}of\text{-}hypreal(hcmod(hcomplex\text{-}of\text{-}hypreal\ x))$
⟨*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*: $!!r.\ hcomplex\text{-}of\text{-}hypreal\ r = HComplex\ r\ 0$
⟨*proof*⟩

lemma *HComplex-add-hcomplex-of-hypreal* [*simp*]:
  $HComplex\ x\ y + hcomplex\text{-}of\text{-}hypreal\ r = HComplex\ (x{+}r)\ y$
⟨*proof*⟩

lemma *hcomplex-of-hypreal-add-HComplex* [*simp*]:
  $hcomplex\text{-}of\text{-}hypreal\ r + HComplex\ x\ y = HComplex\ (r{+}x)\ y$
⟨*proof*⟩

lemma *HComplex-mult-hcomplex-of-hypreal*:
  $HComplex\ x\ y * hcomplex\text{-}of\text{-}hypreal\ r = HComplex\ (x{*}r)\ (y{*}r)$
⟨*proof*⟩

lemma *hcomplex-of-hypreal-mult-HComplex*:
  $hcomplex\text{-}of\text{-}hypreal\ r * HComplex\ x\ y = HComplex\ (r{*}x)\ (r{*}y)$
⟨*proof*⟩

lemma *i-hcomplex-of-hypreal* [*simp*]:
  $!!r.\ iii * hcomplex\text{-}of\text{-}hypreal\ r = HComplex\ 0\ r$
⟨*proof*⟩

lemma *hcomplex-of-hypreal-i* [*simp*]:
$!!r.$ *hcomplex-of-hypreal r* $*$ *iii* $=$ *HComplex 0 r*
⟨*proof*⟩

36.9 Conjugation

lemma *hcnj*: *hcnj* (*star-n X*) $=$ *star-n* (%*n*. *cnj*(*X n*))
⟨*proof*⟩

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*:
  $!!z.\ z * hcnj\ z = hcomplex\text{-}of\text{-}hypreal\ (hRe(z)\ \hat{}\ 2 + hIm(z)\ \hat{}\ 2)$
⟨*proof*⟩

36.10 More Theorems about the Function *hcmod*

lemma *hcomplex-hcmod-eq-zero-cancel* [*simp*]: $!!x.\ (hcmod\ x = 0) = (x = 0)$
⟨*proof*⟩

lemma *hcmod-hcomplex-of-hypreal-of-nat* [*simp*]:
  $hcmod\ (hcomplex\text{-}of\text{-}hypreal(hypreal\text{-}of\text{-}nat\ n)) = hypreal\text{-}of\text{-}nat\ n$
⟨*proof*⟩

lemma *hcmod-hcomplex-of-hypreal-of-hypnat* [*simp*]:
  $hcmod\ (hcomplex\text{-}of\text{-}hypreal(hypreal\text{-}of\text{-}hypnat\ n)) = hypreal\text{-}of\text{-}hypnat\ n$
⟨*proof*⟩

lemma *hcmod-minus* [*simp*]: $!!x.\ hcmod\ (-x) = hcmod(x)$
⟨*proof*⟩

lemma *hcmod-mult-hcnj*: $!!z.\ hcmod(z * hcnj(z)) = hcmod(z)\ \hat{}\ 2$
⟨*proof*⟩

lemma *hcmod-ge-zero* [*simp*]: $!!x.\ (0::hypreal) \leq hcmod\ x$
⟨*proof*⟩

lemma *hrabs-hcmod-cancel* [*simp*]: $abs(hcmod\ x) = hcmod\ x$
⟨*proof*⟩

lemma *hcmod-mult*: $!!x\ y.\ hcmod(x*y) = hcmod(x) * hcmod(y)$
⟨*proof*⟩

lemma *hcmod-add-squared-eq*:
  $!!x\ y.\ hcmod(x + y)\ \hat{}\ 2 = hcmod(x)\ \hat{}\ 2 + hcmod(y)\ \hat{}\ 2 + 2 * hRe(x * hcnj\ y)$
⟨*proof*⟩

lemma *hcomplex-hRe-mult-hcnj-le-hcmod* [*simp*]:
  $!!x\ y.\ hRe(x * hcnj\ y) \leq hcmod(x * hcnj\ y)$
⟨*proof*⟩

lemma *hcomplex-hRe-mult-hcnj-le-hcmod2* [*simp*]:
  $!!x\ y.\ hRe(x * hcnj\ y) \leq hcmod(x * y)$
⟨*proof*⟩

lemma *hcmod-triangle-squared* [*simp*]:
  $!!x\ y.\ hcmod\ (x + y)\ \hat{}\ 2 \leq (hcmod(x) + hcmod(y))\ \hat{}\ 2$
⟨*proof*⟩

lemma *hcmod-triangle-ineq* [*simp*]:
  $!!x\ y.\ hcmod\ (x + y) \le hcmod(x) + hcmod(y)$
⟨*proof*⟩

lemma *hcmod-triangle-ineq2* [*simp*]:
  $!!a\ b.\ hcmod(b + a) - hcmod\ b \le hcmod\ a$
⟨*proof*⟩

lemma *hcmod-diff-commute*: $!!x\ y.\ hcmod\ (x - y) = hcmod\ (y - x)$
⟨*proof*⟩

lemma *hcmod-add-less*:
  $!!x\ y\ r\ s.\ [|\ hcmod\ x < r;\ hcmod\ y < s\ |] ==> hcmod\ (x + y) < r + s$
⟨*proof*⟩

lemma *hcmod-mult-less*:
  $!!x\ y\ r\ s.\ [|\ hcmod\ x < r;\ hcmod\ y < s\ |] ==> hcmod\ (x * y) < r * s$
⟨*proof*⟩

lemma *hcmod-diff-ineq* [*simp*]: $!!a\ b.\ hcmod(a) - hcmod(b) \le hcmod(a + b)$
⟨*proof*⟩

36.11 A Few Nonlinear Theorems

lemma *hcpow*: $star\text{-}n\ X\ hcpow\ star\text{-}n\ Y = star\text{-}n\ (\%n.\ X\ n\ \hat{}\ Y\ n)$
⟨*proof*⟩

lemma *hcomplex-of-hypreal-hyperpow*:
  $!!x\ n.\ hcomplex\text{-}of\text{-}hypreal\ (x\ pow\ n) = (hcomplex\text{-}of\text{-}hypreal\ x)\ hcpow\ n$
⟨*proof*⟩

lemma *hcmod-hcpow*: $!!x\ n.\ hcmod(x\ hcpow\ n) = hcmod(x)\ pow\ n$
⟨*proof*⟩

lemma *hcmod-hcomplex-inverse*: $!!x.\ hcmod(inverse\ x) = inverse(hcmod\ x)$
⟨*proof*⟩

lemma *hcmod-divide*: $hcmod(x/y) = hcmod(x)/(hcmod\ y)$
⟨*proof*⟩

36.12 Exponentiation

lemma *hcomplexpow-0* [*simp*]: $z\ \hat{}\ 0 \quad = (1::hcomplex)$
⟨*proof*⟩

lemma *hcomplexpow-Suc* [*simp*]: $z\ \hat{}\ (Suc\ n) = (z::hcomplex) * (z\ \hat{}\ n)$
⟨*proof*⟩

lemma *hcomplexpow-i-squared* [*simp*]: $iii\ \hat{}\ 2 = -\ 1$

⟨*proof*⟩

lemma *hcomplex-of-hypreal-pow*:
hcomplex-of-hypreal $(x \,\hat{}\, n) = (hcomplex\text{-}of\text{-}hypreal\ x) \,\hat{}\, n$
⟨*proof*⟩

lemma *hcomplex-hcnj-pow*: $hcnj(z \,\hat{}\, n) = hcnj(z) \,\hat{}\, n$
⟨*proof*⟩

lemma *hcmod-hcomplexpow*: $hcmod(x \,\hat{}\, n) = hcmod(x) \,\hat{}\, n$
⟨*proof*⟩

lemma *hcpow-minus*:
$!!x\ n.\ (-x::hcomplex)\ hcpow\ n =$
$(if\ (\ *p*\ even)\ n\ then\ (x\ hcpow\ n)\ else\ -(x\ hcpow\ n))$
⟨*proof*⟩

lemma *hcpow-mult*:
$!!r\ s\ n.\ ((r::hcomplex) * s)\ hcpow\ n = (r\ hcpow\ n) * (s\ hcpow\ n)$
⟨*proof*⟩

lemma *hcpow-zero* [*simp*]: $!!n.\ 0\ hcpow\ (n + 1) = 0$
⟨*proof*⟩

lemma *hcpow-zero2* [*simp*]: $0\ hcpow\ (hSuc\ n) = 0$
⟨*proof*⟩

lemma *hcpow-not-zero* [*simp*,*intro*]:
$!!r\ n.\ r \neq 0 ==> r\ hcpow\ n \neq (0::hcomplex)$
⟨*proof*⟩

lemma *hcpow-zero-zero*: $r\ hcpow\ n = (0::hcomplex) ==> r = 0$
⟨*proof*⟩

lemma *star-n-divide*: $star\text{-}n\ X\ /\ star\text{-}n\ Y = star\text{-}n\ (\%n.\ X\ n\ /\ Y\ n)$
⟨*proof*⟩

36.13 The Function *hsgn*

lemma *hsgn*: $hsgn\ (star\text{-}n\ X) = star\text{-}n\ (\%n.\ sgn\ (X\ n))$
⟨*proof*⟩

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

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

lemma *hsgn-minus*: $!!z.\ hsgn\ (-z) = -\ hsgn(z)$
⟨*proof*⟩

lemma *hsgn-eq*: $!!z.\ hsgn\ z = z\ /\ hcomplex\text{-}of\text{-}hypreal\ (hcmod\ z)$
⟨*proof*⟩

lemma *hcmod-i*: $!!x\ y.\ hcmod\ (HComplex\ x\ y) = (\ *f*\ sqrt)\ (x\ \hat{}\ 2 + y\ \hat{}\ 2)$
⟨*proof*⟩

lemma *hcomplex-eq-cancel-iff1* [*simp*]:
$(hcomplex\text{-}of\text{-}hypreal\ xa = HComplex\ x\ y) = (xa = x\ \&\ y = 0)$
⟨*proof*⟩

lemma *hcomplex-eq-cancel-iff2* [*simp*]:
$(HComplex\ x\ y = hcomplex\text{-}of\text{-}hypreal\ xa) = (x = xa\ \&\ y = 0)$
⟨*proof*⟩

lemma *HComplex-eq-0* [*simp*]: $(HComplex\ x\ y = 0) = (x = 0\ \&\ y = 0)$
⟨*proof*⟩

lemma *HComplex-eq-1* [*simp*]: $(HComplex\ x\ y = 1) = (x = 1\ \&\ y = 0)$
⟨*proof*⟩

lemma *i-eq-HComplex-0-1*: $iii = HComplex\ 0\ 1$
⟨*proof*⟩

lemma *HComplex-eq-i* [*simp*]: $(HComplex\ x\ y = iii) = (x = 0\ \&\ y = 1)$
⟨*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 *real-two-squares-add-zero-iff* [*simp*]: $(x*x + y*y = 0) = ((x::real) = 0\ \&\ y = 0)$
⟨*proof*⟩

lemma *hcomplex-inverse-complex-split*:
$!!x\ y.\ inverse(hcomplex\text{-}of\text{-}hypreal\ x + iii * hcomplex\text{-}of\text{-}hypreal\ y) =$
$hcomplex\text{-}of\text{-}hypreal(x/(x\ \hat{}\ 2 + y\ \hat{}\ 2)) -$
$iii * hcomplex\text{-}of\text{-}hypreal(y/(x\ \hat{}\ 2 + y\ \hat{}\ 2))$
⟨*proof*⟩

lemma *HComplex-inverse*:
$!!x\ y.\ inverse\ (HComplex\ x\ y) =$
$HComplex\ (x/(x\ \hat{}\ 2 + y\ \hat{}\ 2))\ (-y/(x\ \hat{}\ 2 + y\ \hat{}\ 2))$

⟨*proof*⟩

lemma *hRe-mult-i-eq*[*simp*]:
  $!!y.\ hRe\ (iii * hcomplex\text{-}of\text{-}hypreal\ y) = 0$
⟨*proof*⟩

lemma *hIm-mult-i-eq* [*simp*]:
  $!!y.\ hIm\ (iii * hcomplex\text{-}of\text{-}hypreal\ y) = y$
⟨*proof*⟩

lemma *hcmod-mult-i* [*simp*]: $!!y.\ hcmod\ (iii * hcomplex\text{-}of\text{-}hypreal\ y) = abs\ y$
⟨*proof*⟩

lemma *hcmod-mult-i2* [*simp*]: $hcmod\ (hcomplex\text{-}of\text{-}hypreal\ y * iii) = abs\ y$
⟨*proof*⟩

lemma *harg*: $harg\ (star\text{-}n\ X) = star\text{-}n\ (\%n.\ arg\ (X\ n))$
⟨*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*]:
  $y \neq 0 ==> (\ *f*\ cos)\ (harg(HComplex\ 0\ y)) = 0$
⟨*proof*⟩

lemma *hcomplex-of-hypreal-zero-iff* [*simp*]:
  $!!y.\ (hcomplex\text{-}of\text{-}hypreal\ y = 0) = (y = 0)$
⟨*proof*⟩

36.14 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 *lemma-hypreal-P-EX2*:
  $(\exists (x::hypreal)\ y.\ P\ x\ y) =$
  $(\exists f\ g.\ P\ (star\text{-}n\ f)\ (star\text{-}n\ g))$
⟨*proof*⟩

lemma *hcomplex-split-polar*:
$!!z.\ \exists\, r\ a.\ z = \mathit{hcomplex\text{-}of\text{-}hypreal}\ r * (\mathit{HComplex}((\ *f*\ \mathit{cos})\ a)((\ *f*\ \mathit{sin})\ a))$
⟨*proof*⟩

lemma *hcis*: $\mathit{hcis}\ (\mathit{star\text{-}n}\ X) = \mathit{star\text{-}n}\ (\%n.\ \mathit{cis}\ (X\ n))$
⟨*proof*⟩

lemma *hcis-eq*:
$!!a.\ \mathit{hcis}\ a =$
$(\mathit{hcomplex\text{-}of\text{-}hypreal}((\ *f*\ \mathit{cos})\ a) +$
$\mathit{iii} * \mathit{hcomplex\text{-}of\text{-}hypreal}((\ *f*\ \mathit{sin})\ a))$
⟨*proof*⟩

lemma *hrcis*: $\mathit{hrcis}\ (\mathit{star\text{-}n}\ X)\ (\mathit{star\text{-}n}\ Y) = \mathit{star\text{-}n}\ (\%n.\ \mathit{rcis}\ (X\ n)\ (Y\ n))$
⟨*proof*⟩

lemma *hrcis-Ex*: $!!z.\ \exists\, r\ a.\ z = \mathit{hrcis}\ r\ a$
⟨*proof*⟩

lemma *hRe-hcomplex-polar* [*simp*]:
$\mathit{hRe}\ (\mathit{hcomplex\text{-}of\text{-}hypreal}\ r * \mathit{HComplex}\ ((\ *f*\ \mathit{cos})\ a)\ ((\ *f*\ \mathit{sin})\ a)) =$
$r * (\ *f*\ \mathit{cos})\ a$
⟨*proof*⟩

lemma *hRe-hrcis* [*simp*]: $!!r\ a.\ \mathit{hRe}(\mathit{hrcis}\ r\ a) = r * (\ *f*\ \mathit{cos})\ a$
⟨*proof*⟩

lemma *hIm-hcomplex-polar* [*simp*]:
$\mathit{hIm}\ (\mathit{hcomplex\text{-}of\text{-}hypreal}\ r * \mathit{HComplex}\ ((\ *f*\ \mathit{cos})\ a)\ ((\ *f*\ \mathit{sin})\ a)) =$
$r * (\ *f*\ \mathit{sin})\ a$
⟨*proof*⟩

lemma *hIm-hrcis* [*simp*]: $!!r\ a.\ \mathit{hIm}(\mathit{hrcis}\ r\ a) = r * (\ *f*\ \mathit{sin})\ a$
⟨*proof*⟩

lemma *hcmod-unit-one* [*simp*]:
$!!a.\ \mathit{hcmod}\ (\mathit{HComplex}\ ((\ *f*\ \mathit{cos})\ a)\ ((\ *f*\ \mathit{sin})\ a)) = \mathit{1}$
⟨*proof*⟩

lemma *hcmod-complex-polar* [*simp*]:
$\mathit{hcmod}\ (\mathit{hcomplex\text{-}of\text{-}hypreal}\ r * \mathit{HComplex}\ ((\ *f*\ \mathit{cos})\ a)\ ((\ *f*\ \mathit{sin})\ a)) =$
$\mathit{abs}\ r$
⟨*proof*⟩

lemma *hcmod-hrcis* [*simp*]: $!!r\ a.\ \mathit{hcmod}(\mathit{hrcis}\ r\ a) = \mathit{abs}\ r$
⟨*proof*⟩

lemma *hcis-hrcis-eq*: !!a. *hcis* a = *hrcis* 1 a
⟨*proof*⟩
declare *hcis-hrcis-eq* [*symmetric*, *simp*]

lemma *hrcis-mult*:
!!a b $r1$ $r2$. *hrcis* $r1$ a ∗ *hrcis* $r2$ b = *hrcis* ($r1$∗$r2$) (a + b)
⟨*proof*⟩

lemma *hcis-mult*: !!a b. *hcis* a ∗ *hcis* b = *hcis* (a + b)
⟨*proof*⟩

lemma *hcis-zero* [*simp*]: *hcis* 0 = 1
⟨*proof*⟩

lemma *hrcis-zero-mod* [*simp*]: !!a. *hrcis* 0 a = 0
⟨*proof*⟩

lemma *hrcis-zero-arg* [*simp*]: !!r. *hrcis* r 0 = *hcomplex-of-hypreal* r
⟨*proof*⟩

lemma *hcomplex-i-mult-minus* [*simp*]: *iii* ∗ (*iii* ∗ x) = − x
⟨*proof*⟩

lemma *hcomplex-i-mult-minus2* [*simp*]: *iii* ∗ *iii* ∗ x = − x
⟨*proof*⟩

lemma *hcis-hypreal-of-nat-Suc-mult*:
!!a. *hcis* (*hypreal-of-nat* (*Suc* n) ∗ a) =
hcis a ∗ *hcis* (*hypreal-of-nat* n ∗ a)
⟨*proof*⟩

lemma *NSDeMoivre*: (*hcis* a) ^ n = *hcis* (*hypreal-of-nat* n ∗ a)
⟨*proof*⟩

lemma *hcis-hypreal-of-hypnat-Suc-mult*:
!! a n. *hcis* (*hypreal-of-hypnat* (n + 1) ∗ a) =
hcis a ∗ *hcis* (*hypreal-of-hypnat* n ∗ a)
⟨*proof*⟩

lemma *NSDeMoivre-ext*:
!!a n. (*hcis* a) *hcpow* n = *hcis* (*hypreal-of-hypnat* n ∗ a)
⟨*proof*⟩

lemma *NSDeMoivre2*:
!!a r. (*hrcis* r a) ^ n = *hrcis* (r ^ n) (*hypreal-of-nat* n ∗ a)

⟨*proof*⟩

lemma *DeMoivre2-ext*:
$!!\ a\ r\ n.\ (hrcis\ r\ a)\ hcpow\ n = hrcis\ (r\ pow\ n)\ (hypreal\text{-}of\text{-}hypnat\ n * a)$
⟨*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*: $(\ *f*\ cos)\ (hypreal\text{-}of\text{-}hypnat\ n * a) = hRe(hcis\ a\ hcpow\ n)$
⟨*proof*⟩

lemma *sin-n-hIm-hcis-hcpow-n*: $(\ *f*\ sin)\ (hypreal\text{-}of\text{-}hypnat\ n * a) = hIm(hcis\ a\ hcpow\ n)$
⟨*proof*⟩

lemma *hexpi-add*: $!!a\ b.\ hexpi(a + b) = hexpi(a) * hexpi(b)$
⟨*proof*⟩

36.15 *star-of*: the Injection from type *complex* to to *hcomplex*

lemma *inj-hcomplex-of-complex*: *inj*(*hcomplex-of-complex*)
⟨*proof*⟩

lemma *hcomplex-of-complex-i*: $iii = hcomplex\text{-}of\text{-}complex\ ii$
⟨*proof*⟩

lemma *hRe-hcomplex-of-complex*:
$hRe\ (hcomplex\text{-}of\text{-}complex\ z) = hypreal\text{-}of\text{-}real\ (Re\ z)$
⟨*proof*⟩

lemma *hIm-hcomplex-of-complex*:
$hIm\ (hcomplex\text{-}of\text{-}complex\ z) = hypreal\text{-}of\text{-}real\ (Im\ z)$

⟨*proof*⟩

lemma *hcmod-hcomplex-of-complex*:
hcmod (*hcomplex-of-complex* x) = *hypreal-of-real* (*cmod* x)
⟨*proof*⟩

36.16 Numerals and Arithmetic

lemma *hcomplex-number-of-def*: (*number-of* w :: *hcomplex*) == *of-int* (*Rep-Bin* 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*⟩

This theorem is necessary because theorems such as *iszero-number-of-0* only hold for ordered rings. They cannot be generalized to fields in general because they fail for finite fields. They work for type complex because the reals can be embedded in them.

lemma *iszero-hcomplex-number-of* [*simp*]:
iszero (*number-of* w :: *hcomplex*) = *iszero* (*number-of* w :: *real*)
⟨*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*⟩

⟨*ML*⟩

end

37 NSCA: Non-Standard Complex Analysis

theory *NSCA*
imports *NSComplex*
begin

constdefs

CInfinitesimal :: *hcomplex set*
CInfinitesimal == {*x*. ∀ *r* ∈ *Reals*. *0* < *r* --> *hcmod x* < *r*}

capprox :: [*hcomplex*,*hcomplex*] => *bool* (**infixl** @*c*= *50*)
— the "infinitely close" relation
x @*c*= *y* == (*x* − *y*) ∈ *CInfinitesimal*

SComplex :: *hcomplex set*
SComplex == {*x*. ∃ *r*. *x* = *hcomplex-of-complex r*}

CFinite :: *hcomplex set*
CFinite == {*x*. ∃ *r* ∈ *Reals*. *hcmod x* < *r*}

CInfinite :: *hcomplex set*
CInfinite == {*x*. ∀ *r* ∈ *Reals*. *r* < *hcmod x*}

stc :: *hcomplex* => *hcomplex*
— standard part map
stc x == (@*r*. *x* ∈ *CFinite* & *r*:*SComplex* & *r* @*c*= *x*)

cmonad :: *hcomplex* => *hcomplex set*
cmonad x == {*y*. *x* @*c*= *y*}

cgalaxy :: *hcomplex* => *hcomplex set*
cgalaxy x == {*y*. (*x* − *y*) ∈ *CFinite*}

37.1 Closure Laws for SComplex, the Standard Complex Numbers

lemma *SComplex-add*: [| *x* ∈ *SComplex*; *y* ∈ *SComplex* |] ==> *x* + *y* ∈ *SComplex*

⟨*proof*⟩

lemma *SComplex-mult*: $[| \, x \in SComplex; \, y \in SComplex \, |] ==> x * y \in SComplex$
⟨*proof*⟩

lemma *SComplex-inverse*: $x \in SComplex ==> inverse\ x \in SComplex$
⟨*proof*⟩

lemma *SComplex-divide*: $[| \, x \in SComplex; \; y \in SComplex \, |] ==> x/y \in SComplex$
⟨*proof*⟩

lemma *SComplex-minus*: $x \in SComplex ==> -x \in SComplex$
⟨*proof*⟩

lemma *SComplex-minus-iff* [*simp*]: $(-x \in SComplex) = (x \in SComplex)$
⟨*proof*⟩

lemma *SComplex-diff*: $[| \, x \in SComplex; \, y \in SComplex \, |] ==> x - y \in SComplex$
⟨*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-of-complex r*) ∈ *Reals*
⟨*proof*⟩

lemma *SReal-hcmod-number-of* [*simp*]: *hcmod* (*number-of w* ::*hcomplex*) ∈ *Reals*
⟨*proof*⟩

lemma *SReal-hcmod-SComplex*: $x \in SComplex ==> hcmod\ x \in Reals$
⟨*proof*⟩

lemma *SComplex-hcomplex-of-complex* [*simp*]: *hcomplex-of-complex x* ∈ *SComplex*
⟨*proof*⟩

lemma *SComplex-number-of* [*simp*]: (*number-of w* ::*hcomplex*) ∈ *SComplex*
⟨*proof*⟩

lemma *SComplex-divide-number-of*:
r ∈ *SComplex* ==> *r*/(*number-of w*::*hcomplex*) ∈ *SComplex*
⟨*proof*⟩

lemma *SComplex-UNIV-complex*:
{*x*. *hcomplex-of-complex x* ∈ *SComplex*} = (*UNIV*::*complex set*)
⟨*proof*⟩

lemma *SComplex-iff*: (*x* ∈ *SComplex*) = (∃ *y*. *x* = *hcomplex-of-complex y*)

⟨*proof*⟩

lemma *hcomplex-of-complex-image*:
hcomplex-of-complex ‘(*UNIV*::*complex set*) = *SComplex*
⟨*proof*⟩

lemma *inv-hcomplex-of-complex-image*: *inv hcomplex-of-complex* ‘*SComplex* = *UNIV*
⟨*proof*⟩

lemma *SComplex-hcomplex-of-complex-image*:
[| $\exists x.\ x$: *P*; $P \le$ *SComplex* |] ==> $\exists Q.\ P$ = *hcomplex-of-complex* ‘ *Q*
⟨*proof*⟩

lemma *SComplex-SReal-dense*:
[| $x \in$ *SComplex*; $y \in$ *SComplex*; *hcmod* $x <$ *hcmod* y
|] ==> $\exists r \in$ *Reals*. *hcmod* $x<$ r & $r <$ *hcmod* y
⟨*proof*⟩

lemma *SComplex-hcmod-SReal*:
$z \in$ *SComplex* ==> *hcmod* $z \in$ *Reals*
⟨*proof*⟩

lemma *SComplex-zero* [*simp*]: *0* $\in$ *SComplex*
⟨*proof*⟩

lemma *SComplex-one* [*simp*]: *1* $\in$ *SComplex*
⟨*proof*⟩

37.2 The Finite Elements form a Subring

lemma *CFinite-add*: [|$x \in$ *CFinite*; $y \in$ *CFinite*|] ==> $(x+y) \in$ *CFinite*
⟨*proof*⟩

lemma *CFinite-mult*: [|$x \in$ *CFinite*; $y \in$ *CFinite*|] ==> $x*y \in$ *CFinite*
⟨*proof*⟩

lemma *CFinite-minus-iff* [*simp*]: $(-x \in$ *CFinite*$) = (x \in$ *CFinite*$)$
⟨*proof*⟩

lemma *SComplex-subset-CFinite* [*simp*]: *SComplex* $\le$ *CFinite*
⟨*proof*⟩

lemma *HFinite-hcmod-hcomplex-of-complex* [*simp*]:
hcmod (*hcomplex-of-complex* r) $\in$ *HFinite*
⟨*proof*⟩

lemma *CFinite-hcomplex-of-complex* [*simp*]: *hcomplex-of-complex* $x \in$ *CFinite*
⟨*proof*⟩

lemma *CFiniteD*: $x \in CFinite ==> \exists\, t \in Reals.\ hcmod\ x < t$
⟨*proof*⟩

lemma *CFinite-hcmod-iff*: $(x \in CFinite) = (hcmod\ x \in HFinite)$
⟨*proof*⟩

lemma *CFinite-number-of* [*simp*]: *number-of* $w \in CFinite$
⟨*proof*⟩

lemma *CFinite-bounded*: $[|x \in CFinite;\ y \leq hcmod\ x;\ 0 \leq y\ |] ==> y: HFinite$
⟨*proof*⟩

37.3 The Complex Infinitesimals form a Subring

lemma *CInfinitesimal-zero* [*iff*]: $0 \in CInfinitesimal$
⟨*proof*⟩

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

lemma *CInfinitesimal-hcmod-iff*:
$(z \in CInfinitesimal) = (hcmod\ z \in Infinitesimal)$
⟨*proof*⟩

lemma *one-not-CInfinitesimal* [*simp*]: $1 \notin CInfinitesimal$
⟨*proof*⟩

lemma *CInfinitesimal-add*:
$[|\ x \in CInfinitesimal;\ y \in CInfinitesimal\ |] ==> (x+y) \in CInfinitesimal$
⟨*proof*⟩

lemma *CInfinitesimal-minus-iff* [*simp*]:
$(-x{:}CInfinitesimal) = (x{:}CInfinitesimal)$
⟨*proof*⟩

lemma *CInfinitesimal-diff*:
$[|\ x \in CInfinitesimal;\ \ y \in CInfinitesimal\ |] ==> x-y \in CInfinitesimal$
⟨*proof*⟩

lemma *CInfinitesimal-mult*:
$[|\ x \in CInfinitesimal;\ y \in CInfinitesimal\ |] ==> x * y \in CInfinitesimal$
⟨*proof*⟩

lemma *CInfinitesimal-CFinite-mult*:
$[|\ x \in CInfinitesimal;\ y \in CFinite\ |] ==> (x * y) \in CInfinitesimal$
⟨*proof*⟩

lemma *CInfinitesimal-CFinite-mult2*:
$[|\ x \in CInfinitesimal;\ y \in CFinite\ |] ==> (y * x) \in CInfinitesimal$

⟨*proof*⟩

lemma *CInfinite-hcmod-iff*: $(z \in \mathit{CInfinite}) = (\mathit{hcmod}\ z \in \mathit{HInfinite})$
⟨*proof*⟩

lemma *CInfinite-inverse-CInfinitesimal*:
$x \in \mathit{CInfinite} ==> \mathit{inverse}\ x \in \mathit{CInfinitesimal}$
⟨*proof*⟩

lemma *CInfinite-mult*: $[|x \in \mathit{CInfinite};\ y \in \mathit{CInfinite}|] ==> (x*y)\colon \mathit{CInfinite}$
⟨*proof*⟩

lemma *CInfinite-minus-iff* [*simp*]: $(-x \in \mathit{CInfinite}) = (x \in \mathit{CInfinite})$
⟨*proof*⟩

lemma *CFinite-sum-squares*:
$[|a \in \mathit{CFinite};\ b \in \mathit{CFinite};\ c \in \mathit{CFinite}|]$
$==> a*a + b*b + c*c \in \mathit{CFinite}$
⟨*proof*⟩

lemma *not-CInfinitesimal-not-zero*: $x \notin \mathit{CInfinitesimal} ==> x \neq 0$
⟨*proof*⟩

lemma *not-CInfinitesimal-not-zero2*: $x \in \mathit{CFinite} - \mathit{CInfinitesimal} ==> x \neq 0$
⟨*proof*⟩

lemma *CFinite-diff-CInfinitesimal-hcmod*:
$x \in \mathit{CFinite} - \mathit{CInfinitesimal} ==> \mathit{hcmod}\ x \in \mathit{HFinite} - \mathit{Infinitesimal}$
⟨*proof*⟩

lemma *hcmod-less-CInfinitesimal*:
$[|\ e \in \mathit{CInfinitesimal};\ \mathit{hcmod}\ x < \mathit{hcmod}\ e\ |] ==> x \in \mathit{CInfinitesimal}$
⟨*proof*⟩

lemma *hcmod-le-CInfinitesimal*:
$[|\ e \in \mathit{CInfinitesimal};\ \mathit{hcmod}\ x \le \mathit{hcmod}\ e\ |] ==> x \in \mathit{CInfinitesimal}$
⟨*proof*⟩

lemma *CInfinitesimal-interval*:
$[|\ e \in \mathit{CInfinitesimal};$
$e' \in \mathit{CInfinitesimal};$
$\mathit{hcmod}\ e' < \mathit{hcmod}\ x\ ;\ \mathit{hcmod}\ x < \mathit{hcmod}\ e$
$|] ==> x \in \mathit{CInfinitesimal}$
⟨*proof*⟩

lemma *CInfinitesimal-interval2*:
$[|\ e \in \mathit{CInfinitesimal};$
$e' \in \mathit{CInfinitesimal};$
$\mathit{hcmod}\ e' \le \mathit{hcmod}\ x\ ;\ \mathit{hcmod}\ x \le \mathit{hcmod}\ e$

|] ==> $x \in$ *CInfinitesimal*
⟨*proof*⟩

lemma *not-CInfinitesimal-mult*:
[| $x \notin$ *CInfinitesimal*; $y \notin$ *CInfinitesimal*|] ==> $(x*y) \notin$ *CInfinitesimal*
⟨*proof*⟩

lemma *CInfinitesimal-mult-disj*:
$x*y \in$ *CInfinitesimal* ==> $x \in$ *CInfinitesimal* | $y \in$ *CInfinitesimal*
⟨*proof*⟩

lemma *CFinite-CInfinitesimal-diff-mult*:
[| $x \in$ *CFinite* − *CInfinitesimal*; $y \in$ *CFinite* − *CInfinitesimal* |]
==> $x*y \in$ *CFinite* − *CInfinitesimal*
⟨*proof*⟩

lemma *CInfinitesimal-subset-CFinite*: *CInfinitesimal* $\leq$ *CFinite*
⟨*proof*⟩

lemma *CInfinitesimal-hcomplex-of-complex-mult*:
$x \in$ *CInfinitesimal* ==> x * *hcomplex-of-complex* $r \in$ *CInfinitesimal*
⟨*proof*⟩

lemma *CInfinitesimal-hcomplex-of-complex-mult2*:
$x \in$ *CInfinitesimal* ==> *hcomplex-of-complex* $r * x \in$ *CInfinitesimal*
⟨*proof*⟩

37.4 The “Infinitely Close” Relation

lemma *mem-cinfmal-iff*: x:*CInfinitesimal* = (x @c= *0*)
⟨*proof*⟩

lemma *capprox-minus-iff*: (x @c= y) = (x + −y @c= *0*)
⟨*proof*⟩

lemma *capprox-minus-iff2*: (x @c= y) = (−y + x @c= *0*)
⟨*proof*⟩

lemma *capprox-refl* [*simp*]: x @c= x
⟨*proof*⟩

lemma *capprox-sym*: x @c= y ==> y @c= x
⟨*proof*⟩

lemma *capprox-trans*: [| x @c= y; y @c= z |] ==> x @c= z
⟨*proof*⟩

lemma *capprox-trans2*: [| r @c= x; s @c= x |] ==> r @c= s
⟨*proof*⟩

lemma *capprox-trans3*: [| x @c= r; x @c= s|] ==> r @c= s
⟨*proof*⟩

lemma *number-of-capprox-reorient* [*simp*]:
(*number-of* w @c= x) = (x @c= *number-of* w)
⟨*proof*⟩

lemma *CInfinitesimal-capprox-minus*: ($x-y$ ∈ *CInfinitesimal*) = (x @c= y)
⟨*proof*⟩

lemma *capprox-monad-iff*: (x @c= y) = (*cmonad*(x)=*cmonad*(y))
⟨*proof*⟩

lemma *Infinitesimal-capprox*:
[| x ∈ *CInfinitesimal*; y ∈ *CInfinitesimal* |] ==> x @c= y
⟨*proof*⟩

lemma *capprox-add*: [| a @c= b; c @c= d |] ==> $a+c$ @c= $b+d$
⟨*proof*⟩

lemma *capprox-minus*: a @c= b ==> $-a$ @c= $-b$
⟨*proof*⟩

lemma *capprox-minus2*: $-a$ @c= $-b$ ==> a @c= b
⟨*proof*⟩

lemma *capprox-minus-cancel* [*simp*]: ($-a$ @c= $-b$) = (a @c= b)
⟨*proof*⟩

lemma *capprox-add-minus*: [| a @c= b; c @c= d |] ==> $a + -c$ @c= $b + -d$
⟨*proof*⟩

lemma *capprox-mult1*:
[| a @c= b; c ∈ *CFinite*|] ==> $a*c$ @c= $b*c$
⟨*proof*⟩

lemma *capprox-mult2*: [|a @c= b; c ∈ *CFinite*|] ==> $c*a$ @c= $c*b$
⟨*proof*⟩

lemma *capprox-mult-subst*:
[|u @c= $v*x$; x @c= y; v ∈ *CFinite*|] ==> u @c= $v*y$
⟨*proof*⟩

lemma *capprox-mult-subst2*:
[| u @c= $x*v$; x @c= y; v ∈ *CFinite* |] ==> u @c= $y*v$
⟨*proof*⟩

lemma *capprox-mult-subst-SComplex*:

[| u @c= x*hcomplex-of-complex v; x @c= y |]
==> u @c= y*hcomplex-of-complex v
⟨proof⟩

lemma *capprox-eq-imp*: $a = b$ ==> a @c= b
⟨proof⟩

lemma *CInfinitesimal-minus-capprox*: $x \in$ CInfinitesimal ==> $-x$ @c= x
⟨proof⟩

lemma *bex-CInfinitesimal-iff*: ($\exists\, y \in$ CInfinitesimal. $x - z = y$) = (x @c= z)
⟨proof⟩

lemma *bex-CInfinitesimal-iff2*: ($\exists\, y \in$ CInfinitesimal. $x = z + y$) = (x @c= z)
⟨proof⟩

lemma *CInfinitesimal-add-capprox*:
[| $y \in$ CInfinitesimal; $x + y = z$ |] ==> x @c= z
⟨proof⟩

lemma *CInfinitesimal-add-capprox-self*: $y \in$ CInfinitesimal ==> x @c= $x + y$
⟨proof⟩

lemma *CInfinitesimal-add-capprox-self2*: $y \in$ CInfinitesimal ==> x @c= $y + x$
⟨proof⟩

lemma *CInfinitesimal-add-minus-capprox-self*:
$y \in$ CInfinitesimal ==> x @c= $x + -y$
⟨proof⟩

lemma *CInfinitesimal-add-cancel*:
[| $y \in$ CInfinitesimal; $x+y$ @c= z|] ==> x @c= z
⟨proof⟩

lemma *CInfinitesimal-add-right-cancel*:
[| $y \in$ CInfinitesimal; x @c= $z + y$|] ==> x @c= z
⟨proof⟩

lemma *capprox-add-left-cancel*: $d + b$ @c= $d + c$ ==> b @c= c
⟨proof⟩

lemma *capprox-add-right-cancel*: $b + d$ @c= $c + d$ ==> b @c= c
⟨proof⟩

lemma *capprox-add-mono1*: b @c= c ==> $d + b$ @c= $d + c$
⟨proof⟩

lemma *capprox-add-mono2*: b @c= c ==> $b + a$ @c= $c + a$
⟨proof⟩

lemma *capprox-add-left-iff* [*iff*]: $(a + b \; @c= \; a + c) = (b \; @c= \; c)$
⟨*proof*⟩

lemma *capprox-add-right-iff* [*iff*]: $(b + a \; @c= \; c + a) = (b \; @c= \; c)$
⟨*proof*⟩

lemma *capprox-CFinite*: $[| \; x \in CFinite; \; x \; @c= \; y \; |] ==> y \in CFinite$
⟨*proof*⟩

lemma *capprox-hcomplex-of-complex-CFinite*:
$x \; @c= \; hcomplex\text{-}of\text{-}complex \; D ==> x \in CFinite$
⟨*proof*⟩

lemma *capprox-mult-CFinite*:
$[|a \; @c= \; b; \; c \; @c= \; d; \; b \in CFinite; \; d \in CFinite|] ==> a*c \; @c= \; b*d$
⟨*proof*⟩

lemma *capprox-mult-hcomplex-of-complex*:
$[|a \; @c= \; hcomplex\text{-}of\text{-}complex \; b; \; c \; @c= \; hcomplex\text{-}of\text{-}complex \; d \; |]$
$==> a*c \; @c= \; hcomplex\text{-}of\text{-}complex \; b * hcomplex\text{-}of\text{-}complex \; d$
⟨*proof*⟩

lemma *capprox-SComplex-mult-cancel-zero*:
$[| \; a \in SComplex; \; a \neq 0; \; a*x \; @c= \; 0 \; |] ==> x \; @c= \; 0$
⟨*proof*⟩

lemma *capprox-mult-SComplex1*: $[| \; a \in SComplex; \; x \; @c= \; 0 \; |] ==> x*a \; @c= \; 0$
⟨*proof*⟩

lemma *capprox-mult-SComplex2*: $[| \; a \in SComplex; \; x \; @c= \; 0 \; |] ==> a*x \; @c= \; 0$
⟨*proof*⟩

lemma *capprox-mult-SComplex-zero-cancel-iff* [*simp*]:
$[|a \in SComplex; \; a \neq 0 \; |] ==> (a*x \; @c= \; 0) = (x \; @c= \; 0)$
⟨*proof*⟩

lemma *capprox-SComplex-mult-cancel*:
$[| \; a \in SComplex; \; a \neq 0; \; a* \; w \; @c= \; a*z \; |] ==> w \; @c= \; z$
⟨*proof*⟩

lemma *capprox-SComplex-mult-cancel-iff1* [*simp*]:
$[| \; a \in SComplex; \; a \neq 0|] ==> (a* \; w \; @c= \; a*z) = (w \; @c= \; z)$
⟨*proof*⟩

lemma *capprox-hcmod-approx-zero*: $(x \; @c= \; y) = (hcmod \; (y - x) \; @= \; 0)$
⟨*proof*⟩

lemma *capprox-approx-zero-iff*: $(x \; @c= \; 0) = (hcmod \; x \; @= \; 0)$

⟨*proof*⟩

lemma *capprox-minus-zero-cancel-iff* [*simp*]: $(-x\ @c{=}\ 0) = (x\ @c{=}\ 0)$
⟨*proof*⟩

lemma *Infinitesimal-hcmod-add-diff*:
$u\ @c{=}\ 0 ==> hcmod(x + u) - hcmod\ x \in Infinitesimal$
⟨*proof*⟩

lemma *approx-hcmod-add-hcmod*: $u\ @c{=}\ 0 ==> hcmod(x + u)\ @{=}\ hcmod\ x$
⟨*proof*⟩

lemma *capprox-hcmod-approx*: $x\ @c{=}\ y ==> hcmod\ x\ @{=}\ hcmod\ y$
⟨*proof*⟩

37.5 Zero is the Only Infinitesimal Complex Number

lemma *CInfinitesimal-less-SComplex*:
$[|\ x \in SComplex;\ y \in CInfinitesimal;\ 0 < hcmod\ x\ |] ==> hcmod\ y < hcmod\ x$
⟨*proof*⟩

lemma *SComplex-Int-CInfinitesimal-zero*: $SComplex\ Int\ CInfinitesimal = \{0\}$
⟨*proof*⟩

lemma *SComplex-CInfinitesimal-zero*:
$[|\ x \in SComplex;\ x \in CInfinitesimal|] ==> x = 0$
⟨*proof*⟩

lemma *SComplex-CFinite-diff-CInfinitesimal*:
$[|\ x \in SComplex;\ x \neq 0\ |] ==> x \in CFinite - CInfinitesimal$
⟨*proof*⟩

lemma *hcomplex-of-complex-CFinite-diff-CInfinitesimal*:
$hcomplex\text{-}of\text{-}complex\ x \neq 0$
$==> hcomplex\text{-}of\text{-}complex\ x \in CFinite - CInfinitesimal$
⟨*proof*⟩

lemma *hcomplex-of-complex-CInfinitesimal-iff-0* [*iff*]:
$(hcomplex\text{-}of\text{-}complex\ x \in CInfinitesimal) = (x{=}0)$
⟨*proof*⟩

lemma *number-of-not-CInfinitesimal* [*simp*]:
$number\text{-}of\ w \neq (0::hcomplex) ==> number\text{-}of\ w \notin CInfinitesimal$
⟨*proof*⟩

lemma *capprox-SComplex-not-zero*:
$[|\ y \in SComplex;\ x\ @c{=}\ y;\ y{\neq}\ 0\ |] ==> x \neq 0$
⟨*proof*⟩

lemma *CFinite-diff-CInfinitesimal-capprox*:
$[| \ x \ @c{=} \ y; \ y \in \mathit{CFinite} - \mathit{CInfinitesimal} \ |]$
$\mathit{==>} \ x \in \mathit{CFinite} - \mathit{CInfinitesimal}$
⟨*proof*⟩

lemma *CInfinitesimal-ratio*:
$[| \ y \neq 0; \ \ y \in \mathit{CInfinitesimal}; \ \ x/y \in \mathit{CFinite} \ |] \ \mathit{==>} \ x \in \mathit{CInfinitesimal}$
⟨*proof*⟩

lemma *SComplex-capprox-iff*:
$[|x \in \mathit{SComplex}; \ y \in \mathit{SComplex}|] \ \mathit{==>} \ (x \ @c{=} \ y) = (x = y)$
⟨*proof*⟩

lemma *number-of-capprox-iff* [*simp*]:
$(\mathit{number\text{-}of} \ v \ @c{=} \ \mathit{number\text{-}of} \ w) = (\mathit{number\text{-}of} \ v = (\mathit{number\text{-}of} \ w :: \mathit{hcomplex}))$
⟨*proof*⟩

lemma *number-of-CInfinitesimal-iff* [*simp*]:
$(\mathit{number\text{-}of} \ w \in \mathit{CInfinitesimal}) = (\mathit{number\text{-}of} \ w = (0{::}\mathit{hcomplex}))$
⟨*proof*⟩

lemma *hcomplex-of-complex-approx-iff* [*simp*]:
$(\mathit{hcomplex\text{-}of\text{-}complex} \ k \ @c{=} \ \mathit{hcomplex\text{-}of\text{-}complex} \ m) = (k = m)$
⟨*proof*⟩

lemma *hcomplex-of-complex-capprox-number-of-iff* [*simp*]:
$(\mathit{hcomplex\text{-}of\text{-}complex} \ k \ @c{=} \ \mathit{number\text{-}of} \ w) = (k = \mathit{number\text{-}of} \ w)$
⟨*proof*⟩

lemma *capprox-unique-complex*:
$[| \ r \in \mathit{SComplex}; \ s \in \mathit{SComplex}; \ r \ @c{=} \ x; \ s \ @c{=} \ x|] \ \mathit{==>} \ r = s$
⟨*proof*⟩

lemma *hcomplex-capproxD1*:
$\mathit{star\text{-}n} \ X \ @c{=} \ \mathit{star\text{-}n} \ Y$
$\mathit{==>} \ \mathit{star\text{-}n} \ (\%n. \ \mathit{Re}(X \ n)) \ @{=} \ \mathit{star\text{-}n} \ (\%n. \ \mathit{Re}(Y \ n))$
⟨*proof*⟩

lemma *hcomplex-capproxD2*:
$\mathit{star\text{-}n} \ X \ @c{=} \ \mathit{star\text{-}n} \ Y$
$\mathit{==>} \ \mathit{star\text{-}n} \ (\%n. \ \mathit{Im}(X \ n)) \ @{=} \ \mathit{star\text{-}n} \ (\%n. \ \mathit{Im}(Y \ n))$
⟨*proof*⟩

lemma *hcomplex-capproxI*:
$[| \ \mathit{star\text{-}n} \ (\%n. \ \mathit{Re}(X \ n)) \ @{=} \ \mathit{star\text{-}n} \ (\%n. \ \mathit{Re}(Y \ n));$
$\mathit{star\text{-}n} \ (\%n. \ \mathit{Im}(X \ n)) \ @{=} \ \mathit{star\text{-}n} \ (\%n. \ \mathit{Im}(Y \ n))$
$|] \ \mathit{==>} \ \mathit{star\text{-}n} \ X \ @c{=} \ \mathit{star\text{-}n} \ Y$
⟨*proof*⟩

lemma *capprox-approx-iff*:
  (*star-n* X @*c*= *star-n* Y) =
   (*star-n* (%*n*. *Re*(X n)) @= *star-n* (%*n*. *Re*(Y n)) &
    star-n (%*n*. *Im*(X n)) @= *star-n* (%*n*. *Im*(Y n)))
⟨*proof*⟩

lemma *hcomplex-of-hypreal-capprox-iff* [*simp*]:
  (*hcomplex-of-hypreal* x @*c*= *hcomplex-of-hypreal* z) = (x @= z)
⟨*proof*⟩

lemma *CFinite-HFinite-Re*:
  star-n X ∈ *CFinite*
  ==> *star-n* (%*n*. *Re*(X n)) ∈ *HFinite*
⟨*proof*⟩

lemma *CFinite-HFinite-Im*:
  star-n X ∈ *CFinite*
  ==> *star-n* (%*n*. *Im*(X n)) ∈ *HFinite*
⟨*proof*⟩

lemma *HFinite-Re-Im-CFinite*:
  [| *star-n* (%*n*. *Re*(X n)) ∈ *HFinite*;
    star-n (%*n*. *Im*(X n)) ∈ *HFinite*
  |] ==> *star-n* X ∈ *CFinite*
⟨*proof*⟩

lemma *CFinite-HFinite-iff*:
  (*star-n* X ∈ *CFinite*) =
  (*star-n* (%*n*. *Re*(X n)) ∈ *HFinite* &
   star-n (%*n*. *Im*(X n)) ∈ *HFinite*)
⟨*proof*⟩

lemma *SComplex-Re-SReal*:
  star-n X ∈ *SComplex*
  ==> *star-n* (%*n*. *Re*(X n)) ∈ *Reals*
⟨*proof*⟩

lemma *SComplex-Im-SReal*:
  star-n X ∈ *SComplex*
  ==> *star-n* (%*n*. *Im*(X n)) ∈ *Reals*
⟨*proof*⟩

lemma *Reals-Re-Im-SComplex*:
  [| *star-n* (%*n*. *Re*(X n)) ∈ *Reals*;
    star-n (%*n*. *Im*(X n)) ∈ *Reals*
  |] ==> *star-n* X ∈ *SComplex*
⟨*proof*⟩

lemma *SComplex-SReal-iff*:
  (*star-n* $X \in$ *SComplex*) =
   (*star-n* (%*n*. *Re*(X *n*)) $\in$ *Reals* &
    star-n (%*n*. *Im*(X *n*)) $\in$ *Reals*)
⟨*proof*⟩

lemma *CInfinitesimal-Infinitesimal-iff*:
  (*star-n* $X \in$ *CInfinitesimal*) =
   (*star-n* (%*n*. *Re*(X *n*)) $\in$ *Infinitesimal* &
    star-n (%*n*. *Im*(X *n*)) $\in$ *Infinitesimal*)
⟨*proof*⟩

lemma *eq-Abs-star-EX*:
  ($\exists$ *t*. *P t*) = ($\exists$ *X*. *P* (*star-n X*))
⟨*proof*⟩

lemma *eq-Abs-star-Bex*:
  ($\exists$ *t* $\in$ *A*. *P t*) = ($\exists$ *X*. *star-n X* $\in$ *A* & *P* (*star-n X*))
⟨*proof*⟩

lemma *stc-part-Ex*: *x*:*CFinite* ==> $\exists$ *t* $\in$ *SComplex*. *x* @*c*= *t*
⟨*proof*⟩

lemma *stc-part-Ex1*: *x*:*CFinite* ==> *EX*! *t*. *t* $\in$ *SComplex* & *x* @*c*= *t*
⟨*proof*⟩

lemma *CFinite-Int-CInfinite-empty*: *CFinite Int CInfinite* = {}
⟨*proof*⟩

lemma *CFinite-not-CInfinite*: *x* $\in$ *CFinite* ==> *x* $\notin$ *CInfinite*
⟨*proof*⟩

Not sure this is a good idea!

declare *CFinite-Int-CInfinite-empty* [*simp*]

lemma *not-CFinite-CInfinite*: *x*$\notin$ *CFinite* ==> *x* $\in$ *CInfinite*
⟨*proof*⟩

lemma *CInfinite-CFinite-disj*: *x* $\in$ *CInfinite* | *x* $\in$ *CFinite*
⟨*proof*⟩

lemma *CInfinite-CFinite-iff*: (*x* $\in$ *CInfinite*) = (*x* $\notin$ *CFinite*)
⟨*proof*⟩

lemma *CFinite-CInfinite-iff*: (*x* $\in$ *CFinite*) = (*x* $\notin$ *CInfinite*)
⟨*proof*⟩

lemma *CInfinite-diff-CFinite-CInfinitesimal-disj*:

x ∉ *CInfinitesimal* ==> *x* ∈ *CInfinite* | *x* ∈ *CFinite* − *CInfinitesimal*
⟨*proof*⟩

lemma *CFinite-inverse*:
[| *x* ∈ *CFinite*; *x* ∉ *CInfinitesimal* |] ==> *inverse x* ∈ *CFinite*
⟨*proof*⟩

lemma *CFinite-inverse2*: *x* ∈ *CFinite* − *CInfinitesimal* ==> *inverse x* ∈ *CFinite*
⟨*proof*⟩

lemma *CInfinitesimal-inverse-CFinite*:
x ∉ *CInfinitesimal* ==> *inverse*(*x*) ∈ *CFinite*
⟨*proof*⟩

lemma *CFinite-not-CInfinitesimal-inverse*:
x ∈ *CFinite* − *CInfinitesimal* ==> *inverse x* ∈ *CFinite* − *CInfinitesimal*
⟨*proof*⟩

lemma *capprox-inverse*:
[| *x* @c= *y*; *y* ∈ *CFinite* − *CInfinitesimal* |] ==> *inverse x* @c= *inverse y*
⟨*proof*⟩

lemmas *hcomplex-of-complex-capprox-inverse* =
hcomplex-of-complex-CFinite-diff-CInfinitesimal [*THEN* [*2*] *capprox-inverse*]

lemma *inverse-add-CInfinitesimal-capprox*:
[| *x* ∈ *CFinite* − *CInfinitesimal*;
h ∈ *CInfinitesimal* |] ==> *inverse*(*x* + *h*) @c= *inverse x*
⟨*proof*⟩

lemma *inverse-add-CInfinitesimal-capprox2*:
[| *x* ∈ *CFinite* − *CInfinitesimal*;
h ∈ *CInfinitesimal* |] ==> *inverse*(*h* + *x*) @c= *inverse x*
⟨*proof*⟩

lemma *inverse-add-CInfinitesimal-approx-CInfinitesimal*:
[| *x* ∈ *CFinite* − *CInfinitesimal*;
h ∈ *CInfinitesimal* |] ==> *inverse*(*x* + *h*) − *inverse x* @c= *h*
⟨*proof*⟩

lemma *CInfinitesimal-square-iff* [*iff*]:
(*x***x* ∈ *CInfinitesimal*) = (*x* ∈ *CInfinitesimal*)
⟨*proof*⟩

lemma *capprox-CFinite-mult-cancel*:
[| *a* ∈ *CFinite*−*CInfinitesimal*; *a***w* @c= *a***z* |] ==> *w* @c= *z*
⟨*proof*⟩

lemma *capprox-CFinite-mult-cancel-iff1*:
$a \in CFinite - CInfinitesimal ==> (a * w \ @c= a * z) = (w \ @c= z)$
⟨*proof*⟩

37.6 Theorems About Monads

lemma *capprox-cmonad-iff*: $(x \ @c= y) = (cmonad(x) = cmonad(y))$
⟨*proof*⟩

lemma *CInfinitesimal-cmonad-eq*:
$e \in CInfinitesimal ==> cmonad\ (x+e) = cmonad\ x$
⟨*proof*⟩

lemma *mem-cmonad-iff*: $(u \in cmonad\ x) = (-u \in cmonad\ (-x))$
⟨*proof*⟩

lemma *CInfinitesimal-cmonad-zero-iff*: $(x{:}CInfinitesimal) = (x \in cmonad\ 0)$
⟨*proof*⟩

lemma *cmonad-zero-minus-iff*: $(x \in cmonad\ 0) = (-x \in cmonad\ 0)$
⟨*proof*⟩

lemma *cmonad-zero-hcmod-iff*: $(x \in cmonad\ 0) = (hcmod\ x{:}monad\ 0)$
⟨*proof*⟩

lemma *mem-cmonad-self* [*simp*]: $x \in cmonad\ x$
⟨*proof*⟩

37.7 Theorems About Standard Part

lemma *stc-capprox-self*: $x \in CFinite ==> stc\ x \ @c= x$
⟨*proof*⟩

lemma *stc-SComplex*: $x \in CFinite ==> stc\ x \in SComplex$
⟨*proof*⟩

lemma *stc-CFinite*: $x \in CFinite ==> stc\ x \in CFinite$
⟨*proof*⟩

lemma *stc-SComplex-eq* [*simp*]: $x \in SComplex ==> stc\ x = x$
⟨*proof*⟩

lemma *stc-hcomplex-of-complex*:
$stc\ (hcomplex\text{-}of\text{-}complex\ x) = hcomplex\text{-}of\text{-}complex\ x$
⟨*proof*⟩

lemma *stc-eq-capprox*:
$[|\ x \in CFinite;\ y \in CFinite;\ stc\ x = stc\ y\ |] ==> x \ @c= y$
⟨*proof*⟩

lemma *capprox-stc-eq*:
[| $x \in$ *CFinite*; $y \in$ *CFinite*; x @c= y |] ==> *stc* x = *stc* y
⟨*proof*⟩

lemma *stc-eq-capprox-iff*:
[| $x \in$ *CFinite*; $y \in$ *CFinite*|] ==> (x @c= y) = (*stc* x = *stc* y)
⟨*proof*⟩

lemma *stc-CInfinitesimal-add-SComplex*:
[| $x \in$ *SComplex*; $e \in$ *CInfinitesimal* |] ==> *stc*(x + e) = x
⟨*proof*⟩

lemma *stc-CInfinitesimal-add-SComplex2*:
[| $x \in$ *SComplex*; $e \in$ *CInfinitesimal* |] ==> *stc*(e + x) = x
⟨*proof*⟩

lemma *CFinite-stc-CInfinitesimal-add*:
$x \in$ *CFinite* ==> $\exists\, e \in$ *CInfinitesimal*. x = *stc*(x) + e
⟨*proof*⟩

lemma *stc-add*:
[| $x \in$ *CFinite*; $y \in$ *CFinite* |] ==> *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*: $y \in$ *CFinite* ==> *stc*($-y$) = $-$*stc*(y)
⟨*proof*⟩

lemma *stc-diff*:
[| $x \in$ *CFinite*; $y \in$ *CFinite* |] ==> *stc* ($x-y$) = *stc*(x) $-$ *stc*(y)
⟨*proof*⟩

lemma *lemma-stc-mult*:
[| $x \in$ *CFinite*; $y \in$ *CFinite*;
$e \in$ *CInfinitesimal*;
ea: *CInfinitesimal* |]
==> e*y + x*ea + e*ea: *CInfinitesimal*
⟨*proof*⟩

lemma *stc-mult*:
[| $x \in$ *CFinite*; $y \in$ *CFinite* |]

$==> stc\ (x * y) = stc(x) * stc(y)$
⟨*proof*⟩

lemma *stc-CInfinitesimal*: $x \in CInfinitesimal ==> stc\ x = 0$
⟨*proof*⟩

lemma *stc-not-CInfinitesimal*: $stc(x) \neq 0 ==> x \notin CInfinitesimal$
⟨*proof*⟩

lemma *stc-inverse*:
$[|\ x \in CFinite;\ stc\ x \neq 0\ |]$
$==> stc(inverse\ x) = inverse\ (stc\ x)$
⟨*proof*⟩

lemma *stc-divide* [*simp*]:
$[|\ x \in CFinite;\ y \in CFinite;\ stc\ y \neq 0\ |]$
$==> stc(x/y) = (stc\ x)\ /\ (stc\ y)$
⟨*proof*⟩

lemma *stc-idempotent* [*simp*]: $x \in CFinite ==> stc(stc(x)) = stc(x)$
⟨*proof*⟩

lemma *CFinite-HFinite-hcomplex-of-hypreal*:
$z \in HFinite ==> hcomplex\text{-}of\text{-}hypreal\ z \in CFinite$
⟨*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 *CInfinitesimal-hcnj-iff* [*simp*]:
$(hcnj\ z \in CInfinitesimal) = (z \in CInfinitesimal)$
⟨*proof*⟩

lemma *CInfinite-HInfinite-iff*:
$(star\text{-}n\ X \in CInfinite) =$
$(star\text{-}n\ (\%n.\ Re(X\ n)) \in HInfinite\ |$
$star\text{-}n\ (\%n.\ Im(X\ n)) \in HInfinite)$
⟨*proof*⟩

These theorems should probably be deleted

lemma *hcomplex-split-CInfinitesimal-iff*:
$(hcomplex\text{-}of\text{-}hypreal\ x + iii * hcomplex\text{-}of\text{-}hypreal\ y \in CInfinitesimal) =$

$(x \in \mathit{Infinitesimal}\ \&\ y \in \mathit{Infinitesimal})$
⟨*proof*⟩

lemma *hcomplex-split-CFinite-iff*:
$(\mathit{hcomplex\text{-}of\text{-}hypreal}\ x + \mathit{iii} * \mathit{hcomplex\text{-}of\text{-}hypreal}\ y \in \mathit{CFinite}) =$
$(x \in \mathit{HFinite}\ \&\ y \in \mathit{HFinite})$
⟨*proof*⟩

lemma *hcomplex-split-SComplex-iff*:
$(\mathit{hcomplex\text{-}of\text{-}hypreal}\ x + \mathit{iii} * \mathit{hcomplex\text{-}of\text{-}hypreal}\ y \in \mathit{SComplex}) =$
$(x \in \mathit{Reals}\ \&\ y \in \mathit{Reals})$
⟨*proof*⟩

lemma *hcomplex-split-CInfinite-iff*:
$(\mathit{hcomplex\text{-}of\text{-}hypreal}\ x + \mathit{iii} * \mathit{hcomplex\text{-}of\text{-}hypreal}\ y \in \mathit{CInfinite}) =$
$(x \in \mathit{HInfinite} \mid y \in \mathit{HInfinite})$
⟨*proof*⟩

lemma *hcomplex-split-capprox-iff*:
$(\mathit{hcomplex\text{-}of\text{-}hypreal}\ x + \mathit{iii} * \mathit{hcomplex\text{-}of\text{-}hypreal}\ y\ @c{=}$
$\mathit{hcomplex\text{-}of\text{-}hypreal}\ x' + \mathit{iii} * \mathit{hcomplex\text{-}of\text{-}hypreal}\ y') =$
$(x\ @{=}\ x'\ \&\ y\ @{=}\ y')$
⟨*proof*⟩

lemma *complex-seq-to-hcomplex-CInfinitesimal*:
$\forall n.\ \mathit{cmod}\ (X\ n - x) < \mathit{inverse}\ (\mathit{real}\ (\mathit{Suc}\ n)) ==>$
$\mathit{star\text{-}n}\ X - \mathit{hcomplex\text{-}of\text{-}complex}\ x \in \mathit{CInfinitesimal}$
⟨*proof*⟩

lemma *CInfinitesimal-hcomplex-of-hypreal-epsilon* [*simp*]:
$\mathit{hcomplex\text{-}of\text{-}hypreal}\ \mathit{epsilon} \in \mathit{CInfinitesimal}$
⟨*proof*⟩

lemma *hcomplex-of-complex-approx-zero-iff* [*simp*]:
$(\mathit{hcomplex\text{-}of\text{-}complex}\ z\ @c{=}\ \mathit{0}) = (z = \mathit{0})$
⟨*proof*⟩

lemma *hcomplex-of-complex-approx-zero-iff2* [*simp*]:
$(\mathit{0}\ @c{=}\ \mathit{hcomplex\text{-}of\text{-}complex}\ z) = (z = \mathit{0})$
⟨*proof*⟩

⟨*ML*⟩

end

38 CStar: Star-transforms in NSA, Extending Sets of Complex Numbers and Complex Functions

theory *CStar*
imports *NSCA*
begin

38.1 Properties of the *-Transform Applied to Sets of Reals

lemma *STARC-SComplex-subset*: $SComplex \subseteq *s* \ (UNIV :: complex\ set)$
⟨*proof*⟩

lemma *STARC-hcomplex-of-complex-Int*:
$*s*\ X\ Int\ SComplex = hcomplex\text{-}of\text{-}complex\ `\ X$
⟨*proof*⟩

lemma *lemma-not-hcomplexA*:
$x \notin hcomplex\text{-}of\text{-}complex\ `\ A ==> \forall y \in A.\ x \neq hcomplex\text{-}of\text{-}complex\ y$
⟨*proof*⟩

38.2 Theorems about Nonstandard Extensions of Functions

lemma *cstarfun-if-eq*:
$w \neq hcomplex\text{-}of\text{-}complex\ x$
$==> (\ *f*\ (\lambda z.\ if\ z = x\ then\ a\ else\ g\ z))\ w = (\ *f*\ g)\ w$
⟨*proof*⟩

lemma *starfun-capprox*:
$(\ *f*\ f)\ (hcomplex\text{-}of\text{-}complex\ a)\ @c= hcomplex\text{-}of\text{-}complex\ (f\ a)$
⟨*proof*⟩

lemma *starfunC-hcpow*: $(\ *f*\ (\%z.\ z\ \hat{}\ n))\ Z = Z\ hcpow\ hypnat\text{-}of\text{-}nat\ n$
⟨*proof*⟩

lemma *starfun-mult-CFinite-capprox*:
$[|\ (\ *f*\ f)\ y\ @c= l;\ (\ *f*\ g)\ y\ @c= m;\ l\text{: } CFinite;\ m\text{: } CFinite\ |]$
$==>\ (\ *f*\ (\%x.\ f\ x * g\ x))\ y\ @c= l * m$
⟨*proof*⟩

lemma *starfun-add-capprox*:
$[|\ (\ *f*\ f)\ y\ @c= l;\ (\ *f*\ g)\ y\ @c= m\ |]$
$==>\ (\ *f*\ (\%x.\ f\ x + g\ x))\ y\ @c= l + m$
⟨*proof*⟩

lemma *starfunCR-cmod*: $*f*\ cmod = hcmod$
⟨*proof*⟩

38.3 Internal Functions - Some Redundancy With *f* Now

lemma *starfun-n-diff*:
$(*fn* f)\ z - (*fn* g)\ z = (*fn* (\%i\ x.\ f\ i\ x - g\ i\ x))\ z$
⟨*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*:
$((*f* f)\ x\ @c{=}\ z) = (((*f* (\%x.\ Re(f\ x)))\ x\ @{=}\ hRe\ (z))\ \&$
$((*f* (\%x.\ Im(f\ x)))\ x\ @{=}\ hIm\ (z)))$
⟨*proof*⟩

lemma *starfunC-Idfun-capprox*:
$x\ @c{=}\ hcomplex\text{-}of\text{-}complex\ a ==> (*f* (\%x.\ x))\ x\ @c{=}\ hcomplex\text{-}of\text{-}complex$
a
⟨*proof*⟩

⟨*ML*⟩

end

39 CSeries: Finite Summation and Infinite Series for Complex Numbers

theory *CSeries*
imports *CStar*
begin

consts *sumc* :: [*nat*,*nat*,(*nat*=>*complex*)] => *complex*
primrec
sumc-0: $sumc\ m\ 0\ f = 0$
sumc-Suc: $sumc\ m\ (Suc\ n)\ f = (if\ n < m\ then\ 0\ else\ sumc\ m\ n\ f + f(n))$

lemma *sumc-Suc-zero* [*simp*]: $sumc\ (Suc\ n)\ n\ f = 0$
⟨*proof*⟩

lemma *sumc-eq-bounds* [*simp*]: $sumc\ m\ m\ f = 0$
⟨*proof*⟩

lemma *sumc-Suc-eq* [*simp*]: $sumc\ m\ (Suc\ m)\ f = f(m)$

⟨*proof*⟩

lemma *sumc-add-lbound-zero* [*simp*]: *sumc* (*m*+*k*) *k f* = *0*
⟨*proof*⟩

lemma *sumc-add*: *sumc m n f* + *sumc m n g* = *sumc m n* (%*n*. *f n* + *g n*)
⟨*proof*⟩

lemma *sumc-mult*: *r* ∗ *sumc m n f* = *sumc m n* (%*n*. *r* ∗ *f n*)
⟨*proof*⟩

lemma *sumc-split-add* [*rule-format*]:
n < *p* −−> *sumc 0 n f* + *sumc n p f* = *sumc 0 p f*
⟨*proof*⟩

lemma *sumc-split-add-minus*:
n < *p* ==> *sumc 0 p f* + − *sumc 0 n f* = *sumc n p f*
⟨*proof*⟩

lemma *sumc-cmod*: *cmod*(*sumc m n f*) ≤ (∑ *i*=*m*..<*n*. *cmod*(*f i*))
⟨*proof*⟩

lemma *sumc-fun-eq* [*rule-format* (*no-asm*)]:
(∀ *r*. *m* ≤ *r* & *r* < *n* −−> *f r* = *g r*) −−> *sumc m n f* = *sumc m n g*
⟨*proof*⟩

lemma *sumc-const* [*simp*]: *sumc 0 n* (%*i*. *r*) = *complex-of-real* (*real n*) ∗ *r*
⟨*proof*⟩

lemma *sumc-add-mult-const*:
sumc 0 n f + −(*complex-of-real*(*real n*) ∗ *r*) = *sumc 0 n* (%*i*. *f i* + −*r*)
⟨*proof*⟩

lemma *sumc-diff-mult-const*:
sumc 0 n f − (*complex-of-real*(*real n*)∗*r*) = *sumc 0 n* (%*i*. *f i* − *r*)
⟨*proof*⟩

lemma *sumc-less-bounds-zero* [*rule-format*]: *n* < *m* −−> *sumc m n f* = *0*
⟨*proof*⟩

lemma *sumc-minus*: *sumc m n* (%*i*. − *f i*) = − *sumc m n f*
⟨*proof*⟩

lemma *sumc-shift-bounds*: *sumc* (*m*+*k*) (*n*+*k*) *f* = *sumc m n* (%*i*. *f*(*i* + *k*))
⟨*proof*⟩

lemma *sumc-minus-one-complexpow-zero* [*simp*]:
sumc 0 (*2*∗*n*) (%*i*. (−*1*) ˆ *Suc i*) = *0*
⟨*proof*⟩

lemma *sumc-interval-const* [*rule-format* (*no-asm*)]:
$(\forall n.\ m \leq Suc\ n \dashrightarrow f\ n = r)\ \&\ m \leq na$
$\dashrightarrow sumc\ m\ na\ f = (complex\text{-}of\text{-}real(real\ (na - m)) * r)$
⟨*proof*⟩

lemma *sumc-interval-const2* [*rule-format* (*no-asm*)]:
$(\forall n.\ m \leq n \dashrightarrow f\ n = r)\ \&\ m \leq na$
$\dashrightarrow sumc\ m\ na\ f = (complex\text{-}of\text{-}real(real\ (na - m)) * r)$
⟨*proof*⟩

lemma *sumr-cmod-ge-zero* [*iff*]: $0 \leq (\sum n{=}m..{<}n{::}nat.\ cmod\ (f\ n))$
⟨*proof*⟩

lemma *rabs-sumc-cmod-cancel* [*simp*]:
$abs\ (\sum n{=}m..{<}n{::}nat.\ cmod\ (f\ n)) = (\sum n{=}m..{<}n.\ cmod\ (f\ n))$
⟨*proof*⟩

lemma *sumc-one-lb-complexpow-zero* [*simp*]: $sumc\ 1\ n\ (\%n.\ f(n) * 0\ \hat{}\ n) = 0$
⟨*proof*⟩

lemma *sumc-diff*: $sumc\ m\ n\ f - sumc\ m\ n\ g = sumc\ m\ n\ (\%n.\ f\ n - g\ n)$
⟨*proof*⟩

lemma *sumc-subst* [*rule-format* (*no-asm*)]:
$(\forall p.\ (m \leq p\ \&\ p < m + n \dashrightarrow (f\ p = g\ p))) \dashrightarrow sumc\ m\ n\ f = sumc\ m\ n\ g$
⟨*proof*⟩

lemma *sumc-group* [*simp*]:
$sumc\ 0\ n\ (\%m.\ sumc\ (m * k)\ (m{*}k + k)\ f) = sumc\ 0\ (n * k)\ f$
⟨*proof*⟩

⟨*ML*⟩

end

40 CLim: Limits, Continuity and Differentiation for Complex Functions

theory *CLim*
imports *CSeries*
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*⟩

constdefs

```
CLIM :: [complex=>complex,complex,complex] => bool
                          (((-)/ -- (-)/ --C> (-)) [60, 0, 60] 60)
f -- a --C> L ==
  ∀r. 0 < r -->
       (∃s. 0 < s & (∀x. (x ≠ a & (cmod(x - a) < s)
                 --> cmod(f x - L) < r)))

NSCLIM :: [complex=>complex,complex,complex] => bool
                         (((-)/ -- (-)/ --NSC> (-)) [60, 0, 60] 60)
f -- a --NSC> L == (∀x. (x ≠ hcomplex-of-complex a &
                          x @c= hcomplex-of-complex a
                            --> ( *f* f) x @c= hcomplex-of-complex L))


CRLIM :: [complex=>real,complex,real] => bool
                          (((-)/ -- (-)/ --CR> (-)) [60, 0, 60] 60)
f -- a --CR> L ==
  ∀r. 0 < r -->
       (∃s. 0 < s & (∀x. (x ≠ a & (cmod(x - a) < s)
                 --> abs(f x - L) < r)))

NSCRLIM :: [complex=>real,complex,real] => bool
                         (((-)/ -- (-)/ --NSCR> (-)) [60, 0, 60] 60)
f -- a --NSCR> L == (∀x. (x ≠ hcomplex-of-complex a &
                          x @c= hcomplex-of-complex a
                            --> ( *f* f) x @= hypreal-of-real L))


isContc :: [complex=>complex,complex] => bool
```

isContc f a == (*f* −− *a* −−*C*> (*f a*))

isNSContc :: [*complex*=>*complex*,*complex*] => *bool*
isNSContc f a == (∀ *y*. *y* @*c*= *hcomplex-of-complex a* −−>
(∗*f*∗ *f*) *y* @*c*= *hcomplex-of-complex* (*f a*))

isContCR :: [*complex*=>*real*,*complex*] => *bool*
isContCR f a == (*f* −− *a* −−*CR*> (*f a*))

isNSContCR :: [*complex*=>*real*,*complex*] => *bool*
isNSContCR f a == (∀ *y*. *y* @*c*= *hcomplex-of-complex a* −−>
(∗*f*∗ *f*) *y* @= *hypreal-of-real* (*f a*))

cderiv:: [*complex*=>*complex*,*complex*,*complex*] => *bool*
((*CDERIV* (-)/ (-)/ :> (-)) [*60*, *0*, *60*] *60*)
CDERIV f x :> *D* == ((%*h*. (*f*(*x* + *h*) − *f*(*x*))/*h*) −− *0* −−*C*> *D*)

nscderiv :: [*complex*=>*complex*,*complex*,*complex*] => *bool*
((*NSCDERIV* (-)/ (-)/ :> (-)) [*60*, *0*, *60*] *60*)
NSCDERIV f x :> *D* == (∀ *h* ∈ *CInfinitesimal* − {*0*}.
((∗*f*∗ *f*)(*hcomplex-of-complex x* + *h*)
− *hcomplex-of-complex* (*f x*))/*h* @*c*= *hcomplex-of-complex*
D)

cdifferentiable :: [*complex*=>*complex*,*complex*] => *bool*
(**infixl** *cdifferentiable 60*)
f cdifferentiable x == (∃ *D*. *CDERIV f x* :> *D*)

NSCdifferentiable :: [*complex*=>*complex*,*complex*] => *bool*
(**infixl** *NSCdifferentiable 60*)
f NSCdifferentiable x == (∃ *D*. *NSCDERIV f x* :> *D*)

isUContc :: (*complex*=>*complex*) => *bool*
isUContc f == (∀ *r*. *0* < *r* −−>
(∃ *s*. *0* < *s* & (∀ *x y*. *cmod*(*x* − *y*) < *s*
−−> *cmod*(*f x* − *f y*) < *r*)))

isNSUContc :: (*complex*=>*complex*) => *bool*
isNSUContc f == (∀ *x y*. *x* @*c*= *y* −−> (∗*f*∗ *f*) *x* @*c*= (∗*f*∗ *f*) *y*)

40.1 Limit of Complex to Complex Function

lemma *NSCLIM-NSCRLIM-Re*: *f* −− *a* −−*NSC*> *L* ==> (%*x*. *Re*(*f x*)) −− *a* −−*NSCR*> *Re*(*L*)
⟨*proof*⟩

lemma *NSCLIM-NSCRLIM-Im*: $f\ {-}{-}\ a\ {-}{-}NSC{>}\ L\ {=}{=}{>}\ (\%x.\ Im(f\,x))\ {-}{-}\ a\ {-}{-}NSCR{>}\ Im(L)$
⟨*proof*⟩

lemma *CLIM-NSCLIM*:
$f\ {-}{-}\ x\ {-}{-}C{>}\ L\ {=}{=}{>}\ f\ {-}{-}\ x\ {-}{-}NSC{>}\ L$
⟨*proof*⟩

lemma *eq-Abs-star-ALL*: $(\forall\, t.\ P\ t) = (\forall\, X.\ P\ (star\text{-}n\ X))$
⟨*proof*⟩

lemma *lemma-CLIM*:
$\forall\, s.\ 0 < s\ {-}{-}{>}\ (\exists\, xa.\ \ xa \neq x\ \&$
$cmod\ (xa - x) < s\ \ \&\ r \leq cmod\ (f\,xa - L))$
${=}{=}{>}\ \forall\,(n{::}nat).\ \exists\, xa.\ \ xa \neq x\ \&$
$cmod(xa - x) < inverse(real(Suc\ n))\ \&\ r \leq cmod(f\,xa - L)$
⟨*proof*⟩

lemma *lemma-skolemize-CLIM2*:
$\forall\, s.\ 0 < s\ {-}{-}{>}\ (\exists\, xa.\ \ xa \neq x\ \&$
$cmod\ (xa - x) < s\ \ \&\ r \leq cmod\ (f\,xa - L))$
${=}{=}{>}\ \exists\, X.\ \forall\,(n{::}nat).\ X\ n \neq x\ \&$
$cmod(X\ n - x) < inverse(real(Suc\ n))\ \&\ r \leq cmod(f\ (X\ n) - L)$
⟨*proof*⟩

lemma *lemma-csimp*:
$\forall\, n.\ X\ n \neq x\ \&$
$cmod\ (X\ n - x) < inverse\ (real(Suc\ n))\ \&$
$r \leq cmod\ (f\ (X\ n) - L)\ {=}{=}{>}$
$\forall\, n.\ cmod\ (X\ n - x) < inverse\ (real(Suc\ n))$
⟨*proof*⟩

lemma *NSCLIM-CLIM*:
$f\ {-}{-}\ x\ {-}{-}NSC{>}\ L\ {=}{=}{>}\ f\ {-}{-}\ x\ {-}{-}C{>}\ L$
⟨*proof*⟩

First key result

theorem *CLIM-NSCLIM-iff*: $(f\ {-}{-}\ x\ {-}{-}C{>}\ L) = (f\ {-}{-}\ x\ {-}{-}NSC{>}\ L)$
⟨*proof*⟩

40.2 Limit of Complex to Real Function

lemma *CRLIM-NSCRLIM*: $f\ {-}{-}\ x\ {-}{-}CR{>}\ L\ {=}{=}{>}\ f\ {-}{-}\ x\ {-}{-}NSCR{>}\ L$
⟨*proof*⟩

lemma *lemma-CRLIM*:

$\forall s.\ 0 < s \dashrightarrow (\exists xa.\ \ xa \neq x\ \&$
$\quad cmod\ (xa - x) < s\ \ \&\ r \leq abs\ (f\, xa - L))$
$\Longrightarrow \forall (n{::}nat).\ \exists xa.\ \ xa \neq x\ \&$
$\quad\quad cmod(xa - x) < inverse(real(Suc\ n))\ \&\ r \leq abs\ (f\, xa - L)$
$\langle proof \rangle$

lemma *lemma-skolemize-CRLIM2*:
$\forall s.\ 0 < s \dashrightarrow (\exists xa.\ \ xa \neq x\ \&$
$\quad cmod\ (xa - x) < s\ \ \&\ r \leq abs\ (f\, xa - L))$
$\Longrightarrow \exists X.\ \forall (n{::}nat).\ X\ n \neq x\ \&$
$\quad\quad cmod(X\ n - x) < inverse(real(Suc\ n))\ \&\ r \leq abs\ (f\ (X\ n) - L)$
$\langle proof \rangle$

lemma *lemma-crsimp*:
$\forall n.\ X\ n \neq x\ \&$
$\quad cmod\ (X\ n - x) < inverse\ (real(Suc\ n))\ \&$
$\quad r \leq abs\ (f\ (X\ n) - L) \Longrightarrow$
$\forall n.\ cmod\ (X\ n - x) < inverse\ (real(Suc\ n))$
$\langle proof \rangle$

lemma *NSCRLIM-CRLIM*: $f\ {-}{-}\ x\ {-}{-}NSCR{>}\ L \Longrightarrow f\ {-}{-}\ x\ {-}{-}CR{>}\ L$
$\langle proof \rangle$

Second key result

theorem *CRLIM-NSCRLIM-iff*: $(f\ {-}{-}\ x\ {-}{-}CR{>}\ L) = (f\ {-}{-}\ x\ {-}{-}NSCR{>}\ L)$
$\langle proof \rangle$

lemma *CLIM-CRLIM-Re*: $f\ {-}{-}\ a\ {-}{-}C{>}\ L \Longrightarrow (\%x.\ Re(f\, x))\ {-}{-}\ a\ {-}{-}CR{>}\ Re(L)$
$\langle proof \rangle$

lemma *CLIM-CRLIM-Im*: $f\ {-}{-}\ a\ {-}{-}C{>}\ L \Longrightarrow (\%x.\ Im(f\, x))\ {-}{-}\ a\ {-}{-}CR{>}\ Im(L)$
$\langle proof \rangle$

lemma *CLIM-cnj*: $f\ {-}{-}\ a\ {-}{-}C{>}\ L \Longrightarrow (\%x.\ cnj\ (f\, x))\ {-}{-}\ a\ {-}{-}C{>}\ cnj\ L$
$\langle proof \rangle$

lemma *CLIM-cnj-iff*: $((\%x.\ cnj\ (f\, x))\ {-}{-}\ a\ {-}{-}C{>}\ cnj\ L) = (f\ {-}{-}\ a\ {-}{-}C{>}\ L)$
$\langle proof \rangle$

lemma *NSCLIM-add*:
$[|\ f\ {-}{-}\ x\ {-}{-}NSC{>}\ l;\ g\ {-}{-}\ x\ {-}{-}NSC{>}\ m\ |]$
$\Longrightarrow (\%x.\ f(x) + g(x))\ {-}{-}\ x\ {-}{-}NSC{>}\ (l + m)$
$\langle proof \rangle$

lemma *CLIM-add*:
[| f -- x --C> l; g -- x --C> m |]
==> (%x. f(x) + g(x)) -- x --C> (l + m)
⟨*proof*⟩

lemma *NSCLIM-mult*:
[| f -- x --NSC> l; g -- x --NSC> m |]
==> (%x. f(x) * g(x)) -- x --NSC> (l * m)
⟨*proof*⟩

lemma *CLIM-mult*:
[| f -- x --C> l; g -- x --C> m |]
==> (%x. f(x) * g(x)) -- x --C> (l * m)
⟨*proof*⟩

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

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

lemma *NSCLIM-minus*: f -- a --NSC> L ==> (%x. -f(x)) -- a --NSC> -L
⟨*proof*⟩

lemma *CLIM-minus*: f -- a --C> L ==> (%x. -f(x)) -- a --C> -L
⟨*proof*⟩

lemma *NSCLIM-diff*:
[| f -- x --NSC> l; g -- x --NSC> m |]
==> (%x. f(x) - g(x)) -- x --NSC> (l - m)
⟨*proof*⟩

lemma *CLIM-diff*:
[| f -- x --C> l; g -- x --C> m |]
==> (%x. f(x) - g(x)) -- x --C> (l - m)
⟨*proof*⟩

lemma *NSCLIM-inverse*:
$[| f -- a --NSC> L; L \neq 0 |]$
$==> (\%x. inverse(f(x))) -- a --NSC> (inverse L)$
⟨*proof*⟩

lemma *CLIM-inverse*:
$[| f -- a --C> L; L \neq 0 |]$
$==> (\%x. inverse(f(x))) -- a --C> (inverse L)$
⟨*proof*⟩

lemma *NSCLIM-zero*: $f -- a --NSC> l ==> (\%x. f(x) - l) -- a --NSC> 0$
⟨*proof*⟩

lemma *CLIM-zero*: $f -- a --C> l ==> (\%x. f(x) - l) -- a --C> 0$
⟨*proof*⟩

lemma *NSCLIM-zero-cancel*: $(\%x. f(x) - l) -- x --NSC> 0 ==> f -- x --NSC> l$
⟨*proof*⟩

lemma *CLIM-zero-cancel*: $(\%x. f(x) - l) -- x --C> 0 ==> f -- x --C> l$
⟨*proof*⟩

lemma *NSCLIM-not-zero*: $k \neq 0 ==> \sim ((\%x. k) -- x --NSC> 0)$
⟨*proof*⟩

lemmas *NSCLIM-not-zeroE* = *NSCLIM-not-zero* [*THEN notE*, *standard*]

lemma *CLIM-not-zero*: $k \neq 0 ==> \sim ((\%x. k) -- x --C> 0)$
⟨*proof*⟩

lemma *NSCLIM-const-eq*: $(\%x. k) -- x --NSC> L ==> k = L$
⟨*proof*⟩

lemma *CLIM-const-eq*: $(\%x. k) -- x --C> L ==> k = L$
⟨*proof*⟩

lemma *NSCLIM-unique*: $[| f -- x --NSC> L; f -- x --NSC> M |] ==>$

$L = M$
⟨*proof*⟩

lemma *CLIM-unique*: $[| f -- x --C> L; f -- x --C> M |] ==> L = M$
⟨*proof*⟩

lemma *NSCLIM-mult-zero*:
$[| f -- x --NSC> 0; g -- x --NSC> 0 |] ==> (\%x. f(x)*g(x)) -- x --NSC> 0$
⟨*proof*⟩

lemma *CLIM-mult-zero*:
$[| f -- x --C> 0; g -- x --C> 0 |] ==> (\%x. f(x)*g(x)) -- x --C> 0$
⟨*proof*⟩

lemma *NSCLIM-self*: $(\%x. x) -- a --NSC> a$
⟨*proof*⟩

lemma *CLIM-self*: $(\%x. x) -- a --C> a$
⟨*proof*⟩

lemma *NSCLIM-NSCRLIM-iff*:
$(f -- x --NSC> L) = ((\%y. cmod(f\, y - L)) -- x --NSCR> 0)$
⟨*proof*⟩

lemma *CLIM-CRLIM-iff*: $(f -- x --C> L) = ((\%y. cmod(f\, y - L)) -- x --CR> 0)$
⟨*proof*⟩

lemma *NSCLIM-NSCRLIM-iff2*:
$(f -- x --NSC> L) = ((\%y. cmod(f\, y - L)) -- x --NSCR> 0)$
⟨*proof*⟩

lemma *NSCLIM-NSCRLIM-Re-Im-iff*:
$(f -- a --NSC> L) = ((\%x. Re(f\, x)) -- a --NSCR> Re(L)\ \&$
$(\%x. Im(f\, x)) -- a --NSCR> Im(L))$
⟨*proof*⟩

lemma *CLIM-CRLIM-Re-Im-iff*:
$(f -- a --C> L) = ((\%x. Re(f\, x)) -- a --CR> Re(L)\ \&$
$(\%x. Im(f\, x)) -- a --CR> Im(L))$

⟨*proof*⟩

40.3 Continuity

lemma *isNSContcD*:
[| *isNSContc f a*; *y @c= hcomplex-of-complex a* |]
==> (**f* f*) *y @c= hcomplex-of-complex* (*f a*)
⟨*proof*⟩

lemma *isNSContc-NSCLIM*: *isNSContc f a ==> f −− a −−NSC> (f a)*
⟨*proof*⟩

lemma *NSCLIM-isNSContc*:
f −− a −−NSC> (f a) ==> isNSContc f a
⟨*proof*⟩

Nonstandard continuity can be defined using NS Limit in similar fashion to standard definition of continuity

lemma *isNSContc-NSCLIM-iff*: (*isNSContc f a*) = (*f −− a −−NSC>* (*f a*))
⟨*proof*⟩

lemma *isNSContc-CLIM-iff*: (*isNSContc f a*) = (*f −− a −−C>* (*f a*))
⟨*proof*⟩

lemma *isNSContc-isContc-iff*: (*isNSContc f a*) = (*isContc f a*)
⟨*proof*⟩

lemma *isContc-isNSContc*: *isContc f a ==> isNSContc f a*
⟨*proof*⟩

lemma *isNSContc-isContc*: *isNSContc f a ==> isContc f a*
⟨*proof*⟩

Alternative definition of continuity

lemma *NSCLIM-h-iff*: (*f −− a −−NSC> L*) = ((*%h. f*(*a + h*)) *−− 0 −−NSC> L*)
⟨*proof*⟩

lemma *NSCLIM-isContc-iff*:
(*f −− a −−NSC> f a*) = ((*%h. f*(*a + h*)) *−− 0 −−NSC> f a*)
⟨*proof*⟩

lemma *CLIM-isContc-iff*: (*f −− a −−C> f a*) = ((*%h. f*(*a + h*)) *−− 0 −−C> f*(*a*))
⟨*proof*⟩

lemma *isContc-iff*: (*isContc f x*) = ((*%h. f*(*x + h*)) *−− 0 −−C> f*(*x*))
⟨*proof*⟩

lemma *isContc-add*:
[| *isContc f a*; *isContc g a* |] ==> *isContc* (%*x*. *f*(*x*) + *g*(*x*)) *a*
⟨*proof*⟩

lemma *isContc-mult*:
[| *isContc f a*; *isContc g a* |] ==> *isContc* (%*x*. *f*(*x*) ∗ *g*(*x*)) *a*
⟨*proof*⟩

lemma *isContc-o*: [| *isContc f a*; *isContc g* (*f a*) |] ==> *isContc* (*g o f*) *a*
⟨*proof*⟩

lemma *isContc-o2*:
[| *isContc f a*; *isContc g* (*f a*) |] ==> *isContc* (%*x*. *g* (*f x*)) *a*
⟨*proof*⟩

lemma *isNSContc-minus*: *isNSContc f a* ==> *isNSContc* (%*x*. − *f x*) *a*
⟨*proof*⟩

lemma *isContc-minus*: *isContc f a* ==> *isContc* (%*x*. − *f x*) *a*
⟨*proof*⟩

lemma *isContc-inverse*:
[| *isContc f x*; *f x* ≠ *0* |] ==> *isContc* (%*x*. *inverse* (*f x*)) *x*
⟨*proof*⟩

lemma *isNSContc-inverse*:
[| *isNSContc f x*; *f x* ≠ *0* |] ==> *isNSContc* (%*x*. *inverse* (*f x*)) *x*
⟨*proof*⟩

lemma *isContc-diff*:
[| *isContc f a*; *isContc g a* |] ==> *isContc* (%*x*. *f*(*x*) − *g*(*x*)) *a*
⟨*proof*⟩

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

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

40.4 Functions from Complex to Reals

lemma *isNSContCRD*:
[| *isNSContCR f a*; *y* @*c*= *hcomplex-of-complex a* |]
==> (∗*f*∗ *f*) *y* @= *hypreal-of-real* (*f a*)
⟨*proof*⟩

lemma *isNSContCR-NSCRLIM*: *isNSContCR f a* ==> *f* −− *a* −−*NSCR*> (*f*

a)
⟨*proof*⟩

lemma *NSCRLIM-isNSContCR*: *f* −− *a* −−*NSCR*> (*f a*) ==> *isNSContCR f a*
⟨*proof*⟩

lemma *isNSContCR-NSCRLIM-iff*: (*isNSContCR f a*) = (*f* −− *a* −−*NSCR*> (*f a*))
⟨*proof*⟩

lemma *isNSContCR-CRLIM-iff*: (*isNSContCR f a*) = (*f* −− *a* −−*CR*> (*f a*))
⟨*proof*⟩

lemma *isNSContCR-isContCR-iff*: (*isNSContCR f a*) = (*isContCR f a*)
⟨*proof*⟩

lemma *isContCR-isNSContCR*: *isContCR f a* ==> *isNSContCR f a*
⟨*proof*⟩

lemma *isNSContCR-isContCR*: *isNSContCR f a* ==> *isContCR f a*
⟨*proof*⟩

lemma *isNSContCR-cmod* [*simp*]: *isNSContCR cmod* (*a*)
⟨*proof*⟩

lemma *isContCR-cmod* [*simp*]: *isContCR cmod* (*a*)
⟨*proof*⟩

lemma *isContc-isContCR-Re*: *isContc f a* ==> *isContCR* (%*x*. *Re* (*f x*)) *a*
⟨*proof*⟩

lemma *isContc-isContCR-Im*: *isContc f a* ==> *isContCR* (%*x*. *Im* (*f x*)) *a*
⟨*proof*⟩

40.5 Derivatives

lemma *CDERIV-iff*: (*CDERIV f x* :> *D*) = ((%*h*. (*f*(*x* + *h*) − *f*(*x*))/*h*) −− *0* −−*C*> *D*)
⟨*proof*⟩

lemma *CDERIV-NSC-iff*:
(*CDERIV f x* :> *D*) = ((%*h*. (*f*(*x* + *h*) − *f*(*x*))/*h*) −− *0* −−*NSC*> *D*)
⟨*proof*⟩

lemma *CDERIVD*: *CDERIV f x* :> *D* ==> (%*h*. (*f*(*x* + *h*) − *f*(*x*))/*h*) −− *0* −−*C*> *D*
⟨*proof*⟩

lemma *NSC-DERIVD*: *CDERIV f x* :> *D* ==> (%*h*. (*f*(*x* + *h*) − *f*(*x*))/*h*) −− *0* −−*NSC*> *D*
⟨*proof*⟩

Uniqueness

lemma *CDERIV-unique*: [| *CDERIV f x* :> *D*; *CDERIV f x* :> *E* |] ==> *D* = *E*
⟨*proof*⟩

lemma *NSCDeriv-unique*: [| *NSCDERIV f x* :> *D*; *NSCDERIV f x* :> *E* |] ==> *D* = *E*
⟨*proof*⟩

40.6 Differentiability

lemma *CDERIV-CLIM-iff*:
((%*h*. (*f*(*a* + *h*) − *f*(*a*))/*h*) −− *0* −−*C*> *D*) =
((%*x*. (*f*(*x*) − *f*(*a*)) / (*x* − *a*)) −− *a* −−*C*> *D*)
⟨*proof*⟩

lemma *CDERIV-iff2*:
(*CDERIV f x* :> *D*) = ((%*z*. (*f*(*z*) − *f*(*x*)) / (*z* − *x*)) −− *x* −−*C*> *D*)
⟨*proof*⟩

40.7 Equivalence of NS and Standard Differentiation

lemma *NSCDERIV-NSCLIM-iff*:
(*NSCDERIV f x* :> *D*) = ((%*h*. (*f*(*x* + *h*) − *f*(*x*))/*h*) −− *0* −−*NSC*> *D*)
⟨*proof*⟩

lemma *NSCDERIV-NSCLIM-iff2*:
(*NSCDERIV f x* :> *D*) = ((%*z*. (*f*(*z*) − *f*(*x*)) / (*z* − *x*)) −− *x* −−*NSC*> *D*)
⟨*proof*⟩

lemma *NSCDERIV-iff2*:
(*NSCDERIV f x* :> *D*) =
(∀ *xa*. *xa* ≠ *hcomplex-of-complex x* & *xa* @*c*= *hcomplex-of-complex x* −−>
(**f** (%*z*. (*f z* − *f x*) / (*z* − *x*))) *xa* @*c*= *hcomplex-of-complex D*)
⟨*proof*⟩

lemma *NSCDERIV-CDERIV-iff*: (*NSCDERIV f x* :> *D*) = (*CDERIV f x* :> *D*)
⟨*proof*⟩

lemma *NSCDERIV-isNSContc*: *NSCDERIV f x* :> *D* ==> *isNSContc f x*
⟨*proof*⟩

lemma *CDERIV-isContc*: *CDERIV f x* :> *D* ==> *isContc f x*
⟨*proof*⟩

Differentiation rules for combinations of functions follow by clear, straightforard algebraic manipulations

lemma *NSCDERIV-const* [*simp*]: (*NSCDERIV* (%*x*. *k*) *x* :> *0*)
⟨*proof*⟩

lemma *CDERIV-const* [*simp*]: (*CDERIV* (%*x*. *k*) *x* :> *0*)
⟨*proof*⟩

lemma *NSCDERIV-add*:
  [| *NSCDERIV f x* :> *Da*; *NSCDERIV g x* :> *Db* |]
  ==> *NSCDERIV* (%*x*. *f x* + *g x*) *x* :> *Da* + *Db*
⟨*proof*⟩

lemma *CDERIV-add*:
  [| *CDERIV f x* :> *Da*; *CDERIV g x* :> *Db* |]
  ==> *CDERIV* (%*x*. *f x* + *g x*) *x* :> *Da* + *Db*
⟨*proof*⟩

40.8 Lemmas for Multiplication

lemma *lemma-nscderiv1*: ((*a*::*hcomplex*)∗*b*) − (*c*∗*d*) = (*b*∗(*a* − *c*)) + (*c*∗(*b* − *d*))
⟨*proof*⟩

lemma *lemma-nscderiv2*:
  [| (*x* + *y*) / *z* = *hcomplex-of-complex D* + *yb*; *z* ≠ *0*;
    z : *CInfinitesimal*; *yb* : *CInfinitesimal* |]
  ==> *x* + *y* @c= *0*
⟨*proof*⟩

lemma *NSCDERIV-mult*:
  [| *NSCDERIV f x* :> *Da*; *NSCDERIV g x* :> *Db* |]
  ==> *NSCDERIV* (%*x*. *f x* ∗ *g x*) *x* :> (*Da* ∗ *g*(*x*)) + (*Db* ∗ *f*(*x*))
⟨*proof*⟩

lemma *CDERIV-mult*:
  [| *CDERIV f x* :> *Da*; *CDERIV g x* :> *Db* |]
  ==> *CDERIV* (%*x*. *f x* ∗ *g x*) *x* :> (*Da* ∗ *g*(*x*)) + (*Db* ∗ *f*(*x*))
⟨*proof*⟩

lemma *NSCDERIV-cmult*: *NSCDERIV f x* :> *D* ==> *NSCDERIV* (%*x*. *c* ∗ *f x*) *x* :> *c*∗*D*
⟨*proof*⟩

lemma *CDERIV-cmult*: *CDERIV f x* :> *D* ==> *CDERIV* (%*x*. *c* ∗ *f x*) *x* :> *c*∗*D*
⟨*proof*⟩

lemma *NSCDERIV-minus*: *NSCDERIV f x* :> *D* ==> *NSCDERIV* (%*x*. −(*f*

$x)) \ x :> -D$
⟨*proof*⟩

lemma *CDERIV-minus*: *CDERIV f x :> D ==> CDERIV (%x. −(f x)) x :> −D*
⟨*proof*⟩

lemma *NSCDERIV-add-minus*:
[| *NSCDERIV f x :> Da*; *NSCDERIV g x :> Db* |]
==> *NSCDERIV (%x. f x + −g x) x :> Da + −Db*
⟨*proof*⟩

lemma *CDERIV-add-minus*:
[| *CDERIV f x :> Da*; *CDERIV g x :> Db* |]
==> *CDERIV (%x. f x + −g x) x :> Da + −Db*
⟨*proof*⟩

lemma *NSCDERIV-diff*:
[| *NSCDERIV f x :> Da*; *NSCDERIV g x :> Db* |]
==> *NSCDERIV (%x. f x − g x) x :> Da − Db*
⟨*proof*⟩

lemma *CDERIV-diff*:
[| *CDERIV f x :> Da*; *CDERIV g x :> Db* |]
==> *CDERIV (%x. f x − g x) x :> Da − Db*
⟨*proof*⟩

40.9 Chain Rule

lemma *NSCDERIV-zero*:
[| *NSCDERIV g x :> D*;
(*∗f∗ g*) (*hcomplex-of-complex*(*x*) + *xa*) = *hcomplex-of-complex*(*g x*);
xa : *CInfinitesimal*; *xa* ≠ *0*
|] ==> *D* = *0*
⟨*proof*⟩

lemma *NSCDERIV-capprox*:
[| *NSCDERIV f x :> D*; *h*: *CInfinitesimal*; *h* ≠ *0* |]
==> (*∗f∗ f*) (*hcomplex-of-complex*(*x*) + *h*) − *hcomplex-of-complex*(*f x*) @c= *0*
⟨*proof*⟩

lemma *NSCDERIVD1*:
[| *NSCDERIV f* $(g\ x)$:> *Da*;
(∗*f*∗ *g*) (*hcomplex-of-complex*(x) + *xa*) ≠ *hcomplex-of-complex* ($g\ x$);
(∗*f*∗ *g*) (*hcomplex-of-complex*(x) + *xa*) @*c*= *hcomplex-of-complex* ($g\ x$)|]
==> ((∗*f*∗ *f*) ((∗*f*∗ *g*) (*hcomplex-of-complex*(x) + *xa*))
− *hcomplex-of-complex* (f ($g\ x$))) /
((∗*f*∗ *g*) (*hcomplex-of-complex*(x) + *xa*) − *hcomplex-of-complex* ($g\ x$))
@*c*= *hcomplex-of-complex* (*Da*)
⟨*proof*⟩

lemma *NSCDERIVD2*:
[| *NSCDERIV g x* :> *Db*; *xa*: *CInfinitesimal*; *xa* ≠ *0* |]
==> ((∗*f*∗ *g*) (*hcomplex-of-complex x* + *xa*) − *hcomplex-of-complex*($g\ x$)) / *xa*
@*c*= *hcomplex-of-complex* (*Db*)
⟨*proof*⟩

lemma *lemma-complex-chain*: (*z*::*hcomplex*) ≠ *0* ==> *x*∗*y* = (*x*∗*inverse*(*z*))∗(*z*∗*y*)
⟨*proof*⟩

Chain rule

theorem *NSCDERIV-chain*:
[| *NSCDERIV f* ($g\ x$) :> *Da*; *NSCDERIV g x* :> *Db* |]
==> *NSCDERIV* (*f o g*) *x* :> *Da* ∗ *Db*
⟨*proof*⟩

standard version

lemma *CDERIV-chain*:
[| *CDERIV f* ($g\ x$) :> *Da*; *CDERIV g x* :> *Db* |]
==> *CDERIV* (*f o g*) *x* :> *Da* ∗ *Db*
⟨*proof*⟩

lemma *CDERIV-chain2*:
[| *CDERIV f* ($g\ x$) :> *Da*; *CDERIV g x* :> *Db* |]
==> *CDERIV* (%*x*. *f* ($g\ x$)) *x* :> *Da* ∗ *Db*
⟨*proof*⟩

40.10 Differentiation of Natural Number Powers

lemma *NSCDERIV-Id* [*simp*]: *NSCDERIV* (%*x*. *x*) *x* :> *1*

⟨*proof*⟩

lemma *CDERIV-Id* [*simp*]: *CDERIV* (%*x*. *x*) *x* :> *1*
⟨*proof*⟩

lemmas *isContc-Id* = *CDERIV-Id* [*THEN CDERIV-isContc*, *standard*]

derivative of linear multiplication

lemma *CDERIV-cmult-Id* [*simp*]: *CDERIV* (*op* * *c*) *x* :> *c*
⟨*proof*⟩

lemma *NSCDERIV-cmult-Id* [*simp*]: *NSCDERIV* (*op* * *c*) *x* :> *c*
⟨*proof*⟩

lemma *CDERIV-pow* [*simp*]:
  CDERIV (%*x*. *x* ^ *n*) *x* :> (*complex-of-real* (*real n*)) * (*x* ^ (*n* − *Suc 0*))
⟨*proof*⟩

Nonstandard version

lemma *NSCDERIV-pow*:
  NSCDERIV (%*x*. *x* ^ *n*) *x* :> *complex-of-real* (*real n*) * (*x* ^ (*n* − *1*))
⟨*proof*⟩

lemma *lemma-CDERIV-subst*:
  [|*CDERIV f x* :> *D*; *D* = *E*|] ==> *CDERIV f x* :> *E*
⟨*proof*⟩

lemma *CInfinitesimal-add-not-zero*:
  [| *h*: *CInfinitesimal*; *x* ≠ *0* |] ==> *hcomplex-of-complex x* + *h* ≠ *0*
⟨*proof*⟩

Can't relax the premise $x \neq (0::'a)$: it isn't continuous at zero

lemma *NSCDERIV-inverse*:
  x ≠ *0* ==> *NSCDERIV* (%*x*. *inverse*(*x*)) *x* :> (− (*inverse x* ^ *2*))
⟨*proof*⟩

lemma *CDERIV-inverse*:
  x ≠ *0* ==> *CDERIV* (%*x*. *inverse*(*x*)) *x* :> (−(*inverse x* ^ *2*))
⟨*proof*⟩

40.11 Derivative of Reciprocals (Function *inverse*)

lemma *CDERIV-inverse-fun*:
  [| *CDERIV f x* :> *d*; *f*(*x*) ≠ *0* |]
   ==> *CDERIV* (%*x*. *inverse*(*f x*)) *x* :> (− (*d* * *inverse*(*f*(*x*) ^ *2*)))
⟨*proof*⟩

lemma *NSCDERIV-inverse-fun*:

[| *NSCDERIV f x* :> *d*; $f(x) \neq 0$ |]
==> *NSCDERIV* (%*x*. *inverse*(*f x*)) *x* :> (− (*d* ∗ *inverse*(*f*(*x*) ^ *2*)))
⟨*proof*⟩

40.12 Derivative of Quotient

lemma *CDERIV-quotient*:
[| *CDERIV f x* :> *d*; *CDERIV g x* :> *e*; $g(x) \neq 0$ |]
==> *CDERIV* (%*y*. *f*(*y*) / (*g y*)) *x* :> (*d*∗*g*(*x*) − (*e*∗*f*(*x*))) / (*g*(*x*) ^ *2*)
⟨*proof*⟩

lemma *NSCDERIV-quotient*:
[| *NSCDERIV f x* :> *d*; *NSCDERIV g x* :> *e*; $g(x) \neq 0$ |]
==> *NSCDERIV* (%*y*. *f*(*y*) / (*g y*)) *x* :> (*d*∗*g*(*x*) − (*e*∗*f*(*x*))) / (*g*(*x*) ^ *2*)
⟨*proof*⟩

40.13 Caratheodory Formulation of Derivative at a Point: Standard Proof

lemma *CLIM-equal*:
[| ∀ *x*. *x* ≠ *a* −−> (*f x* = *g x*) |] ==> (*f* −− *a* −−*C*> *l*) = (*g* −− *a* −−*C*> *l*)
⟨*proof*⟩

lemma *CLIM-trans*:
[| (%*x*. *f*(*x*) + −*g*(*x*)) −− *a* −−*C*> *0*; *g* −− *a* −−*C*> *l* |] ==> *f* −− *a* −−*C*> *l*
⟨*proof*⟩

lemma *CARAT-CDERIV*:
(*CDERIV f x* :> *l*) =
(∃ *g*. (∀ *z*. *f z* − *f x* = *g z* ∗ (*z* − *x*)) & *isContc g x* & *g x* = *l*)
⟨*proof*⟩

lemma *CARAT-NSCDERIV*:
NSCDERIV f x :> *l* ==>
∃ *g*. (∀ *z*. *f z* − *f x* = *g z* ∗ (*z* − *x*)) & *isNSContc g x* & *g x* = *l*
⟨*proof*⟩

lemma *CARAT-CDERIVD*:
(∀ *z*. *f z* − *f x* = *g z* ∗ (*z* − *x*)) & *isNSContc g x* & *g x* = *l*
==> *NSCDERIV f x* :> *l*
⟨*proof*⟩

⟨*ML*⟩

end

41 Complex-Main: Comprehensive Complex Theory

theory *Complex-Main*
imports *CLim*
begin

end