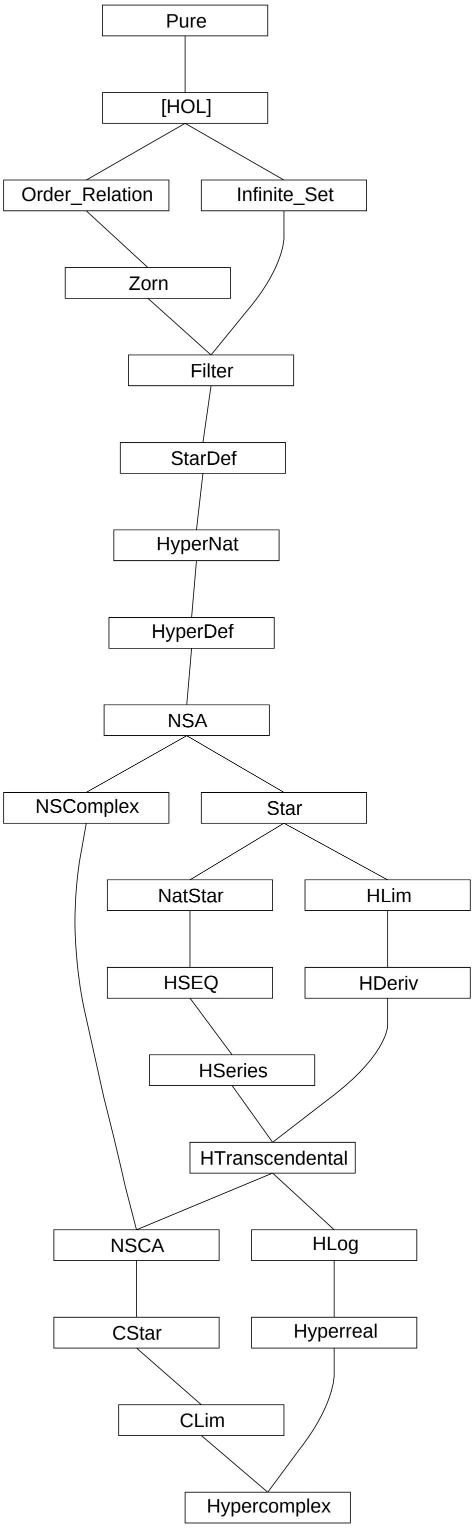
Isabelle/HOL-NSA — Non-Standard Analysis

April 19, 2009

Contents


Pure
[HOL]
Order_Relation
Infinite_Set
Zorn
Filter
StarDef
HyperNat
HyperDef
NSA
NSComplex
Star
NatStar
HLim
HSEQ
HDeriv
HSeries
HTranscendental
NSCA
HLog
CStar
Hyperreal
CLim
Hypercomplex

1 Order-Relation: Orders as Relations

theory *Order-Relation*
imports *Main*
begin

1.1 Orders on a set

definition $preorder\text{-}on\ A\ r \equiv refl\text{-}on\ A\ r \wedge trans\ r$

definition $partial\text{-}order\text{-}on\ A\ r \equiv preorder\text{-}on\ A\ r \wedge antisym\ r$

definition $linear\text{-}order\text{-}on\ A\ r \equiv partial\text{-}order\text{-}on\ A\ r \wedge total\text{-}on\ A\ r$

definition $strict\text{-}linear\text{-}order\text{-}on\ A\ r \equiv trans\ r \wedge irrefl\ r \wedge total\text{-}on\ A\ r$

definition $well\text{-}order\text{-}on\ A\ r \equiv linear\text{-}order\text{-}on\ A\ r \wedge wf(r - Id)$

lemmas *order-on-defs* =
preorder-on-def partial-order-on-def linear-order-on-def
strict-linear-order-on-def well-order-on-def

lemma *preorder-on-empty*[*simp*]: *preorder-on* {} {}
⟨*proof*⟩

lemma *partial-order-on-empty*[*simp*]: *partial-order-on* {} {}
⟨*proof*⟩

lemma *lnear-order-on-empty*[*simp*]: *linear-order-on* {} {}
⟨*proof*⟩

lemma *well-order-on-empty*[*simp*]: *well-order-on* {} {}
⟨*proof*⟩

lemma *preorder-on-converse*[*simp*]: $preorder\text{-}on\ A\ (r\hat{\ }{-1}) = preorder\text{-}on\ A\ r$
⟨*proof*⟩

lemma *partial-order-on-converse*[*simp*]:
$partial\text{-}order\text{-}on\ A\ (r\hat{\ }{-1}) = partial\text{-}order\text{-}on\ A\ r$
⟨*proof*⟩

lemma *linear-order-on-converse*[*simp*]:
$linear\text{-}order\text{-}on\ A\ (r\hat{\ }{-1}) = linear\text{-}order\text{-}on\ A\ r$
⟨*proof*⟩

lemma *strict-linear-order-on-diff-Id*:
$linear\text{-}order\text{-}on\ A\ r \Longrightarrow strict\text{-}linear\text{-}order\text{-}on\ A\ (r{-}Id)$

⟨*proof*⟩

1.2 Orders on the field

abbreviation *Refl* $r \equiv$ *refl-on* (*Field* r) r

abbreviation *Preorder* $r \equiv$ *preorder-on* (*Field* r) r

abbreviation *Partial-order* $r \equiv$ *partial-order-on* (*Field* r) r

abbreviation *Total* $r \equiv$ *total-on* (*Field* r) r

abbreviation *Linear-order* $r \equiv$ *linear-order-on* (*Field* r) r

abbreviation *Well-order* $r \equiv$ *well-order-on* (*Field* r) r

lemma *subset-Image-Image-iff*:
$[\![$ *Preorder* r; $A \subseteq$ *Field* r; $B \subseteq$ *Field* $r]\!] \Longrightarrow$
r `` $A \subseteq r$ `` $B \longleftrightarrow (\forall a{\in}A.\exists b{\in}B.\ (b,a){:}r)$
⟨*proof*⟩

lemma *subset-Image1-Image1-iff*:
$[\![$ *Preorder* r; a : *Field* r; b : *Field* $r]\!] \Longrightarrow r$ `` $\{a\} \subseteq r$ `` $\{b\} \longleftrightarrow (b,a){:}r$
⟨*proof*⟩

lemma *Refl-antisym-eq-Image1-Image1-iff*:
$[\![$*Refl* r; *antisym* r; a:*Field* r; b:*Field* $r]\!] \Longrightarrow r$ `` $\{a\} = r$ `` $\{b\} \longleftrightarrow a{=}b$
⟨*proof*⟩

lemma *Partial-order-eq-Image1-Image1-iff*:
$[\![$*Partial-order* r; a:*Field* r; b:*Field* $r]\!] \Longrightarrow r$ `` $\{a\} = r$ `` $\{b\} \longleftrightarrow a{=}b$
⟨*proof*⟩

1.3 Orders on a type

abbreviation *strict-linear-order* $\equiv$ *strict-linear-order-on UNIV*

abbreviation *linear-order* $\equiv$ *linear-order-on UNIV*

abbreviation *well-order* $r \equiv$ *well-order-on UNIV*

end

2 Zorn: Zorn's Lemma

theory *Zorn*
imports *Order-Relation Main*

begin

definition *chain-subset* :: $'a\ set\ set \Rightarrow bool$ ($chain_{\subseteq}$) **where**
$chain_{\subseteq}\ C \equiv \forall A{\in}C.\forall B{\in}C.\ A \subseteq B \vee B \subseteq A$

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

definition
$chain\ \ ::\ \ 'a\ set\ set => 'a\ set\ set\ set$ **where**
$chain\ S\ = \{F.\ F \subseteq S\ \&\ chain_{\subseteq}\ F\}$

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

definition
$maxchain\ ::\ 'a\ set\ set => 'a\ set\ set\ set$ **where**
$maxchain\ S = \{c.\ c \in chain\ S\ \&\ super\ S\ c = \{\}\}$

definition
$succ\ \ ::\ \ ['a\ set\ set,'a\ set\ set] => 'a\ set\ set$ **where**
$succ\ S\ c =$
$(if\ c \notin chain\ S\ |\ c \in maxchain\ S$
$then\ c\ else\ SOME\ c'.\ c' \in super\ S\ c)$

inductive-set
$TFin :: 'a\ set\ set => 'a\ set\ set\ set$
for $S :: 'a\ set\ set$
where
$succI$: $x \in TFin\ S ==> succ\ S\ x \in TFin\ S$
| $Pow\text{-}UnionI$: $Y \in Pow(TFin\ S) ==> Union(Y) \in TFin\ S$

2.1 Mathematical Preamble

lemma *Union-lemma0*:
$(\forall x \in C.\ x \subseteq A\ |\ B \subseteq x) ==> Union(C) \subseteq A\ |\ B \subseteq Union(C)$
⟨*proof*⟩

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

lemma *Abrial-axiom1*: $x \subseteq succ\ S\ x$
⟨*proof*⟩

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

lemma *TFin-induct*:
assumes H: $n \in TFin\ S$
and I: $!!x.\ x \in TFin\ S ==> P\ x ==> P\ (succ\ S\ x)$
$!!Y.\ Y \subseteq TFin\ S ==> Ball\ Y\ P ==> P(Union\ Y)$
shows $P\ n$ ⟨*proof*⟩

lemma *succ-trans*: $x \subseteq y ==> x \subseteq succ\ S\ y$
⟨*proof*⟩

Lemma 1 of section 3.1

lemma *TFin-linear-lemma1*:
$[|\ n \in TFin\ S;\ \ m \in TFin\ S;$
$\forall x \in TFin\ S.\ x \subseteq m\ --> x = m\ |\ succ\ S\ x \subseteq m$
$|] ==> n \subseteq m\ |\ succ\ S\ m \subseteq n$
⟨*proof*⟩

Lemma 2 of section 3.2

lemma *TFin-linear-lemma2*:
$m \in TFin\ S ==> \forall n \in TFin\ S.\ n \subseteq m\ --> n{=}m\ |\ succ\ S\ n \subseteq m$
⟨*proof*⟩

Re-ordering the premises of Lemma 2

lemma *TFin-subsetD*:
$[|\ n \subseteq m;\ \ m \in TFin\ S;\ \ n \in TFin\ S\ |] ==> n{=}m\ |\ succ\ S\ n \subseteq m$
⟨*proof*⟩

Consequences from section 3.3 – Property 3.2, the ordering is total

lemma *TFin-subset-linear*: $[|\ m \in TFin\ S;\ \ n \in TFin\ S|] ==> n \subseteq m\ |\ m \subseteq n$
⟨*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*⟩

2.2 Hausdorff's Theorem: Every Set Contains a Maximal Chain.

NB: We assume the partial ordering is $\subseteq$, the subset relation!

lemma *empty-set-mem-chain*: $(\{\} :: 'a\ set\ set) \in chain\ S$
⟨*proof*⟩

lemma *super-subset-chain*: $super\ S\ c \subseteq chain\ S$
⟨*proof*⟩

lemma *maxchain-subset-chain*: $maxchain\ S \subseteq chain\ S$
⟨*proof*⟩

lemma *mem-super-Ex*: $c \in chain\ S - maxchain\ S ==> EX\ d.\ d \in super\ S\ c$
⟨*proof*⟩

lemma *select-super*:
$c \in chain\ S - maxchain\ S ==> (\epsilon\, c'.\ c'{:}\ super\ S\ c){:}\ super\ S\ c$
⟨*proof*⟩

lemma *select-not-equals*:
$c \in chain\ S - maxchain\ S ==> (\epsilon\, c'.\ c'{:}\ super\ S\ c) \neq c$
⟨*proof*⟩

lemma *succI3*: $c \in chain\ S - maxchain\ S ==> succ\ S\ c = (\epsilon\, c'.\ c'{:}\ super\ S\ c)$
⟨*proof*⟩

lemma *succ-not-equals*: $c \in chain\ S - maxchain\ S ==> succ\ S\ c \neq c$
⟨*proof*⟩

lemma *TFin-chain-lemma4*: $c \in TFin\ S ==> (c :: 'a\ set\ set){:}\ chain\ S$
⟨*proof*⟩

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

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

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

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

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

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

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

2.4 Alternative version of Zorn's Lemma

lemma *Zorn-Lemma2*:
$\forall\, c \in chain\ S.\ \exists\, y \in S.\ \forall\, x \in c.\ x \subseteq y$

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

Various other lemmas

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

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

definition *Chain* :: $('a{*}'a)set \Rightarrow 'a\ set\ set$ **where**
$Chain\ r \equiv \{A.\ \forall\, a{\in}A. \forall\, b{\in}A.\ (a,b) : r \vee (b,a) \in r\}$

lemma *mono-Chain*: $r \subseteq s \Longrightarrow Chain\ r \subseteq Chain\ s$
⟨*proof*⟩

Zorn's lemma for partial orders:

lemma *Zorns-po-lemma*:
assumes *po*: *Partial-order r* **and** *u*: $\forall\, C{\in}Chain\ r.\ \exists\, u{\in}Field\ r.\ \forall\, a{\in}C.\ (a,u){:}r$
shows $\exists\, m{\in}Field\ r.\ \forall\, a{\in}Field\ r.\ (m,a){:}r \longrightarrow a{=}m$
⟨*proof*⟩

definition *init-seg-of* :: $(('a{*}'a)set * ('a{*}'a)set)set$ **where**
$init\text{-}seg\text{-}of == \{(r,s).\ r \subseteq s \wedge (\forall\, a\ b\ c.\ (a,b){:}s \wedge (b,c){:}r \longrightarrow (a,b){:}r)\}$

abbreviation *initialSegmentOf* :: $('a{*}'a)set \Rightarrow ('a{*}'a)set \Rightarrow bool$
(**infix** *initial'-segment'-of 55*) **where**
$r\ initial\text{-}segment\text{-}of\ s == (r,s){:}init\text{-}seg\text{-}of$

lemma *refl-on-init-seg-of*[*simp*]: *r initial-segment-of r*
⟨*proof*⟩

lemma *trans-init-seg-of*:
$r\ initial\text{-}segment\text{-}of\ s \Longrightarrow s\ initial\text{-}segment\text{-}of\ t \Longrightarrow r\ initial\text{-}segment\text{-}of\ t$
⟨*proof*⟩

lemma *antisym-init-seg-of*:
$r\ initial\text{-}segment\text{-}of\ s \Longrightarrow s\ initial\text{-}segment\text{-}of\ r \Longrightarrow r{=}s$
⟨*proof*⟩

lemma *Chain-init-seg-of-Union*:
$R \in Chain\ init\text{-}seg\text{-}of \Longrightarrow r{\in}R \Longrightarrow r\ initial\text{-}segment\text{-}of \bigcup R$
⟨*proof*⟩

lemma *chain-subset-trans-Union*:
$chain_{\subseteq}\ R \Longrightarrow \forall\, r{\in}R.\ trans\ r \Longrightarrow trans(\bigcup R)$

⟨*proof*⟩

lemma *chain-subset-antisym-Union*:
$chain_{\subseteq}\ R \Longrightarrow \forall r{\in}R.\ antisym\ r \Longrightarrow antisym(\bigcup R)$
⟨*proof*⟩

lemma *chain-subset-Total-Union*:
assumes $chain_{\subseteq}\ R\ \forall r{\in}R.\ Total\ r$
shows $Total\ (\bigcup R)$
⟨*proof*⟩

lemma *wf-Union-wf-init-segs*:
assumes $R \in Chain\ init\text{-}seg\text{-}of$ **and** $\forall r{\in}R.\ wf\ r$ **shows** $wf(\bigcup R)$
⟨*proof*⟩

lemma *initial-segment-of-Diff*:
$p\ initial\text{-}segment\text{-}of\ q \Longrightarrow p - s\ initial\text{-}segment\text{-}of\ q - s$
⟨*proof*⟩

lemma *Chain-inits-DiffI*:
$R \in Chain\ init\text{-}seg\text{-}of \Longrightarrow \{r - s\ |r.\ r \in R\} \in Chain\ init\text{-}seg\text{-}of$
⟨*proof*⟩

theorem *well-ordering*: $\exists r{::}('a{*}'a)set.\ Well\text{-}order\ r \wedge Field\ r = UNIV$
⟨*proof*⟩

corollary *well-order-on*: $\exists r{::}('a{*}'a)set.\ well\text{-}order\text{-}on\ A\ r$
⟨*proof*⟩

end

3 Infinite-Set: Infinite Sets and Related Concepts

theory *Infinite-Set*
imports *Main*
begin

3.1 Infinite Sets

Some elementary facts about infinite sets, mostly by Stefan Merz. Beware! Because "infinite" merely abbreviates a negation, these lemmas may not work well with *blast*.

abbreviation
$infinite :: 'a\ set \Rightarrow bool$ **where**
$infinite\ S == \neg\ finite\ S$

Infinite sets are non-empty, and if we remove some elements from an infinite set, the result is still infinite.

lemma *infinite-imp-nonempty*: *infinite* S ==> $S \neq \{\}$
⟨*proof*⟩

lemma *infinite-remove*:
infinite $S \Longrightarrow$ *infinite* $(S - \{a\})$
⟨*proof*⟩

lemma *Diff-infinite-finite*:
assumes T: *finite* T **and** S: *infinite* S
shows *infinite* $(S - T)$
⟨*proof*⟩

lemma *Un-infinite*: *infinite* $S \Longrightarrow$ *infinite* $(S \cup T)$
⟨*proof*⟩

lemma *infinite-super*:
assumes T: $S \subseteq T$ **and** S: *infinite* S
shows *infinite* T
⟨*proof*⟩

As a concrete example, we prove that the set of natural numbers is infinite.

lemma *finite-nat-bounded*:
assumes S: *finite* $(S::nat\ set)$
shows $\exists\, k.\ S \subseteq \{..<k\}$ (**is** $\exists\, k.$ *?bounded* S k)
⟨*proof*⟩

lemma *finite-nat-iff-bounded*:
finite $(S::nat\ set) = (\exists\, k.\ S \subseteq \{..<k\})$ (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *finite-nat-iff-bounded-le*:
finite $(S::nat\ set) = (\exists\, k.\ S \subseteq \{..k\})$ (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *infinite-nat-iff-unbounded*:
infinite $(S::nat\ set) = (\forall\, m.\ \exists\, n.\ m<n \wedge n \in S)$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *infinite-nat-iff-unbounded-le*:
infinite $(S::nat\ set) = (\forall\, m.\ \exists\, n.\ m \leq n \wedge n \in S)$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

For a set of natural numbers to be infinite, it is enough to know that for any number larger than some k, there is some larger number that is an element of the set.

lemma *unbounded-k-infinite*:

assumes k: $\forall m.\ k<m \longrightarrow (\exists n.\ m<n \wedge n \in S)$
shows *infinite* (S::*nat set*)
⟨*proof*⟩

lemma *nat-infinite* [*simp*]: *infinite* (*UNIV* :: *nat set*)
⟨*proof*⟩

lemma *nat-not-finite* [*elim*]: *finite* (*UNIV*::*nat set*) $\Longrightarrow R$
⟨*proof*⟩

Every infinite set contains a countable subset. More precisely we show that a set S is infinite if and only if there exists an injective function from the naturals into S.

lemma *range-inj-infinite*:
inj (f::*nat* $\Rightarrow$ $'a$) $\Longrightarrow$ *infinite* (*range* f)
⟨*proof*⟩

lemma *int-infinite* [*simp*]:
shows *infinite* (*UNIV*::*int set*)
⟨*proof*⟩

The "only if" direction is harder because it requires the construction of a sequence of pairwise different elements of an infinite set S. The idea is to construct a sequence of non-empty and infinite subsets of S obtained by successively removing elements of S.

lemma *linorder-injI*:
assumes *hyp*: !!x y. $x <$ (y::$'a$::*linorder*) ==> $f\ x \neq f\ y$
shows *inj* f
⟨*proof*⟩

lemma *infinite-countable-subset*:
assumes *inf*: *infinite* (S::$'a$ *set*)
shows $\exists f$. *inj* (f::*nat* $\Rightarrow$ $'a$) $\wedge$ *range* $f \subseteq S$
⟨*proof*⟩

lemma *infinite-iff-countable-subset*:
infinite S = ($\exists f$. *inj* (f::*nat* $\Rightarrow$ $'a$) $\wedge$ *range* $f \subseteq S$)
⟨*proof*⟩

For any function with infinite domain and finite range there is some element that is the image of infinitely many domain elements. In particular, any infinite sequence of elements from a finite set contains some element that occurs infinitely often.

lemma *inf-img-fin-dom*:
assumes *img*: *finite* (f‘A) **and** *dom*: *infinite* A
shows $\exists y \in f$‘A. *infinite* (f −‘ $\{y\}$)
⟨*proof*⟩

lemma *inf-img-fin-domE*:
assumes *finite* $(f`A)$ **and** *infinite* A
obtains y **where** $y \in f`A$ **and** *infinite* $(f\ -`\ \{y\})$
⟨*proof*⟩

3.2 Infinitely Many and Almost All

We often need to reason about the existence of infinitely many (resp., all but finitely many) objects satisfying some predicate, so we introduce corresponding binders and their proof rules.

definition
Inf-many :: $('a \Rightarrow bool) \Rightarrow bool$ (**binder** *INFM* *10*) **where**
Inf-many P = *infinite* $\{x.\ P\ x\}$

definition
Alm-all :: $('a \Rightarrow bool) \Rightarrow bool$ (**binder** *MOST* *10*) **where**
Alm-all $P = (\neg\ (INFM\ x.\ \neg\ P\ x))$

notation (*xsymbols*)
Inf-many (**binder** $\exists_\infty$ *10*) **and**
Alm-all (**binder** $\forall_\infty$ *10*)

notation (*HTML* **output**)
Inf-many (**binder** $\exists_\infty$ *10*) **and**
Alm-all (**binder** $\forall_\infty$ *10*)

lemma *INFM-EX*:
$(\exists_\infty x.\ P\ x) \Longrightarrow (\exists x.\ P\ x)$
⟨*proof*⟩

lemma *MOST-iff-finiteNeg*: $(\forall_\infty x.\ P\ x)$ = *finite* $\{x.\ \neg\ P\ x\}$
⟨*proof*⟩

lemma *ALL-MOST*: $\forall x.\ P\ x \Longrightarrow \forall_\infty x.\ P\ x$
⟨*proof*⟩

lemma *INFM-mono*:
assumes *inf*: $\exists_\infty x.\ P\ x$ **and** q: $\bigwedge x.\ P\ x \Longrightarrow Q\ x$
shows $\exists_\infty x.\ Q\ x$
⟨*proof*⟩

lemma *MOST-mono*: $\forall_\infty x.\ P\ x \Longrightarrow (\bigwedge x.\ P\ x \Longrightarrow Q\ x) \Longrightarrow \forall_\infty x.\ Q\ x$
⟨*proof*⟩

lemma *INFM-disj-distrib*:
$(\exists_\infty x.\ P\ x \lor Q\ x) \longleftrightarrow (\exists_\infty x.\ P\ x) \lor (\exists_\infty x.\ Q\ x)$
⟨*proof*⟩

lemma *MOST-conj-distrib*:

$(\forall_\infty x.\ P\ x \wedge Q\ x) \longleftrightarrow (\forall_\infty x.\ P\ x) \wedge (\forall_\infty x.\ Q\ x)$
⟨*proof*⟩

lemma *MOST-rev-mp*:
assumes $\forall_\infty x.\ P\ x$ **and** $\forall_\infty x.\ P\ x \longrightarrow Q\ x$
shows $\forall_\infty x.\ Q\ x$
⟨*proof*⟩

lemma *not-INFM* [*simp*]: $\neg\ (INFM\ x.\ P\ x) \longleftrightarrow (MOST\ x.\ \neg\ P\ x)$
⟨*proof*⟩

lemma *not-MOST* [*simp*]: $\neg\ (MOST\ x.\ P\ x) \longleftrightarrow (INFM\ x.\ \neg\ P\ x)$
⟨*proof*⟩

lemma *INFM-const* [*simp*]: $(INFM\ x{::}'a.\ P) \longleftrightarrow P \wedge \mathit{infinite}\ (UNIV{::}'a\ set)$
⟨*proof*⟩

lemma *MOST-const* [*simp*]: $(MOST\ x{::}'a.\ P) \longleftrightarrow P \vee \mathit{finite}\ (UNIV{::}'a\ set)$
⟨*proof*⟩

lemma *INFM-nat*: $(\exists_\infty n.\ P\ (n{::}nat)) = (\forall m.\ \exists n.\ m<n \wedge P\ n)$
⟨*proof*⟩

lemma *INFM-nat-le*: $(\exists_\infty n.\ P\ (n{::}nat)) = (\forall m.\ \exists n.\ m\leq n \wedge P\ n)$
⟨*proof*⟩

lemma *MOST-nat*: $(\forall_\infty n.\ P\ (n{::}nat)) = (\exists m.\ \forall n.\ m<n \longrightarrow P\ n)$
⟨*proof*⟩

lemma *MOST-nat-le*: $(\forall_\infty n.\ P\ (n{::}nat)) = (\exists m.\ \forall n.\ m\leq n \longrightarrow P\ n)$
⟨*proof*⟩

3.3 Enumeration of an Infinite Set

The set's element type must be wellordered (e.g. the natural numbers).

consts
enumerate :: *'a::wellorder set => (nat => 'a::wellorder)*
primrec
enumerate-0: *enumerate S 0* $=$ *(LEAST n. n* $\in$ *S)*
enumerate-Suc: *enumerate S (Suc n)* $=$ *enumerate (S* $-$ *{LEAST n. n* $\in$ *S}) n*

lemma *enumerate-Suc'*:
enumerate S (Suc n) $=$ *enumerate (S* $-$ *{enumerate S 0}) n*
⟨*proof*⟩

lemma *enumerate-in-set*: *infinite S* $\Longrightarrow$ *enumerate S n : S*
⟨*proof*⟩

declare *enumerate-0* [*simp del*] *enumerate-Suc* [*simp del*]

lemma *enumerate-step*: *infinite* $S \Longrightarrow$ *enumerate* S $n <$ *enumerate* S (*Suc* n)
⟨*proof*⟩

lemma *enumerate-mono*: $m<n \Longrightarrow$ *infinite* $S \Longrightarrow$ *enumerate* S $m <$ *enumerate* S n
⟨*proof*⟩

3.4 Miscellaneous

A few trivial lemmas about sets that contain at most one element. These simplify the reasoning about deterministic automata.

definition
atmost-one :: $'a$ *set* $\Rightarrow$ *bool* **where**
atmost-one $S = (\forall x\ y.\ x \in S \wedge y \in S \longrightarrow x = y)$

lemma *atmost-one-empty*: $S = \{\} \Longrightarrow$ *atmost-one* S
⟨*proof*⟩

lemma *atmost-one-singleton*: $S = \{x\} \Longrightarrow$ *atmost-one* S
⟨*proof*⟩

lemma *atmost-one-unique* [*elim*]: *atmost-one* $S \Longrightarrow x \in S \Longrightarrow y \in S \Longrightarrow y = x$
⟨*proof*⟩

end

4 Filter: Filters and Ultrafilters

theory *Filter*
imports ~~/src/HOL/Library/Zorn ~~/src/HOL/Library/Infinite-Set
begin

4.1 Definitions and basic properties

4.1.1 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*⟩

4.1.2 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*⟩

4.1.3 Free Ultrafilters

locale *freeultrafilter* = *ultrafilter* +
assumes *infinite*: $A \in F \Longrightarrow$ *infinite* A

lemma (**in** *freeultrafilter*) *finite*: *finite* $A \Longrightarrow A \notin F$
⟨*proof*⟩

lemma (**in** *freeultrafilter*) *singleton*: $\{x\} \notin F$
⟨*proof*⟩

lemma (**in** *freeultrafilter*) *insert-iff* [*simp*]: (*insert* x $A \in F) = (A \in F)$
⟨*proof*⟩

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

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

4.2 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*:
$(\{n.\ P\ n \wedge Q\ n\} \in F) = (\{n.\ P\ n\} \in F \wedge \{n.\ Q\ n\} \in F)$
$\langle proof \rangle$

lemma (**in** *ultrafilter*) *Collect-not*:
$(\{n.\ \neg\ P\ n\} \in F) = (\{n.\ P\ n\} \notin F)$
$\langle proof \rangle$

lemma (**in** *ultrafilter*) *Collect-disj*:
$(\{n.\ P\ n \vee Q\ n\} \in F) = (\{n.\ P\ n\} \in F \vee \{n.\ Q\ n\} \in F)$
$\langle proof \rangle$

lemma (**in** *ultrafilter*) *Collect-all*:
$(\{n.\ \forall x.\ P\ n\ x\} \in F) = (\forall X.\ \{n.\ P\ n\ (X\ n)\} \in F)$
$\langle proof \rangle$

4.3 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*: $UNIV \in F \Longrightarrow A \in \{X.\ \exists f{\in}F.\ A \cap f \subseteq X\}$
$\langle proof \rangle$

lemma *extend-lemma2*: $F \subseteq \{X.\ \exists f{\in}F.\ A \cap f \subseteq X\}$
$\langle proof \rangle$

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*)
$\langle proof \rangle$

lemma (**in** *filter*) *max-filter-ultrafilter*:
assumes *max*: $\bigwedge G.\ [\![filter\ G;\ F \subseteq G]\!] \Longrightarrow F = G$
shows *ultrafilter-axioms F*
$\langle proof \rangle$

lemma (**in** *ultrafilter*) *max-filter*:
assumes G: *filter* G **and** *sub*: $F \subseteq G$ **shows** $F = G$
$\langle proof \rangle$

4.4 Ultrafilter Theorem

A locale makes proof of ultrafilter Theorem more modular

locale $UFT =$

fixes *frechet* :: *'a set set*
and *superfrechet* :: *'a set set set*

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

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

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

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

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

A few properties of free filters

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

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

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

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

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

4.4.1 Unions of chains of superfrechets

In this section we prove that superfrechet is closed with respect to unions of non-empty chains. We must show 1) Union of a chain is a filter, 2) Union of a chain contains frechet.

Number 2 is trivial, but 1 requires us to prove all the filter rules.

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

⟨*proof*⟩

lemma (**in** UFT) *Union-chain-empty*:
$c \in chain\ superfrechet \Longrightarrow \{\} \notin \bigcup c$
⟨*proof*⟩

lemma (**in** UFT) *Union-chain-Int*:
$[\![c \in chain\ superfrechet;\ u \in \bigcup c;\ v \in \bigcup c]\!] \Longrightarrow u \cap v \in \bigcup c$
⟨*proof*⟩

lemma (**in** UFT) *Union-chain-subset*:
$[\![c \in chain\ superfrechet;\ u \in \bigcup c;\ u \subseteq v]\!] \Longrightarrow v \in \bigcup c$
⟨*proof*⟩

lemma (**in** UFT) *Union-chain-filter*:
assumes *chain*: $c \in chain\ superfrechet$ **and** *nonempty*: $c \neq \{\}$
shows *filter* $(\bigcup c)$
⟨*proof*⟩

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

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

4.4.2 Existence of free ultrafilter

lemma (**in** UFT) *max-cofinite-filter-Ex*:
$\exists\, U{\in}superfrechet.\ \forall\, G{\in}superfrechet.\ U \subseteq G \longrightarrow U = G$
⟨*proof*⟩

lemma (**in** UFT) *mem-superfrechet-all-infinite*:
$[\![U \in superfrechet;\ A \in U]\!] \Longrightarrow infinite\ A$
⟨*proof*⟩

There exists a free ultrafilter on any infinite set

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

lemmas *freeultrafilter-Ex* $=$ *UFT.freeultrafilter-ex* [*OF UFT.intro*]

hide (**open**) *const filter*

end

5 StarDef: Construction of Star Types Using Ultrafilters

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

5.1 A Free Ultrafilter over the Naturals

definition
$FreeUltrafilterNat :: nat\ set\ set\ \ (\mathcal{U})$ **where**
$\mathcal{U} = (SOME\ U.\ freeultrafilter\ U)$

lemma *freeultrafilter-FreeUltrafilterNat*: $freeultrafilter\ \mathcal{U}$
⟨*proof*⟩

interpretation *FreeUltrafilterNat*: *freeultrafilter FreeUltrafilterNat*
⟨*proof*⟩

This rule takes the place of the old ultra tactic

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

5.2 Definition of *star* type constructor

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

typedef $'a\ star = (UNIV :: (nat \Rightarrow 'a)\ set)\ //\ starrel$
⟨*proof*⟩

definition
$star\text{-}n :: (nat \Rightarrow 'a) \Rightarrow 'a\ star$ **where**
$star\text{-}n\ X = Abs\text{-}star\ (starrel\ ``\ \{X\})$

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

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

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

Proving that *starrel* is an equivalence relation

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

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

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

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

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

5.3 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*]:
$[\![p \equiv \{n.\ P\ n\} \in \mathcal{U};\ x \equiv \{n.\ X\ n\} \in \mathcal{U};\ y \equiv \{n.\ Y\ n\} \in \mathcal{U}]\!]$
$\Longrightarrow (\mathit{if}\ p\ \mathit{then}\ x\ \mathit{else}\ y) \equiv \{n.\ \mathit{if}\ P\ n\ \mathit{then}\ X\ n\ \mathit{else}\ Y\ n\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-eq* [*transfer-intro*]:
$[\![x \equiv \mathit{star\text{-}n}\ X;\ y \equiv \mathit{star\text{-}n}\ Y]\!] \Longrightarrow x = y \equiv \{n.\ X\ n = Y\ n\} \in \mathcal{U}$
⟨*proof*⟩

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

lemma *transfer-fun-eq* [*transfer-intro*]:
$[\![\bigwedge X.\ f\ (\mathit{star\text{-}n}\ X) = g\ (\mathit{star\text{-}n}\ X)$
$\equiv \{n.\ F\ n\ (X\ n) = G\ n\ (X\ n)\} \in \mathcal{U}]\!]$
$\Longrightarrow f = g \equiv \{n.\ F\ n = G\ n\} \in \mathcal{U}$
⟨*proof*⟩

lemma *transfer-star-n* [*transfer-intro*]: $\mathit{star\text{-}n}\ X \equiv \mathit{star\text{-}n}\ (\lambda n.\ X\ n)$
⟨*proof*⟩

lemma *transfer-bool* [*transfer-intro*]: $p \equiv \{n.\ p\} \in \mathcal{U}$
⟨*proof*⟩

5.4 Standard elements

definition
$\mathit{star\text{-}of} :: {'a} \Rightarrow {'a}\ \mathit{star}$ **where**
$\mathit{star\text{-}of}\ x == \mathit{star\text{-}n}\ (\lambda n.\ x)$

definition
$\mathit{Standard} :: {'a}\ \mathit{star}\ \mathit{set}$ **where**
$\mathit{Standard} = \mathit{range}\ \mathit{star\text{-}of}$

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*⟩

lemma *Standard-star-of* [*simp*]: $star\text{-}of\ x \in Standard$
⟨*proof*⟩

5.5 Internal functions

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

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

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

Transfer tactic should remove occurrences of *Ifun*

⟨*ML*⟩

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

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

lemma *Standard-Ifun* [*simp*]:
$[\![f \in Standard;\ x \in Standard]\!] \Longrightarrow f \star x \in Standard$
⟨*proof*⟩

Nonstandard extensions of functions

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

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

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

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

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

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

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

lemma *Standard-starfun* [*simp*]: $x \in Standard \Longrightarrow starfun\ f\ x \in Standard$
⟨*proof*⟩

lemma *Standard-starfun2* [*simp*]:
$[\![x \in Standard;\ y \in Standard]\!] \Longrightarrow starfun2\ f\ x\ y \in Standard$
⟨*proof*⟩

lemma *Standard-starfun-iff*:
assumes *inj*: $\bigwedge x\ y.\ f\ x = f\ y \Longrightarrow x = y$
shows $(starfun\ f\ x \in Standard) = (x \in Standard)$
⟨*proof*⟩

lemma *Standard-starfun2-iff*:
assumes *inj*: $\bigwedge a\ b\ a'\ b'.\ f\ a\ b = f\ a'\ b' \Longrightarrow a = a' \wedge b = b'$
shows $(starfun2\ f\ x\ y \in Standard) = (x \in Standard \wedge y \in Standard)$
⟨*proof*⟩

5.6 Internal predicates

definition *unstar* :: *bool star* ⇒ *bool* **where**
[*code del*]: $unstar\ b \longleftrightarrow b = star\text{-}of\ True$

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*⟩

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

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

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*⟩

5.7 Internal sets

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

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

Transfer tactic should remove occurrences of *Iset*

⟨*ML*⟩

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

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

lemma *transfer-set-eq* [*transfer-intro*]:
$[\![a \equiv Iset\ (star\text{-}n\ A);\ b \equiv Iset\ (star\text{-}n\ B)]\!]$

$\Longrightarrow a = b \equiv \{n.\ A\ n = B\ n\} \in \mathcal{U}$
⟨proof⟩

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

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

lemma *transfer-Iset* [*transfer-intro*]:
$[\![a \equiv \mathit{star\text{-}n}\ A]\!] \Longrightarrow \mathit{Iset}\ a \equiv \mathit{Iset}\ (\mathit{star\text{-}n}\ (\lambda n.\ A\ n))$
⟨proof⟩

Nonstandard extensions of sets.

definition
starset :: $'a\ \mathit{set} \Rightarrow 'a\ \mathit{star}\ \mathit{set}$ (*$*s*$* - [*80*] *80*) **where**
starset $A = \mathit{Iset}\ (\mathit{star\text{-}of}\ A)$

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

lemma *starset-mem*: $(\mathit{star\text{-}of}\ x \in {*s*}\ A) = (x \in A)$
⟨proof⟩

lemma *starset-UNIV*: ${*s*}\ (\mathit{UNIV}{::}'a\ \mathit{set}) = (\mathit{UNIV}{::}'a\ \mathit{star}\ \mathit{set})$
⟨proof⟩

lemma *starset-empty*: ${*s*}\ \{\} = \{\}$
⟨proof⟩

lemma *starset-insert*: ${*s*}\ (\mathit{insert}\ x\ A) = \mathit{insert}\ (\mathit{star\text{-}of}\ x)\ (\ {*s*}\ A)$
⟨proof⟩

lemma *starset-Un*: ${*s*}\ (A \cup B) = {*s*}\ A \cup {*s*}\ B$
⟨proof⟩

lemma *starset-Int*: ${*s*}\ (A \cap B) = {*s*}\ A \cap {*s*}\ B$
⟨proof⟩

lemma *starset-Compl*: ${*s*}\ {-A} = -(\ {*s*}\ A)$
⟨proof⟩

lemma *starset-diff*: ${*s*}\ (A - B) = {*s*}\ A - {*s*}\ B$
⟨proof⟩

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

⟨*proof*⟩

lemma *starset-vimage*: $*s* \ (f -` A) = ( *f* \ f) -` (*s* \ A)$
⟨*proof*⟩

lemma *starset-subset*: $(*s* \ A \subseteq *s* \ B) = (A \subseteq B)$
⟨*proof*⟩

lemma *starset-eq*: $(*s* \ A = *s* \ B) = (A = B)$
⟨*proof*⟩

lemmas *starset-simps* [*simp*] =
starset-mem *starset-UNIV*
starset-empty *starset-insert*
starset-Un *starset-Int*
starset-Compl *starset-diff*
starset-image *starset-vimage*
starset-subset *starset-eq*

5.8 Syntactic classes

instantiation *star* :: (*zero*) *zero*
begin

definition
star-zero-def [*code del*]: $0 \equiv$ *star-of 0*

instance ⟨*proof*⟩

end

instantiation *star* :: (*one*) *one*
begin

definition
star-one-def [*code del*]: $1 \equiv$ *star-of 1*

instance ⟨*proof*⟩

end

instantiation *star* :: (*plus*) *plus*
begin

definition
star-add-def [*code del*]: $(op \ +) \equiv *f2* \ (op \ +)$

instance ⟨*proof*⟩

end

instantiation *star* :: (*times*) *times*
begin

definition
star-mult-def [*code del*]: $(op\ *) \equiv *f2*\ (op\ *)$

instance ⟨*proof*⟩

end

instantiation *star* :: (*uminus*) *uminus*
begin

definition
star-minus-def [*code del*]: $uminus \equiv *f*\ uminus$

instance ⟨*proof*⟩

end

instantiation *star* :: (*minus*) *minus*
begin

definition
star-diff-def [*code del*]: $(op\ -) \equiv *f2*\ (op\ -)$

instance ⟨*proof*⟩

end

instantiation *star* :: (*abs*) *abs*
begin

definition
star-abs-def: $abs \equiv *f*\ abs$

instance ⟨*proof*⟩

end

instantiation *star* :: (*sgn*) *sgn*
begin

definition
star-sgn-def: $sgn \equiv *f*\ sgn$

instance ⟨*proof*⟩

end

instantiation *star* :: (*inverse*) *inverse*
begin

definition
star-divide-def: $(op\ /) \equiv *f2*\ (op\ /)$

definition
star-inverse-def: $inverse \equiv *f*\ inverse$

instance ⟨*proof*⟩

end

instantiation *star* :: (*number*) *number*
begin

definition
star-number-def: $number\text{-}of\ b \equiv star\text{-}of\ (number\text{-}of\ b)$

instance ⟨*proof*⟩

end

instance *star* :: (*Ring-and-Field.dvd*) *Ring-and-Field.dvd* ⟨*proof*⟩

instantiation *star* :: (*Divides.div*) *Divides.div*
begin

definition
star-div-def: $(op\ div) \equiv *f2*\ (op\ div)$

definition
star-mod-def: $(op\ mod) \equiv *f2*\ (op\ mod)$

instance ⟨*proof*⟩

end

instantiation *star* :: (*power*) *power*
begin

definition
star-power-def: $(op\ \hat{}\,) \equiv \lambda x\ n.\ (\ *f*\ (\lambda x.\ x\ \hat{}\ n))\ x$

instance ⟨*proof*⟩

end

instantiation *star* :: (*ord*) *ord*
begin

definition
star-le-def: $(op \le) \equiv *p2* (op \le)$

definition
star-less-def: $(op <) \equiv *p2* (op <)$

instance ⟨*proof*⟩

end

lemmas *star-class-defs* [*transfer-unfold*] =
star-zero-def *star-one-def* *star-number-def*
star-add-def *star-diff-def* *star-minus-def*
star-mult-def *star-divide-def* *star-inverse-def*
star-le-def *star-less-def* *star-abs-def* *star-sgn-def*
star-div-def *star-mod-def* *star-power-def*

Class operations preserve standard elements

lemma *Standard-zero*: $0 \in Standard$
⟨*proof*⟩

lemma *Standard-one*: $1 \in Standard$
⟨*proof*⟩

lemma *Standard-number-of*: $number\text{-}of\ b \in Standard$
⟨*proof*⟩

lemma *Standard-add*: $[\![x \in Standard; y \in Standard]\!] \Longrightarrow x + y \in Standard$
⟨*proof*⟩

lemma *Standard-diff*: $[\![x \in Standard; y \in Standard]\!] \Longrightarrow x - y \in Standard$
⟨*proof*⟩

lemma *Standard-minus*: $x \in Standard \Longrightarrow - x \in Standard$
⟨*proof*⟩

lemma *Standard-mult*: $[\![x \in Standard; y \in Standard]\!] \Longrightarrow x * y \in Standard$
⟨*proof*⟩

lemma *Standard-divide*: $[\![x \in Standard; y \in Standard]\!] \Longrightarrow x / y \in Standard$
⟨*proof*⟩

lemma *Standard-inverse*: $x \in Standard \Longrightarrow inverse\ x \in Standard$
⟨*proof*⟩

lemma *Standard-abs*: $x \in \mathit{Standard} \Longrightarrow \mathit{abs}\ x \in \mathit{Standard}$
⟨*proof*⟩

lemma *Standard-div*: $[\![x \in \mathit{Standard};\ y \in \mathit{Standard}]\!] \Longrightarrow x\ \mathit{div}\ y \in \mathit{Standard}$
⟨*proof*⟩

lemma *Standard-mod*: $[\![x \in \mathit{Standard};\ y \in \mathit{Standard}]\!] \Longrightarrow x\ \mathit{mod}\ y \in \mathit{Standard}$
⟨*proof*⟩

lemma *Standard-power*: $x \in \mathit{Standard} \Longrightarrow x\ \hat{}\ n \in \mathit{Standard}$
⟨*proof*⟩

lemmas *Standard-simps* [*simp*] =
Standard-zero *Standard-one* *Standard-number-of*
Standard-add *Standard-diff* *Standard-minus*
Standard-mult *Standard-divide* *Standard-inverse*
Standard-abs *Standard-div* *Standard-mod*
Standard-power

star-of preserves class operations

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

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

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

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

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

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

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

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

lemma *star-of-power*: $\mathit{star\text{-}of}\ (x\ \hat{}\ n) = \mathit{star\text{-}of}\ x\ \hat{}\ n$
⟨*proof*⟩

lemma *star-of-abs*: $\mathit{star\text{-}of}\ (\mathit{abs}\ x) = \mathit{abs}\ (\mathit{star\text{-}of}\ x)$

⟨*proof*⟩

star-of preserves numerals

lemma *star-of-zero*: *star-of 0 = 0*
⟨*proof*⟩

lemma *star-of-one*: *star-of 1 = 1*
⟨*proof*⟩

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

star-of preserves orderings

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

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

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

As above, for 0

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

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

As above, for 1

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

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

As above, for numerals

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

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

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

5.9 Ordering and lattice classes

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

instantiation *star* :: (*lower-semilattice*) *lower-semilattice*
begin

definition
star-inf-def [*transfer-unfold*]: *inf* ≡ ∗*f2*∗ *inf*

instance
⟨*proof*⟩

end

instantiation *star* :: (*upper-semilattice*) *upper-semilattice*
begin

definition
star-sup-def [*transfer-unfold*]: *sup* ≡ ∗*f2*∗ *sup*

instance
⟨*proof*⟩

end

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

instance *star* :: (*distrib-lattice*) *distrib-lattice*
⟨*proof*⟩

lemma *Standard-inf* [*simp*]:
$[\![x \in \mathit{Standard};\ y \in \mathit{Standard}]\!] \Longrightarrow \mathit{inf}\ x\ y \in \mathit{Standard}$
⟨*proof*⟩

lemma *Standard-sup* [*simp*]:
$[\![x \in \mathit{Standard};\ y \in \mathit{Standard}]\!] \Longrightarrow \mathit{sup}\ x\ y \in \mathit{Standard}$
⟨*proof*⟩

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

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

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

lemma *star-max-def* [*transfer-unfold*]: $\mathit{max} = {*}\mathit{f2}{*}\ \mathit{max}$
⟨*proof*⟩

lemma *star-min-def* [*transfer-unfold*]: $\mathit{min} = {*}\mathit{f2}{*}\ \mathit{min}$
⟨*proof*⟩

lemma *Standard-max* [*simp*]:
$[\![x \in \mathit{Standard};\ y \in \mathit{Standard}]\!] \Longrightarrow \mathit{max}\ x\ y \in \mathit{Standard}$
⟨*proof*⟩

lemma *Standard-min* [*simp*]:
$[\![x \in \mathit{Standard};\ y \in \mathit{Standard}]\!] \Longrightarrow \mathit{min}\ x\ y \in \mathit{Standard}$
⟨*proof*⟩

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

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

5.10 Ordered group classes

instance *star* :: (*semigroup-add*) *semigroup-add*
⟨*proof*⟩

instance *star* :: (*ab-semigroup-add*) *ab-semigroup-add*
⟨*proof*⟩

instance *star* :: (*semigroup-mult*) *semigroup-mult*
⟨*proof*⟩

instance *star* :: (*ab-semigroup-mult*) *ab-semigroup-mult*
⟨*proof*⟩

instance *star* :: (*comm-monoid-add*) *comm-monoid-add*
⟨*proof*⟩

instance *star* :: (*monoid-mult*) *monoid-mult*
⟨*proof*⟩

instance *star* :: (*comm-monoid-mult*) *comm-monoid-mult*
⟨*proof*⟩

instance *star* :: (*cancel-semigroup-add*) *cancel-semigroup-add*
⟨*proof*⟩

instance *star* :: (*cancel-ab-semigroup-add*) *cancel-ab-semigroup-add*
⟨*proof*⟩

instance *star* :: (*cancel-comm-monoid-add*) *cancel-comm-monoid-add* ⟨*proof*⟩

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

instance *star* :: (*pordered-ab-semigroup-add*) *pordered-ab-semigroup-add*
⟨*proof*⟩

instance *star* :: (*pordered-cancel-ab-semigroup-add*) *pordered-cancel-ab-semigroup-add*
⟨*proof*⟩

instance *star* :: (*pordered-ab-semigroup-add-imp-le*) *pordered-ab-semigroup-add-imp-le*
⟨*proof*⟩

instance *star* :: (*pordered-comm-monoid-add*) *pordered-comm-monoid-add* ⟨*proof*⟩
instance *star* :: (*pordered-ab-group-add*) *pordered-ab-group-add* ⟨*proof*⟩

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

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

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

5.11 Ring and field classes

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

instance *star* :: (*semiring-0*) *semiring-0*
⟨*proof*⟩

instance *star* :: (*semiring-0-cancel*) *semiring-0-cancel* ⟨*proof*⟩

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

instance *star* :: (*comm-semiring-0*) *comm-semiring-0* ⟨*proof*⟩
instance *star* :: (*comm-semiring-0-cancel*) *comm-semiring-0-cancel* ⟨*proof*⟩

instance *star* :: (*zero-neq-one*) *zero-neq-one*
⟨*proof*⟩

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

instance *star* :: (*no-zero-divisors*) *no-zero-divisors*
⟨*proof*⟩

instance *star* :: (*semiring-1-cancel*) *semiring-1-cancel* ⟨*proof*⟩
instance *star* :: (*comm-semiring-1-cancel*) *comm-semiring-1-cancel* ⟨*proof*⟩
instance *star* :: (*ring*) *ring* ⟨*proof*⟩
instance *star* :: (*comm-ring*) *comm-ring* ⟨*proof*⟩
instance *star* :: (*ring-1*) *ring-1* ⟨*proof*⟩
instance *star* :: (*comm-ring-1*) *comm-ring-1* ⟨*proof*⟩
instance *star* :: (*ring-no-zero-divisors*) *ring-no-zero-divisors* ⟨*proof*⟩
instance *star* :: (*ring-1-no-zero-divisors*) *ring-1-no-zero-divisors* ⟨*proof*⟩
instance *star* :: (*idom*) *idom* ⟨*proof*⟩

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

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

instance *star* :: (*division-by-zero*) *division-by-zero*
⟨*proof*⟩

instance *star* :: (*pordered-semiring*) *pordered-semiring*
⟨*proof*⟩

instance *star* :: (*pordered-cancel-semiring*) *pordered-cancel-semiring* ⟨*proof*⟩

instance *star* :: (*ordered-semiring-strict*) *ordered-semiring-strict*

⟨*proof*⟩

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

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

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

instance *star* :: (*pordered-ring*) *pordered-ring* ⟨*proof*⟩
instance *star* :: (*pordered-ring-abs*) *pordered-ring-abs*
⟨*proof*⟩
instance *star* :: (*lordered-ring*) *lordered-ring* ⟨*proof*⟩

instance *star* :: (*abs-if*) *abs-if*
⟨*proof*⟩

instance *star* :: (*sgn-if*) *sgn-if*
⟨*proof*⟩

instance *star* :: (*ordered-ring-strict*) *ordered-ring-strict* ⟨*proof*⟩
instance *star* :: (*pordered-comm-ring*) *pordered-comm-ring* ⟨*proof*⟩

instance *star* :: (*ordered-semidom*) *ordered-semidom*
⟨*proof*⟩

instance *star* :: (*ordered-idom*) *ordered-idom* ⟨*proof*⟩
instance *star* :: (*ordered-field*) *ordered-field* ⟨*proof*⟩

5.12 Power classes

Proving the class axiom *power-Suc* for type *'a star* is a little tricky, because it quantifies over values of type *nat*. The transfer principle does not handle quantification over non-star types in general, but we can work around this by fixing an arbitrary *nat* value, and then applying the transfer principle.

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

5.13 Number classes

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

lemma *Standard-of-nat* [*simp*]: *of-nat n* ∈ *Standard*
⟨*proof*⟩

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

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

lemma *Standard-of-int* [*simp*]: *of-int z* ∈ *Standard*
⟨*proof*⟩

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

instance *star* :: (*semiring-char-0*) *semiring-char-0*
⟨*proof*⟩

instance *star* :: (*ring-char-0*) *ring-char-0* ⟨*proof*⟩

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

5.14 Finite class

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

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

end

6 HyperNat: Hypernatural numbers

theory *HyperNat*
imports *StarDef*
begin

types *hypnat* = *nat star*

abbreviation
  hypnat-of-nat :: *nat* => *nat star* **where**
  hypnat-of-nat == *star-of*

definition
  hSuc :: *hypnat* => *hypnat* **where**
  hSuc-def [*transfer-unfold*, *code del*]: *hSuc* = *f* *Suc*

6.1 Properties Transferred from Naturals

lemma *hSuc-not-zero* [*iff*]: $\bigwedge m.\ hSuc\ m \neq 0$
⟨*proof*⟩

lemma *zero-not-hSuc* [*iff*]: $\bigwedge m.\ 0 \neq hSuc\ m$
⟨*proof*⟩

lemma *hSuc-hSuc-eq* [*iff*]: $\bigwedge m\ n.\ (hSuc\ m = hSuc\ n) = (m = n)$
⟨*proof*⟩

lemma *zero-less-hSuc* [*iff*]: $\bigwedge n.\ 0 < hSuc\ n$
⟨*proof*⟩

lemma *hypnat-minus-zero* [*simp*]: $!!z.\ z - z = (0::hypnat)$
⟨*proof*⟩

lemma *hypnat-diff-0-eq-0* [*simp*]: $!!n.\ (0::hypnat) - n = 0$
⟨*proof*⟩

lemma *hypnat-add-is-0* [*iff*]: $!!m\ n.\ (m+n = (0::hypnat)) = (m=0\ \&\ n=0)$
⟨*proof*⟩

lemma *hypnat-diff-diff-left*: $!!i\ j\ k.\ (i::hypnat) - j - k = i - (j+k)$
⟨*proof*⟩

lemma *hypnat-diff-commute*: $!!i\ j\ k.\ (i::hypnat) - j - k = i-k-j$
⟨*proof*⟩

lemma *hypnat-diff-add-inverse* [*simp*]: $!!m\ n.\ ((n::hypnat) + m) - n = m$
⟨*proof*⟩

lemma *hypnat-diff-add-inverse2* [*simp*]: $!!m\ n.\ ((m::hypnat) + n) - n = m$
⟨*proof*⟩

lemma *hypnat-diff-cancel* [*simp*]: $!!k\ m\ n.\ ((k::hypnat) + m) - (k+n) = m - n$
⟨*proof*⟩

lemma *hypnat-diff-cancel2* [*simp*]: $!!k\ m\ n.\ ((m::hypnat) + k) - (n+k) = m - n$
⟨*proof*⟩

lemma *hypnat-diff-add-0* [*simp*]: $!!m\ n.\ (n::hypnat) - (n+m) = (0::hypnat)$
⟨*proof*⟩

lemma *hypnat-diff-mult-distrib*: $!!k\ m\ n.\ ((m::hypnat) - n) * k = (m * k) - (n * k)$
⟨*proof*⟩

lemma *hypnat-diff-mult-distrib2*: $!!k\ m\ n.\ (k::hypnat) * (m - n) = (k * m) - (k * n)$

⟨*proof*⟩

lemma *hypnat-le-zero-cancel* [*iff*]: $!!n.\ (n \le (0::hypnat)) = (n = 0)$
⟨*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*]: $!!n.\ \tilde{}\ n < (0::hypnat)$
⟨*proof*⟩

lemma *hypnat-less-one* [*iff*]:
$!!n.\ (n < (1::hypnat)) = (n{=}0)$
⟨*proof*⟩

lemma *hypnat-add-diff-inverse*: $!!m\ n.\ \tilde{}\ m{<}n \ {=}{=}{>}\ n{+}(m{-}n) = (m::hypnat)$
⟨*proof*⟩

lemma *hypnat-le-add-diff-inverse* [*simp*]: $!!m\ n.\ n \le m \ {=}{=}{>}\ n{+}(m{-}n) = (m::hypnat)$
⟨*proof*⟩

lemma *hypnat-le-add-diff-inverse2* [*simp*]: $!!m\ n.\ n{\le}m \ {=}{=}{>}\ (m{-}n){+}n = (m::hypnat)$
⟨*proof*⟩

declare *hypnat-le-add-diff-inverse2* [*OF order-less-imp-le*]

lemma *hypnat-le0* [*iff*]: $!!n.\ (0::hypnat) \le n$
⟨*proof*⟩

lemma *hypnat-le-add1* [*simp*]: $!!x\ n.\ (x::hypnat) \le x + n$
⟨*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*]: $!!n.\ (n \ne 0) = (0 < (n::hypnat))$
⟨*proof*⟩

lemma *hypnat-gt-zero-iff*: $((0::hypnat) < n) = ((1::hypnat) \le n)$
⟨*proof*⟩

lemma *hypnat-gt-zero-iff2*: $(0 < n) = (\exists\, m.\ n = m + (1::hypnat))$
⟨*proof*⟩

lemma *hypnat-add-self-not-less*: ~ $(x + y < (x::hypnat))$
⟨*proof*⟩

lemma *hypnat-diff-split*:
$P(a - b::hypnat) = ((a<b \text{ --> } P\ 0)\ \&\ (ALL\ d.\ a = b + d \text{ --> } P\ d))$
— elimination of − on *hypnat*
⟨*proof*⟩

6.2 Properties of the set of embedded natural numbers

lemma *of-nat-eq-star-of* [*simp*]: *of-nat* = *star-of*
⟨*proof*⟩

lemma *Nats-eq-Standard*: (*Nats* :: *nat star set*) = *Standard*
⟨*proof*⟩

lemma *hypnat-of-nat-mem-Nats* [*simp*]: *hypnat-of-nat* $n \in$ *Nats*
⟨*proof*⟩

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

lemma *hypnat-of-nat-Suc* [*simp*]:
hypnat-of-nat (*Suc n*) = *hypnat-of-nat* n + (*1*::*hypnat*)
⟨*proof*⟩

lemma *of-nat-eq-add* [*rule-format*]:
$\forall$ *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*⟩

6.3 Infinite Hypernatural Numbers – *HNatInfinite*

definition

HNatInfinite :: *hypnat set* **where**
HNatInfinite = $\{n.\ n \notin$ *Nats*$\}$

lemma *Nats-not-HNatInfinite-iff*: ($x \in$ *Nats*) = ($x \notin$ *HNatInfinite*)
⟨*proof*⟩

lemma *HNatInfinite-not-Nats-iff*: ($x \in$ *HNatInfinite*) = ($x \notin$ *Nats*)
⟨*proof*⟩

lemma *star-of-neq-HNatInfinite*: $N \in$ *HNatInfinite* $\Longrightarrow$ *star-of* $n \neq N$
⟨*proof*⟩

lemma *star-of-Suc-lessI*:
$\bigwedge N.\ [\![star\text{-}of\ n < N;\ star\text{-}of\ (Suc\ n) \neq N]\!] \Longrightarrow star\text{-}of\ (Suc\ n) < N$
⟨*proof*⟩

lemma *star-of-less-HNatInfinite*:
assumes N: $N \in HNatInfinite$
shows $star\text{-}of\ n < N$
⟨*proof*⟩

lemma *star-of-le-HNatInfinite*: $N \in HNatInfinite \Longrightarrow star\text{-}of\ n \leq N$
⟨*proof*⟩

6.3.1 Closure Rules

lemma *Nats-less-HNatInfinite*: $[\![x \in Nats;\ y \in HNatInfinite]\!] \Longrightarrow x < y$
⟨*proof*⟩

lemma *Nats-le-HNatInfinite*: $[\![x \in Nats;\ y \in HNatInfinite]\!] \Longrightarrow x \leq y$
⟨*proof*⟩

lemma *zero-less-HNatInfinite*: $x \in HNatInfinite \Longrightarrow 0 < x$
⟨*proof*⟩

lemma *one-less-HNatInfinite*: $x \in HNatInfinite \Longrightarrow 1 < x$
⟨*proof*⟩

lemma *one-le-HNatInfinite*: $x \in HNatInfinite \Longrightarrow 1 \leq x$
⟨*proof*⟩

lemma *zero-not-mem-HNatInfinite* [*simp*]: $0 \notin HNatInfinite$
⟨*proof*⟩

lemma *Nats-downward-closed*:
$[\![x \in Nats;\ (y::hypnat) \leq x]\!] \Longrightarrow y \in Nats$
⟨*proof*⟩

lemma *HNatInfinite-upward-closed*:
$[\![x \in HNatInfinite;\ x \leq y]\!] \Longrightarrow y \in HNatInfinite$
⟨*proof*⟩

lemma *HNatInfinite-add*: $x \in HNatInfinite \Longrightarrow x + y \in HNatInfinite$
⟨*proof*⟩

lemma *HNatInfinite-add-one*: $x \in HNatInfinite \Longrightarrow x + 1 \in HNatInfinite$
⟨*proof*⟩

lemma *HNatInfinite-diff*:
$[\![x \in HNatInfinite;\ y \in Nats]\!] \Longrightarrow x - y \in HNatInfinite$
⟨*proof*⟩

lemma *HNatInfinite-is-Suc*: $x \in HNatInfinite$ ==> $\exists\, y.\ x = y + (1::hypnat)$
⟨*proof*⟩

6.4 Existence of an infinite hypernatural number

definition

whn :: *hypnat* **where**
hypnat-omega-def: $whn = star\text{-}n\ (\%n::nat.\ n)$

lemma *hypnat-of-nat-neq-whn*: $hypnat\text{-}of\text{-}nat\ n \neq whn$
⟨*proof*⟩

lemma *whn-neq-hypnat-of-nat*: $whn \neq hypnat\text{-}of\text{-}nat\ n$
⟨*proof*⟩

lemma *whn-not-Nats* [*simp*]: $whn \notin Nats$
⟨*proof*⟩

lemma *HNatInfinite-whn* [*simp*]: $whn \in HNatInfinite$
⟨*proof*⟩

lemma *lemma-unbounded-set* [*simp*]: $\{n::nat.\ m < n\} \in FreeUltrafilterNat$
⟨*proof*⟩

lemma *Compl-Collect-le*: $- \{n::nat.\ N \leq n\} = \{n.\ n < N\}$
⟨*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 *Nats-less-whn*: $n \in Nats \Longrightarrow n < whn$
⟨*proof*⟩

lemma *Nats-le-whn*: $n \in Nats \Longrightarrow n \leq whn$
⟨*proof*⟩

lemma *hypnat-of-nat-less-whn* [*simp*]: $hypnat\text{-}of\text{-}nat\ n < whn$
⟨*proof*⟩

lemma *hypnat-of-nat-le-whn* [*simp*]: $hypnat\text{-}of\text{-}nat\ n \leq whn$
⟨*proof*⟩

lemma *hypnat-zero-less-hypnat-omega* [*simp*]: $0 < whn$

⟨*proof*⟩

lemma *hypnat-one-less-hypnat-omega* [*simp*]: $1 < whn$
⟨*proof*⟩

6.4.1 Alternative characterization of the set of infinite hypernaturals

$HNatInfinite = \{N. \forall n \in \mathbb{N}. n < N\}$

lemma *HNatInfinite-FreeUltrafilterNat-lemma*:
assumes $\forall N::nat. \{n. f\ n \neq N\} \in FreeUltrafilterNat$
shows $\{n. N < f\ n\} \in FreeUltrafilterNat$
⟨*proof*⟩

lemma *HNatInfinite-iff*: $HNatInfinite = \{N. \forall n \in Nats. n < N\}$
⟨*proof*⟩

6.4.2 Alternative Characterization of *HNatInfinite* using Free Ultrafilter

lemma *HNatInfinite-FreeUltrafilterNat*:
star-n $X \in HNatInfinite ==> \forall u. \{n. u < X\ n\}$: *FreeUltrafilterNat*
⟨*proof*⟩

lemma *FreeUltrafilterNat-HNatInfinite*:
$\forall u. \{n. u < X\ n\}$: *FreeUltrafilterNat* $==>$ *star-n* $X \in HNatInfinite$
⟨*proof*⟩

lemma *HNatInfinite-FreeUltrafilterNat-iff*:
(*star-n* $X \in HNatInfinite) = (\forall u. \{n. u < X\ n\}$: *FreeUltrafilterNat*)
⟨*proof*⟩

6.5 Embedding of the Hypernaturals into other types

definition
of-hypnat :: *hypnat* ⇒ *'a::semiring-1-cancel star* **where**
of-hypnat-def [*transfer-unfold*, *code del*]: *of-hypnat* = **f* of-nat*

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

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

lemma *of-hypnat-hSuc*: $\bigwedge m.$ *of-hypnat* (*hSuc m*) $= 1 +$ *of-hypnat m*
⟨*proof*⟩

lemma *of-hypnat-add* [*simp*]:
$\bigwedge m\ n.$ *of-hypnat* $(m + n) =$ *of-hypnat m* $+$ *of-hypnat n*

⟨proof⟩

lemma *of-hypnat-mult* [*simp*]:
$\bigwedge m\ n.\ \textit{of-hypnat}\ (m * n) = \textit{of-hypnat}\ m * \textit{of-hypnat}\ n$
⟨proof⟩

lemma *of-hypnat-less-iff* [*simp*]:
$\bigwedge m\ n.\ (\textit{of-hypnat}\ m < (\textit{of-hypnat}\ n::'a::\textit{ordered-semidom star})) = (m < n)$
⟨proof⟩

lemma *of-hypnat-0-less-iff* [*simp*]:
$\bigwedge n.\ (0 < (\textit{of-hypnat}\ n::'a::\textit{ordered-semidom star})) = (0 < n)$
⟨proof⟩

lemma *of-hypnat-less-0-iff* [*simp*]:
$\bigwedge m.\ \neg\ (\textit{of-hypnat}\ m::'a::\textit{ordered-semidom star}) < 0$
⟨proof⟩

lemma *of-hypnat-le-iff* [*simp*]:
$\bigwedge m\ n.\ (\textit{of-hypnat}\ m \le (\textit{of-hypnat}\ n::'a::\textit{ordered-semidom star})) = (m \le n)$
⟨proof⟩

lemma *of-hypnat-0-le-iff* [*simp*]:
$\bigwedge n.\ 0 \le (\textit{of-hypnat}\ n::'a::\textit{ordered-semidom star})$
⟨proof⟩

lemma *of-hypnat-le-0-iff* [*simp*]:
$\bigwedge m.\ ((\textit{of-hypnat}\ m::'a::\textit{ordered-semidom star}) \le 0) = (m = 0)$
⟨proof⟩

lemma *of-hypnat-eq-iff* [*simp*]:
$\bigwedge m\ n.\ (\textit{of-hypnat}\ m = (\textit{of-hypnat}\ n::'a::\textit{ordered-semidom star})) = (m = n)$
⟨proof⟩

lemma *of-hypnat-eq-0-iff* [*simp*]:
$\bigwedge m.\ ((\textit{of-hypnat}\ m::'a::\textit{ordered-semidom star}) = 0) = (m = 0)$
⟨proof⟩

lemma *HNatInfinite-of-hypnat-gt-zero*:
$N \in \textit{HNatInfinite} \Longrightarrow (0::'a::\textit{ordered-semidom star}) < \textit{of-hypnat}\ N$
⟨proof⟩

end

7 HyperDef: Construction of Hyperreals Using Ultrafilters

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

types *hypreal* = *real star*

abbreviation
  hypreal-of-real :: *real* => *real star* **where**
  hypreal-of-real == *star-of*

abbreviation
  hypreal-of-hypnat :: *hypnat* ⇒ *hypreal* **where**
  hypreal-of-hypnat ≡ *of-hypnat*

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

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

notation (*xsymbols*)
  omega (ω) **and**
  epsilon (ε)

notation (*HTML* **output**)
  omega (ω) **and**
  epsilon (ε)

7.1 Real vector class instances

instantiation *star* :: (*scaleR*) *scaleR*
begin

definition
  star-scaleR-def [*transfer-unfold*, *code del*]: *scaleR r* ≡ *∗f∗* (*scaleR r*)

instance ⟨*proof*⟩

end

lemma *Standard-scaleR* [*simp*]: $x \in$ *Standard* $\Longrightarrow$ *scaleR r x* $\in$ *Standard*

⟨*proof*⟩

lemma *star-of-scaleR* [*simp*]: *star-of* (*scaleR r x*) = *scaleR r* (*star-of x*)
⟨*proof*⟩

instance *star* :: (*real-vector*) *real-vector*
⟨*proof*⟩

instance *star* :: (*real-algebra*) *real-algebra*
⟨*proof*⟩

instance *star* :: (*real-algebra-1*) *real-algebra-1* ⟨*proof*⟩

instance *star* :: (*real-div-algebra*) *real-div-algebra* ⟨*proof*⟩

instance *star* :: (*field-char-0*) *field-char-0* ⟨*proof*⟩

instance *star* :: (*real-field*) *real-field* ⟨*proof*⟩

lemma *star-of-real-def* [*transfer-unfold*]: *of-real r* = *star-of* (*of-real r*)
⟨*proof*⟩

lemma *Standard-of-real* [*simp*]: *of-real r* ∈ *Standard*
⟨*proof*⟩

lemma *star-of-of-real* [*simp*]: *star-of* (*of-real r*) = *of-real r*
⟨*proof*⟩

lemma *of-real-eq-star-of* [*simp*]: *of-real* = *star-of*
⟨*proof*⟩

lemma *Reals-eq-Standard*: (*Reals* :: *hypreal set*) = *Standard*
⟨*proof*⟩

7.2 Injection from *hypreal*

definition
of-hypreal :: *hypreal* ⇒ *'a::real-algebra-1 star* **where**
[*transfer-unfold*, *code del*]: *of-hypreal* = *f* *of-real*

lemma *Standard-of-hypreal* [*simp*]:
r ∈ *Standard* ⟹ *of-hypreal r* ∈ *Standard*
⟨*proof*⟩

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

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

lemma *of-hypreal-add* [*simp*]:
$\bigwedge x\ y.\ \textit{of-hypreal}\ (x + y) = \textit{of-hypreal}\ x + \textit{of-hypreal}\ y$
⟨*proof*⟩

lemma *of-hypreal-minus* [*simp*]: $\bigwedge x.\ \textit{of-hypreal}\ (-\ x) = -\ \textit{of-hypreal}\ x$
⟨*proof*⟩

lemma *of-hypreal-diff* [*simp*]:
$\bigwedge x\ y.\ \textit{of-hypreal}\ (x - y) = \textit{of-hypreal}\ x - \textit{of-hypreal}\ y$
⟨*proof*⟩

lemma *of-hypreal-mult* [*simp*]:
$\bigwedge x\ y.\ \textit{of-hypreal}\ (x * y) = \textit{of-hypreal}\ x * \textit{of-hypreal}\ y$
⟨*proof*⟩

lemma *of-hypreal-inverse* [*simp*]:
$\bigwedge x.\ \textit{of-hypreal}\ (\textit{inverse}\ x) =$
$\textit{inverse}\ (\textit{of-hypreal}\ x :: {'}a{::}\{\textit{real-div-algebra},\textit{division-by-zero}\}\ \textit{star})$
⟨*proof*⟩

lemma *of-hypreal-divide* [*simp*]:
$\bigwedge x\ y.\ \textit{of-hypreal}\ (x\ /\ y) =$
$(\textit{of-hypreal}\ x\ /\ \textit{of-hypreal}\ y :: {'}a{::}\{\textit{real-field},\textit{division-by-zero}\}\ \textit{star})$
⟨*proof*⟩

lemma *of-hypreal-eq-iff* [*simp*]:
$\bigwedge x\ y.\ (\textit{of-hypreal}\ x = \textit{of-hypreal}\ y) = (x = y)$
⟨*proof*⟩

lemma *of-hypreal-eq-0-iff* [*simp*]:
$\bigwedge x.\ (\textit{of-hypreal}\ x = 0) = (x = 0)$
⟨*proof*⟩

7.3 Properties of *starrel*

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

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

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

7.4 *hypreal-of-real*: the Injection from *real* to *hypreal*

lemma *inj-star-of*: *inj star-of*
⟨*proof*⟩

lemma *mem-Rep-star-iff*: $(X \in \mathit{Rep\text{-}star}\ x) = (x = \mathit{star\text{-}n}\ X)$
⟨*proof*⟩

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

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

7.5 Properties of *star-n*

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

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

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

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

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

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

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

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

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

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

⟨*proof*⟩

7.6 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*⟩

7.7 Existence of Infinite Hyperreal Number

Existence of infinite number not corresponding to any real number. Use assumption that member $\mathcal{U}$ is not finite.

A few lemmas first

lemma *lemma-omega-empty-singleton-disj*: $\{n::nat.\ x = real\ n\} = \{\}\ |$
$(\exists\, y.\ \{n::nat.\ x = real\ n\} = \{y\})$
⟨*proof*⟩

lemma *lemma-finite-omega-set*: *finite* $\{n::nat.\ x = real\ n\}$
⟨*proof*⟩

lemma *not-ex-hypreal-of-real-eq-omega*:
$\sim (\exists\, x.\ hypreal\text{-}of\text{-}real\ x = omega)$
⟨*proof*⟩

lemma *hypreal-of-real-not-eq-omega*: $hypreal\text{-}of\text{-}real\ x \neq omega$
⟨*proof*⟩

Existence of infinitesimal number also not corresponding to any real number

lemma *lemma-epsilon-empty-singleton-disj*:
$\{n::nat.\ x = inverse(real(Suc\ n))\} = \{\}\ |$
$(\exists\, y.\ \{n::nat.\ x = inverse(real(Suc\ n))\} = \{y\})$
⟨*proof*⟩

lemma *lemma-finite-epsilon-set*: *finite* $\{n.\ x = inverse(real(Suc\ n))\}$
⟨*proof*⟩

lemma *not-ex-hypreal-of-real-eq-epsilon*: $\sim (\exists\, x.\ hypreal\text{-}of\text{-}real\ x = epsilon)$

⟨*proof*⟩

lemma *hypreal-of-real-not-eq-epsilon*: *hypreal-of-real* $x \neq$ *epsilon*
⟨*proof*⟩

lemma *hypreal-epsilon-not-zero*: *epsilon* $\neq 0$
⟨*proof*⟩

lemma *hypreal-epsilon-inverse-omega*: *epsilon* = *inverse*(*omega*)
⟨*proof*⟩

lemma *hypreal-epsilon-gt-zero*: $0 <$ *epsilon*
⟨*proof*⟩

7.8 Absolute Value Function for the Hyperreals

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

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

lemma *hrabs-disj*: *abs* $x = (x$::'*a*::*abs-if*) | *abs* $x = -x$
⟨*proof*⟩

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

7.9 Embedding the Naturals into the Hyperreals

abbreviation
hypreal-of-nat :: *nat* => *hypreal* **where**
hypreal-of-nat == *of-nat*

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

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

lemma *hypreal-of-nat*:
hypreal-of-nat m = *star-n* (%n. *real* m)

⟨*proof*⟩

⟨*ML*⟩

7.10 Exponentials on the Hyperreals

lemma *hpowr-0* [*simp*]: $r \mathbin{\hat{}} 0 \quad = (1::hypreal)$
⟨*proof*⟩

lemma *hpowr-Suc* [*simp*]: $r \mathbin{\hat{}} (Suc\ n) = (r::hypreal) * (r \mathbin{\hat{}} n)$
⟨*proof*⟩

lemma *hrealpow-two*: $(r::hypreal) \mathbin{\hat{}} Suc\ (Suc\ 0) = r * r$
⟨*proof*⟩

lemma *hrealpow-two-le* [*simp*]: $(0::hypreal) \leq r \mathbin{\hat{}} Suc\ (Suc\ 0)$
⟨*proof*⟩

lemma *hrealpow-two-le-add-order* [*simp*]:
$(0::hypreal) \leq u \mathbin{\hat{}} Suc\ (Suc\ 0) + v \mathbin{\hat{}} Suc\ (Suc\ 0)$
⟨*proof*⟩

lemma *hrealpow-two-le-add-order2* [*simp*]:
$(0::hypreal) \leq u \mathbin{\hat{}} Suc\ (Suc\ 0) + v \mathbin{\hat{}} Suc\ (Suc\ 0) + w \mathbin{\hat{}} Suc\ (Suc\ 0)$
⟨*proof*⟩

lemma *hypreal-add-nonneg-eq-0-iff*:
$[|\ 0 \leq x;\ 0 \leq y\ |] ==> (x+y = 0) = (x = 0\ \&\ y = (0::hypreal))$
⟨*proof*⟩

FIXME: DELETE THESE

lemma *hypreal-three-squares-add-zero-iff*:
$(x*x + y*y + z*z = 0) = (x = 0\ \&\ y = 0\ \&\ z = (0::hypreal))$
⟨*proof*⟩

lemma *hrealpow-three-squares-add-zero-iff* [*simp*]:
$(x \mathbin{\hat{}} Suc\ (Suc\ 0) + y \mathbin{\hat{}} Suc\ (Suc\ 0) + z \mathbin{\hat{}} Suc\ (Suc\ 0) = (0::hypreal)) =$
$(x = 0\ \&\ y = 0\ \&\ z = 0)$
⟨*proof*⟩

lemma *hrabs-hrealpow-two* [*simp*]:
$abs(x \mathbin{\hat{}} Suc\ (Suc\ 0)) = (x::hypreal) \mathbin{\hat{}} Suc\ (Suc\ 0)$
⟨*proof*⟩

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

lemma *two-hrealpow-gt* [*simp*]: $hypreal\text{-}of\text{-}nat\ n < 2\ \hat{}\ n$
⟨*proof*⟩

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

lemma *hrealpow-sum-square-expand*:
$(x + (y{::}hypreal))\ \hat{}\ Suc\ (Suc\ 0) =$
$x\ \hat{}\ Suc\ (Suc\ 0) + y\ \hat{}\ Suc\ (Suc\ 0) + (hypreal\text{-}of\text{-}nat\ (Suc\ (Suc\ 0))){*}x{*}y$
⟨*proof*⟩

lemma *power-hypreal-of-real-number-of*:
$(number\text{-}of\ v :: hypreal)\ \hat{}\ n = hypreal\text{-}of\text{-}real\ ((number\text{-}of\ v)\ \hat{}\ n)$
⟨*proof*⟩
declare *power-hypreal-of-real-number-of* [*of - number-of w*, *standard*, *simp*]

7.11 Powers with Hypernatural Exponents

definition $pow :: [{}'a{::}power\ star,\ nat\ star] \Rightarrow {}'a\ star$ (**infixr** *pow 80*) **where**
hyperpow-def [*transfer-unfold*, *code del*]: $R\ pow\ N = (\ {*}f2{*}\ op\ \hat{}\ )\ R\ N$

lemma *Standard-hyperpow* [*simp*]:
$[\![r \in Standard;\ n \in Standard]\!] \Longrightarrow r\ pow\ n \in Standard$
⟨*proof*⟩

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

lemma *hyperpow-zero* [*simp*]:
$\bigwedge n.\ (0{::}{}'a{::}\{recpower,semiring\text{-}0\}\ star)\ pow\ (n + (1{::}hypnat)) = 0$
⟨*proof*⟩

lemma *hyperpow-not-zero*:
$\bigwedge r\ n.\ r \neq (0{::}{}'a{::}\{recpower,field\}\ star) ==> r\ pow\ n \neq 0$
⟨*proof*⟩

lemma *hyperpow-inverse*:
$\bigwedge r\ n.\ r \neq (0{::}{}'a{::}\{recpower,division\text{-}by\text{-}zero,field\}\ star)$
$\Longrightarrow inverse\ (r\ pow\ n) = (inverse\ r)\ pow\ n$
⟨*proof*⟩

lemma *hyperpow-hrabs*:
$\bigwedge r\ n.\ abs\ (r{::}{}'a{::}\{recpower,ordered\text{-}idom\}\ star)\ pow\ n = abs\ (r\ pow\ n)$
⟨*proof*⟩

lemma *hyperpow-add*:

$\bigwedge r\ n\ m.\ (r{::}'a{::}recpower\ star)\ pow\ (n + m) = (r\ pow\ n) * (r\ pow\ m)$
$\langle proof\rangle$

lemma *hyperpow-one* [*simp*]:
$\bigwedge r.\ (r{::}'a{::}recpower\ star)\ pow\ (1{::}hypnat) = r$
$\langle proof\rangle$

lemma *hyperpow-two*:
$\bigwedge r.\ (r{::}'a{::}recpower\ star)\ pow\ ((1{::}hypnat) + (1{::}hypnat)) = r * r$
$\langle proof\rangle$

lemma *hyperpow-gt-zero*:
$\bigwedge r\ n.\ (0{::}'a{::}\{recpower,ordered\text{-}semidom\}\ star) < r \Longrightarrow 0 < r\ pow\ n$
$\langle proof\rangle$

lemma *hyperpow-ge-zero*:
$\bigwedge r\ n.\ (0{::}'a{::}\{recpower,ordered\text{-}semidom\}\ star) \le r \Longrightarrow 0 \le r\ pow\ n$
$\langle proof\rangle$

lemma *hyperpow-le*:
$\bigwedge x\ y\ n.\ [\![(0{::}'a{::}\{recpower,ordered\text{-}semidom\}\ star) < x;\ x \le y]\!]$
$\Longrightarrow x\ pow\ n \le y\ pow\ n$
$\langle proof\rangle$

lemma *hyperpow-eq-one* [*simp*]:
$\bigwedge n.\ 1\ pow\ n = (1{::}'a{::}recpower\ star)$
$\langle proof\rangle$

lemma *hrabs-hyperpow-minus-one* [*simp*]:
$\bigwedge n.\ abs(-1\ pow\ n) = (1{::}'a{::}\{number\text{-}ring,recpower,ordered\text{-}idom\}\ star)$
$\langle proof\rangle$

lemma *hyperpow-mult*:
$\bigwedge r\ s\ n.\ (r * s{::}'a{::}\{comm\text{-}monoid\text{-}mult,recpower\}\ star)\ pow\ n$
$= (r\ pow\ n) * (s\ pow\ n)$
$\langle proof\rangle$

lemma *hyperpow-two-le* [*simp*]:
$(0{::}'a{::}\{recpower,ordered\text{-}ring\text{-}strict\}\ star) \le r\ pow\ (1 + 1)$
$\langle proof\rangle$

lemma *hrabs-hyperpow-two* [*simp*]:
$abs(x\ pow\ (1 + 1)) =$
$(x{::}'a{::}\{recpower,ordered\text{-}ring\text{-}strict\}\ star)\ pow\ (1 + 1)$
$\langle proof\rangle$

lemma *hyperpow-two-hrabs* [*simp*]:
$abs(x{::}'a{::}\{recpower,ordered\text{-}idom\}\ star)\ pow\ (1 + 1) = x\ pow\ (1 + 1)$
$\langle proof\rangle$

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

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

lemma *hyperpow-two-gt-one*:
$\bigwedge r::'a::\{recpower,ordered\text{-}semidom\} \ star. \ 1 < r \Longrightarrow 1 < r \ pow \ (1 + 1)$
⟨*proof*⟩

lemma *hyperpow-two-ge-one*:
$\bigwedge r::'a::\{recpower,ordered\text{-}semidom\} \ star. \ 1 \le r \Longrightarrow 1 \le r \ pow \ (1 + 1)$
⟨*proof*⟩

lemma *two-hyperpow-ge-one* [*simp*]: $(1::hypreal) \le 2 \ pow \ n$
⟨*proof*⟩

lemma *hyperpow-minus-one2* [*simp*]:
$!!n. \ -1 \ pow \ ((1 + 1)*n) = (1::hypreal)$
⟨*proof*⟩

lemma *hyperpow-less-le*:
$!!r \ n \ N. \ [|(0::hypreal) \le r; \ r \le 1; \ n < N|] ==> r \ pow \ N \le r \ pow \ n$
⟨*proof*⟩

lemma *hyperpow-SHNat-le*:
$[| \ 0 \le r; \ \ r \le (1::hypreal); \ \ N \in HNatInfinite \ |]$
$==> ALL \ n: Nats. \ r \ pow \ N \le r \ pow \ n$
⟨*proof*⟩

lemma *hyperpow-realpow*:
$(hypreal\text{-}of\text{-}real \ r) \ pow \ (hypnat\text{-}of\text{-}nat \ n) = hypreal\text{-}of\text{-}real \ (r \ \hat{} \ n)$
⟨*proof*⟩

lemma *hyperpow-SReal* [*simp*]:
$(hypreal\text{-}of\text{-}real \ r) \ pow \ (hypnat\text{-}of\text{-}nat \ n) \in Reals$
⟨*proof*⟩

lemma *hyperpow-zero-HNatInfinite* [*simp*]:
$N \in HNatInfinite ==> (0::hypreal) \ pow \ N = 0$
⟨*proof*⟩

lemma *hyperpow-le-le*:
$[| \ (0::hypreal) \le r; \ r \le 1; \ n \le N \ |] ==> r \ pow \ N \le r \ pow \ n$
⟨*proof*⟩

lemma *hyperpow-Suc-le-self2*:
$[| \ (0::hypreal) \le r; \ r < 1 \ |] ==> r \ pow \ (n + (1::hypnat)) \le r$
⟨*proof*⟩

lemma *hyperpow-hypnat-of-nat*: $\bigwedge x.\ x\ pow\ hypnat\text{-}of\text{-}nat\ n = x\ \hat{}\ n$
⟨*proof*⟩

lemma *of-hypreal-hyperpow*:
$\bigwedge x\ n.\ of\text{-}hypreal\ (x\ pow\ n) =$
$(of\text{-}hypreal\ x{::}'a{::}\{real\text{-}algebra\text{-}1, recpower\}\ star)\ pow\ n$
⟨*proof*⟩

end

8 NSA: Infinite Numbers, Infinitesimals, Infinitely Close Relation

theory *NSA*
imports *HyperDef RComplete*
begin

definition
hnorm :: *'a::norm star* ⇒ *real star* **where**
[*transfer-unfold*]: *hnorm* = ∗*f*∗ *norm*

definition
Infinitesimal :: (*'a::real-normed-vector*) *star set* **where**
[*code del*]: *Infinitesimal* = {*x*. ∀ *r* ∈ *Reals*. *0* < *r* −−> *hnorm x* < *r*}

definition
HFinite :: (*'a::real-normed-vector*) *star set* **where**
[*code del*]: *HFinite* = {*x*. ∃ *r* ∈ *Reals*. *hnorm x* < *r*}

definition
HInfinite :: (*'a::real-normed-vector*) *star set* **where**
[*code del*]: *HInfinite* = {*x*. ∀ *r* ∈ *Reals*. *r* < *hnorm x*}

definition
approx :: [*'a::real-normed-vector star*, *'a star*] => *bool* (**infixl** @= *50*) **where**
— the 'infinitely close' relation
(*x* @= *y*) = ((*x* − *y*) ∈ *Infinitesimal*)

definition
st :: *hypreal* => *hypreal* **where**
— the standard part of a hyperreal
st = (%*x*. @*r*. *x* ∈ *HFinite* & *r* ∈ *Reals* & *r* @= *x*)

definition
monad :: *'a::real-normed-vector star* => *'a star set* **where**
monad x = {*y*. *x* @= *y*}

definition
galaxy :: $'a$::*real-normed-vector star* => $'a$ *star set* **where**
galaxy $x = \{y.\ (x + -y) \in HFinite\}$

notation (*xsymbols*)
approx (**infixl** $\approx$ *50*)

notation (*HTML* **output**)
approx (**infixl** $\approx$ *50*)

lemma *SReal-def*: *Reals* == $\{x.\ \exists\, r.\ x = hypreal\text{-}of\text{-}real\ r\}$
⟨*proof*⟩

8.1 Nonstandard Extension of the Norm Function

definition
scaleHR :: *real star* $\Rightarrow$ $'a$ *star* $\Rightarrow$ $'a$::*real-normed-vector star* **where**
[*transfer-unfold*, *code del*]: *scaleHR* = *starfun2 scaleR*

lemma *Standard-hnorm* [*simp*]: $x \in Standard \Longrightarrow hnorm\ x \in Standard$
⟨*proof*⟩

lemma *star-of-norm* [*simp*]: *star-of* (*norm* x) = *hnorm* (*star-of* x)
⟨*proof*⟩

lemma *hnorm-ge-zero* [*simp*]:
$\bigwedge x$::$'a$::*real-normed-vector star*. $0 \leq hnorm\ x$
⟨*proof*⟩

lemma *hnorm-eq-zero* [*simp*]:
$\bigwedge x$::$'a$::*real-normed-vector star*. $(hnorm\ x = 0) = (x = 0)$
⟨*proof*⟩

lemma *hnorm-triangle-ineq*:
$\bigwedge x\ y$::$'a$::*real-normed-vector star*. $hnorm\ (x + y) \leq hnorm\ x + hnorm\ y$
⟨*proof*⟩

lemma *hnorm-triangle-ineq3*:
$\bigwedge x\ y$::$'a$::*real-normed-vector star*. $|hnorm\ x - hnorm\ y| \leq hnorm\ (x - y)$
⟨*proof*⟩

lemma *hnorm-scaleR*:
$\bigwedge x$::$'a$::*real-normed-vector star*.
$hnorm\ (a *_R x) = |star\text{-}of\ a| * hnorm\ x$
⟨*proof*⟩

lemma *hnorm-scaleHR*:
$\bigwedge a$ (x::$'a$::*real-normed-vector star*).
$hnorm\ (scaleHR\ a\ x) = |a| * hnorm\ x$

⟨*proof*⟩

lemma *hnorm-mult-ineq*:
$\bigwedge x\ y::'a::real\text{-}normed\text{-}algebra\ star.\ hnorm\ (x * y) \leq hnorm\ x * hnorm\ y$
⟨*proof*⟩

lemma *hnorm-mult*:
$\bigwedge x\ y::'a::real\text{-}normed\text{-}div\text{-}algebra\ star.\ hnorm\ (x * y) = hnorm\ x * hnorm\ y$
⟨*proof*⟩

lemma *hnorm-hyperpow*:
$\bigwedge(x::'a::\{real\text{-}normed\text{-}div\text{-}algebra,recpower\}\ star)\ n.$
$hnorm\ (x\ pow\ n) = hnorm\ x\ pow\ n$
⟨*proof*⟩

lemma *hnorm-one* [*simp*]:
$hnorm\ (1::'a::real\text{-}normed\text{-}div\text{-}algebra\ star) = 1$
⟨*proof*⟩

lemma *hnorm-zero* [*simp*]:
$hnorm\ (0::'a::real\text{-}normed\text{-}vector\ star) = 0$
⟨*proof*⟩

lemma *zero-less-hnorm-iff* [*simp*]:
$\bigwedge x::'a::real\text{-}normed\text{-}vector\ star.\ (0 < hnorm\ x) = (x \neq 0)$
⟨*proof*⟩

lemma *hnorm-minus-cancel* [*simp*]:
$\bigwedge x::'a::real\text{-}normed\text{-}vector\ star.\ hnorm\ (-\ x) = hnorm\ x$
⟨*proof*⟩

lemma *hnorm-minus-commute*:
$\bigwedge a\ b::'a::real\text{-}normed\text{-}vector\ star.\ hnorm\ (a - b) = hnorm\ (b - a)$
⟨*proof*⟩

lemma *hnorm-triangle-ineq2*:
$\bigwedge a\ b::'a::real\text{-}normed\text{-}vector\ star.\ hnorm\ a - hnorm\ b \leq hnorm\ (a - b)$
⟨*proof*⟩

lemma *hnorm-triangle-ineq4*:
$\bigwedge a\ b::'a::real\text{-}normed\text{-}vector\ star.\ hnorm\ (a - b) \leq hnorm\ a + hnorm\ b$
⟨*proof*⟩

lemma *abs-hnorm-cancel* [*simp*]:
$\bigwedge a::'a::real\text{-}normed\text{-}vector\ star.\ |hnorm\ a| = hnorm\ a$
⟨*proof*⟩

lemma *hnorm-of-hypreal* [*simp*]:
$\bigwedge r.\ hnorm\ (of\text{-}hypreal\ r::'a::real\text{-}normed\text{-}algebra\text{-}1\ star) = |r|$

⟨*proof*⟩

lemma *nonzero-hnorm-inverse*:
$\bigwedge a::'a::real\text{-}normed\text{-}div\text{-}algebra\ star.$
$a \neq 0 \Longrightarrow hnorm\ (inverse\ a) = inverse\ (hnorm\ a)$
⟨*proof*⟩

lemma *hnorm-inverse*:
$\bigwedge a::'a::\{real\text{-}normed\text{-}div\text{-}algebra,division\text{-}by\text{-}zero\}\ star.$
$hnorm\ (inverse\ a) = inverse\ (hnorm\ a)$
⟨*proof*⟩

lemma *hnorm-divide*:
$\bigwedge a\ b::'a::\{real\text{-}normed\text{-}field,division\text{-}by\text{-}zero\}\ star.$
$hnorm\ (a\ /\ b) = hnorm\ a\ /\ hnorm\ b$
⟨*proof*⟩

lemma *hypreal-hnorm-def* [*simp*]:
$\bigwedge r::hypreal.\ hnorm\ r = |r|$
⟨*proof*⟩

lemma *hnorm-add-less*:
$\bigwedge(x::'a::real\text{-}normed\text{-}vector\ star)\ y\ r\ s.$
$[\![hnorm\ x < r;\ hnorm\ y < s]\!] \Longrightarrow hnorm\ (x + y) < r + s$
⟨*proof*⟩

lemma *hnorm-mult-less*:
$\bigwedge(x::'a::real\text{-}normed\text{-}algebra\ star)\ y\ r\ s.$
$[\![hnorm\ x < r;\ hnorm\ y < s]\!] \Longrightarrow hnorm\ (x * y) < r * s$
⟨*proof*⟩

lemma *hnorm-scaleHR-less*:
$[\![|x| < r;\ hnorm\ y < s]\!] \Longrightarrow hnorm\ (scaleHR\ x\ y) < r * s$
⟨*proof*⟩

8.2 Closure Laws for the Standard Reals

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

lemma *Reals-add-cancel*: $[\![x + y \in Reals;\ y \in Reals]\!] \Longrightarrow x \in Reals$
⟨*proof*⟩

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

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

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

epsilon is not in Reals because it is an infinitesimal

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

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

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

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

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

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

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

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

Completeness of Reals, but both lemmas are unused.

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

8.3 Set of Finite Elements is a Subring of the Extended Reals

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

lemma *HFinite-mult*:
fixes $x\ y :: 'a::real\text{-}normed\text{-}algebra\ star$

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

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

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

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

lemma *SReal-subset-HFinite*: $(Reals::hypreal\ set) \subseteq HFinite$
⟨*proof*⟩

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

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

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

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

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

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

lemma *hrealpow-HFinite*:
fixes $x :: 'a::\{real\text{-}normed\text{-}algebra,recpower\}\ star$
shows $x \in HFinite ==> x\ \hat{}\ n \in HFinite$
⟨*proof*⟩

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

8.4 Set of Infinitesimals is a Subring of the Hyperreals

lemma *InfinitesimalI*:

$(\bigwedge r.\ [\![r \in \mathbb{R};\ 0 < r]\!] \Longrightarrow hnorm\ x < r) \Longrightarrow x \in Infinitesimal$
⟨proof⟩

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

lemma *InfinitesimalI2*:
$(\bigwedge r.\ 0 < r \Longrightarrow hnorm\ x < star\text{-}of\ r) \Longrightarrow x \in Infinitesimal$
⟨proof⟩

lemma *InfinitesimalD2*:
$[\![x \in Infinitesimal;\ 0 < r]\!] \Longrightarrow hnorm\ x < star\text{-}of\ r$
⟨proof⟩

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

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

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

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

lemma *Infinitesimal-hnorm-iff*:
$(hnorm\ x \in Infinitesimal) = (x \in Infinitesimal)$
⟨proof⟩

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

lemma *Infinitesimal-of-hypreal-iff* [*simp*]:
$((of\text{-}hypreal\ x::'a::real\text{-}normed\text{-}algebra\text{-}1\ star) \in Infinitesimal) =$
$(x \in Infinitesimal)$
⟨proof⟩

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

lemma *Infinitesimal-mult*:
fixes $x\ y$ :: $'a::real\text{-}normed\text{-}algebra\ star$
shows [|$x \in Infinitesimal$; $y \in Infinitesimal$|] ==> $(x * y) \in Infinitesimal$
⟨proof⟩

lemma *Infinitesimal-HFinite-mult*:
fixes $x\ y$ :: $'a$::*real-normed-algebra star*
shows $[|\ x \in Infinitesimal;\ y \in HFinite\ |] ==> (x * y) \in Infinitesimal$
⟨*proof*⟩

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

lemma *Infinitesimal-HFinite-mult2*:
fixes $x\ y$ :: $'a$::*real-normed-algebra star*
shows $[|\ x \in Infinitesimal;\ y \in HFinite\ |] ==> (y * x) \in Infinitesimal$
⟨*proof*⟩

lemma *Infinitesimal-scaleR2*:
$x \in Infinitesimal ==> a *_R x \in Infinitesimal$
⟨*proof*⟩

lemma *Compl-HFinite*: $-\ HFinite = HInfinite$
⟨*proof*⟩

lemma *HInfinite-inverse-Infinitesimal*:
fixes x :: $'a$::*real-normed-div-algebra star*
shows $x \in HInfinite ==> inverse\ x \in Infinitesimal$
⟨*proof*⟩

lemma *HInfiniteI*: $(\bigwedge r.\ r \in \mathbb{R} \Longrightarrow r < hnorm\ x) \Longrightarrow x \in HInfinite$
⟨*proof*⟩

lemma *HInfiniteD*: $[\![x \in HInfinite;\ r \in \mathbb{R}]\!] \Longrightarrow r < hnorm\ x$
⟨*proof*⟩

lemma *HInfinite-mult*:
fixes $x\ y$ :: $'a$::*real-normed-div-algebra star*
shows $[|x \in HInfinite;\ y \in HInfinite|] ==> (x*y) \in HInfinite$
⟨*proof*⟩

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

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

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

lemma *HInfinite-add-gt-zero*:
  $[|(x::hypreal) \in HInfinite;\ 0 < y;\ 0 < x|] ==> (x + y): HInfinite$
$\langle proof \rangle$

lemma *HInfinite-minus-iff*: $(-x \in HInfinite) = (x \in HInfinite)$
$\langle proof \rangle$

lemma *HInfinite-add-le-zero*:
  $[|(x::hypreal) \in HInfinite;\ y \le 0;\ x \le 0|] ==> (x + y): HInfinite$
$\langle proof \rangle$

lemma *HInfinite-add-lt-zero*:
  $[|(x::hypreal) \in HInfinite;\ y < 0;\ x < 0|] ==> (x + y): HInfinite$
$\langle proof \rangle$

lemma *HFinite-sum-squares*:
 fixes $a\ b\ c :: 'a::real\text{-}normed\text{-}algebra\ star$
 shows $[|a: HFinite;\ b: HFinite;\ c: HFinite|]$
  $==> a*a + b*b + c*c \in HFinite$
$\langle proof \rangle$

lemma *not-Infinitesimal-not-zero*: $x \notin Infinitesimal ==> x \neq 0$
$\langle proof \rangle$

lemma *not-Infinitesimal-not-zero2*: $x \in HFinite - Infinitesimal ==> x \neq 0$
$\langle proof \rangle$

lemma *HFinite-diff-Infinitesimal-hrabs*:
 $(x::hypreal) \in HFinite - Infinitesimal ==> abs\ x \in HFinite - Infinitesimal$
$\langle proof \rangle$

lemma *hnorm-le-Infinitesimal*:
 $[\![e \in Infinitesimal;\ hnorm\ x \le e]\!] \Longrightarrow x \in Infinitesimal$
$\langle proof \rangle$

lemma *hnorm-less-Infinitesimal*:
 $[\![e \in Infinitesimal;\ hnorm\ x < e]\!] \Longrightarrow x \in Infinitesimal$
$\langle proof \rangle$

lemma *hrabs-le-Infinitesimal*:
  $[|\ e \in Infinitesimal;\ abs\ (x::hypreal) \le e\ |] ==> x \in Infinitesimal$
$\langle proof \rangle$

lemma *hrabs-less-Infinitesimal*:
  $[|\ e \in Infinitesimal;\ abs\ (x::hypreal) < e\ |] ==> x \in Infinitesimal$
$\langle proof \rangle$

lemma *Infinitesimal-interval*:
  $[|\ e \in Infinitesimal;\ e' \in Infinitesimal;\ e' < x\ ;\ x < e\ |]$

$==>$ $(x{::}hypreal) \in Infinitesimal$
⟨proof⟩

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

lemma *lemma-Infinitesimal-hyperpow*:
$[| \ (x{::}hypreal) \in Infinitesimal; \ 0 < N \ |] ==> abs \ (x \ pow \ N) \leq abs \ x$
⟨proof⟩

lemma *Infinitesimal-hyperpow*:
$[| \ (x{::}hypreal) \in Infinitesimal; \ 0 < N \ |] ==> x \ pow \ N \in Infinitesimal$
⟨proof⟩

lemma *hrealpow-hyperpow-Infinitesimal-iff*:
$(x \ \hat{} \ n \in Infinitesimal) = (x \ pow \ (hypnat\text{-}of\text{-}nat \ n) \in Infinitesimal)$
⟨proof⟩

lemma *Infinitesimal-hrealpow*:
$[| \ (x{::}hypreal) \in Infinitesimal; \ 0 < n \ |] ==> x \ \hat{} \ n \in Infinitesimal$
⟨proof⟩

lemma *not-Infinitesimal-mult*:
fixes $x \ y :: 'a{::}real\text{-}normed\text{-}div\text{-}algebra \ star$
shows $[| \ x \notin Infinitesimal; \ \ y \notin Infinitesimal|] ==> (x{*}y) \notin Infinitesimal$
⟨proof⟩

lemma *Infinitesimal-mult-disj*:
fixes $x \ y :: 'a{::}real\text{-}normed\text{-}div\text{-}algebra \ star$
shows $x{*}y \in Infinitesimal ==> x \in Infinitesimal \ | \ y \in Infinitesimal$
⟨proof⟩

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

lemma *HFinite-Infinitesimal-diff-mult*:
fixes $x \ y :: 'a{::}real\text{-}normed\text{-}div\text{-}algebra \ star$
shows $[| \ x \in HFinite - Infinitesimal;$
$y \in HFinite - Infinitesimal$
$|] ==> (x{*}y) \in HFinite - Infinitesimal$
⟨proof⟩

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

lemma *Infinitesimal-star-of-mult*:
fixes x :: *'a::real-normed-algebra star*
shows $x \in$ *Infinitesimal* ==> $x *$ *star-of* $r \in$ *Infinitesimal*
⟨*proof*⟩

lemma *Infinitesimal-star-of-mult2*:
fixes x :: *'a::real-normed-algebra star*
shows $x \in$ *Infinitesimal* ==> *star-of* $r * x \in$ *Infinitesimal*
⟨*proof*⟩

8.5 The Infinitely Close Relation

lemma *mem-infmal-iff*: ($x \in$ *Infinitesimal*) = (x @= *0*)
⟨*proof*⟩

lemma *approx-minus-iff*: (x @= y) = ($x - y$ @= *0*)
⟨*proof*⟩

lemma *approx-minus-iff2*: (x @= y) = ($-y + x$ @= *0*)
⟨*proof*⟩

lemma *approx-refl* [*iff*]: x @= x
⟨*proof*⟩

lemma *hypreal-minus-distrib1*: $-(y + -(x$::*'a::ab-group-add*$))$ = $x + -y$
⟨*proof*⟩

lemma *approx-sym*: x @= y ==> y @= x
⟨*proof*⟩

lemma *approx-trans*: [| x @= y; y @= z |] ==> x @= z
⟨*proof*⟩

lemma *approx-trans2*: [| r @= x; s @= x |] ==> r @= s
⟨*proof*⟩

lemma *approx-trans3*: [| x @= r; x @= s|] ==> r @= s
⟨*proof*⟩

lemma *number-of-approx-reorient*: (*number-of* w @= x) = (x @= *number-of* w)
⟨*proof*⟩

lemma *zero-approx-reorient*: (*0* @= x) = (x @= *0*)
⟨*proof*⟩

lemma *one-approx-reorient*: (*1* @= x) = (x @= *1*)
⟨*proof*⟩

⟨*ML*⟩

lemma *Infinitesimal-approx-minus*: $(x-y \in Infinitesimal) = (x \ @= \ y)$
⟨*proof*⟩

lemma *approx-monad-iff*: $(x \ @= \ y) = (monad(x)=monad(y))$
⟨*proof*⟩

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

lemma *approx-add*: $[| \ a \ @= \ b; \ c \ @= \ d \ |] ==> a+c \ @= \ b+d$
⟨*proof*⟩

lemma *approx-minus*: $a \ @= \ b ==> -a \ @= \ -b$
⟨*proof*⟩

lemma *approx-minus2*: $-a \ @= \ -b ==> a \ @= \ b$
⟨*proof*⟩

lemma *approx-minus-cancel* [*simp*]: $(-a \ @= \ -b) = (a \ @= \ b)$
⟨*proof*⟩

lemma *approx-add-minus*: $[| \ a \ @= \ b; \ c \ @= \ d \ |] ==> a + -c \ @= \ b + -d$
⟨*proof*⟩

lemma *approx-diff*: $[| \ a \ @= \ b; \ c \ @= \ d \ |] ==> a - c \ @= \ b - d$
⟨*proof*⟩

lemma *approx-mult1*:
 fixes $a \ b \ c$ :: *'a::real-normed-algebra star*
 shows $[| \ a \ @= \ b; \ c\text{: } HFinite|] ==> a*c \ @= \ b*c$
⟨*proof*⟩

lemma *approx-mult2*:
 fixes $a \ b \ c$ :: *'a::real-normed-algebra star*
 shows $[|a \ @= \ b; \ c\text{: } HFinite|] ==> c*a \ @= \ c*b$
⟨*proof*⟩

lemma *approx-mult-subst*:
 fixes $u \ v \ x \ y$ :: *'a::real-normed-algebra star*
 shows $[|u \ @= \ v*x; \ x \ @= \ y; \ v \in HFinite|] ==> u \ @= \ v*y$
⟨*proof*⟩

lemma *approx-mult-subst2*:
 fixes $u \ v \ x \ y$ :: *'a::real-normed-algebra star*
 shows $[| \ u \ @= \ x*v; \ x \ @= \ y; \ v \in HFinite \ |] ==> u \ @= \ y*v$
⟨*proof*⟩

lemma *approx-mult-subst-star-of*:
  fixes *u x y* :: *'a::real-normed-algebra star*
  shows [| *u* @= *x*∗*star-of v*; *x* @= *y* |] ==> *u* @= *y*∗*star-of v*
⟨*proof*⟩

lemma *approx-eq-imp*: *a* = *b* ==> *a* @= *b*
⟨*proof*⟩

lemma *Infinitesimal-minus-approx*: *x* ∈ *Infinitesimal* ==> −*x* @= *x*
⟨*proof*⟩

lemma *bex-Infinitesimal-iff*: (∃ *y* ∈ *Infinitesimal*. *x* − *z* = *y*) = (*x* @= *z*)
⟨*proof*⟩

lemma *bex-Infinitesimal-iff2*: (∃ *y* ∈ *Infinitesimal*. *x* = *z* + *y*) = (*x* @= *z*)
⟨*proof*⟩

lemma *Infinitesimal-add-approx*: [| *y* ∈ *Infinitesimal*; *x* + *y* = *z* |] ==> *x* @= *z*
⟨*proof*⟩

lemma *Infinitesimal-add-approx-self*: *y* ∈ *Infinitesimal* ==> *x* @= *x* + *y*
⟨*proof*⟩

lemma *Infinitesimal-add-approx-self2*: *y* ∈ *Infinitesimal* ==> *x* @= *y* + *x*
⟨*proof*⟩

lemma *Infinitesimal-add-minus-approx-self*: *y* ∈ *Infinitesimal* ==> *x* @= *x* + −*y*
⟨*proof*⟩

lemma *Infinitesimal-add-cancel*: [| *y* ∈ *Infinitesimal*; *x*+*y* @= *z*|] ==> *x* @= *z*
⟨*proof*⟩

lemma *Infinitesimal-add-right-cancel*:
     [| *y* ∈ *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-star-of-HFinite*: $x \; @= \; star\text{-}of \; D \; ==> \; x \in HFinite$
⟨*proof*⟩

lemma *approx-mult-HFinite*:
fixes $a \; b \; c \; d$ :: $'a$::*real-normed-algebra star*
shows $[|a \; @= \; b; \; c \; @= \; d; \; b\text{: } HFinite; \; d\text{: } HFinite|] \; ==> \; a{*}c \; @= \; b{*}d$
⟨*proof*⟩

lemma *scaleHR-left-diff-distrib*:
$\bigwedge a \; b \; x. \; scaleHR \; (a - b) \; x = scaleHR \; a \; x - scaleHR \; b \; x$
⟨*proof*⟩

lemma *approx-scaleR1*:
$[| \; a \; @= \; star\text{-}of \; b; \; c\text{: } HFinite|] \; ==> \; scaleHR \; a \; c \; @= \; b *_R c$
⟨*proof*⟩

lemma *approx-scaleR2*:
$a \; @= \; b \; ==> \; c *_R a \; @= \; c *_R b$
⟨*proof*⟩

lemma *approx-scaleR-HFinite*:
$[|a \; @= \; star\text{-}of \; b; \; c \; @= \; d; \; d\text{: } HFinite|] \; ==> \; scaleHR \; a \; c \; @= \; b *_R d$
⟨*proof*⟩

lemma *approx-mult-star-of*:
fixes $a \; c$ :: $'a$::*real-normed-algebra star*
shows $[|a \; @= \; star\text{-}of \; b; \; c \; @= \; star\text{-}of \; d \; |]$
$==> \; a{*}c \; @= \; star\text{-}of \; b{*}star\text{-}of \; d$
⟨*proof*⟩

lemma *approx-SReal-mult-cancel-zero*:
$[| \; (a::hypreal) \in Reals; \; a \neq 0; \; a{*}x \; @= \; 0 \; |] \; ==> \; x \; @= \; 0$
⟨*proof*⟩

lemma *approx-mult-SReal1*: $[| \; (a::hypreal) \in Reals; \; x \; @= \; 0 \; |] \; ==> \; x{*}a \; @= \; 0$
⟨*proof*⟩

lemma *approx-mult-SReal2*: $[| \; (a::hypreal) \in Reals; \; x \; @= \; 0 \; |] \; ==> \; a{*}x \; @= \; 0$
⟨*proof*⟩

lemma *approx-mult-SReal-zero-cancel-iff* [*simp*]:
$[|(a::hypreal) \in Reals;\ a \neq 0\ |] ==> (a*x\ @=\ 0) = (x\ @=\ 0)$
⟨*proof*⟩

lemma *approx-SReal-mult-cancel*:
$[|\ (a::hypreal) \in Reals;\ a \neq 0;\ a*\ w\ @=\ a*z\ |] ==> w\ @=\ z$
⟨*proof*⟩

lemma *approx-SReal-mult-cancel-iff1* [*simp*]:
$[|\ (a::hypreal) \in Reals;\ a \neq 0|] ==> (a*\ w\ @=\ a*z) = (w\ @=\ z)$
⟨*proof*⟩

lemma *approx-le-bound*: $[|\ (z::hypreal) \leq f;\ f\ @=\ g;\ g \leq z\ |] ==> f\ @=\ z$
⟨*proof*⟩

lemma *approx-hnorm*:
fixes $x\ y :: 'a::real\text{-}normed\text{-}vector\ star$
shows $x \approx y \Longrightarrow hnorm\ x \approx hnorm\ y$
⟨*proof*⟩

8.6 Zero is the Only Infinitesimal that is also a Real

lemma *Infinitesimal-less-SReal*:
$[|\ (x::hypreal) \in Reals;\ y \in Infinitesimal;\ 0 < x\ |] ==> y < x$
⟨*proof*⟩

lemma *Infinitesimal-less-SReal2*:
$(y::hypreal) \in Infinitesimal ==> \forall r \in Reals.\ 0 < r\ -->\ y < r$
⟨*proof*⟩

lemma *SReal-not-Infinitesimal*:
$[|\ 0 < y;\ \ (y::hypreal) \in Reals|] ==> y \notin Infinitesimal$
⟨*proof*⟩

lemma *SReal-minus-not-Infinitesimal*:
$[|\ y < 0;\ \ (y::hypreal) \in Reals\ |] ==> y \notin Infinitesimal$
⟨*proof*⟩

lemma *SReal-Int-Infinitesimal-zero*: $Reals\ Int\ Infinitesimal = \{0::hypreal\}$
⟨*proof*⟩

lemma *SReal-Infinitesimal-zero*:
$[|\ (x::hypreal) \in Reals;\ x \in Infinitesimal|] ==> x = 0$
⟨*proof*⟩

lemma *SReal-HFinite-diff-Infinitesimal*:
$[|\ (x::hypreal) \in Reals;\ x \neq 0\ |] ==> x \in HFinite - Infinitesimal$
⟨*proof*⟩

lemma *hypreal-of-real-HFinite-diff-Infinitesimal*:
  hypreal-of-real $x \neq 0$ ==> *hypreal-of-real* $x \in$ *HFinite* $-$ *Infinitesimal*
⟨*proof*⟩

lemma *star-of-Infinitesimal-iff-0* [*iff*]:
  (*star-of* $x \in$ *Infinitesimal*) = (x = *0*)
⟨*proof*⟩

lemma *star-of-HFinite-diff-Infinitesimal*:
  $x \neq 0$ ==> *star-of* $x \in$ *HFinite* $-$ *Infinitesimal*
⟨*proof*⟩

lemma *number-of-not-Infinitesimal* [*simp*]:
  number-of $w \neq$ (*0*::*hypreal*) ==> (*number-of* w :: *hypreal*) $\notin$ *Infinitesimal*
⟨*proof*⟩

lemma *one-not-Infinitesimal* [*simp*]:
  (*1*::′*a*::{*real-normed-vector*,*zero-neq-one*} *star*) $\notin$ *Infinitesimal*
⟨*proof*⟩

lemma *approx-SReal-not-zero*:
  [| (y::*hypreal*) $\in$ *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*:
  fixes x y :: ′*a*::{*real-normed-div-algebra*,*field*} *star*
  shows [| $y \neq$ *0*; $y \in$ *Infinitesimal*; $x/y \in$ *HFinite* |]
    ==> $x \in$ *Infinitesimal*
⟨*proof*⟩

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

8.7 Uniqueness: Two Infinitely Close Reals are Equal

lemma *star-of-approx-iff* [*simp*]: (*star-of* x @= *star-of* y) = (x = y)
⟨*proof*⟩

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

lemma *number-of-approx-iff* [*simp*]:
(*number-of v* @= (*number-of w* :: *'a*::{*number*,*real-normed-vector*} *star*)) =
(*number-of v* = (*number-of w* :: *'a*))
⟨*proof*⟩

lemma [*simp*]:
(*number-of w* @= (*0*::*'a*::{*number*,*real-normed-vector*} *star*)) =
(*number-of w* = (*0*::*'a*))
((*0*::*'a*::{*number*,*real-normed-vector*} *star*) @= *number-of w*) =
(*number-of w* = (*0*::*'a*))
(*number-of w* @= (*1*::*'b*::{*number*,*one*,*real-normed-vector*} *star*)) =
(*number-of w* = (*1*::*'b*))
((*1*::*'b*::{*number*,*one*,*real-normed-vector*} *star*) @= *number-of w*) =
(*number-of w* = (*1*::*'b*))
~ (*0* @= (*1*::*'c*::{*zero-neq-one*,*real-normed-vector*} *star*))
~ (*1* @= (*0*::*'c*::{*zero-neq-one*,*real-normed-vector*} *star*))
⟨*proof*⟩

lemma *star-of-approx-number-of-iff* [*simp*]:
(*star-of k* @= *number-of w*) = (*k* = *number-of w*)
⟨*proof*⟩

lemma *star-of-approx-zero-iff* [*simp*]: (*star-of k* @= *0*) = (*k* = *0*)
⟨*proof*⟩

lemma *star-of-approx-one-iff* [*simp*]: (*star-of k* @= *1*) = (*k* = *1*)
⟨*proof*⟩

lemma *approx-unique-real*:
[| (*r*::*hypreal*) ∈ *Reals*; *s* ∈ *Reals*; *r* @= *x*; *s* @= *x*|] ==> *r* = *s*
⟨*proof*⟩

8.8 Existence of Unique Real Infinitely Close

8.8.1 Lifting of the Ub and Lub Properties

lemma *hypreal-of-real-isUb-iff*:
(*isUb* (*Reals*) (*hypreal-of-real* ' *Q*) (*hypreal-of-real Y*)) =
(*isUb* (*UNIV* :: *real set*) *Q Y*)
⟨*proof*⟩

lemma *hypreal-of-real-isLub1*:
isLub Reals (*hypreal-of-real* ' *Q*) (*hypreal-of-real Y*)
==> *isLub* (*UNIV* :: *real set*) *Q Y*
⟨*proof*⟩

lemma *hypreal-of-real-isLub2*:
isLub (*UNIV* :: *real set*) *Q Y*

$\implies$ *isLub Reals (hypreal-of-real ‘ Q) (hypreal-of-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-of-real Yo)*
⟨*proof*⟩

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

lemma *lemma-isLub-hypreal-of-real2*:
$\exists$ *Yo. isLub Reals P (hypreal-of-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*⟩

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

lemma *lemma-st-part-ub*:
$(x::hypreal) \in HFinite$ ==> $\exists u.$ *isUb Reals* $\{s.\ s \in Reals\ \&\ s < x\}\ u$
⟨*proof*⟩

lemma *lemma-st-part-nonempty*:
$(x::hypreal) \in HFinite$ ==> $\exists y.\ y \in \{s.\ s \in Reals\ \&\ s < x\}$
⟨*proof*⟩

lemma *lemma-st-part-subset*: $\{s.\ s \in Reals\ \&\ s < x\} \subseteq Reals$
⟨*proof*⟩

lemma *lemma-st-part-lub*:
$(x::hypreal) \in HFinite$ ==> $\exists t.$ *isLub Reals* $\{s.\ s \in Reals\ \&\ s < x\}\ t$
⟨*proof*⟩

lemma *lemma-hypreal-le-left-cancel*: $((t::hypreal) + r \le t) = (r \le 0)$
⟨*proof*⟩

lemma *lemma-st-part-le1*:

[| $(x{::}hypreal) \in HFinite$; $isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t$;
  $r \in Reals$; $0 < r$ |] ==> $x \le t + r$
⟨*proof*⟩

lemma *hypreal-setle-less-trans*:
  [| $S *<= (x{::}hypreal)$; $x < y$ |] ==> $S *<= y$
⟨*proof*⟩

lemma *hypreal-gt-isUb*:
  [| $isUb\ R\ S\ (x{::}hypreal)$; $x < y$; $y \in R$ |] ==> $isUb\ R\ S\ y$
⟨*proof*⟩

lemma *lemma-st-part-gt-ub*:
  [| $(x{::}hypreal) \in HFinite$; $x < y$; $y \in Reals$ |]
  ==> $isUb\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ y$
⟨*proof*⟩

lemma *lemma-minus-le-zero*: $t \le t + -r$ ==> $r \le (0{::}hypreal)$
⟨*proof*⟩

lemma *lemma-st-part-le2*:
  [| $(x{::}hypreal) \in HFinite$;
  $isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t$;
  $r \in Reals$; $0 < r$ |]
  ==> $t + -r \le x$
⟨*proof*⟩

lemma *lemma-st-part1a*:
  [| $(x{::}hypreal) \in HFinite$;
  $isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t$;
  $r \in Reals$; $0 < r$ |]
  ==> $x + -t \le r$
⟨*proof*⟩

lemma *lemma-st-part2a*:
  [| $(x{::}hypreal) \in HFinite$;
  $isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t$;
  $r \in Reals$; $0 < r$ |]
  ==> $-(x + -t) \le r$
⟨*proof*⟩

lemma *lemma-SReal-ub*:
  $(x{::}hypreal) \in Reals$ ==> $isUb\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ x$
⟨*proof*⟩

lemma *lemma-SReal-lub*:
  $(x{::}hypreal) \in Reals$ ==> $isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ x$
⟨*proof*⟩

lemma *lemma-st-part-not-eq1*:
$[| (x::hypreal) \in HFinite;$
$isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ 0 < r\ |]$
$==> x + -t \neq r$
⟨*proof*⟩

lemma *lemma-st-part-not-eq2*:
$[| (x::hypreal) \in HFinite;$
$isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ 0 < r\ |]$
$==> -(x + -t) \neq r$
⟨*proof*⟩

lemma *lemma-st-part-major*:
$[| (x::hypreal) \in HFinite;$
$isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t;$
$r \in Reals;\ 0 < r\ |]$
$==> abs\ (x - t) < r$
⟨*proof*⟩

lemma *lemma-st-part-major2*:
$[| (x::hypreal) \in HFinite;\ isLub\ Reals\ \{s.\ s \in Reals\ \&\ s < x\}\ t\ |]$
$==> \forall r \in Reals.\ 0 < r --> abs\ (x - t) < r$
⟨*proof*⟩

Existence of real and Standard Part Theorem

lemma *lemma-st-part-Ex*:
$(x::hypreal) \in HFinite$
$==> \exists t \in Reals.\ \forall r \in Reals.\ 0 < r --> abs\ (x - t) < r$
⟨*proof*⟩

lemma *st-part-Ex*:
$(x::hypreal) \in HFinite ==> \exists t \in Reals.\ x\ @= t$
⟨*proof*⟩

There is a unique real infinitely close

lemma *st-part-Ex1*: $x \in HFinite ==> EX!\ t::hypreal.\ t \in Reals\ \&\ x\ @= t$
⟨*proof*⟩

8.9 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* | $x \in$ *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* | $x \in$ *HFinite* − *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-inverse*:
 fixes x :: *'a::real-normed-div-algebra star*
 shows [| $x \in$ *HFinite*; $x \notin$ *Infinitesimal* |] $==>$ *inverse* $x \in$ *HFinite*
⟨*proof*⟩

lemma *HFinite-inverse2*:
 fixes x :: *'a::real-normed-div-algebra star*
 shows $x \in$ *HFinite* − *Infinitesimal* $==>$ *inverse* $x \in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-inverse-HFinite*:
 fixes x :: *'a::real-normed-div-algebra star*
 shows $x \notin$ *Infinitesimal* $==>$ *inverse*(x) $\in$ *HFinite*
⟨*proof*⟩

lemma *HFinite-not-Infinitesimal-inverse*:
 fixes x :: *'a::real-normed-div-algebra star*
 shows $x \in$ *HFinite* − *Infinitesimal* $==>$ *inverse* $x \in$ *HFinite* − *Infinitesimal*
⟨*proof*⟩

lemma *approx-inverse*:
 fixes x y :: *'a::real-normed-div-algebra star*
 shows
  [| x @= y; $y \in$ *HFinite* − *Infinitesimal* |]
  $==>$ *inverse* x @= *inverse* y
⟨*proof*⟩

lemmas *star-of-approx-inverse* = *star-of-HFinite-diff-Infinitesimal* [*THEN* [*2*] *approx-inverse*]
lemmas *hypreal-of-real-approx-inverse* = *hypreal-of-real-HFinite-diff-Infinitesimal*
[*THEN* [*2*] *approx-inverse*]

lemma *inverse-add-Infinitesimal-approx*:
fixes *x h* :: *'a::real-normed-div-algebra star*
shows
[| *x* ∈ *HFinite* − *Infinitesimal*;
h ∈ *Infinitesimal* |] ==> *inverse*(*x* + *h*) @= *inverse x*
⟨*proof*⟩

lemma *inverse-add-Infinitesimal-approx2*:
fixes *x h* :: *'a::real-normed-div-algebra star*
shows
[| *x* ∈ *HFinite* − *Infinitesimal*;
h ∈ *Infinitesimal* |] ==> *inverse*(*h* + *x*) @= *inverse x*
⟨*proof*⟩

lemma *inverse-add-Infinitesimal-approx-Infinitesimal*:
fixes *x h* :: *'a::real-normed-div-algebra star*
shows
[| *x* ∈ *HFinite* − *Infinitesimal*;
h ∈ *Infinitesimal* |] ==> *inverse*(*x* + *h*) − *inverse x* @= *h*
⟨*proof*⟩

lemma *Infinitesimal-square-iff*:
fixes *x* :: *'a::real-normed-div-algebra star*
shows (*x* ∈ *Infinitesimal*) = (*x*∗*x* ∈ *Infinitesimal*)
⟨*proof*⟩
declare *Infinitesimal-square-iff* [*symmetric*, *simp*]

lemma *HFinite-square-iff* [*simp*]:
fixes *x* :: *'a::real-normed-div-algebra star*
shows (*x*∗*x* ∈ *HFinite*) = (*x* ∈ *HFinite*)
⟨*proof*⟩

lemma *HInfinite-square-iff* [*simp*]:
fixes *x* :: *'a::real-normed-div-algebra star*
shows (*x*∗*x* ∈ *HInfinite*) = (*x* ∈ *HInfinite*)
⟨*proof*⟩

lemma *approx-HFinite-mult-cancel*:
fixes *a w z* :: *'a::real-normed-div-algebra star*
shows [| *a*: *HFinite*−*Infinitesimal*; *a*∗ *w* @= *a*∗*z* |] ==> *w* @= *z*
⟨*proof*⟩

lemma *approx-HFinite-mult-cancel-iff1*:
fixes *a w z* :: *'a::real-normed-div-algebra star*
shows *a*: *HFinite*−*Infinitesimal* ==> (*a* ∗ *w* @= *a* ∗ *z*) = (*w* @= *z*)
⟨*proof*⟩

lemma *HInfinite-HFinite-add-cancel*:

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

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

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

lemma *Infinitesimal-inverse-HInfinite*:
fixes x :: *'a::real-normed-div-algebra star*
shows [| $x \in$ *Infinitesimal*; $x \neq 0$ |] ==> *inverse* $x \in$ *HInfinite*
⟨*proof*⟩

lemma *HInfinite-HFinite-not-Infinitesimal-mult*:
fixes x y :: *'a::real-normed-div-algebra star*
shows [| $x \in$ *HInfinite*; $y \in$ *HFinite* $-$ *Infinitesimal* |]
==> $x * y \in$ *HInfinite*
⟨*proof*⟩

lemma *HInfinite-HFinite-not-Infinitesimal-mult2*:
fixes x y :: *'a::real-normed-div-algebra star*
shows [| $x \in$ *HInfinite*; $y \in$ *HFinite* $-$ *Infinitesimal* |]
==> $y * x \in$ *HInfinite*
⟨*proof*⟩

lemma *HInfinite-gt-SReal*:
[| (x::*hypreal*) $\in$ *HInfinite*; $0 < x$; $y \in$ *Reals* |] ==> $y < x$
⟨*proof*⟩

lemma *HInfinite-gt-zero-gt-one*:
[| (x::*hypreal*) $\in$ *HInfinite*; $0 < x$ |] ==> $1 < x$
⟨*proof*⟩

lemma *not-HInfinite-one* [*simp*]: $1 \notin$ *HInfinite*
⟨*proof*⟩

lemma *approx-hrabs-disj*: *abs* (x::*hypreal*) @= x | *abs* x @= $-x$
⟨*proof*⟩

8.10 Theorems about Monads

lemma *monad-hrabs-Un-subset*: *monad* (*abs* x) $\le$ *monad*(x::*hypreal*) *Un* *monad*($-x$)
⟨*proof*⟩

lemma *Infinitesimal-monad-eq*: $e \in$ *Infinitesimal* ==> *monad* ($x+e$) = *monad* x

⟨*proof*⟩

lemma *mem-monad-iff*: $(u \in monad\ x) = (-u \in monad\ (-x))$
⟨*proof*⟩

lemma *Infinitesimal-monad-zero-iff*: $(x \in Infinitesimal) = (x \in monad\ 0)$
⟨*proof*⟩

lemma *monad-zero-minus-iff*: $(x \in monad\ 0) = (-x \in monad\ 0)$
⟨*proof*⟩

lemma *monad-zero-hrabs-iff*: $((x::hypreal) \in monad\ 0) = (abs\ x \in monad\ 0)$
⟨*proof*⟩

lemma *mem-monad-self* [*simp*]: $x \in monad\ x$
⟨*proof*⟩

8.11 Proof that $x \approx y$ implies $|x| \approx |y|$

lemma *approx-subset-monad*: x @= y ==> $\{x,y\} \leq monad\ x$
⟨*proof*⟩

lemma *approx-subset-monad2*: x @= y ==> $\{x,y\} \leq monad\ y$
⟨*proof*⟩

lemma *mem-monad-approx*: $u \in monad\ x$ ==> x @= u
⟨*proof*⟩

lemma *approx-mem-monad*: x @= u ==> $u \in monad\ x$
⟨*proof*⟩

lemma *approx-mem-monad2*: x @= u ==> $x \in monad\ u$
⟨*proof*⟩

lemma *approx-mem-monad-zero*: [| x @= y;$x \in monad\ 0$ |] ==> $y \in monad\ 0$
⟨*proof*⟩

lemma *Infinitesimal-approx-hrabs*:
[| x @= y; $(x::hypreal) \in Infinitesimal$ |] ==> $abs\ x$ @= $abs\ y$
⟨*proof*⟩

lemma *less-Infinitesimal-less*:
[| $0 < x$; $(x::hypreal) \notin Infinitesimal$; e :*Infinitesimal* |] ==> $e < x$
⟨*proof*⟩

lemma *Ball-mem-monad-gt-zero*:
[| $0 < (x::hypreal)$; $x \notin Infinitesimal$; $u \in monad\ x$ |] ==> $0 < u$
⟨*proof*⟩

lemma *Ball-mem-monad-less-zero*:
  [| (x::hypreal) < 0; x ∉ Infinitesimal; u ∈ monad x |] ==> u < 0
⟨proof⟩

lemma *lemma-approx-gt-zero*:
  [|0 < (x::hypreal); x ∉ Infinitesimal; x @= y|] ==> 0 < y
⟨proof⟩

lemma *lemma-approx-less-zero*:
  [|(x::hypreal) < 0; x ∉ Infinitesimal; x @= y|] ==> y < 0
⟨proof⟩

theorem *approx-hrabs*: (x::hypreal) @= y ==> abs x @= abs y
⟨proof⟩

lemma *approx-hrabs-zero-cancel*: abs(x::hypreal) @= 0 ==> x @= 0
⟨proof⟩

lemma *approx-hrabs-add-Infinitesimal*:
 (e::hypreal) ∈ Infinitesimal ==> abs x @= abs(x+e)
⟨proof⟩

lemma *approx-hrabs-add-minus-Infinitesimal*:
  (e::hypreal) ∈ Infinitesimal ==> abs x @= abs(x + −e)
⟨proof⟩

lemma *hrabs-add-Infinitesimal-cancel*:
  [| (e::hypreal) ∈ Infinitesimal; e′ ∈ Infinitesimal;
    abs(x+e) = abs(y+e′)|] ==> abs x @= abs y
⟨proof⟩

lemma *hrabs-add-minus-Infinitesimal-cancel*:
  [| (e::hypreal) ∈ Infinitesimal; e′ ∈ Infinitesimal;
    abs(x + −e) = abs(y + −e′)|] ==> abs x @= abs y
⟨proof⟩

8.12 More *HFinite* and *Infinitesimal* Theorems

lemma *Infinitesimal-add-hypreal-of-real-less*:
  [| x < y; u ∈ Infinitesimal |]
   ==> hypreal-of-real x + u < hypreal-of-real y
⟨proof⟩

lemma *Infinitesimal-add-hrabs-hypreal-of-real-less*:
  [| x ∈ Infinitesimal; abs(hypreal-of-real r) < hypreal-of-real y |]
   ==> abs (hypreal-of-real r + x) < hypreal-of-real y
⟨proof⟩

lemma *Infinitesimal-add-hrabs-hypreal-of-real-less2*:

[| $x \in$ *Infinitesimal*; *abs*(*hypreal-of-real* r) $<$ *hypreal-of-real* y |]
==> *abs* (x + *hypreal-of-real* r) $<$ *hypreal-of-real* y
⟨*proof*⟩

lemma *hypreal-of-real-le-add-Infininitesimal-cancel*:
[| $u \in$ *Infinitesimal*; $v \in$ *Infinitesimal*;
hypreal-of-real x + $u \leq$ *hypreal-of-real* y + v |]
==> *hypreal-of-real* $x \leq$ *hypreal-of-real* y
⟨*proof*⟩

lemma *hypreal-of-real-le-add-Infininitesimal-cancel2*:
[| $u \in$ *Infinitesimal*; $v \in$ *Infinitesimal*;
hypreal-of-real x + $u \leq$ *hypreal-of-real* y + v |]
==> $x \leq y$
⟨*proof*⟩

lemma *hypreal-of-real-less-Infinitesimal-le-zero*:
[| *hypreal-of-real* x $<$ e; $e \in$ *Infinitesimal* |] ==> *hypreal-of-real* $x \leq$ *0*
⟨*proof*⟩

lemma *Infinitesimal-add-not-zero*:
[| $h \in$ *Infinitesimal*; $x \neq$ *0* |] ==> *star-of* x + $h \neq$ *0*
⟨*proof*⟩

lemma *Infinitesimal-square-cancel* [*simp*]:
(x::*hypreal*)*x + y*y $\in$ *Infinitesimal* ==> x*x $\in$ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-square-cancel* [*simp*]:
(x::*hypreal*)*x + y*y $\in$ *HFinite* ==> x*x $\in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-square-cancel2* [*simp*]:
(x::*hypreal*)*x + y*y $\in$ *Infinitesimal* ==> y*y $\in$ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-square-cancel2* [*simp*]:
(x::*hypreal*)*x + y*y $\in$ *HFinite* ==> y*y $\in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-sum-square-cancel* [*simp*]:
(x::*hypreal*)*x + y*y + z*z $\in$ *Infinitesimal* ==> x*x $\in$ *Infinitesimal*
⟨*proof*⟩

lemma *HFinite-sum-square-cancel* [*simp*]:
(x::*hypreal*)*x + y*y + z*z $\in$ *HFinite* ==> x*x $\in$ *HFinite*
⟨*proof*⟩

lemma *Infinitesimal-sum-square-cancel2* [*simp*]:
$(y::hypreal)*y + x*x + z*z \in Infinitesimal ==> x*x \in Infinitesimal$
⟨*proof*⟩

lemma *HFinite-sum-square-cancel2* [*simp*]:
$(y::hypreal)*y + x*x + z*z \in HFinite ==> x*x \in HFinite$
⟨*proof*⟩

lemma *Infinitesimal-sum-square-cancel3* [*simp*]:
$(z::hypreal)*z + y*y + x*x \in Infinitesimal ==> x*x \in Infinitesimal$
⟨*proof*⟩

lemma *HFinite-sum-square-cancel3* [*simp*]:
$(z::hypreal)*z + y*y + x*x \in HFinite ==> x*x \in HFinite$
⟨*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*⟩

8.13 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-unique*: $[\![r \in \mathbb{R};\ r \approx x]\!] \Longrightarrow st\ x = r$
⟨*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*: $[\![x \in HFinite; y \in HFinite]\!] \Longrightarrow st\ (x + y) = st\ x + st\ y$
⟨*proof*⟩

lemma *st-number-of* [*simp*]: st (*number-of* w) = *number-of* w
⟨*proof*⟩

lemma [*simp*]: $st\ 0 = 0\ st\ 1 = 1$
⟨*proof*⟩

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

lemma *st-diff*: $[\![x \in HFinite; y \in HFinite]\!] \Longrightarrow st\ (x - y) = st\ x - st\ y$
⟨*proof*⟩

lemma *st-mult*: $[\![x \in HFinite; y \in HFinite]\!] \Longrightarrow st\ (x * y) = st\ x * st\ y$
⟨*proof*⟩

lemma *st-Infinitesimal*: $x \in Infinitesimal$ ==> $st\ x = 0$
⟨*proof*⟩

lemma *st-not-Infinitesimal*: $st(x) \neq 0$ ==> $x \notin Infinitesimal$
⟨*proof*⟩

lemma *st-inverse*:
[| $x \in HFinite$; $st\ x \neq 0$ |]
==> $st(inverse\ x) = inverse\ (st\ x)$

⟨*proof*⟩

lemma *st-divide* [*simp*]:
$[| \ x \in HFinite; \ y \in HFinite; \ st \ y \neq 0 \ |]$
$==> st(x/y) = (st \ x) \ / \ (st \ y)$
⟨*proof*⟩

lemma *st-idempotent* [*simp*]: $x \in HFinite ==> st(st(x)) = st(x)$
⟨*proof*⟩

lemma *Infinitesimal-add-st-less*:
$[| \ x \in HFinite; \ y \in HFinite; \ u \in Infinitesimal; \ st \ x < st \ y \ |]$
$==> st \ x + u < st \ y$
⟨*proof*⟩

lemma *Infinitesimal-add-st-le-cancel*:
$[| \ x \in HFinite; \ y \in HFinite;$
$u \in Infinitesimal; \ st \ x \leq st \ y + u$
$|] ==> st \ x \leq st \ y$
⟨*proof*⟩

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

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

lemma *st-zero-ge*: $[| \ x \leq 0; \ \ x \in HFinite \ |] ==> st \ x \leq 0$
⟨*proof*⟩

lemma *st-hrabs*: $x \in HFinite ==> abs(st \ x) = st(abs \ x)$
⟨*proof*⟩

8.14 Alternative Definitions using Free Ultrafilter

8.14.1 *HFinite*

lemma *HFinite-FreeUltrafilterNat*:
$star\text{-}n \ X \in HFinite$
$==> \exists \, u. \ \{n. \ norm \ (X \ n) < u\} \in FreeUltrafilterNat$
⟨*proof*⟩

lemma *FreeUltrafilterNat-HFinite*:
$\exists \, u. \ \{n. \ norm \ (X \ n) < u\} \in FreeUltrafilterNat$
$==> \ star\text{-}n \ X \in HFinite$
⟨*proof*⟩

lemma *HFinite-FreeUltrafilterNat-iff*:
$(star\text{-}n \ X \in HFinite) = (\exists \, u. \ \{n. \ norm \ (X \ n) < u\} \in FreeUltrafilterNat)$
⟨*proof*⟩

8.14.2 *HInfinite*

lemma *lemma-Compl-eq*: $- \{n.\ u < norm\ (xa\ n)\} = \{n.\ norm\ (xa\ n) \leq u\}$
⟨*proof*⟩

lemma *lemma-Compl-eq2*: $- \{n.\ norm\ (xa\ n) < u\} = \{n.\ u \leq norm\ (xa\ n)\}$
⟨*proof*⟩

lemma *lemma-Int-eq1*:
$\{n.\ norm\ (xa\ n) \leq u\}\ Int\ \{n.\ u \leq norm\ (xa\ n)\}$
$= \{n.\ norm(xa\ n) = u\}$
⟨*proof*⟩

lemma *lemma-FreeUltrafilterNat-one*:
$\{n.\ norm\ (xa\ n) = u\} \leq \{n.\ norm\ (xa\ n) < u + (1{::}real)\}$
⟨*proof*⟩

lemma *FreeUltrafilterNat-const-Finite*:
$\{n.\ norm\ (X\ n) = u\} \in FreeUltrafilterNat ==> star\text{-}n\ X \in HFinite$
⟨*proof*⟩

lemma *HInfinite-FreeUltrafilterNat*:
$star\text{-}n\ X \in HInfinite ==> \forall u.\ \{n.\ u < norm\ (X\ n)\} \in FreeUltrafilterNat$
⟨*proof*⟩

lemma *lemma-Int-HI*:
$\{n.\ norm\ (Xa\ n) < u\}\ Int\ \{n.\ X\ n = Xa\ n\} \subseteq \{n.\ norm\ (X\ n) < (u{::}real)\}$
⟨*proof*⟩

lemma *lemma-Int-HIa*: $\{n.\ u < norm\ (X\ n)\}\ Int\ \{n.\ norm\ (X\ n) < u\} = \{\}$
⟨*proof*⟩

lemma *FreeUltrafilterNat-HInfinite*:
$\forall u.\ \{n.\ u < norm\ (X\ n)\} \in FreeUltrafilterNat ==> star\text{-}n\ X \in HInfinite$
⟨*proof*⟩

lemma *HInfinite-FreeUltrafilterNat-iff*:
$(star\text{-}n\ X \in HInfinite) = (\forall u.\ \{n.\ u < norm\ (X\ n)\} \in FreeUltrafilterNat)$
⟨*proof*⟩

8.14.3 *Infinitesimal*

lemma *ball-SReal-eq*: $(\forall x{::}hypreal \in Reals.\ P\ x) = (\forall x{::}real.\ P\ (star\text{-}of\ x))$
⟨*proof*⟩

lemma *Infinitesimal-FreeUltrafilterNat*:
$star\text{-}n\ X \in Infinitesimal ==> \forall u{>}0.\ \{n.\ norm\ (X\ n) < u\} \in \mathcal{U}$
⟨*proof*⟩

lemma *FreeUltrafilterNat-Infinitesimal*:
$\forall\, u{>}0.\ \{n.\ norm\ (X\ n) < u\} \in \mathcal{U} \Longrightarrow star\text{-}n\ X \in Infinitesimal$
⟨*proof*⟩

lemma *Infinitesimal-FreeUltrafilterNat-iff*:
$(star\text{-}n\ X \in Infinitesimal) = (\forall\, u{>}0.\ \{n.\ norm\ (X\ n) < u\} \in \mathcal{U})$
⟨*proof*⟩

lemma *lemma-Infinitesimal*:
$(\forall\, r.\ 0 < r \dashrightarrow x < r) = (\forall\, n.\ x < inverse(real\ (Suc\ n)))$
⟨*proof*⟩

lemma *lemma-Infinitesimal2*:
$(\forall\, r \in Reals.\ 0 < r \dashrightarrow x < r) =$
$(\forall\, n.\ x < inverse(hypreal\text{-}of\text{-}nat\ (Suc\ n)))$
⟨*proof*⟩

lemma *Infinitesimal-hypreal-of-nat-iff*:
$Infinitesimal = \{x.\ \forall\, n.\ hnorm\ x < inverse\ (hypreal\text{-}of\text{-}nat\ (Suc\ n))\}$
⟨*proof*⟩

8.15 Proof that ω is an infinite number

It will follow that epsilon is an infinitesimal number.

lemma *Suc-Un-eq*: $\{n.\ n < Suc\ m\} = \{n.\ n < m\}\ Un\ \{n.\ n = m\}$
⟨*proof*⟩

lemma *finite-nat-segment*: *finite* $\{n{::}nat.\ n < m\}$
⟨*proof*⟩

lemma *finite-real-of-nat-segment*: *finite* $\{n{::}nat.\ real\ n < real\ (m{::}nat)\}$
⟨*proof*⟩

lemma *finite-real-of-nat-less-real*: *finite* $\{n{::}nat.\ real\ n < u\}$
⟨*proof*⟩

lemma *lemma-real-le-Un-eq*:
$\{n.\ f\ n \le u\} = \{n.\ f\ n < u\}\ Un\ \{n.\ u = (f\ n :: real)\}$
⟨*proof*⟩

lemma *finite-real-of-nat-le-real*: *finite* $\{n{::}nat.\ real\ n \le u\}$
⟨*proof*⟩

lemma *finite-rabs-real-of-nat-le-real*: *finite* $\{n{::}nat.\ abs(real\ n) \le u\}$
⟨*proof*⟩

lemma *rabs-real-of-nat-le-real-FreeUltrafilterNat*:
$\{n.\ abs(real\ n) \leq u\} \notin FreeUltrafilterNat$
⟨*proof*⟩

lemma *FreeUltrafilterNat-nat-gt-real*: $\{n.\ u < real\ n\} \in FreeUltrafilterNat$
⟨*proof*⟩

lemma *Compl-real-le-eq*: $-\ \{n{::}nat.\ real\ n \leq u\} = \{n.\ u < real\ n\}$
⟨*proof*⟩

ω is a member of *HInfinite*

lemma *FreeUltrafilterNat-omega*: $\{n.\ u < real\ n\} \in FreeUltrafilterNat$
⟨*proof*⟩

theorem *HInfinite-omega* [*simp*]: $omega \in HInfinite$
⟨*proof*⟩

lemma *Infinitesimal-epsilon* [*simp*]: $epsilon \in Infinitesimal$
⟨*proof*⟩

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

lemma *epsilon-approx-zero* [*simp*]: $epsilon\ @{=}\ 0$
⟨*proof*⟩

lemma *real-of-nat-less-inverse-iff*:
$0 < u\ \ {=}{=}{>}\ (u < inverse\ (real(Suc\ n))) = (real(Suc\ n) < inverse\ u)$
⟨*proof*⟩

lemma *finite-inverse-real-of-posnat-gt-real*:
$0 < u\ {=}{=}{>}\ finite\ \{n.\ u < inverse(real(Suc\ n))\}$
⟨*proof*⟩

lemma *lemma-real-le-Un-eq2*:
$\{n.\ u \leq inverse(real(Suc\ n))\} =$
$\{n.\ u < inverse(real(Suc\ n))\}\ Un\ \{n.\ u = inverse(real(Suc\ n))\}$
⟨*proof*⟩

lemma *real-of-nat-inverse-eq-iff*:
$(u = inverse\ (real(Suc\ n))) = (real(Suc\ n) = inverse\ u)$
⟨*proof*⟩

lemma *lemma-finite-omega-set2*: *finite* $\{n::nat.\ u = inverse(real(Suc\ n))\}$
⟨*proof*⟩

lemma *finite-inverse-real-of-posnat-ge-real*:
$0 < u ==> finite\ \{n.\ u \leq inverse(real(Suc\ n))\}$
⟨*proof*⟩

lemma *inverse-real-of-posnat-ge-real-FreeUltrafilterNat*:
$0 < u ==> \{n.\ u \leq inverse(real(Suc\ n))\} \notin FreeUltrafilterNat$
⟨*proof*⟩

lemma *Compl-le-inverse-eq*:
$- \{n.\ u \leq inverse(real(Suc\ n))\} =$
$\{n.\ inverse(real(Suc\ n)) < u\}$
⟨*proof*⟩

lemma *FreeUltrafilterNat-inverse-real-of-posnat*:
$0 < u ==>$
$\{n.\ inverse(real(Suc\ n)) < u\} \in FreeUltrafilterNat$
⟨*proof*⟩

Example of an hypersequence (i.e. an extended standard sequence) whose term with an hypernatural suffix is an infinitesimal i.e. the whn'nth term of the hypersequence is a member of Infinitesimal

lemma *SEQ-Infinitesimal*:
$(\ *f*\ (\%n::nat.\ inverse(real(Suc\ n))))\ whn : Infinitesimal$
⟨*proof*⟩

Example where we get a hyperreal from a real sequence for which a particular property holds. The theorem is used in proofs about equivalence of nonstandard and standard neighbourhoods. Also used for equivalence of nonstandard ans standard definitions of pointwise limit.

lemma *real-seq-to-hypreal-Infinitesimal*:
$\forall n.\ norm(X\ n - x) < inverse(real(Suc\ n))$
$==> star\text{-}n\ X - star\text{-}of\ x \in Infinitesimal$
⟨*proof*⟩

lemma *real-seq-to-hypreal-approx*:
$\forall n.\ norm(X\ n - x) < inverse(real(Suc\ n))$
$==> star\text{-}n\ X\ @= star\text{-}of\ x$
⟨*proof*⟩

lemma *real-seq-to-hypreal-approx2*:
$\forall n.\ norm(x - X\ n) < inverse(real(Suc\ n))$
$==> star\text{-}n\ X\ @= star\text{-}of\ x$
⟨*proof*⟩

lemma *real-seq-to-hypreal-Infinitesimal2*:
  $\forall\, n.\ norm(X\ n\ -\ Y\ n) < inverse(real(Suc\ n))$
  $==> star\text{-}n\ X\ -\ star\text{-}n\ Y \in Infinitesimal$
⟨*proof*⟩

end

9 NSComplex: Nonstandard Complex Numbers

theory *NSComplex*
imports *Complex ../Hyperreal/NSA*
begin

types *hcomplex = complex star*

abbreviation
  hcomplex-of-complex :: *complex => complex star* **where**
  hcomplex-of-complex == star-of

abbreviation
  hcmod :: *complex star => real star* **where**
  hcmod == hnorm

definition
  hRe :: *hcomplex => hypreal* **where**
  [*code del*]: *hRe = *f* Re*

definition
  hIm :: *hcomplex => hypreal* **where**
  [*code del*]: *hIm = *f* Im*

definition
  iii :: *hcomplex* **where**
  iii = star-of ii

definition
  hcnj :: *hcomplex => hcomplex* **where**
  [*code del*]: *hcnj = *f* cnj*

definition
hsgn :: *hcomplex* => *hcomplex* **where**
[*code del*]: *hsgn* = **f** *sgn*

definition
harg :: *hcomplex* => *hypreal* **where**
[*code del*]: *harg* = **f** *arg*

definition

hcis :: *hypreal* => *hcomplex* **where**
[*code del*]:*hcis* = **f** *cis*

abbreviation
hcomplex-of-hypreal :: *hypreal* ⇒ *hcomplex* **where**
hcomplex-of-hypreal ≡ *of-hypreal*

definition

hrcis :: [*hypreal*, *hypreal*] => *hcomplex* **where**
[*code del*]: *hrcis* = **f2** *rcis*

definition
hexpi :: *hcomplex* => *hcomplex* **where**
[*code del*]: *hexpi* = **f** *expi*

definition
HComplex :: [*hypreal*,*hypreal*] => *hcomplex* **where**
[*code del*]: *HComplex* = **f2** *Complex*

lemmas *hcomplex-defs* [*transfer-unfold*] =
hRe-def hIm-def iii-def hcnj-def hsgn-def harg-def hcis-def
hrcis-def hexpi-def HComplex-def

lemma *Standard-hRe* [*simp*]: $x \in Standard \Longrightarrow hRe\ x \in Standard$
⟨*proof*⟩

lemma *Standard-hIm* [*simp*]: $x \in Standard \Longrightarrow hIm\ x \in Standard$
⟨*proof*⟩

lemma *Standard-iii* [*simp*]: $iii \in Standard$
⟨*proof*⟩

lemma *Standard-hcnj* [*simp*]: $x \in Standard \Longrightarrow hcnj\ x \in Standard$

⟨*proof*⟩

lemma *Standard-hsgn* [*simp*]: $x \in Standard \Longrightarrow hsgn\ x \in Standard$
⟨*proof*⟩

lemma *Standard-harg* [*simp*]: $x \in Standard \Longrightarrow harg\ x \in Standard$
⟨*proof*⟩

lemma *Standard-hcis* [*simp*]: $r \in Standard \Longrightarrow hcis\ r \in Standard$
⟨*proof*⟩

lemma *Standard-hexpi* [*simp*]: $x \in Standard \Longrightarrow hexpi\ x \in Standard$
⟨*proof*⟩

lemma *Standard-hrcis* [*simp*]:
$[\![r \in Standard; s \in Standard]\!] \Longrightarrow hrcis\ r\ s \in Standard$
⟨*proof*⟩

lemma *Standard-HComplex* [*simp*]:
$[\![r \in Standard; s \in Standard]\!] \Longrightarrow HComplex\ r\ s \in Standard$
⟨*proof*⟩

lemma *hcmod-def*: $hcmod = *f*\ cmod$
⟨*proof*⟩

9.1 Properties of Nonstandard Real and Imaginary Parts

lemma *hcomplex-hRe-hIm-cancel-iff*:
$!!w\ z.\ (w=z) = (hRe(w) = hRe(z)\ \&\ hIm(w) = hIm(z))$
⟨*proof*⟩

lemma *hcomplex-equality* [*intro?*]:
$!!z\ w.\ hRe\ z = hRe\ w ==> hIm\ z = hIm\ w ==> z = w$
⟨*proof*⟩

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

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

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

lemma *hcomplex-hIm-one* [*simp*]: $hIm\ 1 = 0$
⟨*proof*⟩

9.2 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⟩

9.3 More Minus Laws

lemma *hRe-minus*: $!!z.\ hRe(-z) = -\ hRe(z)$
⟨proof⟩

lemma *hIm-minus*: $!!z.\ hIm(-z) = -\ hIm(z)$
⟨proof⟩

lemma *hcomplex-add-minus-eq-minus*:
$x + y = (0::hcomplex) ==> x = -y$
⟨proof⟩

lemma *hcomplex-i-mult-eq* [*simp*]: $iii * iii = -\ 1$
⟨proof⟩

lemma *hcomplex-i-mult-left* [*simp*]: $!!z.\ iii * (iii * z) = -z$
⟨proof⟩

lemma *hcomplex-i-not-zero* [*simp*]: $iii \neq 0$
⟨proof⟩

9.4 More Multiplication Laws

lemma *hcomplex-mult-minus-one*: $-\ 1 * (z::hcomplex) = -z$
⟨proof⟩

lemma *hcomplex-mult-minus-one-right*: $(z::hcomplex) * -\ 1 = -z$
⟨proof⟩

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

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

9.5 Subraction and Division

lemma *hcomplex-diff-eq-eq* [*simp*]: $((x::hcomplex) - y = z) = (x = z + y)$

⟨proof⟩

9.6 Embedding Properties for *hcomplex-of-hypreal* Map

lemma *hRe-hcomplex-of-hypreal* [*simp*]: $!!z.\ hRe(hcomplex\text{-}of\text{-}hypreal\ z) = z$
⟨*proof*⟩

lemma *hIm-hcomplex-of-hypreal* [*simp*]: $!!z.\ hIm(hcomplex\text{-}of\text{-}hypreal\ z) = 0$
⟨*proof*⟩

lemma *hcomplex-of-hypreal-epsilon-not-zero* [*simp*]:
$hcomplex\text{-}of\text{-}hypreal\ epsilon \neq 0$
⟨*proof*⟩

9.7 HComplex theorems

lemma *hRe-HComplex* [*simp*]: $!!x\ y.\ hRe\ (HComplex\ x\ y) = x$
⟨*proof*⟩

lemma *hIm-HComplex* [*simp*]: $!!x\ y.\ hIm\ (HComplex\ x\ y) = y$
⟨*proof*⟩

lemma *hcomplex-surj* [*simp*]: $!!z.\ HComplex\ (hRe\ z)\ (hIm\ z) = z$
⟨*proof*⟩

lemma *hcomplex-induct* [*case-names rect*]:
$(\bigwedge x\ y.\ P\ (HComplex\ x\ y)) ==> P\ z$
⟨*proof*⟩

9.8 Modulus (Absolute Value) of Nonstandard Complex Number

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*]:
$!!x\ y\ r.\ HComplex\ x\ y + hcomplex\text{-}of\text{-}hypreal\ r = HComplex\ (x{+}r)\ y$
⟨*proof*⟩

lemma *hcomplex-of-hypreal-add-HComplex* [*simp*]:
$!!r\ x\ y.\ hcomplex\text{-}of\text{-}hypreal\ r + HComplex\ x\ y = HComplex\ (r{+}x)\ y$
⟨*proof*⟩

lemma *HComplex-mult-hcomplex-of-hypreal*:
$!!x\ y\ r.\ HComplex\ x\ y * hcomplex\text{-}of\text{-}hypreal\ r = HComplex\ (x{*}r)\ (y{*}r)$
⟨*proof*⟩

lemma *hcomplex-of-hypreal-mult-HComplex*:
$!!r\ x\ y.\ hcomplex\text{-}of\text{-}hypreal\ r * HComplex\ x\ y = HComplex\ (r{*}x)\ (r{*}y)$
⟨*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*]:
$!!r.\ hcomplex\text{-}of\text{-}hypreal\ r * iii = HComplex\ 0\ r$
⟨*proof*⟩

9.9 Conjugation

lemma *hcomplex-hcnj-cancel-iff* [*iff*]: $!!x\ y.\ (hcnj\ x = hcnj\ y) = (x = y)$
⟨*proof*⟩

lemma *hcomplex-hcnj-hcnj* [*simp*]: $!!z.\ hcnj\ (hcnj\ z) = z$
⟨*proof*⟩

lemma *hcomplex-hcnj-hcomplex-of-hypreal* [*simp*]:
$!!x.\ hcnj\ (hcomplex\text{-}of\text{-}hypreal\ x) = hcomplex\text{-}of\text{-}hypreal\ x$
⟨*proof*⟩

lemma *hcomplex-hmod-hcnj* [*simp*]: $!!z.\ hcmod\ (hcnj\ z) = hcmod\ z$
⟨*proof*⟩

lemma *hcomplex-hcnj-minus*: $!!z.\ hcnj\ (-z) = -\ hcnj\ z$
⟨*proof*⟩

lemma *hcomplex-hcnj-inverse*: $!!z.\ hcnj(inverse\ z) = inverse(hcnj\ z)$
⟨*proof*⟩

lemma *hcomplex-hcnj-add*: $!!w\ z.\ hcnj(w + z) = hcnj(w) + hcnj(z)$
⟨*proof*⟩

lemma *hcomplex-hcnj-diff*: $!!w\ z.\ hcnj(w - z) = hcnj(w) - hcnj(z)$
⟨*proof*⟩

lemma *hcomplex-hcnj-mult*: $!!w\ z.\ hcnj(w * z) = hcnj(w) * hcnj(z)$
⟨*proof*⟩

lemma *hcomplex-hcnj-divide*: $!!w\ z.\ hcnj(w\ /\ z) = (hcnj\ w)/(hcnj\ z)$
⟨*proof*⟩

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

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

lemma *hcomplex-hcnj-zero-iff* [*iff*]: $!!z.\ (hcnj\ z = 0) = (z = 0)$
⟨*proof*⟩

lemma *hcomplex-mult-hcnj*:
$!!z.\ z * hcnj\ z = hcomplex\text{-}of\text{-}hypreal\ (hRe(z)\ \hat{}\ 2 + hIm(z)\ \hat{}\ 2)$
⟨*proof*⟩

9.10 More Theorems about the Function *hcmod*

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

lemma *hcmod-hcomplex-of-hypreal-of-hypnat* [*simp*]:
$hcmod\ (hcomplex\text{-}of\text{-}hypreal(hypreal\text{-}of\text{-}hypnat\ n)) = hypreal\text{-}of\text{-}hypnat\ n$
⟨*proof*⟩

lemma *hcmod-mult-hcnj*: $!!z.\ hcmod(z * hcnj(z)) = hcmod(z)\ \hat{}\ 2$
⟨*proof*⟩

lemma *hcmod-triangle-ineq2* [*simp*]:
$!!a\ b.\ hcmod(b + a) - hcmod\ b \le hcmod\ a$
⟨*proof*⟩

lemma *hcmod-diff-ineq* [*simp*]: $!!a\ b.\ hcmod(a) - hcmod(b) \le hcmod(a + b)$

⟨*proof*⟩

9.11 Exponentiation

lemma *hcomplexpow-0* [*simp*]: $z \;\hat{}\; 0 \quad = (1::hcomplex)$
⟨*proof*⟩

lemma *hcomplexpow-Suc* [*simp*]: $z \;\hat{}\; (Suc\ n) = (z::hcomplex) * (z \;\hat{}\; n)$
⟨*proof*⟩

lemma *hcomplexpow-i-squared* [*simp*]: $iii \;\hat{}\; 2 = -1$
⟨*proof*⟩

lemma *hcomplex-of-hypreal-pow*:
$!!x.\ hcomplex\text{-}of\text{-}hypreal\ (x \;\hat{}\; n) = (hcomplex\text{-}of\text{-}hypreal\ x) \;\hat{}\; n$
⟨*proof*⟩

lemma *hcomplex-hcnj-pow*: $!!z.\ hcnj(z \;\hat{}\; n) = hcnj(z) \;\hat{}\; n$
⟨*proof*⟩

lemma *hcmod-hcomplexpow*: $!!x.\ hcmod(x \;\hat{}\; n) = hcmod(x) \;\hat{}\; n$
⟨*proof*⟩

lemma *hcpow-minus*:
$!!x\ n.\ (-x::hcomplex)\ pow\ n =$
$(if\ (\ *p*\ even)\ n\ then\ (x\ pow\ n)\ else\ -(x\ pow\ n))$
⟨*proof*⟩

lemma *hcpow-mult*:
$!!r\ s\ n.\ ((r::hcomplex) * s)\ pow\ n = (r\ pow\ n) * (s\ pow\ n)$
⟨*proof*⟩

lemma *hcpow-zero2* [*simp*]:
$\bigwedge n.\ 0\ pow\ (hSuc\ n) = (0::'a::\{recpower,semiring\text{-}0\}\ star)$
⟨*proof*⟩

lemma *hcpow-not-zero* [*simp,intro*]:
$!!r\ n.\ r \neq 0 ==> r\ pow\ n \neq (0::hcomplex)$
⟨*proof*⟩

lemma *hcpow-zero-zero*: $r\ pow\ n = (0::hcomplex) ==> r = 0$
⟨*proof*⟩

9.12 The Function *hsgn*

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

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

lemma *hsgn-minus*: $!!z.\ hsgn\ (-z) = -\ hsgn(z)$
⟨*proof*⟩

lemma *hsgn-eq*: $!!z.\ hsgn\ z = z\ /\ hcomplex\text{-}of\text{-}hypreal\ (hcmod\ z)$
⟨*proof*⟩

lemma *hcmod-i*: $!!x\ y.\ hcmod\ (HComplex\ x\ y) = (\ *f*\ sqrt)\ (x\ \hat{}\ 2 + y\ \hat{}\ 2)$
⟨*proof*⟩

lemma *hcomplex-eq-cancel-iff1* [*simp*]:
$(hcomplex\text{-}of\text{-}hypreal\ xa = HComplex\ x\ y) = (xa = x\ \&\ y = 0)$
⟨*proof*⟩

lemma *hcomplex-eq-cancel-iff2* [*simp*]:
$(HComplex\ x\ y = hcomplex\text{-}of\text{-}hypreal\ xa) = (x = xa\ \&\ y = 0)$
⟨*proof*⟩

lemma *HComplex-eq-0* [*simp*]: $!!x\ y.\ (HComplex\ x\ y = 0) = (x = 0\ \&\ y = 0)$
⟨*proof*⟩

lemma *HComplex-eq-1* [*simp*]: $!!x\ y.\ (HComplex\ x\ y = 1) = (x = 1\ \&\ y = 0)$
⟨*proof*⟩

lemma *i-eq-HComplex-0-1*: $iii = HComplex\ 0\ 1$
⟨*proof*⟩

lemma *HComplex-eq-i* [*simp*]: $!!x\ y.\ (HComplex\ x\ y = iii) = (x = 0\ \&\ y = 1)$
⟨*proof*⟩

lemma *hRe-hsgn* [*simp*]: $!!z.\ hRe(hsgn\ z) = hRe(z)/hcmod\ z$
⟨*proof*⟩

lemma *hIm-hsgn* [*simp*]: $!!z.\ hIm(hsgn\ z) = hIm(z)/hcmod\ z$
⟨*proof*⟩

lemma *hcomplex-inverse-complex-split*:
$!!x\ y.\ inverse(hcomplex\text{-}of\text{-}hypreal\ x + iii * hcomplex\text{-}of\text{-}hypreal\ y) =$
$hcomplex\text{-}of\text{-}hypreal(x/(x\ \hat{}\ 2 + y\ \hat{}\ 2)) -$
$iii * hcomplex\text{-}of\text{-}hypreal(y/(x\ \hat{}\ 2 + y\ \hat{}\ 2))$
⟨*proof*⟩

lemma *HComplex-inverse*:
$!!x\ y.\ inverse\ (HComplex\ x\ y) =$
$HComplex\ (x/(x\ \hat{}\ 2 + y\ \hat{}\ 2))\ (-y/(x\ \hat{}\ 2 + y\ \hat{}\ 2))$
⟨*proof*⟩

lemma *hRe-mult-i-eq*[*simp*]:
$!!y.\ hRe\ (iii * hcomplex\text{-}of\text{-}hypreal\ y) = 0$

⟨*proof*⟩

lemma *hIm-mult-i-eq* [*simp*]:
!!*y*. *hIm* (*iii* ∗ *hcomplex-of-hypreal y*) = *y*
⟨*proof*⟩

lemma *hcmod-mult-i* [*simp*]: !!*y*. *hcmod* (*iii* ∗ *hcomplex-of-hypreal y*) = *abs y*
⟨*proof*⟩

lemma *hcmod-mult-i2* [*simp*]: !!*y*. *hcmod* (*hcomplex-of-hypreal y* ∗ *iii*) = *abs y*
⟨*proof*⟩

lemma *cos-harg-i-mult-zero-pos*:
!!*y*. *0* < *y* ==> (∗*f*∗ *cos*) (*harg*(*HComplex 0 y*)) = *0*
⟨*proof*⟩

lemma *cos-harg-i-mult-zero-neg*:
!!*y*. *y* < *0* ==> (∗*f*∗ *cos*) (*harg*(*HComplex 0 y*)) = *0*
⟨*proof*⟩

lemma *cos-harg-i-mult-zero* [*simp*]:
!!*y*. *y* ≠ *0* ==> (∗*f*∗ *cos*) (*harg*(*HComplex 0 y*)) = *0*
⟨*proof*⟩

lemma *hcomplex-of-hypreal-zero-iff* [*simp*]:
!!*y*. (*hcomplex-of-hypreal y* = *0*) = (*y* = *0*)
⟨*proof*⟩

9.13 Polar Form for Nonstandard Complex Numbers

lemma *complex-split-polar2*:
∀ *n*. ∃ *r a*. (*z n*) = *complex-of-real r* ∗ (*Complex* (*cos a*) (*sin a*))
⟨*proof*⟩

lemma *hcomplex-split-polar*:
!!*z*. ∃ *r a*. *z* = *hcomplex-of-hypreal r* ∗ (*HComplex*((∗*f*∗ *cos*) *a*)((∗*f*∗ *sin*) *a*))
⟨*proof*⟩

lemma *hcis-eq*:
!!*a*. *hcis a* =
(*hcomplex-of-hypreal*((∗*f*∗ *cos*) *a*) +
iii ∗ *hcomplex-of-hypreal*((∗*f*∗ *sin*) *a*))
⟨*proof*⟩

lemma *hrcis-Ex*: !!*z*. ∃ *r a*. *z* = *hrcis r a*

⟨*proof*⟩

lemma *hRe-hcomplex-polar* [*simp*]:
!!*r a*. *hRe* (*hcomplex-of-hypreal r* ∗ *HComplex* ((∗*f*∗ *cos*) *a*) ((∗*f*∗ *sin*) *a*)) =
r ∗ (∗*f*∗ *cos*) *a*
⟨*proof*⟩

lemma *hRe-hrcis* [*simp*]: !!*r a*. *hRe*(*hrcis r a*) = *r* ∗ (∗*f*∗ *cos*) *a*
⟨*proof*⟩

lemma *hIm-hcomplex-polar* [*simp*]:
!!*r a*. *hIm* (*hcomplex-of-hypreal r* ∗ *HComplex* ((∗*f*∗ *cos*) *a*) ((∗*f*∗ *sin*) *a*)) =
r ∗ (∗*f*∗ *sin*) *a*
⟨*proof*⟩

lemma *hIm-hrcis* [*simp*]: !!*r a*. *hIm*(*hrcis r a*) = *r* ∗ (∗*f*∗ *sin*) *a*
⟨*proof*⟩

lemma *hcmod-unit-one* [*simp*]:
!!*a*. *hcmod* (*HComplex* ((∗*f*∗ *cos*) *a*) ((∗*f*∗ *sin*) *a*)) = *1*
⟨*proof*⟩

lemma *hcmod-complex-polar* [*simp*]:
!!*r a*. *hcmod* (*hcomplex-of-hypreal r* ∗ *HComplex* ((∗*f*∗ *cos*) *a*) ((∗*f*∗ *sin*) *a*)) =
abs r
⟨*proof*⟩

lemma *hcmod-hrcis* [*simp*]: !!*r a*. *hcmod*(*hrcis r a*) = *abs r*
⟨*proof*⟩

lemma *hcis-hrcis-eq*: !!*a*. *hcis a* = *hrcis 1 a*
⟨*proof*⟩
declare *hcis-hrcis-eq* [*symmetric*, *simp*]

lemma *hrcis-mult*:
!!*a b r1 r2*. *hrcis r1 a* ∗ *hrcis r2 b* = *hrcis* (*r1*∗*r2*) (*a* + *b*)
⟨*proof*⟩

lemma *hcis-mult*: !!*a b*. *hcis a* ∗ *hcis b* = *hcis* (*a* + *b*)
⟨*proof*⟩

lemma *hcis-zero* [*simp*]: *hcis 0* = *1*
⟨*proof*⟩

lemma *hrcis-zero-mod* [*simp*]: !!*a*. *hrcis 0 a* = *0*

⟨*proof*⟩

lemma *hrcis-zero-arg* [*simp*]: $!!r.\ hrcis\ r\ 0 = hcomplex\text{-}of\text{-}hypreal\ r$
⟨*proof*⟩

lemma *hcomplex-i-mult-minus* [*simp*]: $!!x.\ iii * (iii * x) = -\ x$
⟨*proof*⟩

lemma *hcomplex-i-mult-minus2* [*simp*]: $iii * iii * x = -\ x$
⟨*proof*⟩

lemma *hcis-hypreal-of-nat-Suc-mult*:
$!!a.\ hcis\ (hypreal\text{-}of\text{-}nat\ (Suc\ n) * a) =$
$hcis\ a * hcis\ (hypreal\text{-}of\text{-}nat\ n * a)$
⟨*proof*⟩

lemma *NSDeMoivre*: $!!a.\ (hcis\ a)\ \hat{}\ n = hcis\ (hypreal\text{-}of\text{-}nat\ n * a)$
⟨*proof*⟩

lemma *hcis-hypreal-of-hypnat-Suc-mult*:
$!!\ a\ n.\ hcis\ (hypreal\text{-}of\text{-}hypnat\ (n + 1) * a) =$
$hcis\ a * hcis\ (hypreal\text{-}of\text{-}hypnat\ n * a)$
⟨*proof*⟩

lemma *NSDeMoivre-ext*:
$!!a\ n.\ (hcis\ a)\ pow\ n = hcis\ (hypreal\text{-}of\text{-}hypnat\ n * a)$
⟨*proof*⟩

lemma *NSDeMoivre2*:
$!!a\ r.\ (hrcis\ r\ a)\ \hat{}\ n = hrcis\ (r\ \hat{}\ n)\ (hypreal\text{-}of\text{-}nat\ n * a)$
⟨*proof*⟩

lemma *DeMoivre2-ext*:
$!!\ a\ r\ n.\ (hrcis\ r\ a)\ pow\ n = hrcis\ (r\ pow\ n)\ (hypreal\text{-}of\text{-}hypnat\ n * a)$
⟨*proof*⟩

lemma *hcis-inverse* [*simp*]: $!!a.\ inverse(hcis\ a) = hcis\ (-a)$
⟨*proof*⟩

lemma *hrcis-inverse*: $!!a\ r.\ inverse(hrcis\ r\ a) = hrcis\ (inverse\ r)\ (-a)$
⟨*proof*⟩

lemma *hRe-hcis* [*simp*]: $!!a.\ hRe(hcis\ a) = (\ *f*\ cos)\ a$
⟨*proof*⟩

lemma *hIm-hcis* [*simp*]: $!!a.\ hIm(hcis\ a) = (\ *f*\ sin)\ a$
⟨*proof*⟩

lemma *cos-n-hRe-hcis-pow-n*: $(\ *f*\ cos)\ (hypreal\text{-}of\text{-}nat\ n * a) = hRe(hcis\ a\ \hat{}\ n)$

⟨*proof*⟩

lemma *sin-n-hIm-hcis-pow-n*: $(*f* \ sin) \ (hypreal\text{-}of\text{-}nat \ n * a) = hIm(hcis \ a \ \hat{} \ n)$
⟨*proof*⟩

lemma *cos-n-hRe-hcis-hcpow-n*: $(*f* \ cos) \ (hypreal\text{-}of\text{-}hypnat \ n * a) = hRe(hcis \ a \ pow \ n)$
⟨*proof*⟩

lemma *sin-n-hIm-hcis-hcpow-n*: $(*f* \ sin) \ (hypreal\text{-}of\text{-}hypnat \ n * a) = hIm(hcis \ a \ pow \ n)$
⟨*proof*⟩

lemma *hexpi-add*: $!!a \ b. \ hexpi(a + b) = hexpi(a) * hexpi(b)$
⟨*proof*⟩

9.14 *hcomplex-of-complex*: the Injection from type *complex* to to *hcomplex*

lemma *inj-hcomplex-of-complex*: $inj(hcomplex\text{-}of\text{-}complex)$

⟨*proof*⟩

lemma *hcomplex-of-complex-i*: $iii = hcomplex\text{-}of\text{-}complex \ ii$
⟨*proof*⟩

lemma *hRe-hcomplex-of-complex*:
$hRe \ (hcomplex\text{-}of\text{-}complex \ z) = hypreal\text{-}of\text{-}real \ (Re \ z)$
⟨*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*:
$hcmod \ (hcomplex\text{-}of\text{-}complex \ x) = hypreal\text{-}of\text{-}real \ (cmod \ x)$
⟨*proof*⟩

9.15 Numerals and Arithmetic

lemma *hcomplex-number-of-def*: $(number\text{-}of \ w :: hcomplex) == of\text{-}int \ w$
⟨*proof*⟩

lemma *hcomplex-of-hypreal-eq-hcomplex-of-complex*:
$hcomplex\text{-}of\text{-}hypreal \ (hypreal\text{-}of\text{-}real \ x) =$
$hcomplex\text{-}of\text{-}complex \ (complex\text{-}of\text{-}real \ x)$
⟨*proof*⟩

lemma *hcomplex-hypreal-number-of*:

hcomplex-of-complex (*number-of* w) = *hcomplex-of-hypreal*(*number-of* w)
⟨*proof*⟩

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

lemma *hcomplex-number-of-hcmod* [*simp*]:
hcmod(*number-of* v :: *hcomplex*) = *abs* (*number-of* v :: *hypreal*)
⟨*proof*⟩

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

lemma *hcomplex-number-of-hIm* [*simp*]:
hIm(*number-of* v :: *hcomplex*) = *0*
⟨*proof*⟩

end

10 Star: Star-Transforms in Non-Standard Analysis

theory *Star*
imports *NSA*
begin

definition

starset-n :: (*nat* => *'a set*) => *'a star set* (*∗sn∗* - [*80*] *80*) **where**
∗sn∗ As = *Iset* (*star-n As*)

definition
InternalSets :: *'a star set set* **where**
[*code del*]: *InternalSets* = $\{X.\ \exists As.\ X = *sn* As\}$

definition

is-starext :: [*'a star* => *'a star*, *'a* => *'a*] => *bool* **where**
[*code del*]: *is-starext F f* = ($\forall x\ y.\ \exists X \in$ *Rep-star*(x). $\exists Y \in$ *Rep-star*(y).
((y = (F x)) = ($\{n.\ Y\ n = f(X\ n)\}$: *FreeUltrafilterNat*)))

definition

starfun-n :: (*nat* => (*'a* => *'b*)) => *'a star* => *'b star* (*∗fn∗* - [*80*] *80*) **where**
∗fn∗ F = *Ifun* (*star-n F*)

definition
InternalFuns :: ('a star => 'b star) set **where**
[code del]:*InternalFuns* = $\{X.\ \exists F.\ X = *fn*\ F\}$

lemma *no-choice*: $\forall x.\ \exists y.\ Q\ x\ y$ ==> $\exists (f :: 'a => nat).\ \forall x.\ Q\ x\ (f\ x)$
⟨proof⟩

10.1 Properties of the Star-transform Applied to Sets of Reals

lemma *STAR-star-of-image-subset*: *star-of* ' A <= *s* A
⟨proof⟩

lemma *STAR-hypreal-of-real-Int*: *s* X Int Reals = *hypreal-of-real* ' X
⟨proof⟩

lemma *STAR-star-of-Int*: *s* X Int Standard = *star-of* ' X
⟨proof⟩

lemma *lemma-not-hyprealA*: $x \notin$ *hypreal-of-real* ' A ==> $\forall y \in A.\ x \neq$ *hypreal-of-real* y
⟨proof⟩

lemma *lemma-not-starA*: $x \notin$ *star-of* ' A ==> $\forall y \in A.\ x \neq$ *star-of* y
⟨proof⟩

lemma *lemma-Compl-eq*: $- \{n.\ X\ n = xa\} = \{n.\ X\ n \neq xa\}$
⟨proof⟩

lemma *STAR-real-seq-to-hypreal*:
$\forall n.\ (X\ n) \notin M$ ==> *star-n* X $\notin$ *s* M
⟨proof⟩

lemma *STAR-singleton*: *s* $\{x\}$ = $\{$*star-of* x$\}$
⟨proof⟩

lemma *STAR-not-mem*: $x \notin F$ ==> *star-of* x $\notin$ *s* F
⟨proof⟩

lemma *STAR-subset-closed*: [| x : *s* A; A <= B |] ==> x : *s* B
⟨proof⟩

Nonstandard extension of a set (defined using a constant sequence) as a special case of an internal set

lemma *starset-n-starset*: $\forall\, n.\ (As\ n = A) ==> *sn*\ As = *s*\ A$
$\langle proof \rangle$

lemma *starfun-n-starfun*: $\forall\, n.\ (F\ n = f) ==> *fn*\ F = *f*\ f$
$\langle proof \rangle$

lemma *hrabs-is-starext-rabs*: *is-starext abs abs*
$\langle proof \rangle$

Nonstandard extension of functions

lemma *starfun*:
$(\ *f*\ f)\ (star\text{-}n\ X) = star\text{-}n\ (\%n.\ f\ (X\ n))$
$\langle proof \rangle$

lemma *starfun-if-eq*:
$!!w.\ w \neq star\text{-}of\ x$
$==> (\ *f*\ (\lambda z.\ if\ z = x\ then\ a\ else\ g\ z))\ w = (\ *f*\ g)\ w$
$\langle proof \rangle$

lemma *starfun-mult*: $!!x.\ (\ *f*\ f)\ x * (\ *f*\ g)\ x = (\ *f*\ (\%x.\ f\ x * g\ x))\ x$
$\langle proof \rangle$
declare *starfun-mult* [*symmetric*, *simp*]

lemma *starfun-add*: $!!x.\ (\ *f*\ f)\ x + (\ *f*\ g)\ x = (\ *f*\ (\%x.\ f\ x + g\ x))\ x$
$\langle proof \rangle$
declare *starfun-add* [*symmetric*, *simp*]

lemma *starfun-minus*: $!!x.\ -\ (\ *f*\ f)\ x = (\ *f*\ (\%x.\ -\ f\ x))\ x$
$\langle proof \rangle$
declare *starfun-minus* [*symmetric*, *simp*]

lemma *starfun-add-minus*: $!!x.\ (\ *f*\ f)\ x + -(\ *f*\ g)\ x = (\ *f*\ (\%x.\ f\ x + -g$

x)) *x*
⟨*proof*⟩
declare *starfun-add-minus* [*symmetric*, *simp*]

lemma *starfun-diff*: !!*x*. (**f** *f*) *x* − (**f** *g*) *x* = (**f** (%*x*. *f x* − *g x*)) *x*
⟨*proof*⟩
declare *starfun-diff* [*symmetric*, *simp*]

lemma *starfun-o2*: (%*x*. (**f** *f*) ((**f** *g*) *x*)) = **f** (%*x*. *f* (*g x*))
⟨*proof*⟩

lemma *starfun-o*: (**f** *f*) *o* (**f** *g*) = (**f** (*f o g*))
⟨*proof*⟩

NS extension of constant function

lemma *starfun-const-fun* [*simp*]: !!*x*. (**f** (%*x*. *k*)) *x* = *star-of k*
⟨*proof*⟩

the NS extension of the identity function

lemma *starfun-Id* [*simp*]: !!*x*. (**f** (%*x*. *x*)) *x* = *x*
⟨*proof*⟩

lemma *starfun-Idfun-approx*:
  x @= *star-of a* ==> (**f** (%*x*. *x*)) *x* @= *star-of a*
⟨*proof*⟩

The Star-function is a (nonstandard) extension of the function

lemma *is-starext-starfun*: *is-starext* (**f** *f*) *f*
⟨*proof*⟩

Any nonstandard extension is in fact the Star-function

lemma *is-starfun-starext*: *is-starext F f* ==> *F* = **f** *f*
⟨*proof*⟩

lemma *is-starext-starfun-iff*: (*is-starext F f*) = (*F* = **f** *f*)
⟨*proof*⟩

extented function has same solution as its standard version for real arguments. i.e they are the same for all real arguments

lemma *starfun-eq*: (**f** *f*) (*star-of a*) = *star-of* (*f a*)
⟨*proof*⟩

lemma *starfun-approx*: (**f** *f*) (*star-of a*) @= *star-of* (*f a*)
⟨*proof*⟩

lemma *starfun-lambda-cancel*:
$!!x'. (*f* (\%h. f (x + h))) x' = (*f* f) (star\text{-}of\ x + x')$
⟨proof⟩

lemma *starfun-lambda-cancel2*:
$(*f* (\%h. f(g(x + h)))) x' = (*f* (f\ o\ g)) (star\text{-}of\ x + x')$
⟨proof⟩

lemma *starfun-mult-HFinite-approx*:
fixes $l\ m :: 'a::real\text{-}normed\text{-}algebra\ star$
shows $[| (*f* f) x @= l; (*f* g) x @= m;$
$l: HFinite; m: HFinite$
$|] ==> (*f* (\%x. f\ x * g\ x)) x @= l * m$
⟨proof⟩

lemma *starfun-add-approx*: $[| (*f* f) x @= l; (*f* g) x @= m$
$|] ==> (*f* (\%x. f\ x + g\ x)) x @= l + m$
⟨proof⟩

Examples: hrabs is nonstandard extension of rabs inverse is nonstandard extension of inverse

lemma *starfun-rabs-hrabs*: $*f*\ abs = abs$
⟨proof⟩

lemma *starfun-inverse-inverse* [*simp*]: $(*f* inverse) x = inverse(x)$
⟨proof⟩

lemma *starfun-inverse*: $!!x. inverse ((*f* f) x) = (*f* (\%x. inverse (f\ x))) x$
⟨proof⟩
declare *starfun-inverse* [*symmetric*, *simp*]

lemma *starfun-divide*: $!!x. (*f* f) x / (*f* g) x = (*f* (\%x. f\ x / g\ x)) x$
⟨proof⟩
declare *starfun-divide* [*symmetric*, *simp*]

lemma *starfun-inverse2*: $!!x. inverse ((*f* f) x) = (*f* (\%x. inverse (f\ x))) x$
⟨proof⟩

General lemma/theorem needed for proofs in elementary topology of the reals

lemma *starfun-mem-starset*:
$!!x. (*f* f) x : *s* A ==> x : *s* \{x. f\ x \in A\}$
⟨proof⟩

Alternative definition for hrabs with rabs function applied entrywise to equivalence class representative. This is easily proved using starfun and ns extension thm

lemma *hypreal-hrabs*:

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

nonstandard extension of set through nonstandard extension of rabs function i.e hrabs. A more general result should be where we replace rabs by some arbitrary function f and hrabs by its NS extenson. See second NS set extension below.

lemma *STAR-rabs-add-minus*:
$*s*\ \{x.\ abs\ (x + -\ y) < r\} =$
$\{x.\ abs(x + -star\text{-}of\ y) < star\text{-}of\ r\}$
⟨proof⟩

lemma *STAR-starfun-rabs-add-minus*:
$*s*\ \{x.\ abs\ (f\ x + -\ y) < r\} =$
$\{x.\ abs((\ *f*\ f)\ x + -star\text{-}of\ y) < star\text{-}of\ r\}$
⟨proof⟩

Another characterization of Infinitesimal and one of @= relation. In this theory since *hypreal-hrabs* proved here. Maybe move both theorems??

lemma *Infinitesimal-FreeUltrafilterNat-iff2*:
$(star\text{-}n\ X \in Infinitesimal) =$
$(\forall\ m.\ \{n.\ norm(X\ n) < inverse(real(Suc\ m))\}$
$\in\ FreeUltrafilterNat)$
⟨proof⟩

lemma *HNatInfinite-inverse-Infinitesimal* [*simp*]:
$n \in HNatInfinite ==> inverse\ (hypreal\text{-}of\text{-}hypnat\ n) \in Infinitesimal$
⟨proof⟩

lemma *approx-FreeUltrafilterNat-iff*: $star\text{-}n\ X\ @{=}\ star\text{-}n\ Y =$
$(\forall\ r{>}0.\ \{n.\ norm\ (X\ n - \ Y\ n) < r\} : FreeUltrafilterNat)$
⟨proof⟩

lemma *approx-FreeUltrafilterNat-iff2*: $star\text{-}n\ X\ @{=}\ star\text{-}n\ Y =$
$(\forall\ m.\ \{n.\ norm\ (X\ n - \ Y\ n) <$
$inverse(real(Suc\ m))\} : FreeUltrafilterNat)$
⟨proof⟩

lemma *inj-starfun*: *inj starfun*
⟨proof⟩

end

11 NatStar: Star-transforms for the Hypernaturals

theory *NatStar*

imports *Star*
begin

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

lemma *starset-n-Un*: $*sn*\ (\%n.\ (A\ n)\ Un\ (B\ n)) = *sn*\ A\ Un\ *sn*\ B$
⟨*proof*⟩

lemma *InternalSets-Un*:
$[|\ X \in InternalSets;\ Y \in InternalSets\ |]$
$==> (X\ Un\ Y) \in InternalSets$
⟨*proof*⟩

lemma *starset-n-Int*:
$*sn*\ (\%n.\ (A\ n)\ Int\ (B\ n)) = *sn*\ A\ Int\ *sn*\ B$
⟨*proof*⟩

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

lemma *starset-n-Compl*: $*sn*\ ((\%n.\ -\ A\ n)) = -(\ *sn*\ A)$
⟨*proof*⟩

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

lemma *starset-n-diff*: $*sn*\ (\%n.\ (A\ n) - (B\ n)) = *sn*\ A - *sn*\ B$
⟨*proof*⟩

lemma *InternalSets-diff*:
$[|\ X \in InternalSets;\ Y \in InternalSets\ |]$
$==> (X\ -\ Y) \in InternalSets$
⟨*proof*⟩

lemma *NatStar-SHNat-subset*: $Nats \le *s*\ (UNIV :: nat\ set)$
⟨*proof*⟩

lemma *NatStar-hypreal-of-real-Int*:
$*s*\ X\ Int\ Nats = hypnat\text{-}of\text{-}nat\ `\ X$
⟨*proof*⟩

lemma *starset-starset-n-eq*: $*s*\ X = *sn*\ (\%n.\ X)$
⟨*proof*⟩

lemma *InternalSets-starset-n* [*simp*]: $(\ *s*\ X) \in InternalSets$
⟨*proof*⟩

lemma *InternalSets-UNIV-diff*:
$X \in InternalSets ==> UNIV - X \in InternalSets$
$\langle proof \rangle$

11.1 Nonstandard Extensions of Functions

Example of transfer of a property from reals to hyperreals — used for limit comparison of sequences

lemma *starfun-le-mono*:
$\forall n.\ N \le n --> f\ n \le g\ n$
$==> \forall n.\ hypnat\text{-}of\text{-}nat\ N \le n --> (\ *f*\ f)\ n \le (\ *f*\ g)\ n$
$\langle proof \rangle$

lemma *starfun-less-mono*:
$\forall n.\ N \le n --> f\ n < g\ n$
$==> \forall n.\ hypnat\text{-}of\text{-}nat\ N \le n --> (\ *f*\ f)\ n < (\ *f*\ g)\ n$
$\langle proof \rangle$

Nonstandard extension when we increment the argument by one

lemma *starfun-shift-one*:
$!!N.\ (\ *f*\ (\%n.\ f\ (Suc\ n)))\ N = (\ *f*\ f)\ (N + (1::hypnat))$
$\langle proof \rangle$

Nonstandard extension with absolute value

lemma *starfun-abs*: $!!N.\ (\ *f*\ (\%n.\ abs\ (f\ n)))\ N = abs((\ *f*\ f)\ N)$
$\langle proof \rangle$

The hyperpow function as a nonstandard extension of realpow

lemma *starfun-pow*: $!!N.\ (\ *f*\ (\%n.\ r\ \hat{}\ n))\ N = (hypreal\text{-}of\text{-}real\ r)\ pow\ N$
$\langle proof \rangle$

lemma *starfun-pow2*:
$!!N.\ (\ *f*\ (\%n.\ (X\ n)\ \hat{}\ m))\ N = (\ *f*\ X)\ N\ pow\ hypnat\text{-}of\text{-}nat\ m$
$\langle proof \rangle$

lemma *starfun-pow3*: $!!R.\ (\ *f*\ (\%r.\ r\ \hat{}\ n))\ R = (R)\ pow\ hypnat\text{-}of\text{-}nat\ n$
$\langle proof \rangle$

The *hypreal-of-hypnat* function as a nonstandard extension of *real-of-nat*

lemma *starfunNat-real-of-nat*: $(\ *f*\ real) = hypreal\text{-}of\text{-}hypnat$
$\langle proof \rangle$

lemma *starfun-inverse-real-of-nat-eq*:
$N \in HNatInfinite$
$==> (\ *f*\ (\%x::nat.\ inverse(real\ x)))\ N = inverse(hypreal\text{-}of\text{-}hypnat\ N)$
$\langle proof \rangle$

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*⟩

11.2 Nonstandard Characterization of Induction

lemma *hypnat-induct-obj*:
$!!n.\ ((*p* P)\ (0::hypnat)\ \&$
$(\forall n.\ (*p* P)(n) --> (*p* P)(n + 1)))$
$--> (*p* P)(n)$
⟨*proof*⟩

lemma *hypnat-induct*:
$!!n.\ [|\ (*p* P)\ (0::hypnat);$

$!!n.\ (\ *p*\ P)(n) ==> (\ *p*\ P)(n + 1)|]$
$==> (\ *p*\ P)(n)$
⟨*proof*⟩

lemma *starP2-eq-iff*: $(\ *p2*\ (op\ =)) = (op\ =)$
⟨*proof*⟩

lemma *starP2-eq-iff2*: $(\ *p2*\ (\%x\ y.\ x = y))\ X\ Y = (X = Y)$
⟨*proof*⟩

lemma *nonempty-nat-set-Least-mem*:
$c \in (S :: nat\ set) ==> (LEAST\ n.\ n \in S) \in S$
⟨*proof*⟩

lemma *nonempty-set-star-has-least*:
$!!S::nat\ set\ star.\ Iset\ S \neq \{\} ==> \exists\, n \in Iset\ S.\ \forall\, m \in Iset\ S.\ n \leq m$
⟨*proof*⟩

lemma *nonempty-InternalNatSet-has-least*:
$[|\ (S::hypnat\ set) \in InternalSets;\ S \neq \{\}\ |] ==> \exists\, n \in S.\ \forall\, m \in S.\ n \leq m$
⟨*proof*⟩

Goldblatt page 129 Thm 11.3.2

lemma *internal-induct-lemma*:
$!!X::nat\ set\ star.\ [|\ (0::hypnat) \in Iset\ X;\ \forall\, n.\ n \in Iset\ X\ -->\ n + 1 \in Iset\ X\ |]$
$==> Iset\ X = (UNIV:: hypnat\ set)$
⟨*proof*⟩

lemma *internal-induct*:
$[|\ X \in InternalSets;\ (0::hypnat) \in X;\ \forall\, n.\ n \in X\ -->\ n + 1 \in X\ |]$
$==> X = (UNIV:: hypnat\ set)$
⟨*proof*⟩

end

12 HSEQ: Sequences and Convergence (Nonstandard)

theory *HSEQ*
imports *SEQ NatStar*
begin

definition
NSLIMSEQ :: $[nat => 'a::real\text{-}normed\text{-}vector,\ 'a] => bool$
$(((\text{-})/\ ----NS>\ (\text{-}))\ [60,\ 60]\ 60)$ **where**

— Nonstandard definition of convergence of sequence
[*code del*]: X $----NS>$ L = $(\forall N \in HNatInfinite.\ (\ *f*\ X)\ N \approx star\text{-}of\ L)$

definition
nslim :: (*nat* => $'a$::*real-normed-vector*) => $'a$ **where**
— Nonstandard definition of limit using choice operator
nslim X = (*THE L. X* $----NS>$ *L*)

definition
NSconvergent :: (*nat* => $'a$::*real-normed-vector*) => *bool* **where**
— Nonstandard definition of convergence
NSconvergent X = $(\exists L.\ X$ $----NS>$ $L)$

definition
NSBseq :: (*nat* => $'a$::*real-normed-vector*) => *bool* **where**
— Nonstandard definition for bounded sequence
[*code del*]: *NSBseq X* = $(\forall N \in HNatInfinite.\ (\ *f*\ X)\ N : HFinite)$

definition
NSCauchy :: (*nat* => $'a$::*real-normed-vector*) => *bool* **where**
— Nonstandard definition
[*code del*]: *NSCauchy X* = $(\forall M \in HNatInfinite.\ \forall N \in HNatInfinite.\ (\ *f*\ X)\ M \approx (\ *f*\ X)\ N)$

12.1 Limits of Sequences

lemma *NSLIMSEQ-iff*:
$(X$ $----NS>$ $L) = (\forall N \in HNatInfinite.\ (\ *f*\ X)\ N \approx star\text{-}of\ L)$
⟨*proof*⟩

lemma *NSLIMSEQ-I*:
$(\bigwedge N.\ N \in HNatInfinite \Longrightarrow starfun\ X\ N \approx star\text{-}of\ L) \Longrightarrow X$ $----NS>$ L
⟨*proof*⟩

lemma *NSLIMSEQ-D*:
$[\![X$ $----NS>$ $L;\ N \in HNatInfinite]\!] \Longrightarrow starfun\ X\ N \approx star\text{-}of\ L$
⟨*proof*⟩

lemma *NSLIMSEQ-const*: (%*n. k*) $----NS>$ *k*
⟨*proof*⟩

lemma *NSLIMSEQ-add*:
[| X $----NS>$ a; Y $----NS>$ b |] ==> (%*n. X n* + *Y n*) $----NS>$ $a + b$
⟨*proof*⟩

lemma *NSLIMSEQ-add-const*: f $----NS>$ a ==> (%*n*.(*f n* + *b*)) $----NS>$ $a + b$
⟨*proof*⟩

lemma *NSLIMSEQ-mult*:
fixes $a\ b :: 'a::real\text{-}normed\text{-}algebra$
shows $[|\ X\ ----NS>\ a;\ Y\ ----NS>\ b\ |] ==> (\%n.\ X\ n * Y\ n)\ ----NS>\ a * b$
⟨*proof*⟩

lemma *NSLIMSEQ-minus*: $X\ ----NS>\ a ==> (\%n.\ -(X\ n))\ ----NS>\ -a$
⟨*proof*⟩

lemma *NSLIMSEQ-minus-cancel*: $(\%n.\ -(X\ n))\ ----NS>\ -a ==> X\ ----NS>\ a$
⟨*proof*⟩

lemma *NSLIMSEQ-add-minus*:
$[|\ X\ ----NS>\ a;\ Y\ ----NS>\ b\ |] ==> (\%n.\ X\ n + -Y\ n)\ ----NS>\ a + -b$
⟨*proof*⟩

lemma *NSLIMSEQ-diff*:
$[|\ X\ ----NS>\ a;\ Y\ ----NS>\ b\ |] ==> (\%n.\ X\ n -\ Y\ n)\ ----NS>\ a - b$
⟨*proof*⟩

lemma *NSLIMSEQ-diff-const*: $f\ ----NS>\ a ==> (\%n.(f\ n - b))\ ----NS>\ a - b$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse*:
fixes $a :: 'a::real\text{-}normed\text{-}div\text{-}algebra$
shows $[|\ X\ ----NS>\ a;\ a\ \tilde{}{=}\ 0\ |] ==> (\%n.\ inverse(X\ n))\ ----NS>\ inverse(a)$
⟨*proof*⟩

lemma *NSLIMSEQ-mult-inverse*:
fixes $a\ b :: 'a::real\text{-}normed\text{-}field$
shows
$[|\ X\ ----NS>\ a;\ Y\ ----NS>\ b;\ b\ \tilde{}{=}\ 0\ |] ==> (\%n.\ X\ n\ /\ Y\ n)\ ----NS>\ a/b$
⟨*proof*⟩

lemma *starfun-hnorm*: $\bigwedge x.\ hnorm\ ((\ *f*\ f)\ x) = (\ *f*\ (\lambda x.\ norm\ (f\ x)))\ x$
⟨*proof*⟩

lemma *NSLIMSEQ-norm*: $X\ ----NS>\ a \Longrightarrow (\lambda n.\ norm\ (X\ n))\ ----NS>\ norm\ a$
⟨*proof*⟩

Uniqueness of limit

lemma *NSLIMSEQ-unique*: [| X ––––*NS*> a; X ––––*NS*> b |] ==> $a = b$
⟨*proof*⟩

lemma *NSLIMSEQ-pow* [*rule-format*]:
fixes a :: *'a*::{*real-normed-algebra*,*recpower*}
shows (X ––––*NS*> a) ––> ((%n. (X n) ^ m) ––––*NS*> a ^ m)
⟨*proof*⟩

We can now try and derive a few properties of sequences, starting with the limit comparison property for sequences.

lemma *NSLIMSEQ-le*:
[| f ––––*NS*> l; g ––––*NS*> m;
$\exists N. \forall n \geq N. f(n) \leq g(n)$
|] ==> $l \leq$ (m::*real*)
⟨*proof*⟩

lemma *NSLIMSEQ-le-const*: [| X ––––*NS*> (r::*real*); $\forall n.\ a \leq X\ n$ |] ==> $a \leq r$
⟨*proof*⟩

lemma *NSLIMSEQ-le-const2*: [| X ––––*NS*> (r::*real*); $\forall n.\ X\ n \leq a$ |] ==> $r \leq a$
⟨*proof*⟩

Shift a convergent series by 1: By the equivalence between Cauchiness and convergence and because the successor of an infinite hypernatural is also infinite.

lemma *NSLIMSEQ-Suc*: f ––––*NS*> l ==> (%n. f(*Suc* n)) ––––*NS*> l
⟨*proof*⟩

lemma *NSLIMSEQ-imp-Suc*: (%n. f(*Suc* n)) ––––*NS*> l ==> f ––––*NS*> l
⟨*proof*⟩

lemma *NSLIMSEQ-Suc-iff*: ((%n. f(*Suc* n)) ––––*NS*> l) = (f ––––*NS*> l)
⟨*proof*⟩

12.1.1 Equivalence of *LIMSEQ* and *NSLIMSEQ*

lemma *LIMSEQ-NSLIMSEQ*:
assumes X: X ––––> L **shows** X ––––*NS*> L
⟨*proof*⟩

lemma *NSLIMSEQ-LIMSEQ*:
assumes X: X ––––*NS*> L **shows** X ––––> L
⟨*proof*⟩

theorem *LIMSEQ-NSLIMSEQ-iff*: (f ––––> L) = (f ––––*NS*> L)
⟨*proof*⟩

12.1.2 Derived theorems about *NSLIMSEQ*

We prove the NS version from the standard one, since the NS proof seems more complicated than the standard one above!

lemma *NSLIMSEQ-norm-zero*: $((\lambda n.\ norm\ (X\ n)) \ {----}NS{>}\ 0) = (X\ {----}NS{>}\ 0)$
⟨*proof*⟩

lemma *NSLIMSEQ-rabs-zero*: $((\%n.\ |f\ n|)\ {----}NS{>}\ 0) = (f\ {----}NS{>}\ (0{::}real))$
⟨*proof*⟩

Generalization to other limits

lemma *NSLIMSEQ-imp-rabs*: $f\ {----}NS{>}\ (l{::}real) ==> (\%n.\ |f\ n|)\ {----}NS{>}\ |l|$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse-zero*:
$\forall\, y{::}real.\ \exists\, N.\ \forall\, n \geq N.\ y < f(n)$
$==> (\%n.\ inverse(f\ n))\ {----}NS{>}\ 0$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse-real-of-nat*: $(\%n.\ inverse(real(Suc\ n)))\ {----}NS{>}\ 0$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse-real-of-nat-add*:
$(\%n.\ r + inverse(real(Suc\ n)))\ {----}NS{>}\ r$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse-real-of-nat-add-minus*:
$(\%n.\ r + -inverse(real(Suc\ n)))\ {----}NS{>}\ r$
⟨*proof*⟩

lemma *NSLIMSEQ-inverse-real-of-nat-add-minus-mult*:
$(\%n.\ r*(\ 1 + -inverse(real(Suc\ n))))\ {----}NS{>}\ r$
⟨*proof*⟩

12.2 Convergence

lemma *nslimI*: $X\ {----}NS{>}\ L ==> nslim\ X = L$
⟨*proof*⟩

lemma *lim-nslim-iff*: $lim\ X = nslim\ X$
⟨*proof*⟩

lemma *NSconvergentD*: $NSconvergent\ X ==> \exists\, L.\ (X\ {----}NS{>}\ L)$
⟨*proof*⟩

lemma *NSconvergentI*: $(X\ {----}NS{>}\ L) ==> NSconvergent\ X$
⟨*proof*⟩

lemma *convergent-NSconvergent-iff*: *convergent X* = *NSconvergent X*
⟨*proof*⟩

lemma *NSconvergent-NSLIMSEQ-iff*: *NSconvergent X* = (*X* −−−−*NS*> *nslim X*)
⟨*proof*⟩

12.3 Bounded Monotonic Sequences

lemma *NSBseqD*: [| *NSBseq X*; *N*: *HNatInfinite* |] ==> (∗*f*∗ *X*) *N* : *HFinite*
⟨*proof*⟩

lemma *Standard-subset-HFinite*: *Standard* ⊆ *HFinite*
⟨*proof*⟩

lemma *NSBseqD2*: *NSBseq X* $\Longrightarrow$ (∗*f*∗ *X*) *N* ∈ *HFinite*
⟨*proof*⟩

lemma *NSBseqI*: ∀ *N* ∈ *HNatInfinite*. (∗*f*∗ *X*) *N* : *HFinite* ==> *NSBseq X*
⟨*proof*⟩

The standard definition implies the nonstandard definition

lemma *Bseq-NSBseq*: *Bseq X* ==> *NSBseq X*
⟨*proof*⟩

The nonstandard definition implies the standard definition

lemma *SReal-less-omega*: $r \in \mathbb{R} \Longrightarrow r < \omega$
⟨*proof*⟩

lemma *NSBseq-Bseq*: *NSBseq X* $\Longrightarrow$ *Bseq X*
⟨*proof*⟩

Equivalence of nonstandard and standard definitions for a bounded sequence

lemma *Bseq-NSBseq-iff*: (*Bseq X*) = (*NSBseq X*)
⟨*proof*⟩

A convergent sequence is bounded: Boundedness as a necessary condition for convergence. The nonstandard version has no existential, as usual

lemma *NSconvergent-NSBseq*: *NSconvergent X* ==> *NSBseq X*
⟨*proof*⟩

Standard Version: easily now proved using equivalence of NS and standard definitions

lemma *convergent-Bseq*: *convergent X* ==> *Bseq X*
⟨*proof*⟩

12.3.1 Upper Bounds and Lubs of Bounded Sequences

lemma *NSBseq-isUb*: *NSBseq X* $==>$ $\exists$ *U::real. isUb UNIV* $\{x.\ \exists n.\ X\ n = x\}$ *U*
⟨*proof*⟩

lemma *NSBseq-isLub*: *NSBseq X* $==>$ $\exists$ *U::real. isLub UNIV* $\{x.\ \exists n.\ X\ n = x\}$ *U*
⟨*proof*⟩

12.3.2 A Bounded and Monotonic Sequence Converges

The best of both worlds: Easier to prove this result as a standard theorem and then use equivalence to "transfer" it into the equivalent nonstandard form if needed!

lemma *Bmonoseq-NSLIMSEQ*: $\forall n \geq m.\ X\ n = X\ m$ $==>$ $\exists L.\ (X$ ----*NS*> *L*)
⟨*proof*⟩

lemma *NSBseq-mono-NSconvergent*:
[| *NSBseq X*; $\forall m.\ \forall n \geq m.\ X\ m \leq X\ n$ |] $==>$ *NSconvergent* (*X::nat=>real*)
⟨*proof*⟩

12.4 Cauchy Sequences

lemma *NSCauchyI*:
$(\bigwedge M\ N.\ [\![M \in HNatInfinite;\ N \in HNatInfinite]\!] \Longrightarrow starfun\ X\ M \approx starfun\ X\ N)$
$\Longrightarrow$ *NSCauchy X*
⟨*proof*⟩

lemma *NSCauchyD*:
$[\![NSCauchy\ X;\ M \in HNatInfinite;\ N \in HNatInfinite]\!]$
$\Longrightarrow starfun\ X\ M \approx starfun\ X\ N$
⟨*proof*⟩

12.4.1 Equivalence Between NS and Standard

lemma *Cauchy-NSCauchy*:
assumes *X*: *Cauchy X* **shows** *NSCauchy X*
⟨*proof*⟩

lemma *NSCauchy-Cauchy*:
assumes *X*: *NSCauchy X* **shows** *Cauchy X*
⟨*proof*⟩

theorem *NSCauchy-Cauchy-iff*: *NSCauchy X* = *Cauchy X*
⟨*proof*⟩

12.4.2 Cauchy Sequences are Bounded

A Cauchy sequence is bounded – nonstandard version

lemma *NSCauchy-NSBseq*: *NSCauchy X ==> NSBseq X*
⟨*proof*⟩

12.4.3 Cauchy Sequences are Convergent

Equivalence of Cauchy criterion and convergence: We will prove this using our NS formulation which provides a much easier proof than using the standard definition. We do not need to use properties of subsequences such as boundedness, monotonicity etc... Compare with Harrison's corresponding proof in HOL which is much longer and more complicated. Of course, we do not have problems which he encountered with guessing the right instantiations for his 'espsilon-delta' proof(s) in this case since the NS formulations do not involve existential quantifiers.

lemma *NSconvergent-NSCauchy*: *NSconvergent X* $\Longrightarrow$ *NSCauchy X*
⟨*proof*⟩

lemma *real-NSCauchy-NSconvergent*:
 fixes $X :: nat \Rightarrow real$
 shows *NSCauchy X* $\Longrightarrow$ *NSconvergent X*
⟨*proof*⟩

lemma *NSCauchy-NSconvergent*:
 fixes $X :: nat \Rightarrow {}'a::banach$
 shows *NSCauchy X* $\Longrightarrow$ *NSconvergent X*
⟨*proof*⟩

lemma *NSCauchy-NSconvergent-iff*:
 fixes $X :: nat \Rightarrow {}'a::banach$
 shows *NSCauchy X = NSconvergent X*
⟨*proof*⟩

12.5 Power Sequences

The sequence x ^ n tends to 0 if $(0::'a) \leq x$ and $x < (1::'a)$. Proof will use (NS) Cauchy equivalence for convergence and also fact that bounded and monotonic sequence converges.

We now use NS criterion to bring proof of theorem through

lemma *NSLIMSEQ-realpow-zero*:
 $[|\ 0 \leq (x::real);\ x < 1\ |] ==> (\%n.\ x$ ^ $n)$ ––––*NS*> *0*
⟨*proof*⟩

lemma *NSLIMSEQ-rabs-realpow-zero*: $|c| < (1::real) ==> (\%n.\ |c|$ ^ $n)$ ––––*NS*> *0*

⟨*proof*⟩

lemma *NSLIMSEQ-rabs-realpow-zero2*: $|c| < (1::real) ==> (\%n.\ c\ \hat{}\ n) ----NS>$
0
⟨*proof*⟩

end

13 HSeries: Finite Summation and Infinite Series for Hyperreals

theory *HSeries*
imports *Series HSEQ*
begin

definition
sumhr :: (*hypnat* ∗ *hypnat* ∗ (*nat=>real*)) => *hypreal* **where**
[*code del*]: *sumhr* =
(%(*M*,*N*,*f*). *starfun2* (%*m n*. *setsum f* {*m*..<*n*}) *M N*)

definition
NSsums :: [*nat=>real*,*real*] => *bool* (**infixr** *NSsums 80*) **where**
f NSsums s = (%*n*. *setsum f* {*0*..<*n*}) −−−−*NS*> *s*

definition
NSsummable :: (*nat=>real*) => *bool* **where**
[*code del*]: *NSsummable f* = (∃ *s*. *f NSsums s*)

definition
NSsuminf :: (*nat=>real*) => *real* **where**
NSsuminf f = (*THE s*. *f NSsums s*)

lemma *sumhr-app*: *sumhr*(*M*,*N*,*f*) = (∗*f2*∗ (λ*m n*. *setsum f* {*m*..<*n*})) *M N*
⟨*proof*⟩

Base case in definition of *sumr*

lemma *sumhr-zero* [*simp*]: !!*m*. *sumhr* (*m*,*0*,*f*) = *0*
⟨*proof*⟩

Recursive case in definition of *sumr*

lemma *sumhr-if*:
!!*m n*. *sumhr*(*m*,*n*+*1*,*f*) =
(*if n* + *1* ≤ *m then 0 else sumhr*(*m*,*n*,*f*) + (∗*f*∗ *f*) *n*)
⟨*proof*⟩

lemma *sumhr-Suc-zero* [*simp*]: !!n. *sumhr* (n + *1*, n, f) = *0*
⟨*proof*⟩

lemma *sumhr-eq-bounds* [*simp*]: !!n. *sumhr* (n,n,f) = *0*
⟨*proof*⟩

lemma *sumhr-Suc* [*simp*]: !!m. *sumhr* (m,m + *1*,f) = ($*f*$ f) m
⟨*proof*⟩

lemma *sumhr-add-lbound-zero* [*simp*]: !!k m. *sumhr*(m+k,k,f) = *0*
⟨*proof*⟩

lemma *sumhr-add*:
  !!m n. *sumhr* (m,n,f) + *sumhr*(m,n,g) = *sumhr*(m,n,%i. f i + g i)
⟨*proof*⟩

lemma *sumhr-mult*:
  !!m n. *hypreal-of-real* r $*$ *sumhr*(m,n,f) = *sumhr*(m,n,%n. r $*$ f n)
⟨*proof*⟩

lemma *sumhr-split-add*:
  !!n p. n < p ==> *sumhr*(*0*,n,f) + *sumhr*(n,p,f) = *sumhr*(*0*,p,f)
⟨*proof*⟩

lemma *sumhr-split-diff*: n<p ==> *sumhr*(*0*,p,f) − *sumhr*(*0*,n,f) = *sumhr*(n,p,f)
⟨*proof*⟩

lemma *sumhr-hrabs*: !!m n. *abs*(*sumhr*(m,n,f)) ≤ *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*:
     !!n. *sumhr*(*0*, n, %i. r) = *hypreal-of-hypnat* n $*$ *hypreal-of-real* r
⟨*proof*⟩

lemma *sumhr-less-bounds-zero* [*simp*]: !!m n. n < m ==> *sumhr*(m,n,f) = *0*
⟨*proof*⟩

lemma *sumhr-minus*: !!m n. *sumhr*(m, n, %i. − f i) = − *sumhr*(m, n, f)
⟨*proof*⟩

lemma *sumhr-shift-bounds*:
  !!m n. *sumhr*(m+*hypnat-of-nat* k,n+*hypnat-of-nat* k,f) =

sumhr$(m,n,\%i.\ f(i + k))$
⟨*proof*⟩

13.1 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-of-hypnat whn*
⟨*proof*⟩

lemma *sumhr-hypreal-omega-minus-one*: *sumhr*$(0,\ whn,\ \%i.\ 1) = omega - 1$
⟨*proof*⟩

lemma *sumhr-minus-one-realpow-zero* [*simp*]:
$!!N.\ sumhr(0,\ N + N,\ \%i.\ (-1)\ \hat{}\ (i+1)) = 0$
⟨*proof*⟩

lemma *sumhr-interval-const*:
$(\forall n.\ m \leq Suc\ n \ {-}{-}{>}\ f\ n = r)\ \&\ m \leq na$
$==>$ *sumhr*(*hypnat-of-nat m*,*hypnat-of-nat na*,*f*) =
(*hypreal-of-nat* $(na - m) *$ *hypreal-of-real r*)
⟨*proof*⟩

lemma *starfunNat-sumr*: $!!N.\ (\ *f*\ (\%n.\ setsum\ f\ \{0..{<}n\}))\ N = sumhr(0,N,f)$
⟨*proof*⟩

lemma *sumhr-hrabs-approx* [*simp*]: $sumhr(0,\ M,\ f)\ @{=}\ sumhr(0,\ N,\ f)$
$==>\ abs\ (sumhr(M,\ N,\ f))\ @{=}\ 0$
⟨*proof*⟩

lemma *sums-NSsums-iff*: (*f sums l*) = (*f NSsums l*)
⟨*proof*⟩

lemma *summable-NSsummable-iff*: (*summable f*) = (*NSsummable f*)
⟨*proof*⟩

lemma *suminf-NSsuminf-iff*: (*suminf f*) = (*NSsuminf f*)
⟨*proof*⟩

lemma *NSsums-NSsummable*: *f NSsums l* ==> *NSsummable f*
⟨*proof*⟩

lemma *NSsummable-NSsums*: *NSsummable f* ==> *f NSsums* (*NSsuminf f*)
⟨*proof*⟩

lemma *NSsums-unique*: *f NSsums s* ==> (*s* = *NSsuminf f*)
⟨*proof*⟩

lemma *NSseries-zero*:
$\forall m.\ n \leq Suc\ m \text{ --> } f(m) = 0 \text{ ==> } f\ NSsums\ (setsum\ f\ \{0..<n\})$
⟨*proof*⟩

lemma *NSsummable-NSCauchy*:
NSsummable f =
$(\forall M \in HNatInfinite.\ \forall N \in HNatInfinite.\ abs\ (sumhr(M,N,f))\ @{=}\ 0)$
⟨*proof*⟩

Terms of a convergent series tend to zero

lemma *NSsummable-NSLIMSEQ-zero*: *NSsummable f* ==> *f* ----NS> *0*
⟨*proof*⟩

Nonstandard comparison test

lemma *NSsummable-comparison-test*:
$[|\ \exists N.\ \forall n.\ N \leq n \text{ --> } abs(f\ n) \leq g\ n;\ NSsummable\ g\ |] \text{ ==> } NSsummable$
f
⟨*proof*⟩

lemma *NSsummable-rabs-comparison-test*:
$[|\ \exists N.\ \forall n.\ N \leq n \text{ --> } abs(f\ n) \leq g\ n;\ NSsummable\ g\ |]$
==> *NSsummable* (%*k. abs* (*f k*))
⟨*proof*⟩

end

14 HLim: Limits and Continuity (Nonstandard)

theory *HLim*
imports *Star Lim*
begin

Nonstandard Definitions

definition
NSLIM :: ['*a*::*real-normed-vector* => '*b*::*real-normed-vector*, '*a*, '*b*] => *bool*
(((-)/ -- (-)/ --NS> (-)) [*60*, *0*, *60*] *60*) **where**
[*code del*]: *f* -- *a* --*NS*> *L* =
$(\forall x.\ (x \neq star\text{-}of\ a\ \&\ x\ @{=}\ star\text{-}of\ a \text{ --> } (\ {*}f{*}\ f)\ x\ @{=}\ star\text{-}of\ L))$

definition
isNSCont :: ['*a*::*real-normed-vector* => '*b*::*real-normed-vector*, '*a*] => *bool* **where**
— NS definition dispenses with limit notions
[*code del*]: $isNSCont\ f\ a = (\forall y.\ y\ @{=}\ star\text{-}of\ a \text{ --> }$
$(\ {*}f{*}\ f)\ y\ @{=}\ star\text{-}of\ (f\ a))$

definition

$isNSUCont$ :: $['a::real\text{-}normed\text{-}vector => 'b::real\text{-}normed\text{-}vector] => bool$ **where**
$[code\ del]$: $isNSUCont\ f = (\forall x\ y.\ x\ @= y\ --> (\ *f*\ f)\ x\ @= (\ *f*\ f)\ y)$

14.1 Limits of Functions

lemma *NSLIM-I*:
$(\bigwedge x.\ [\![x \neq star\text{-}of\ a;\ x \approx star\text{-}of\ a]\!] \Longrightarrow starfun\ f\ x \approx star\text{-}of\ L)$
$\Longrightarrow f\ --\ a\ --NS> L$
⟨proof⟩

lemma *NSLIM-D*:
$[\![f\ --\ a\ --NS> L;\ x \neq star\text{-}of\ a;\ x \approx star\text{-}of\ a]\!]$
$\Longrightarrow starfun\ f\ x \approx star\text{-}of\ L$
⟨proof⟩

Proving properties of limits using nonstandard definition. The properties hold for standard limits as well!

lemma *NSLIM-mult*:
fixes $l\ m$ :: $'a::real\text{-}normed\text{-}algebra$
shows $[|\ f\ --\ x\ --NS> l;\ g\ --\ x\ --NS> m\ |]$
$==> (\%x.\ f(x) * g(x))\ --\ x\ --NS> (l * m)$
⟨proof⟩

lemma *starfun-scaleR* $[simp]$:
$starfun\ (\lambda x.\ f\ x *_R g\ x) = (\lambda x.\ scaleHR\ (starfun\ f\ x)\ (starfun\ g\ x))$
⟨proof⟩

lemma *NSLIM-scaleR*:
$[|\ f\ --\ x\ --NS> l;\ g\ --\ x\ --NS> m\ |]$
$==> (\%x.\ f(x) *_R g(x))\ --\ x\ --NS> (l *_R m)$
⟨proof⟩

lemma *NSLIM-add*:
$[|\ f\ --\ x\ --NS> l;\ g\ --\ x\ --NS> m\ |]$
$==> (\%x.\ f(x) + g(x))\ --\ x\ --NS> (l + m)$
⟨proof⟩

lemma *NSLIM-const* $[simp]$: $(\%x.\ k)\ --\ x\ --NS> k$
⟨proof⟩

lemma *NSLIM-minus*: $f\ --\ a\ --NS> L ==> (\%x.\ -f(x))\ --\ a\ --NS> -L$
⟨proof⟩

lemma *NSLIM-diff*:
$[\![f\ --\ x\ --NS> l;\ g\ --\ x\ --NS> m]\!] \Longrightarrow (\lambda x.\ f\ x - g\ x)\ --\ x\ --NS> (l - m)$
⟨proof⟩

lemma *NSLIM-add-minus*: $[|\ f\ --\ x\ --NS> l;\ g\ --\ x\ --NS> m\ |] ==>$

$(\%x.\ f(x) + -g(x))$ -- x --NS> $(l + -m)$
⟨proof⟩

lemma *NSLIM-inverse*:
fixes L :: 'a::real-normed-div-algebra
shows [| f -- a --NS> L; $L \neq 0$ |]
==> $(\%x.\ inverse(f(x)))$ -- a --NS> $(inverse\ L)$
⟨proof⟩

lemma *NSLIM-zero*:
assumes f: f -- a --NS> l **shows** $(\%x.\ f(x) - l)$ -- a --NS> 0
⟨proof⟩

lemma *NSLIM-zero-cancel*: $(\%x.\ f(x) - l)$ -- x --NS> 0 ==> f -- x --NS> l
⟨proof⟩

lemma *NSLIM-const-not-eq*:
fixes a :: 'a::real-normed-algebra-1
shows $k \neq L \Longrightarrow \neg\ (\lambda x.\ k)$ -- a --NS> L
⟨proof⟩

lemma *NSLIM-not-zero*:
fixes a :: 'a::real-normed-algebra-1
shows $k \neq 0 \Longrightarrow \neg\ (\lambda x.\ k)$ -- a --NS> 0
⟨proof⟩

lemma *NSLIM-const-eq*:
fixes a :: 'a::real-normed-algebra-1
shows $(\lambda x.\ k)$ -- a --NS> $L \Longrightarrow k = L$
⟨proof⟩

lemma *NSLIM-unique*:
fixes a :: 'a::real-normed-algebra-1
shows $[\![f$ -- a --NS> L; f -- a --NS> $M]\!] \Longrightarrow L = M$
⟨proof⟩

lemma *NSLIM-mult-zero*:
fixes $f\ g$ :: 'a::real-normed-vector ⇒ 'b::real-normed-algebra
shows [| f -- x --NS> 0; g -- x --NS> 0 |] ==> $(\%x.\ f(x)*g(x))$ -- x --NS> 0
⟨proof⟩

lemma *NSLIM-self*: $(\%x.\ x)$ -- a --NS> a
⟨proof⟩

14.1.1 Equivalence of *LIM* and *NSLIM*

lemma *LIM-NSLIM*:

assumes *f*: f −− a −−> L **shows** f −− a −−*NS*> L
⟨*proof*⟩

lemma *NSLIM-LIM*:
assumes *f*: f −− a −−*NS*> L **shows** f −− a −−> L
⟨*proof*⟩

theorem *LIM-NSLIM-iff*: (f −− x −−> L) = (f −− x −−*NS*> L)
⟨*proof*⟩

14.2 Continuity

lemma *isNSContD*:
⟦*isNSCont f a*; $y \approx$ *star-of a*⟧ $\Longrightarrow$ (∗*f*∗ f) $y \approx$ *star-of* (f a)
⟨*proof*⟩

lemma *isNSCont-NSLIM*: *isNSCont f a* ==> f −− a −−*NS*> (f a)
⟨*proof*⟩

lemma *NSLIM-isNSCont*: f −− a −−*NS*> (f a) ==> *isNSCont f a*
⟨*proof*⟩

NS continuity can be defined using NS Limit in similar fashion to standard def of continuity

lemma *isNSCont-NSLIM-iff*: (*isNSCont f a*) = (f −− a −−*NS*> (f a))
⟨*proof*⟩

Hence, NS continuity can be given in terms of standard limit

lemma *isNSCont-LIM-iff*: (*isNSCont f a*) = (f −− a −−> (f a))
⟨*proof*⟩

Moreover, it’s trivial now that NS continuity is equivalent to standard continuity

lemma *isNSCont-isCont-iff*: (*isNSCont f a*) = (*isCont f a*)
⟨*proof*⟩

Standard continuity ==¿ NS continuity

lemma *isCont-isNSCont*: *isCont f a* ==> *isNSCont f a*
⟨*proof*⟩

NS continuity ==¿ Standard continuity

lemma *isNSCont-isCont*: *isNSCont f a* ==> *isCont f a*
⟨*proof*⟩

Alternative definition of continuity

lemma *NSLIM-h-iff*: (f −− a −−*NS*> L) = ((%h. $f(a + h)$) −− *0* −−*NS*> L)
⟨*proof*⟩

lemma *NSLIM-isCont-iff*: (*f* −− *a* −−*NS*> *f a*) = ((%*h*. *f*(*a* + *h*)) −− *0* −−*NS*> *f a*)
⟨*proof*⟩

lemma *isNSCont-minus*: *isNSCont f a* ==> *isNSCont* (%*x*. − *f x*) *a*
⟨*proof*⟩

lemma *isNSCont-inverse*:
  fixes *f* :: ′*a*::*real-normed-vector* ⇒ ′*b*::*real-normed-div-algebra*
  shows [| *isNSCont f x*; *f x* ≠ *0* |] ==> *isNSCont* (%*x*. *inverse* (*f x*)) *x*
⟨*proof*⟩

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

lemma *isNSCont-abs* [*simp*]: *isNSCont abs* (*a*::*real*)
⟨*proof*⟩

14.3 Uniform Continuity

lemma *isNSUContD*: [| *isNSUCont f*; *x* ≈ *y*|] ==> (∗*f*∗ *f*) *x* ≈ (∗*f*∗ *f*) *y*
⟨*proof*⟩

lemma *isUCont-isNSUCont*:
  fixes *f* :: ′*a*::*real-normed-vector* ⇒ ′*b*::*real-normed-vector*
  assumes *f*: *isUCont f* **shows** *isNSUCont f*
⟨*proof*⟩

lemma *isNSUCont-isUCont*:
  fixes *f* :: ′*a*::*real-normed-vector* ⇒ ′*b*::*real-normed-vector*
  assumes *f*: *isNSUCont f* **shows** *isUCont f*
⟨*proof*⟩

end

15 HDeriv: Differentiation (Nonstandard)

theory *HDeriv*
imports *Deriv HLim*
begin

Nonstandard Definitions

definition
  nsderiv :: [′*a*::*real-normed-field* ⇒ ′*a*, ′*a*, ′*a*] ⇒ *bool*
    ((*NSDERIV* (-)/ (-)/ :> (-)) [*1000*, *1000*, *60*] *60*) **where**
  NSDERIV f x :> *D* = (∀ *h* ∈ *Infinitesimal* − {*0*}.
    ((∗*f*∗ *f*)(*star-of x* + *h*)
     − *star-of* (*f x*))/*h* @= *star-of D*)

definition
NSdifferentiable :: $['a::real\text{-}normed\text{-}field \Rightarrow 'a,\ 'a] \Rightarrow bool$
(**infixl** *NSdifferentiable 60*) **where**
f *NSdifferentiable* $x = (\exists D.\ NSDERIV\ f\ x :> D)$

definition
increment :: [*real=>real,real,hypreal*] => *hypreal* **where**
[*code del*]: *increment f x h* = (@*inc*. *f NSdifferentiable x* &
inc = (**f* f*)(*hypreal-of-real x* + *h*) − *hypreal-of-real* (*f x*))

15.1 Derivatives

lemma *DERIV-NS-iff*:
(*DERIV f x* :> *D*) = ((%*h*. (*f*(*x* + *h*) − *f*(*x*))/*h*) −− *0* −−*NS*> *D*)
⟨*proof*⟩

lemma *NS-DERIV-D*: *DERIV f x* :> *D* ==> (%*h*. (*f*(*x* + *h*) − *f*(*x*))/*h*) −− *0* −−*NS*> *D*
⟨*proof*⟩

lemma *hnorm-of-hypreal*:
$\bigwedge r.$ *hnorm* ((**f* of-real*) *r*::'*a*::*real-normed-div-algebra star*) = |*r*|
⟨*proof*⟩

lemma *Infinitesimal-of-hypreal*:
$x \in$ *Infinitesimal* $\Longrightarrow$
((**f* of-real*) *x*::'*a*::*real-normed-div-algebra star*) $\in$ *Infinitesimal*
⟨*proof*⟩

lemma *of-hypreal-eq-0-iff*:
$\bigwedge x.$ ((**f* of-real*) *x* = (*0*::'*a*::*real-algebra-1 star*)) = (*x* = *0*)
⟨*proof*⟩

lemma *NSDeriv-unique*:
[| *NSDERIV f x* :> *D*; *NSDERIV f x* :> *E* |] ==> *D* = *E*
⟨*proof*⟩

First NSDERIV in terms of NSLIM

first equivalence

lemma *NSDERIV-NSLIM-iff*:
(*NSDERIV f x* :> *D*) = ((%*h*. (*f*(*x* + *h*) − *f*(*x*))/*h*) −− *0* −−*NS*> *D*)
⟨*proof*⟩

second equivalence

lemma *NSDERIV-NSLIM-iff2*:
(*NSDERIV f x* :> *D*) = ((%*z*. (*f*(*z*) − *f*(*x*)) / (*z*−*x*)) −− *x* −−*NS*> *D*)
⟨*proof*⟩

lemma *NSDERIV-iff2*:
$(NSDERIV\ f\ x :> D) =$
$(\forall\ w.$
$w \neq star\text{-}of\ x\ \&\ w \approx star\text{-}of\ x\ --> $
$(\ *f*\ (\%z.\ (f\ z\ -\ f\ x)\ /\ (z{-}x)))\ w \approx star\text{-}of\ D)$
⟨*proof*⟩

lemma *hypreal-not-eq-minus-iff*:
$(x \neq a) = (x\ -\ a \neq (0::'a::ab\text{-}group\text{-}add))$
⟨*proof*⟩

lemma *NSDERIVD5*:
$(NSDERIV\ f\ x :> D) ==>$
$(\forall\ u.\ u \approx hypreal\text{-}of\text{-}real\ x\ -->$
$(\ *f*\ (\%z.\ f\ z\ -\ f\ x))\ u \approx hypreal\text{-}of\text{-}real\ D\ *\ (u\ -\ hypreal\text{-}of\text{-}real\ x))$
⟨*proof*⟩

lemma *NSDERIVD4*:
$(NSDERIV\ f\ x :> D) ==>$
$(\forall\ h \in Infinitesimal.$
$((\ *f*\ f)(hypreal\text{-}of\text{-}real\ x\ +\ h)\ -$
$hypreal\text{-}of\text{-}real\ (f\ x))\approx (hypreal\text{-}of\text{-}real\ D)\ *\ h)$
⟨*proof*⟩

lemma *NSDERIVD3*:
$(NSDERIV\ f\ x :> D) ==>$
$(\forall\ h \in Infinitesimal\ -\ \{0\}.$
$((\ *f*\ f)(hypreal\text{-}of\text{-}real\ x\ +\ h)\ -$
$hypreal\text{-}of\text{-}real\ (f\ x))\approx (hypreal\text{-}of\text{-}real\ D)\ *\ h)$
⟨*proof*⟩

Differentiability implies continuity nice and simple "algebraic" proof

lemma *NSDERIV-isNSCont*: $NSDERIV\ f\ x :> D ==> isNSCont\ f\ x$
⟨*proof*⟩

Differentiation rules for combinations of functions follow from clear, straightforard, algebraic manipulations

Constant function

lemma *NSDERIV-const* [*simp*]: $(NSDERIV\ (\%x.\ k)\ x :> 0)$
⟨*proof*⟩

Sum of functions- proved easily

lemma *NSDERIV-add*: $[|\ NSDERIV\ f\ x :> Da;\ \ NSDERIV\ g\ x :> Db\ |]$
$==> NSDERIV\ (\%x.\ f\ x\ +\ g\ x)\ x :> Da\ +\ Db$

⟨*proof*⟩

Product of functions - Proof is trivial but tedious and long due to rearrangement of terms

lemma *lemma-nsderiv1*:
  fixes *a b c d* :: *'a::comm-ring star*
  shows $(a*b) - (c*d) = (b*(a - c)) + (c*(b - d))$
⟨*proof*⟩

lemma *lemma-nsderiv2*:
  fixes *x y z* :: *'a::real-normed-field star*
  shows [| $(x - y) / z = $ *star-of D* $+ yb$; $z \neq 0$;
      $z \in$ *Infinitesimal*; $yb \in$ *Infinitesimal* |]
    ==> $x - y \approx 0$
⟨*proof*⟩

lemma *NSDERIV-mult*: [| *NSDERIV f x* :> *Da*; *NSDERIV g x* :> *Db* |]
    ==> *NSDERIV* (%*x*. *f x* ∗ *g x*) *x* :> $(Da * g(x)) + (Db * f(x))$
⟨*proof*⟩

Multiplying by a constant

lemma *NSDERIV-cmult*: *NSDERIV f x* :> *D*
    ==> *NSDERIV* (%*x*. *c* ∗ *f x*) *x* :> *c*∗*D*
⟨*proof*⟩

Negation of function

lemma *NSDERIV-minus*: *NSDERIV f x* :> *D* ==> *NSDERIV* (%*x*. −(*f x*)) *x* :> −*D*
⟨*proof*⟩

Subtraction

lemma *NSDERIV-add-minus*: [| *NSDERIV f x* :> *Da*; *NSDERIV g x* :> *Db* |] ==> *NSDERIV* (%*x*. *f x* + −*g x*) *x* :> *Da* + −*Db*
⟨*proof*⟩

lemma *NSDERIV-diff*:
   [| *NSDERIV f x* :> *Da*; *NSDERIV g x* :> *Db* |]
    ==> *NSDERIV* (%*x*. *f x* − *g x*) *x* :> *Da*−*Db*
⟨*proof*⟩

Similarly to the above, the chain rule admits an entirely straightforward derivation. Compare this with Harrison's HOL proof of the chain rule, which proved to be trickier and required an alternative characterisation of differentiability- the so-called Carathedory derivative. Our main problem is manipulation of terms.

lemma *NSDERIV-zero*:
    [| *NSDERIV g x* :> *D*;

(*f* g) (star-of x + xa) = star-of (g x);
xa ∈ Infinitesimal;
xa ≠ 0
|] ==> D = 0
⟨proof⟩

lemma *NSDERIV-approx*:
[| NSDERIV f x :> D; h ∈ Infinitesimal; h ≠ 0 |]
==> (*f* f) (star-of x + h) − star-of (f x) ≈ 0
⟨proof⟩

lemma *NSDERIVD1*: [| NSDERIV f (g x) :> Da;
(*f* g) (star-of(x) + xa) ≠ star-of (g x);
(*f* g) (star-of(x) + xa) ≈ star-of (g x)
|] ==> ((*f* f) ((*f* g) (star-of(x) + xa))
− star-of (f (g x)))
/ ((*f* g) (star-of(x) + xa) − star-of (g x))
≈ star-of(Da)
⟨proof⟩

lemma *NSDERIVD2*: [| NSDERIV g x :> Db; xa ∈ Infinitesimal; xa ≠ 0 |]
==> ((*f* g) (star-of(x) + xa) − star-of(g x)) / xa
≈ star-of(Db)
⟨proof⟩

lemma *lemma-chain*: (z::'a::real-normed-field star) ≠ 0 ==> x*y = (x*inverse(z))*(z*y)
⟨proof⟩

This proof uses both definitions of differentiability.

lemma *NSDERIV-chain*: [| NSDERIV f (g x) :> Da; NSDERIV g x :> Db |]
==> NSDERIV (f o g) x :> Da * Db
⟨proof⟩

Differentiation of natural number powers

lemma *NSDERIV-Id* [simp]: NSDERIV (%x. x) x :> 1
⟨proof⟩

lemma *NSDERIV-cmult-Id* [simp]: NSDERIV (op * c) x :> c
⟨proof⟩

lemma *NSDERIV-inverse*:
fixes x :: 'a::{real-normed-field,recpower}
shows x ≠ 0 ==> NSDERIV (%x. inverse(x)) x :> (− (inverse x ^ Suc (Suc

0)))
⟨*proof*⟩

15.1.1 Equivalence of NS and Standard definitions

lemma *divideR-eq-divide*: $x /_R\ y = x\ /\ y$
⟨*proof*⟩

Now equivalence between NSDERIV and DERIV

lemma *NSDERIV-DERIV-iff*: (*NSDERIV f x* :> *D*) = (*DERIV f x* :> *D*)
⟨*proof*⟩

lemma *NSDERIV-pow*: *NSDERIV* (%*x*. *x* ^ *n*) *x* :> *real n* ∗ (*x* ^ (*n* − *Suc 0*))
⟨*proof*⟩

Derivative of inverse

lemma *NSDERIV-inverse-fun*:
  fixes *x* :: *'a*::{*real-normed-field*,*recpower*}
  shows [| *NSDERIV f x* :> *d*; *f*(*x*) ≠ *0* |]
      ==> *NSDERIV* (%*x*. *inverse*(*f x*)) *x* :> (− (*d* ∗ *inverse*(*f*(*x*) ^ *Suc* (*Suc*
0))))
⟨*proof*⟩

Derivative of quotient

lemma *NSDERIV-quotient*:
  fixes *x* :: *'a*::{*real-normed-field*,*recpower*}
  shows [| *NSDERIV f x* :> *d*; *NSDERIV g x* :> *e*; *g*(*x*) ≠ *0* |]
      ==> *NSDERIV* (%*y*. *f*(*y*) / (*g y*)) *x* :> (*d*∗*g*(*x*)
                  − (*e*∗*f*(*x*))) / (*g*(*x*) ^ *Suc* (*Suc 0*))
⟨*proof*⟩

lemma *CARAT-NSDERIV*: *NSDERIV f x* :> *l* ==>
    ∃ *g*. (∀ *z*. *f z* − *f x* = *g z* ∗ (*z*−*x*)) & *isNSCont g x* & *g x* = *l*
⟨*proof*⟩

lemma *hypreal-eq-minus-iff3*: (*x* = *y* + *z*) = (*x* + −*z* = (*y*::*hypreal*))
⟨*proof*⟩

lemma *CARAT-DERIVD*:
  assumes *all*: ∀ *z*. *f z* − *f x* = *g z* ∗ (*z*−*x*)
    and *nsc*: *isNSCont g x*
  shows *NSDERIV f x* :> *g x*
⟨*proof*⟩

15.1.2 Differentiability predicate

lemma *NSdifferentiableD*: *f NSdifferentiable x* ==> ∃ *D*. *NSDERIV f x* :> *D*
⟨*proof*⟩

lemma *NSdifferentiableI*: *NSDERIV f x :> D ==> f NSdifferentiable x*
⟨*proof*⟩

15.2 (NS) Increment

lemma *incrementI*:
f NSdifferentiable x ==>
*increment f x h = (*f* f) (hypreal-of-real(x) + h) −*
hypreal-of-real (f x)
⟨*proof*⟩

lemma *incrementI2*: *NSDERIV f x :> D ==>*
*increment f x h = (*f* f) (hypreal-of-real(x) + h) −*
hypreal-of-real (f x)
⟨*proof*⟩

lemma *increment-thm*: [| *NSDERIV f x :> D*; $h \in$ *Infinitesimal*; $h \neq 0$ |]
==> ∃ *e* ∈ *Infinitesimal*. *increment f x h = hypreal-of-real(D)*h + e*h*
⟨*proof*⟩

lemma *increment-thm2*:
[| *NSDERIV f x :> D*; $h \approx 0$; $h \neq 0$ |]
==> ∃ *e* ∈ *Infinitesimal*. *increment f x h =*
*hypreal-of-real(D)*h + e*h*
⟨*proof*⟩

lemma *increment-approx-zero*: [| *NSDERIV f x :> D*; $h \approx 0$; $h \neq 0$ |]
==> *increment f x h* ≈ 0
⟨*proof*⟩

end

16 HTranscendental: Nonstandard Extensions of Transcendental Functions

theory *HTranscendental*
imports *Transcendental HSeries HDeriv*
begin

definition
exphr :: *real => hypreal* **where**
— define exponential function using standard part
*exphr x = st(sumhr (0, whn, %n. inverse(real (fact n)) * (x ˆ n)))*

definition
sinhr :: *real* => *hypreal* **where**
sinhr *x* = *st*(*sumhr* (*0*, *whn*, %*n*. (*if even*(*n*) *then 0 else*
((−*1*) ^ ((*n* − *1*) *div 2*))/(*real* (*fact n*))) ∗ (*x* ^ *n*)))

definition
coshr :: *real* => *hypreal* **where**
coshr *x* = *st*(*sumhr* (*0*, *whn*, %*n*. (*if even*(*n*) *then*
((−*1*) ^ (*n div 2*))/(*real* (*fact n*)) *else 0*) ∗ *x* ^ *n*))

16.1 Nonstandard Extension of Square Root Function

lemma *STAR-sqrt-zero* [*simp*]: (∗*f*∗ *sqrt*) *0* = *0*
⟨*proof*⟩

lemma *STAR-sqrt-one* [*simp*]: (∗*f*∗ *sqrt*) *1* = *1*
⟨*proof*⟩

lemma *hypreal-sqrt-pow2-iff*: ((∗*f*∗ *sqrt*)(*x*) ^ *2* = *x*) = (*0* ≤ *x*)
⟨*proof*⟩

lemma *hypreal-sqrt-gt-zero-pow2*: !!*x*. *0* < *x* ==> (∗*f*∗ *sqrt*) (*x*) ^ *2* = *x*
⟨*proof*⟩

lemma *hypreal-sqrt-pow2-gt-zero*: *0* < *x* ==> *0* < (∗*f*∗ *sqrt*) (*x*) ^ *2*
⟨*proof*⟩

lemma *hypreal-sqrt-not-zero*: *0* < *x* ==> (∗*f*∗ *sqrt*) (*x*) ≠ *0*
⟨*proof*⟩

lemma *hypreal-inverse-sqrt-pow2*:
0 < *x* ==> *inverse* ((∗*f*∗ *sqrt*)(*x*)) ^ *2* = *inverse x*
⟨*proof*⟩

lemma *hypreal-sqrt-mult-distrib*:
!!*x y*. [|*0* < *x*; *0* <*y* |] ==>
(∗*f*∗ *sqrt*)(*x*∗*y*) = (∗*f*∗ *sqrt*)(*x*) ∗ (∗*f*∗ *sqrt*)(*y*)
⟨*proof*⟩

lemma *hypreal-sqrt-mult-distrib2*:
[|*0*≤*x*; *0*≤*y* |] ==>
(∗*f*∗ *sqrt*)(*x*∗*y*) = (∗*f*∗ *sqrt*)(*x*) ∗ (∗*f*∗ *sqrt*)(*y*)
⟨*proof*⟩

lemma *hypreal-sqrt-approx-zero* [*simp*]:
0 < *x* ==> ((∗*f*∗ *sqrt*)(*x*) @= *0*) = (*x* @= *0*)
⟨*proof*⟩

lemma *hypreal-sqrt-approx-zero2* [*simp*]:

$0 \le x$ ==> ((*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*: $0 \le x$ ==> $0 \le$ (*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-of-nat 2)) = abs(x)
⟨proof⟩

lemma *star-sqrt-HFinite*: $[\![x \in HFinite;\ 0 \le x]\!] \Longrightarrow$ (*f* sqrt) $x \in$ HFinite
⟨proof⟩

lemma *st-hypreal-sqrt*:
[| $x \in$ HFinite; $0 \le x$ |] ==> st((*f* sqrt) x) = (*f* sqrt)(st x)
⟨proof⟩

lemma *hypreal-sqrt-sum-squares-ge1* [*simp*]: !!x y. $x \le$ (*f* sqrt)(x ^ 2 + y ^ 2)
⟨proof⟩

lemma *HFinite-hypreal-sqrt*:
[| $0 \le x$; $x \in$ HFinite |] ==> (*f* sqrt) $x \in$ HFinite
⟨proof⟩

lemma *HFinite-hypreal-sqrt-imp-HFinite*:
[| $0 \le x$; (*f* sqrt) $x \in$ HFinite |] ==> $x \in$ HFinite
⟨proof⟩

lemma *HFinite-hypreal-sqrt-iff* [*simp*]:
$0 \le x$ ==> ((*f* sqrt) $x \in$ HFinite) = ($x \in$ HFinite)
⟨proof⟩

lemma *HFinite-sqrt-sum-squares* [*simp*]:
((*f* *sqrt*)(*x**x* + *y**y*) ∈ *HFinite*) = (*x**x* + *y**y* ∈ *HFinite*)
⟨*proof*⟩

lemma *Infinitesimal-hypreal-sqrt*:
[| *0* ≤ *x*; *x* ∈ *Infinitesimal* |] ==> (*f* *sqrt*) *x* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *Infinitesimal-hypreal-sqrt-imp-Infinitesimal*:
[| *0* ≤ *x*; (*f* *sqrt*) *x* ∈ *Infinitesimal* |] ==> *x* ∈ *Infinitesimal*
⟨*proof*⟩

lemma *Infinitesimal-hypreal-sqrt-iff* [*simp*]:
0 ≤ *x* ==> ((*f* *sqrt*) *x* ∈ *Infinitesimal*) = (*x* ∈ *Infinitesimal*)
⟨*proof*⟩

lemma *Infinitesimal-sqrt-sum-squares* [*simp*]:
((*f* *sqrt*)(*x**x* + *y**y*) ∈ *Infinitesimal*) = (*x**x* + *y**y* ∈ *Infinitesimal*)
⟨*proof*⟩

lemma *HInfinite-hypreal-sqrt*:
[| *0* ≤ *x*; *x* ∈ *HInfinite* |] ==> (*f* *sqrt*) *x* ∈ *HInfinite*
⟨*proof*⟩

lemma *HInfinite-hypreal-sqrt-imp-HInfinite*:
[| *0* ≤ *x*; (*f* *sqrt*) *x* ∈ *HInfinite* |] ==> *x* ∈ *HInfinite*
⟨*proof*⟩

lemma *HInfinite-hypreal-sqrt-iff* [*simp*]:
0 ≤ *x* ==> ((*f* *sqrt*) *x* ∈ *HInfinite*) = (*x* ∈ *HInfinite*)
⟨*proof*⟩

lemma *HInfinite-sqrt-sum-squares* [*simp*]:
((*f* *sqrt*)(*x**x* + *y**y*) ∈ *HInfinite*) = (*x**x* + *y**y* ∈ *HInfinite*)
⟨*proof*⟩

lemma *HFinite-exp* [*simp*]:
sumhr (*0*, *whn*, %*n*. *inverse* (*real* (*fact n*)) * *x* ^ *n*) ∈ *HFinite*
⟨*proof*⟩

lemma *exphr-zero* [*simp*]: *exphr 0* = *1*
⟨*proof*⟩

lemma *coshr-zero* [*simp*]: *coshr 0* = *1*
⟨*proof*⟩

lemma *STAR-exp-zero-approx-one* [*simp*]: (*f* *exp*) (*0*::*hypreal*) @= *1*
⟨*proof*⟩

lemma *STAR-exp-Infinitesimal*: $x \in$ *Infinitesimal* ==> ($*f*$ *exp*) (x::*hypreal*) @= *1*
⟨*proof*⟩

lemma *STAR-exp-epsilon* [*simp*]: ($*f*$ *exp*) *epsilon* @= *1*
⟨*proof*⟩

lemma *STAR-exp-add*: !!x y. ($*f*$ *exp*)(x + y) = ($*f*$ *exp*) x $*$ ($*f*$ *exp*) y
⟨*proof*⟩

lemma *exphr-hypreal-of-real-exp-eq*: *exphr* x = *hypreal-of-real* (*exp* x)
⟨*proof*⟩

lemma *starfun-exp-ge-add-one-self* [*simp*]: !!x::*hypreal*. $0 \le x$ ==> ($1 + x$) $\le$ ($*f*$ *exp*) x
⟨*proof*⟩

lemma *starfun-exp-HInfinite*:
[| $x \in$ *HInfinite*; $0 \le x$ |] ==> ($*f*$ *exp*) (x::*hypreal*) $\in$ *HInfinite*
⟨*proof*⟩

lemma *starfun-exp-minus*: !!x. ($*f*$ *exp*) ($-x$) = *inverse*(($*f*$ *exp*) x)
⟨*proof*⟩

lemma *starfun-exp-Infinitesimal*:
[| $x \in$ *HInfinite*; $x \le 0$ |] ==> ($*f*$ *exp*) (x::*hypreal*) $\in$ *Infinitesimal*
⟨*proof*⟩

lemma *starfun-exp-gt-one* [*simp*]: !!x::*hypreal*. $0 < x$ ==> $1 <$ ($*f*$ *exp*) x
⟨*proof*⟩

lemma *starfun-ln-exp* [*simp*]: !!x. ($*f*$ *ln*) (($*f*$ *exp*) x) = x
⟨*proof*⟩

lemma *starfun-exp-ln-iff* [*simp*]: !!x. (($*f*$ *exp*)(($*f*$ *ln*) x) = x) = ($0 < x$)
⟨*proof*⟩

lemma *starfun-exp-ln-eq*: !!u x. ($*f*$ *exp*) u = x ==> ($*f*$ *ln*) x = u
⟨*proof*⟩

lemma *starfun-ln-less-self* [*simp*]: !!x. $0 < x$ ==> ($*f*$ *ln*) $x < x$
⟨*proof*⟩

lemma *starfun-ln-ge-zero* [*simp*]: !!x. $1 \le x$ ==> $0 \le$ ($*f*$ *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-abs-exp-cancel*: $\bigwedge x$. |($*f*$ *exp*) (x::*hypreal*)| = ($*f*$ *exp*) x
⟨*proof*⟩

lemma *starfun-exp-less-mono*: $\bigwedge x$ y::*hypreal*. $x < y \Longrightarrow$ ($*f*$ *exp*) $x <$ ($*f*$ *exp*) y
⟨*proof*⟩

lemma *starfun-exp-HFinite*: $x \in$ *HFinite* ==> ($*f*$ *exp*) (x::*hypreal*) $\in$ *HFinite*
⟨*proof*⟩

lemma *starfun-exp-add-HFinite-Infinitesimal-approx*:
[|$x \in$ *Infinitesimal*; $z \in$ *HFinite* |] ==> ($*f*$ *exp*) (z + x::*hypreal*) @= ($*f*$ *exp*) z
⟨*proof*⟩

lemma *starfun-ln-HInfinite*:
[| $x \in$ *HInfinite*; $0 < x$ |] ==> ($*f*$ *ln*) $x \in$ *HInfinite*
⟨*proof*⟩

lemma *starfun-exp-HInfinite-Infinitesimal-disj*:
$x \in$ *HInfinite* ==> ($*f*$ *exp*) $x \in$ *HInfinite* | ($*f*$ *exp*) (x::*hypreal*) $\in$ *Infinitesimal*
⟨*proof*⟩

lemma *starfun-ln-HFinite-not-Infinitesimal*:
[| $x \in$ *HFinite* $-$ *Infinitesimal*; $0 < x$ |] ==> ($*f*$ *ln*) $x \in$ *HFinite*
⟨*proof*⟩

lemma *starfun-ln-Infinitesimal-HInfinite*:
[| $x \in$ *Infinitesimal*; $0 < x$ |] ==> ($*f*$ *ln*) $x \in$ *HInfinite*
⟨*proof*⟩

lemma *starfun-ln-less-zero*: !!x. [| $0 < x$; $x < 1$ |] ==> ($*f*$ *ln*) $x < 0$
⟨*proof*⟩

lemma *starfun-ln-Infinitesimal-less-zero*:

*[| x ∈ Infinitesimal; 0 < x |] ==> (*f* ln) x < 0*
⟨*proof*⟩

lemma *starfun-ln-HInfinite-gt-zero*:
*[| x ∈ HInfinite; 0 < x |] ==> 0 < (*f* ln) x*
⟨*proof*⟩

lemma *HFinite-sin* [*simp*]:
sumhr (0, whn, %n. (if even(n) then 0 else
*(−1 ^ ((n − 1) div 2))/(real (fact n))) * x ^ n)*
∈ HFinite
⟨*proof*⟩

lemma *STAR-sin-zero* [*simp*]: *(*f* sin) 0 = 0*
⟨*proof*⟩

lemma *STAR-sin-Infinitesimal* [*simp*]: *x ∈ Infinitesimal ==> (*f* sin) x @= x*
⟨*proof*⟩

lemma *HFinite-cos* [*simp*]:
sumhr (0, whn, %n. (if even(n) then
(−1 ^ (n div 2))/(real (fact n)) else
*0) * x ^ n) ∈ HFinite*
⟨*proof*⟩

lemma *STAR-cos-zero* [*simp*]: *(*f* cos) 0 = 1*
⟨*proof*⟩

lemma *STAR-cos-Infinitesimal* [*simp*]: *x ∈ Infinitesimal ==> (*f* cos) x @= 1*
⟨*proof*⟩

lemma *STAR-tan-zero* [*simp*]: *(*f* tan) 0 = 0*
⟨*proof*⟩

lemma *STAR-tan-Infinitesimal*: *x ∈ Infinitesimal ==> (*f* tan) x @= x*
⟨*proof*⟩

lemma *STAR-sin-cos-Infinitesimal-mult*:
*x ∈ Infinitesimal ==> (*f* sin) x * (*f* cos) x @= x*
⟨*proof*⟩

lemma *HFinite-pi*: *hypreal-of-real pi ∈ HFinite*
⟨*proof*⟩

lemma *lemma-split-hypreal-of-real*:
N ∈ *HNatInfinite*
==> *hypreal-of-real a* =
hypreal-of-hypnat N ∗ (*inverse*(*hypreal-of-hypnat N*) ∗ *hypreal-of-real a*)
⟨*proof*⟩

lemma *STAR-sin-Infinitesimal-divide*:
[|*x* ∈ *Infinitesimal*; *x* ≠ *0* |] ==> (∗*f*∗ *sin*) *x*/*x* @= *1*
⟨*proof*⟩

lemma *lemma-sin-pi*:
n ∈ *HNatInfinite*
==> (∗*f*∗ *sin*) (*inverse* (*hypreal-of-hypnat n*))/(*inverse* (*hypreal-of-hypnat n*)) @= *1*
⟨*proof*⟩

lemma *STAR-sin-inverse-HNatInfinite*:
n ∈ *HNatInfinite*
==> (∗*f*∗ *sin*) (*inverse* (*hypreal-of-hypnat n*)) ∗ *hypreal-of-hypnat n* @= *1*
⟨*proof*⟩

lemma *Infinitesimal-pi-divide-HNatInfinite*:
N ∈ *HNatInfinite*
==> *hypreal-of-real pi*/(*hypreal-of-hypnat N*) ∈ *Infinitesimal*
⟨*proof*⟩

lemma *pi-divide-HNatInfinite-not-zero* [*simp*]:
N ∈ *HNatInfinite* ==> *hypreal-of-real pi*/(*hypreal-of-hypnat N*) ≠ *0*
⟨*proof*⟩

lemma *STAR-sin-pi-divide-HNatInfinite-approx-pi*:
n ∈ *HNatInfinite*
==> (∗*f*∗ *sin*) (*hypreal-of-real pi*/(*hypreal-of-hypnat n*)) ∗ *hypreal-of-hypnat n*
@= *hypreal-of-real pi*
⟨*proof*⟩

lemma *STAR-sin-pi-divide-HNatInfinite-approx-pi2*:
n ∈ *HNatInfinite*
==> *hypreal-of-hypnat n* ∗
(∗*f*∗ *sin*) (*hypreal-of-real pi*/(*hypreal-of-hypnat n*))
@= *hypreal-of-real pi*
⟨*proof*⟩

lemma *starfunNat-pi-divide-n-Infinitesimal*:

$N \in HNatInfinite ==> (*f* (\%x.\ pi\ /\ real\ x))\ N \in Infinitesimal$
⟨proof⟩

lemma *STAR-sin-pi-divide-n-approx*:
$N \in HNatInfinite ==>$
$(*f* sin) ((*f* (\%x.\ pi\ /\ real\ x))\ N)\ @=$
$hypreal\text{-}of\text{-}real\ pi/(hypreal\text{-}of\text{-}hypnat\ N)$
⟨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*:
$x \in Infinitesimal ==> (*f* cos)\ x\ @=\ 1 - x\ \hat{}\ 2$
⟨proof⟩

lemma *STAR-cos-Infinitesimal-approx2*:
$x \in Infinitesimal ==> (*f* cos)\ x\ @=\ 1 - (x\ \hat{}\ 2)/2$
⟨proof⟩

end

17 NSCA: Non-Standard Complex Analysis

theory *NSCA*
imports *NSComplex HTranscendental*
begin

abbreviation

SComplex :: *hcomplex set* **where**
SComplex ≡ *Standard*

definition — standard part map
stc :: *hcomplex* => *hcomplex* **where**
[*code del*]: $stc\ x = (SOME\ r.\ x \in HFinite\ \&\ r{:}SComplex\ \&\ r\ @=\ x)$

17.1 Closure Laws for SComplex, the Standard Complex Numbers

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

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

lemma *SReal-hcmod-hcomplex-of-complex* [*simp*]:
hcmod (*hcomplex-of-complex* r) $\in$ *Reals*
⟨*proof*⟩

lemma *SReal-hcmod-number-of* [*simp*]: *hcmod* (*number-of* w ::*hcomplex*) $\in$ *Reals*
⟨*proof*⟩

lemma *SReal-hcmod-SComplex*: $x \in SComplex ==>$ *hcmod* $x \in$ *Reals*
⟨*proof*⟩

lemma *SComplex-divide-number-of*:
$r \in SComplex ==> r/($*number-of* w::*hcomplex*$) \in SComplex$
⟨*proof*⟩

lemma *SComplex-UNIV-complex*:
$\{x.$ *hcomplex-of-complex* $x \in SComplex\} = ($*UNIV*::*complex set*$)$
⟨*proof*⟩

lemma *SComplex-iff*: $(x \in SComplex) = (\exists\, y.\ x =$ *hcomplex-of-complex* $y)$
⟨*proof*⟩

lemma *hcomplex-of-complex-image*:
hcomplex-of-complex ‘(*UNIV*::*complex set*) = *SComplex*
⟨*proof*⟩

lemma *inv-hcomplex-of-complex-image*: *inv hcomplex-of-complex* ‘*SComplex* = *UNIV*
⟨*proof*⟩

lemma *SComplex-hcomplex-of-complex-image*:
$[|\ \exists\, x.\ x$: P; $P \le SComplex\ |] ==> \exists\, Q.\ P =$ *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*⟩

17.2 The Finite Elements form a Subring

lemma *HFinite-hcmod-hcomplex-of-complex* [*simp*]:
$hcmod\ (hcomplex\text{-}of\text{-}complex\ r) \in HFinite$
⟨*proof*⟩

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

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

17.3 The Complex Infinitesimals form a Subring

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

lemma *Infinitesimal-hcmod-iff*:
$(z \in Infinitesimal) = (hcmod\ z \in Infinitesimal)$
⟨*proof*⟩

lemma *HInfinite-hcmod-iff*: $(z \in HInfinite) = (hcmod\ z \in HInfinite)$
⟨*proof*⟩

lemma *HFinite-diff-Infinitesimal-hcmod*:
$x \in HFinite - Infinitesimal ==> hcmod\ x \in HFinite - Infinitesimal$
⟨*proof*⟩

lemma *hcmod-less-Infinitesimal*:
$[|\ e \in Infinitesimal;\ hcmod\ x < hcmod\ e\ |] ==> x \in Infinitesimal$
⟨*proof*⟩

lemma *hcmod-le-Infinitesimal*:
$[|\ e \in Infinitesimal;\ hcmod\ x \le hcmod\ e\ |] ==> x \in Infinitesimal$
⟨*proof*⟩

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

lemma *Infinitesimal-interval2-hcmod*:
$[|\ e \in Infinitesimal;$
$e' \in Infinitesimal;$
$hcmod\ e' \le hcmod\ x\ ;\ hcmod\ x \le hcmod\ e$
$|] ==> x \in Infinitesimal$
⟨*proof*⟩

17.4 The "Infinitely Close" Relation

lemma *approx-SComplex-mult-cancel-zero*:
$[| \ a \in SComplex; \ a \neq 0; \ a*x \ @= \ 0 \ |] ==> x \ @= \ 0$
⟨*proof*⟩

lemma *approx-mult-SComplex1*: $[| \ a \in SComplex; \ x \ @= \ 0 \ |] ==> x*a \ @= \ 0$
⟨*proof*⟩

lemma *approx-mult-SComplex2*: $[| \ a \in SComplex; \ x \ @= \ 0 \ |] ==> a*x \ @= \ 0$
⟨*proof*⟩

lemma *approx-mult-SComplex-zero-cancel-iff* [*simp*]:
$[|a \in SComplex; \ a \neq 0 \ |] ==> (a*x \ @= \ 0) = (x \ @= \ 0)$
⟨*proof*⟩

lemma *approx-SComplex-mult-cancel*:
$[| \ a \in SComplex; \ a \neq 0; \ a* \ w \ @= \ a*z \ |] ==> w \ @= \ z$
⟨*proof*⟩

lemma *approx-SComplex-mult-cancel-iff1* [*simp*]:
$[| \ a \in SComplex; \ a \neq 0|] ==> (a* \ w \ @= \ a*z) = (w \ @= \ z)$
⟨*proof*⟩

lemma *approx-hcmod-approx-zero*: $(x \ @= \ y) = (hcmod \ (y - x) \ @= \ 0)$
⟨*proof*⟩

lemma *approx-approx-zero-iff*: $(x \ @= \ 0) = (hcmod \ x \ @= \ 0)$
⟨*proof*⟩

lemma *approx-minus-zero-cancel-iff* [*simp*]: $(-x \ @= \ 0) = (x \ @= \ 0)$
⟨*proof*⟩

lemma *Infinitesimal-hcmod-add-diff*:
$u \ @= \ 0 ==> hcmod(x + u) - hcmod \ x \in Infinitesimal$
⟨*proof*⟩

lemma *approx-hcmod-add-hcmod*: $u \ @= \ 0 ==> hcmod(x + u) \ @= \ hcmod \ x$
⟨*proof*⟩

17.5 Zero is the Only Infinitesimal Complex Number

lemma *Infinitesimal-less-SComplex*:
$[| \ x \in SComplex; \ y \in Infinitesimal; \ 0 < hcmod \ x \ |] ==> hcmod \ y < hcmod \ x$
⟨*proof*⟩

lemma *SComplex-Int-Infinitesimal-zero*: $SComplex \ Int \ Infinitesimal = \{0\}$
⟨*proof*⟩

lemma *SComplex-Infinitesimal-zero*:
  $[| \ x \in SComplex; \ x \in Infinitesimal|] ==> x = 0$
⟨*proof*⟩

lemma *SComplex-HFinite-diff-Infinitesimal*:
  $[| \ x \in SComplex; \ x \neq 0 \ |] ==> x \in HFinite - Infinitesimal$
⟨*proof*⟩

lemma *hcomplex-of-complex-HFinite-diff-Infinitesimal*:
  $hcomplex\text{-}of\text{-}complex\ x \neq 0$
  $==> hcomplex\text{-}of\text{-}complex\ x \in HFinite - Infinitesimal$
⟨*proof*⟩

lemma *number-of-not-Infinitesimal* [*simp*]:
  $number\text{-}of\ w \neq (0{::}hcomplex) ==> (number\text{-}of\ w{::}hcomplex) \notin Infinitesimal$
⟨*proof*⟩

lemma *approx-SComplex-not-zero*:
  $[| \ y \in SComplex; \ x \ @= \ y; \ y \neq \ 0 \ |] ==> x \neq 0$
⟨*proof*⟩

lemma *SComplex-approx-iff*:
  $[|x \in SComplex; \ y \in SComplex|] ==> (x \ @= \ y) = (x = y)$
⟨*proof*⟩

lemma *number-of-Infinitesimal-iff* [*simp*]:
  $((number\text{-}of\ w :: hcomplex) \in Infinitesimal) =$
  $(number\text{-}of\ w = (0{::}hcomplex))$
⟨*proof*⟩

lemma *approx-unique-complex*:
  $[| \ r \in SComplex; \ s \in SComplex; \ r \ @= \ x; \ s \ @= \ x|] ==> r = s$
⟨*proof*⟩

17.6 Properties of *hRe*, *hIm* and *HComplex*

lemma *abs-hRe-le-hcmod*: $\bigwedge x. \ |hRe\ x| \leq hcmod\ x$
⟨*proof*⟩

lemma *abs-hIm-le-hcmod*: $\bigwedge x. \ |hIm\ x| \leq hcmod\ x$
⟨*proof*⟩

lemma *Infinitesimal-hRe*: $x \in Infinitesimal \Longrightarrow hRe\ x \in Infinitesimal$
⟨*proof*⟩

lemma *Infinitesimal-hIm*: $x \in Infinitesimal \Longrightarrow hIm\ x \in Infinitesimal$
⟨*proof*⟩

lemma *real-sqrt-lessI*: $[\![0 < u; x < u^2]\!] \Longrightarrow sqrt\ x < u$

$\langle proof \rangle$

lemma *hypreal-sqrt-lessI*:
$\bigwedge x\ u.\ [\![0 < u; x < u^2]\!] \Longrightarrow (\ {*f*}\ sqrt)\ x < u$
$\langle proof \rangle$

lemma *hypreal-sqrt-ge-zero*: $\bigwedge x.\ 0 \leq x \Longrightarrow 0 \leq (\ {*f*}\ sqrt)\ x$
$\langle proof \rangle$

lemma *Infinitesimal-sqrt*:
$[\![x \in Infinitesimal;\ 0 \leq x]\!] \Longrightarrow (\ {*f*}\ sqrt)\ x \in Infinitesimal$
$\langle proof \rangle$

lemma *Infinitesimal-HComplex*:
$[\![x \in Infinitesimal;\ y \in Infinitesimal]\!] \Longrightarrow HComplex\ x\ y \in Infinitesimal$
$\langle proof \rangle$

lemma *hcomplex-Infinitesimal-iff*:
$(x \in Infinitesimal) = (hRe\ x \in Infinitesimal \wedge hIm\ x \in Infinitesimal)$
$\langle proof \rangle$

lemma *hRe-diff* [*simp*]: $\bigwedge x\ y.\ hRe\ (x - y) = hRe\ x - hRe\ y$
$\langle proof \rangle$

lemma *hIm-diff* [*simp*]: $\bigwedge x\ y.\ hIm\ (x - y) = hIm\ x - hIm\ y$
$\langle proof \rangle$

lemma *approx-hRe*: $x \approx y \Longrightarrow hRe\ x \approx hRe\ y$
$\langle proof \rangle$

lemma *approx-hIm*: $x \approx y \Longrightarrow hIm\ x \approx hIm\ y$
$\langle proof \rangle$

lemma *approx-HComplex*:
$[\![a \approx b;\ c \approx d]\!] \Longrightarrow HComplex\ a\ c \approx HComplex\ b\ d$
$\langle proof \rangle$

lemma *hcomplex-approx-iff*:
$(x \approx y) = (hRe\ x \approx hRe\ y \wedge hIm\ x \approx hIm\ y)$
$\langle proof \rangle$

lemma *HFinite-hRe*: $x \in HFinite \Longrightarrow hRe\ x \in HFinite$
$\langle proof \rangle$

lemma *HFinite-hIm*: $x \in HFinite \Longrightarrow hIm\ x \in HFinite$
$\langle proof \rangle$

lemma *HFinite-HComplex*:
$[\![x \in \mathit{HFinite};\ y \in \mathit{HFinite}]\!] \Longrightarrow \mathit{HComplex}\ x\ y \in \mathit{HFinite}$
⟨*proof*⟩

lemma *hcomplex-HFinite-iff*:
$(x \in \mathit{HFinite}) = (\mathit{hRe}\ x \in \mathit{HFinite} \wedge \mathit{hIm}\ x \in \mathit{HFinite})$
⟨*proof*⟩

lemma *hcomplex-HInfinite-iff*:
$(x \in \mathit{HInfinite}) = (\mathit{hRe}\ x \in \mathit{HInfinite} \vee \mathit{hIm}\ x \in \mathit{HInfinite})$
⟨*proof*⟩

lemma *hcomplex-of-hypreal-approx-iff* [*simp*]:
$(\mathit{hcomplex\text{-}of\text{-}hypreal}\ x\ @{=}\ \mathit{hcomplex\text{-}of\text{-}hypreal}\ z) = (x\ @{=}\ z)$
⟨*proof*⟩

lemma *Standard-HComplex*:
$[\![x \in \mathit{Standard};\ y \in \mathit{Standard}]\!] \Longrightarrow \mathit{HComplex}\ x\ y \in \mathit{Standard}$
⟨*proof*⟩

lemma *stc-part-Ex*: $x{:}\mathit{HFinite} ==> \exists\, t \in \mathit{SComplex}.\ x\ @{=}\ t$
⟨*proof*⟩

lemma *stc-part-Ex1*: $x{:}\mathit{HFinite} ==> \mathit{EX}!\ t.\ t \in \mathit{SComplex}\ \&\ \ x\ @{=}\ t$
⟨*proof*⟩

lemmas *hcomplex-of-complex-approx-inverse* =
hcomplex-of-complex-HFinite-diff-Infinitesimal [*THEN* [*2*] *approx-inverse*]

17.7 Theorems About Monads

lemma *monad-zero-hcmod-iff*: $(x \in \mathit{monad}\ 0) = (\mathit{hcmod}\ x{:}\mathit{monad}\ 0)$
⟨*proof*⟩

17.8 Theorems About Standard Part

lemma *stc-approx-self*: $x \in \mathit{HFinite} ==> \mathit{stc}\ x\ @{=}\ x$
⟨*proof*⟩

lemma *stc-SComplex*: $x \in \mathit{HFinite} ==> \mathit{stc}\ x \in \mathit{SComplex}$
⟨*proof*⟩

lemma *stc-HFinite*: $x \in \mathit{HFinite} ==> \mathit{stc}\ x \in \mathit{HFinite}$
⟨*proof*⟩

lemma *stc-unique*: $[\![y \in \mathit{SComplex};\ y \approx x]\!] \Longrightarrow \mathit{stc}\ x = y$
⟨*proof*⟩

lemma *stc-SComplex-eq* [*simp*]: $x \in \mathit{SComplex} ==> \mathit{stc}\ x = x$

⟨*proof*⟩

lemma *stc-hcomplex-of-complex*:
stc (*hcomplex-of-complex* x) = *hcomplex-of-complex* x
⟨*proof*⟩

lemma *stc-eq-approx*:
[| $x \in$ *HFinite*; $y \in$ *HFinite*; *stc* x = *stc* y |] ==> x @= y
⟨*proof*⟩

lemma *approx-stc-eq*:
[| $x \in$ *HFinite*; $y \in$ *HFinite*; x @= y |] ==> *stc* x = *stc* y
⟨*proof*⟩

lemma *stc-eq-approx-iff*:
[| $x \in$ *HFinite*; $y \in$ *HFinite*|] ==> (x @= y) = (*stc* x = *stc* y)
⟨*proof*⟩

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

lemma *stc-Infinitesimal-add-SComplex2*:
[| $x \in$ *SComplex*; $e \in$ *Infinitesimal* |] ==> *stc*(e + x) = x
⟨*proof*⟩

lemma *HFinite-stc-Infinitesimal-add*:
$x \in$ *HFinite* ==> $\exists\, e \in$ *Infinitesimal*. x = *stc*(x) + e
⟨*proof*⟩

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

lemma *stc-number-of* [*simp*]: *stc* (*number-of* w) = *number-of* w
⟨*proof*⟩

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

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

lemma *stc-minus*: $y \in$ *HFinite* ==> *stc*($-y$) = $-$*stc*(y)
⟨*proof*⟩

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

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

lemma *stc-Infinitesimal*: $x \in Infinitesimal ==> stc\ x = 0$
⟨*proof*⟩

lemma *stc-not-Infinitesimal*: $stc(x) \neq 0 ==> x \notin Infinitesimal$
⟨*proof*⟩

lemma *stc-inverse*:
$[| x \in HFinite; stc\ x \neq 0 |]$
$==> stc(inverse\ x) = inverse\ (stc\ x)$
⟨*proof*⟩

lemma *stc-divide* [*simp*]:
$[| x \in HFinite; y \in HFinite; stc\ y \neq 0 |]$
$==> stc(x/y) = (stc\ x) / (stc\ y)$
⟨*proof*⟩

lemma *stc-idempotent* [*simp*]: $x \in HFinite ==> stc(stc(x)) = stc(x)$
⟨*proof*⟩

lemma *HFinite-HFinite-hcomplex-of-hypreal*:
$z \in HFinite ==> hcomplex\text{-}of\text{-}hypreal\ z \in HFinite$
⟨*proof*⟩

lemma *SComplex-SReal-hcomplex-of-hypreal*:
$x \in Reals ==> hcomplex\text{-}of\text{-}hypreal\ x \in SComplex$
⟨*proof*⟩

lemma *stc-hcomplex-of-hypreal*:
$z \in HFinite ==> stc(hcomplex\text{-}of\text{-}hypreal\ z) = hcomplex\text{-}of\text{-}hypreal\ (st\ z)$
⟨*proof*⟩

lemma *Infinitesimal-hcnj-iff* [*simp*]:
$(hcnj\ z \in Infinitesimal) = (z \in Infinitesimal)$
⟨*proof*⟩

lemma *Infinitesimal-hcomplex-of-hypreal-epsilon* [*simp*]:
$hcomplex\text{-}of\text{-}hypreal\ epsilon \in Infinitesimal$
⟨*proof*⟩

end

18 CStar: Star-transforms in NSA, Extending Sets of Complex Numbers and Complex Functions

theory *CStar*
imports *NSCA*
begin

18.1 Properties of the *-Transform Applied to Sets of Reals

lemma *STARC-hcomplex-of-complex-Int*:
$*s*$ *X Int SComplex* $=$ *hcomplex-of-complex* ' *X*
$\langle proof \rangle$

lemma *lemma-not-hcomplexA*:
$x \notin$ *hcomplex-of-complex* ' A $==>$ $\forall\, y \in A.\ x \neq$ *hcomplex-of-complex* y
$\langle proof \rangle$

18.2 Theorems about Nonstandard Extensions of Functions

lemma *starfunC-hcpow*: $!!Z.\ (\ *f*\ (\%z.\ z\ \hat{}\ n))\ Z = Z$ *pow hypnat-of-nat* n
$\langle proof \rangle$

lemma *starfunCR-cmod*: $*f*$ *cmod* $=$ *hcmod*
$\langle proof \rangle$

18.3 Internal Functions - Some Redundancy With *f* Now

lemma *starfun-Re*: $(\ *f*\ (\lambda x.\ Re\ (f\,x))) = (\lambda x.\ hRe\ ((\ *f*\ f)\ x))$
$\langle proof \rangle$

lemma *starfun-Im*: $(\ *f*\ (\lambda x.\ Im\ (f\,x))) = (\lambda x.\ hIm\ ((\ *f*\ f)\ x))$
$\langle proof \rangle$

lemma *starfunC-eq-Re-Im-iff*:
$((\ *f*\ f)\ x = z) = (((\ *f*\ (\%x.\ Re(f\,x)))\ x = hRe\ (z))\ \&$
$((\ *f*\ (\%x.\ Im(f\,x)))\ x = hIm\ (z)))$
$\langle proof \rangle$

lemma *starfunC-approx-Re-Im-iff*:
$((\ *f*\ f)\ x\ @{=}\ z) = (((\ *f*\ (\%x.\ Re(f\,x)))\ x\ @{=}\ hRe\ (z))\ \&$
$((\ *f*\ (\%x.\ Im(f\,x)))\ x\ @{=}\ hIm\ (z)))$
$\langle proof \rangle$

end

19 CLim: Limits, Continuity and Differentiation for Complex Functions

theory *CLim*
imports *CStar*
begin

declare *hypreal-epsilon-not-zero* [*simp*]

lemma *lemma-complex-mult-inverse-squared* [*simp*]:
$x \neq (0::complex) \Longrightarrow (x * inverse(x)\ \hat{}\ 2) = inverse\ x$
⟨*proof*⟩

Changing the quantified variable. Install earlier?

lemma *all-shift*: $(\forall\, x::'a::comm\text{-}ring\text{-}1.\ P\ x) = (\forall\, x.\ P\ (x-a))$
⟨*proof*⟩

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

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

19.1 Limit of Complex to Complex Function

lemma *NSLIM-Re*: $f\ --\ a\ --NS{>}\ L ==> (\%x.\ Re(f\,x))\ --\ a\ --NS{>}\ Re(L)$
⟨*proof*⟩

lemma *NSLIM-Im*: $f\ --\ a\ --NS{>}\ L ==> (\%x.\ Im(f\,x))\ --\ a\ --NS{>}\ Im(L)$
⟨*proof*⟩

lemma *LIM-Re*: $f\ --\ a\ --{>}\ L ==> (\%x.\ Re(f\,x))\ --\ a\ --{>}\ Re(L)$
⟨*proof*⟩

lemma *LIM-Im*: $f\ --\ a\ --{>}\ L ==> (\%x.\ Im(f\,x))\ --\ a\ --{>}\ Im(L)$
⟨*proof*⟩

lemma *LIM-cnj*: $f\ --\ a\ --{>}\ L ==> (\%x.\ cnj\ (f\,x))\ --\ a\ --{>}\ cnj\ L$
⟨*proof*⟩

lemma *LIM-cnj-iff*: $((\%x.\ cnj\ (f\,x))\ --\ a\ --{>}\ cnj\ L) = (f\ --\ a\ --{>}\ L)$
⟨*proof*⟩

lemma *starfun-norm*: $(\ {*}f{*}\ (\lambda x.\ norm\ (f\,x))) = (\lambda x.\ hnorm\ ((\ {*}f{*}\ f)\ x))$
⟨*proof*⟩

lemma *star-of-Re* [*simp*]: *star-of* (*Re x*) = *hRe* (*star-of x*)
⟨*proof*⟩

lemma *star-of-Im* [*simp*]: *star-of* (*Im x*) = *hIm* (*star-of x*)
⟨*proof*⟩

lemma *NSCLIM-NSCRLIM-iff*:
(*f* −− *x* −−*NS*> *L*) = ((%*y*. *cmod*(*f y* − *L*)) −− *x* −−*NS*> *0*)
⟨*proof*⟩

lemma *CLIM-CRLIM-iff*: (*f* −− *x* −−> *L*) = ((%*y*. *cmod*(*f y* − *L*)) −− *x* −−> *0*)
⟨*proof*⟩

lemma *NSCLIM-NSCRLIM-iff2*:
(*f* −− *x* −−*NS*> *L*) = ((%*y*. *cmod*(*f y* − *L*)) −− *x* −−*NS*> *0*)
⟨*proof*⟩

lemma *NSLIM-NSCRLIM-Re-Im-iff*:
(*f* −− *a* −−*NS*> *L*) = ((%*x*. *Re*(*f x*)) −− *a* −−*NS*> *Re*(*L*) &
(%*x*. *Im*(*f x*)) −− *a* −−*NS*> *Im*(*L*))
⟨*proof*⟩

lemma *LIM-CRLIM-Re-Im-iff*:
(*f* −− *a* −−> *L*) = ((%*x*. *Re*(*f x*)) −− *a* −−> *Re*(*L*) &
(%*x*. *Im*(*f x*)) −− *a* −−> *Im*(*L*))
⟨*proof*⟩

19.2 Continuity

lemma *NSLIM-isContc-iff*:
(*f* −− *a* −−*NS*> *f a*) = ((%*h*. *f*(*a* + *h*)) −− *0* −−*NS*> *f a*)
⟨*proof*⟩

19.3 Functions from Complex to Reals

lemma *isNSContCR-cmod* [*simp*]: *isNSCont cmod* (*a*)
⟨*proof*⟩

lemma *isContCR-cmod* [*simp*]: *isCont cmod* (*a*)
⟨*proof*⟩

lemma *isCont-Re*: *isCont f a* ==> *isCont* (%*x*. *Re* (*f x*)) *a*
⟨*proof*⟩

lemma *isCont-Im*: *isCont f a* ==> *isCont* (%*x*. *Im* (*f x*)) *a*
⟨*proof*⟩

19.4 Differentiation of Natural Number Powers

lemma *CDERIV-pow* [*simp*]:
  DERIV (%*x*. *x* ˆ *n*) *x* :> (*complex-of-real* (*real n*)) ∗ (*x* ˆ (*n* − *Suc 0*))
⟨*proof*⟩

Nonstandard version

lemma *NSCDERIV-pow*:
  NSDERIV (%*x*. *x* ˆ *n*) *x* :> *complex-of-real* (*real n*) ∗ (*x* ˆ (*n* − *1*))
⟨*proof*⟩

Can't relax the premise $x \neq (0{::}'a)$: it isn't continuous at zero

lemma *NSCDERIV-inverse*:
  (*x*::*complex*) ≠ *0* ==> *NSDERIV* (%*x*. *inverse*(*x*)) *x* :> (− (*inverse x* ˆ *2*))
⟨*proof*⟩

lemma *CDERIV-inverse*:
  (*x*::*complex*) ≠ *0* ==> *DERIV* (%*x*. *inverse*(*x*)) *x* :> (−(*inverse x* ˆ *2*))
⟨*proof*⟩

19.5 Derivative of Reciprocals (Function *inverse*)

lemma *CDERIV-inverse-fun*:
  [| *DERIV f x* :> *d*; *f*(*x*) ≠ (*0*::*complex*) |]
  ==> *DERIV* (%*x*. *inverse*(*f x*)) *x* :> (− (*d* ∗ *inverse*(*f*(*x*) ˆ *2*)))
⟨*proof*⟩

lemma *NSCDERIV-inverse-fun*:
  [| *NSDERIV f x* :> *d*; *f*(*x*) ≠ (*0*::*complex*) |]
  ==> *NSDERIV* (%*x*. *inverse*(*f x*)) *x* :> (− (*d* ∗ *inverse*(*f*(*x*) ˆ *2*)))
⟨*proof*⟩

19.6 Derivative of Quotient

lemma *CDERIV-quotient*:
  [| *DERIV f x* :> *d*; *DERIV g x* :> *e*; *g*(*x*) ≠ (*0*::*complex*) |]
  ==> *DERIV* (%*y*. *f*(*y*) / (*g y*)) *x* :> (*d*∗*g*(*x*) − (*e*∗*f*(*x*))) / (*g*(*x*) ˆ *2*)
⟨*proof*⟩

lemma *NSCDERIV-quotient*:
  [| *NSDERIV f x* :> *d*; *NSDERIV g x* :> *e*; *g*(*x*) ≠ (*0*::*complex*) |]
  ==> *NSDERIV* (%*y*. *f*(*y*) / (*g y*)) *x* :> (*d*∗*g*(*x*) − (*e*∗*f*(*x*))) / (*g*(*x*) ˆ *2*)
⟨*proof*⟩

19.7 Caratheodory Formulation of Derivative at a Point: Standard Proof

lemma *CARAT-CDERIVD*:
  (∀ *z*. *f z* − *f x* = *g z* ∗ (*z* − *x*)) & *isNSCont g x* & *g x* = *l*
  ==> *NSDERIV f x* :> *l*

⟨*proof*⟩

end

20 HLog: Logarithms: Non-Standard Version

theory *HLog*
imports *Log HTranscendental*
begin

lemma *epsilon-ge-zero* [*simp*]: *0* ≤ *epsilon*
⟨*proof*⟩

lemma *hpfinite-witness*: *epsilon* : {*x*. *0* ≤ *x* & *x* : *HFinite*}
⟨*proof*⟩

definition
powhr :: [*hypreal*,*hypreal*] => *hypreal* (**infixr** *powhr 80*) **where**
[*transfer-unfold*, *code del*]: *x powhr a* = *starfun2* (*op powr*) *x a*

definition
hlog :: [*hypreal*,*hypreal*] => *hypreal* **where**
[*transfer-unfold*, *code del*]: *hlog a x* = *starfun2 log a x*

lemma *powhr*: (*star-n X*) *powhr* (*star-n Y*) = *star-n* (%*n*. (*X n*) *powr* (*Y n*))
⟨*proof*⟩

lemma *powhr-one-eq-one* [*simp*]: !!*a*. *1 powhr a* = *1*
⟨*proof*⟩

lemma *powhr-mult*:
!!*a x y*. [| *0* < *x*; *0* < *y* |] ==> (*x* ∗ *y*) *powhr a* = (*x powhr a*) ∗ (*y powhr a*)
⟨*proof*⟩

lemma *powhr-gt-zero* [*simp*]: !!*a x*. *0* < *x powhr a*
⟨*proof*⟩

lemma *powhr-not-zero* [*simp*]: *x powhr a* ≠ *0*
⟨*proof*⟩

lemma *powhr-divide*:
!!*a x y*. [| *0* < *x*; *0* < *y* |] ==> (*x* / *y*) *powhr a* = (*x powhr a*)/(*y powhr a*)
⟨*proof*⟩

lemma *powhr-add*: !!*a b x*. *x powhr* (*a* + *b*) = (*x powhr a*) ∗ (*x powhr b*)

⟨*proof*⟩

lemma *powhr-powhr*: $!!a\ b\ x.\ (x\ powhr\ a)\ powhr\ b = x\ powhr\ (a * b)$
⟨*proof*⟩

lemma *powhr-powhr-swap*: $!!a\ b\ x.\ (x\ powhr\ a)\ powhr\ b = (x\ powhr\ b)\ powhr\ a$
⟨*proof*⟩

lemma *powhr-minus*: $!!a\ x.\ x\ powhr\ (-a) = inverse\ (x\ powhr\ a)$
⟨*proof*⟩

lemma *powhr-minus-divide*: $x\ powhr\ (-a) = 1/(x\ powhr\ a)$
⟨*proof*⟩

lemma *powhr-less-mono*: $!!a\ b\ x.\ [|\ a < b;\ 1 < x\ |] ==> x\ powhr\ a < x\ powhr\ b$
⟨*proof*⟩

lemma *powhr-less-cancel*: $!!a\ b\ x.\ [|\ x\ powhr\ a < x\ powhr\ b;\ 1 < x\ |] ==> a < b$
⟨*proof*⟩

lemma *powhr-less-cancel-iff* [*simp*]:
$1 < x ==> (x\ powhr\ a < x\ powhr\ b) = (a < b)$
⟨*proof*⟩

lemma *powhr-le-cancel-iff* [*simp*]:
$1 < x ==> (x\ powhr\ a \le x\ powhr\ b) = (a \le b)$
⟨*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*:
$!!a\ x.\ [|\ 0\ <\ a;\ a \neq 1\ |]\ ==>\ hlog\ a\ x\ =\ (\ *f*\ ln)\ x\ /\ (\ *f*\ ln)\ a$
$\langle proof \rangle$

lemma *hlog-eq-div-starfun-ln-mult-hlog*:
$!!a\ b\ x.\ [|\ 0\ <\ a;\ a \neq 1;\ 0\ <\ b;\ b \neq 1;\ 0\ <\ x\ |]$
$==>\ hlog\ a\ x\ =\ ((\ *f*\ ln)\ b/(\ *f*ln)\ a)\ *\ hlog\ b\ x$
$\langle proof \rangle$

lemma *powhr-as-starfun*: $!!a\ x.\ x\ powhr\ a\ =\ (\ *f*\ exp)\ (a\ *\ (\ *f*\ ln)\ x)$
$\langle proof \rangle$

lemma *HInfinite-powhr*:
$[|\ x\ :\ HInfinite;\ 0\ <\ x;\ a\ :\ HFinite\ -\ Infinitesimal;$
$0\ <\ a\ |]\ ==>\ x\ powhr\ a\ :\ HInfinite$
$\langle proof \rangle$

lemma *hlog-hrabs-HInfinite-Infinitesimal*:
$[|\ x\ :\ HFinite\ -\ Infinitesimal;\ a\ :\ HInfinite;\ 0\ <\ a\ |]$
$==>\ hlog\ a\ (abs\ x)\ :\ Infinitesimal$
$\langle proof \rangle$

lemma *hlog-HInfinite-as-starfun*:
$[|\ a\ :\ HInfinite;\ 0\ <\ a\ |]\ ==>\ hlog\ a\ x\ =\ (\ *f*\ ln)\ x\ /\ (\ *f*\ ln)\ a$
$\langle proof \rangle$

lemma *hlog-one* [*simp*]: $!!a.\ hlog\ a\ 1\ =\ 0$
$\langle proof \rangle$

lemma *hlog-eq-one* [*simp*]: $!!a.\ [|\ 0\ <\ a;\ a \neq 1\ |]\ ==>\ hlog\ a\ a\ =\ 1$
$\langle proof \rangle$

lemma *hlog-inverse*:
$[|\ 0\ <\ a;\ a \neq 1;\ 0\ <\ x\ |]\ ==>\ hlog\ a\ (inverse\ x)\ =\ -\ hlog\ a\ x$
$\langle proof \rangle$

lemma *hlog-divide*:
$[|\ 0\ <\ a;\ a \neq 1;\ 0\ <\ x;\ 0\ <\ y|]\ ==>\ hlog\ a\ (x/y)\ =\ hlog\ a\ x\ -\ hlog\ a\ y$
$\langle proof \rangle$

lemma *hlog-less-cancel-iff* [*simp*]:
$!!a\ x\ y.\ [|\ 1\ <\ a;\ 0\ <\ x;\ 0\ <\ y\ |]\ ==>\ (hlog\ a\ x\ <\ hlog\ a\ y)\ =\ (x\ <\ y)$
$\langle proof \rangle$

lemma *hlog-le-cancel-iff* [*simp*]:
$[|\ 1\ <\ a;\ 0\ <\ x;\ 0\ <\ y\ |]\ ==>\ (hlog\ a\ x\ \leq\ hlog\ a\ y)\ =\ (x\ \leq\ y)$
$\langle proof \rangle$

end

theory *Hyperreal*
imports *Ln Deriv Taylor Integration HLog*
begin

end

theory *Hypercomplex*
imports *CLim Hyperreal*
begin

end