

Isabelle/HOL-NSA — Non-Standard Analysis

January 18, 2026

Contents

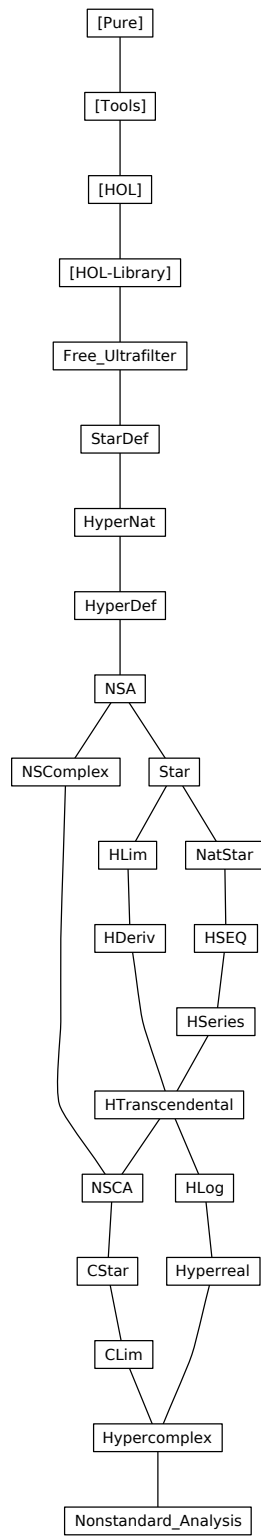
1	Filters and Ultrafilters	3
1.1	Definitions and basic properties	3
1.1.1	Ultrafilters	3
1.2	Maximal filter = Ultrafilter	3
1.3	Ultrafilter Theorem	4
1.3.1	Free Ultrafilters	5
2	Construction of Star Types Using Ultrafilters	6
2.1	A Free Ultrafilter over the Naturals	7
2.2	Definition of <i>star</i> type constructor	7
2.3	Transfer principle	8
2.4	Standard elements	10
2.5	Internal functions	10
2.6	Internal predicates	12
2.7	Internal sets	13
2.8	Syntactic classes	14
2.9	Ordering and lattice classes	18
2.10	Ordered group classes	20
2.11	Ring and field classes	21
2.12	Power	23
2.13	Number classes	24
2.14	Finite class	25
3	Hypersnatural numbers	25
3.1	Properties Transferred from Naturals	26
3.2	Properties of the set of embedded natural numbers	28
3.3	Infinite Hypersnatural Numbers – <i>HNatInfinite</i>	29
3.3.1	Closure Rules	30
3.4	Existence of an infinite hypersnatural number	30
3.4.1	Alternative characterization of the set of infinite hypersnaturals	31

3.4.2	Alternative Characterization of <i>HNatInfinite</i> using Free Ultrafilter	32
3.5	Embedding of the Hypernaturals into other types	32
4	Construction of Hyperreals Using Ultrafilters	33
4.1	Real vector class instances	34
4.2	Injection from <i>hypreal</i>	35
4.3	Properties of <i>starrel</i>	36
4.4	<i>hypreal-of-real</i> : the Injection from <i>real</i> to <i>hypreal</i>	36
4.5	Properties of <i>star-n</i>	36
4.6	Existence of Infinite Hyperreal Number	37
4.7	Embedding the Naturals into the Hyperreals	38
4.8	Exponentials on the Hyperreals	39
4.9	Powers with Hypernatural Exponents	39
5	Infinite Numbers, Infinitesimals, Infinitely Close Relation	42
5.1	Nonstandard Extension of the Norm Function	42
5.2	Closure Laws for the Standard Reals	45
5.3	Set of Finite Elements is a Subring of the Extended Reals	45
5.4	Set of Infinitesimals is a Subring of the Hyperreals	47
5.5	The Infinitely Close Relation	53
5.6	Zero is the Only Infinitesimal that is also a Real	58
6	Standard Part Theorem	60
6.1	Uniqueness: Two Infinitely Close Reals are Equal	60
6.2	Existence of Unique Real Infinitely Close	61
6.2.1	Lifting of the Ub and Lub Properties	61
6.3	Finite, Infinite and Infinitesimal	64
6.4	Theorems about Monads	67
6.5	Proof that $x \approx y$ implies $ x \approx y $	67
6.6	More <i>HFinite</i> and <i>Infinitesimal</i> Theorems	69
6.7	Theorems about Standard Part	70
6.8	Alternative Definitions using Free Ultrafilter	72
6.8.1	<i>HFinite</i>	72
6.8.2	<i>HInfinite</i>	73
6.8.3	<i>Infinitesimal</i>	74
6.9	Proof that ω is an infinite number	75
7	Nonstandard Complex Numbers	77
7.0.1	Real and Imaginary parts	78
7.0.2	Imaginary unit	78
7.0.3	Complex conjugate	78
7.0.4	Argand	78
7.0.5	Injection from hyperreals	78

7.0.6	$e^{\wedge}(x + iy)$	78
7.1	Properties of Nonstandard Real and Imaginary Parts	79
7.2	Addition for Nonstandard Complex Numbers	80
7.3	More Minus Laws	80
7.4	More Multiplication Laws	80
7.5	Subtraction and Division	81
7.6	Embedding Properties for <i>hcomplex-of-hypreal</i> Map	81
7.7	<i>HComplex</i> theorems	81
7.8	Modulus (Absolute Value) of Nonstandard Complex Number	82
7.9	Conjugation	83
7.10	More Theorems about the Function <i>hcmo</i> d	84
7.11	Exponentiation	84
7.12	The Function <i>hsgn</i>	85
7.12.1	<i>harg</i>	86
7.13	Polar Form for Nonstandard Complex Numbers	86
7.14	<i>hcomplex-of-complex</i> : the Injection from type <i>complex</i> to to <i>hcomplex</i>	89
7.15	Numerals and Arithmetic	89
8	Star-Transforms in Non-Standard Analysis	90
8.1	Preamble - Pulling \exists over \forall	90
8.2	Properties of the Star-transform Applied to Sets of Reals	90
8.3	Theorems about nonstandard extensions of functions	91
9	Star-transforms for the Hypernaturals	96
9.1	Nonstandard Extensions of Functions	97
9.2	Nonstandard Characterization of Induction	99
10	Sequences and Convergence (Nonstandard)	100
10.1	Limits of Sequences	101
10.1.1	Equivalence of <i>LIMSEQ</i> and <i>NSLIMSEQ</i>	103
10.1.2	Derived theorems about <i>NSLIMSEQ</i>	104
10.2	Convergence	105
10.3	Bounded Monotonic Sequences	105
10.3.1	Upper Bounds and Lubs of Bounded Sequences	107
10.3.2	A Bounded and Monotonic Sequence Converges	107
10.4	Cauchy Sequences	107
10.4.1	Equivalence Between NS and Standard	108
10.4.2	Cauchy Sequences are Bounded	109
10.4.3	Cauchy Sequences are Convergent	109
10.5	Power Sequences	110

11 Finite Summation and Infinite Series for Hyperreals	111
11.1 Nonstandard Sums	112
11.2 Infinite sums: Standard and NS theorems	113
12 Limits and Continuity (Nonstandard)	114
12.1 Limits of Functions	115
12.1.1 Equivalence of <i>filterlim</i> and <i>NSLIM</i>	117
12.2 Continuity	118
12.3 Uniform Continuity	119
13 Differentiation (Nonstandard)	121
13.1 Derivatives	121
13.2 Lemmas	124
13.2.1 Equivalence of NS and Standard definitions	126
13.2.2 Differentiability predicate	127
13.3 (NS) Increment	127
14 Nonstandard Extensions of Transcendental Functions	128
14.1 Nonstandard Extension of Square Root Function	129
14.2 Proving $\sin^*(1/n) \times 1/(1/n) \approx 1$ for $n = \infty$	137
15 Non-Standard Complex Analysis	140
15.1 Closure Laws for SComplex, the Standard Complex Numbers	140
15.2 The Finite Elements form a Subring	141
15.3 The Complex Infinitesimals form a Subring	141
15.4 The “Infinitely Close” Relation	142
15.5 Zero is the Only Infinitesimal Complex Number	142
15.6 Properties of <i>hRe</i> , <i>hIm</i> and <i>HComplex</i>	143
15.7 Theorems About Monads	145
15.8 Theorems About Standard Part	145
16 Star-transforms in NSA, Extending Sets of Complex Numbers and Complex Functions	148
16.1 Properties of the *-Transform Applied to Sets of Reals	148
16.2 Theorems about Nonstandard Extensions of Functions	148
16.3 Internal Functions - Some Redundancy With <i>*f*</i> Now	148
17 Limits, Continuity and Differentiation for Complex Functions	148
17.1 Limit of Complex to Complex Function	149
17.2 Continuity	150
17.3 Functions from Complex to Reals	150
17.4 Differentiation of Natural Number Powers	151
17.5 Derivative of Reciprocals (Function <i>inverse</i>)	151
17.6 Derivative of Quotient	151

17.7 Caratheodory Formulation of Derivative at a Point: Standard Proof	152
18 Logarithms: Non-Standard Version	152



1 Filters and Ultrafilters

```
theory Free-Ultrafilter
  imports HOL-Library.Infinite-Set
begin
```

1.1 Definitions and basic properties

1.1.1 Ultrafilters

```
locale ultrafilter =
  fixes F :: 'a filter
  assumes proper: F ≠ bot
  assumes ultra: eventually P F ∨ eventually (λx. ¬ P x) F
begin

lemma eventually-imp-frequently: frequently P F ⟹ eventually P F
  using ultra[of P] by (simp add: frequently-def)

lemma frequently-eq-eventually: frequently P F = eventually P F
  using eventually-imp-frequently eventually-frequently[OF proper] ..

lemma eventually-disj-iff: eventually (λx. P x ∨ Q x) F ⟷ eventually P F ∨
  eventually Q F
  unfolding frequently-eq-eventually[symmetric] frequently-disj-iff ..

lemma eventually-all-iff: eventually (λx. ∀ y. P x y) F = (∀ Y. eventually (λx. P
  x (Y x)) F)
  using frequently-all[of P F] by (simp add: frequently-eq-eventually)

lemma eventually-imp-iff: eventually (λx. P x ⟶ Q x) F ⟷ (eventually P F
  ⟶ eventually Q F)
  using frequently-imp-iff[of P Q F] by (simp add: frequently-eq-eventually)

lemma eventually-iff-iff: eventually (λx. P x ⟷ Q x) F ⟷ (eventually P F
  ⟷ eventually Q F)
  unfolding iff-conv-conj-imp eventually-conj-iff eventually-imp-iff by simp

lemma eventually-not-iff: eventually (λx. ¬ P x) F ⟷ ¬ eventually P F
  unfolding not-eventually frequently-eq-eventually ..

end
```

1.2 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 $F = G$

Lemma 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-filter*: $\text{frequently } P \ F \implies \inf F \ (\text{principal } \{x. P \ x\}) \neq \text{bot}$
by (*simp add: trivial-limit-def eventually-inf-principal not-eventually*)

lemma *max-filter-ultrafilter*:

assumes $F \neq \text{bot}$

assumes *max*: $\bigwedge G. G \neq \text{bot} \implies G \leq F \implies F = G$

shows *ultrafilter* F

proof

show *eventually* $P \ F \vee (\forall_F x \text{ in } F. \neg P \ x)$ **for** P

proof (*rule disjCI*)

assume $\neg (\forall_F x \text{ in } F. \neg P \ x)$

then have $\inf F \ (\text{principal } \{x. P \ x\}) \neq \text{bot}$

by (*simp add: not-eventually extend-filter*)

then have $F: F = \inf F \ (\text{principal } \{x. P \ x\})$

by (*rule max*) *simp*

show *eventually* $P \ F$

by (*subst F*) (*simp add: eventually-inf-principal*)

qed

qed fact

lemma *le-filter-frequently*: $F \leq G \longleftrightarrow (\forall P. \text{frequently } P \ F \longrightarrow \text{frequently } P \ G)$

unfolding *frequently-def le-filter-def*

apply *auto*

apply (*erule-tac x= $\lambda x. \neg P \ x$ in allE*)

apply *auto*

done

lemma (*in ultrafilter*) *max-filter*:

assumes $G: G \neq \text{bot}$

and *sub*: $G \leq F$

shows $F = G$

proof (*rule antisym*)

show $F \leq G$

using *sub*

by (*auto simp: le-filter-frequently[of F] frequently-eq-eventually le-filter-def[of G]*)

intro!: eventually-frequently G proper)

qed fact

1.3 Ultrafilter Theorem

lemma *ex-max-ultrafilter*:

fixes $F :: 'a \text{ filter}$

assumes $F: F \neq \text{bot}$

shows $\exists U \leq F. \text{ultrafilter } U$

proof –

let $?X = \{G. G \neq \text{bot} \wedge G \leq F\}$

let $?R = \{(b, a). a \neq \text{bot} \wedge a \leq b \wedge b \leq F\}$


```

have bot-notin-R:  $c \in \text{Chains } ?R \implies \text{bot} \notin c$  for  $c$ 
  by (auto simp: Chains-def)

have [simp]:  $\text{Field } ?R = ?X$ 
  by (auto simp: Field-def bot-unique)

have  $\exists m \in \text{Field } ?R. \forall a \in \text{Field } ?R. (m, a) \in ?R \longrightarrow a = m$  (is  $\exists m \in ?A. ?B m$ )
proof (rule Zorns-po-lemma)
  show Partial-order  $?R$ 
    by (auto simp: partial-order-on-def preorder-on-def
      antisym-def refl-on-def trans-def Field-def bot-unique)
  show  $\exists u \in \text{Field } ?R. \forall a \in C. (a, u) \in ?R$  if  $C: C \in \text{Chains } ?R$  for  $C$ 
  proof (simp, intro exI conjI ballI)
    have  $\text{Inf } C \neq \text{bot} \implies \text{Inf } C \leq F$  if  $C \neq \{\}$ 
    proof -
      from  $C$  that have  $\text{Inf } C = \text{bot} \iff (\exists x \in C. x = \text{bot})$ 
      unfolding trivial-limit-def by (intro eventually-Inf-base) (auto simp:
Chains-def)
      with  $C$  show  $\text{Inf } C \neq \text{bot}$ 
      by (simp add: bot-notin-R)
      from that obtain  $x$  where  $x \in C$  by auto
      with  $C$  show  $\text{Inf } C \leq F$ 
      by (auto intro!: Inf-lower2[of  $x$ ] simp: Chains-def)
    qed
  qed
  then have [simp]:  $\inf F (\text{Inf } C) = (\text{if } C = \{\} \text{ then } F \text{ else } \text{Inf } C)$ 
    using  $C$  by (auto simp add: inf-absorb2)
  from  $C$  show  $\inf F (\text{Inf } C) \neq \text{bot}$ 
    by (simp add: F Inf-C)
  from  $C$  show  $\inf F (\text{Inf } C) \leq F$ 
    by (simp add: Chains-def Inf-C F)
  with  $C$  show  $\inf F (\text{Inf } C) \leq x \wedge x \leq F$  if  $x \in C$  for  $x$ 
    using that by (auto intro: Inf-lower simp: Chains-def)
  qed
qed
then obtain  $U$  where  $U: U \in ?A \wedge ?B U$  ..
show ?thesis
proof
  from  $U$  show  $U \leq F \wedge \text{ultrafilter } U$ 
    by (auto intro!: max-filter-ultrafilter)
  qed
qed

```

1.3.1 Free Ultrafilters

There exists a free ultrafilter on any infinite set.

```

locale freeultrafilter = ultrafilter +
  assumes infinite:  $\text{eventually } P F \implies \text{infinite } \{x. P x\}$ 
begin

```

```

lemma finite: finite  $\{x. P\ x\} \implies \neg \text{eventually } P\ F$ 
  by (erule contrapos-pn) (erule infinite)

lemma finite': finite  $\{x. \neg P\ x\} \implies \text{eventually } P\ F$ 
  by (drule finite) (simp add: not-eventually frequently-eq-eventually)

lemma le-cofinite:  $F \leq \text{cofinite}$ 
  by (intro filter-leI)
  (auto simp add: eventually-cofinite not-eventually frequently-eq-eventually dest!:
finite)

lemma singleton:  $\neg \text{eventually } (\lambda x. x = a)\ F$ 
  by (rule finite) simp

lemma singleton':  $\neg \text{eventually } ((=)\ a)\ F$ 
  by (rule finite) simp

lemma ultrafilter: ultrafilter  $F \ ..$ 

end

lemma freeultrafilter-Ex:
  assumes [simp]: infinite ( $UNIV :: 'a\ set$ )
  shows  $\exists U :: 'a\ filter. \text{freeultrafilter } U$ 
proof –
  from ex-max-ultrafilter[of cofinite :: 'a filter]
  obtain  $U :: 'a\ filter$  where  $U \leq \text{cofinite ultrafilter } U$ 
  by auto
  interpret ultrafilter  $U$  by fact
  have freeultrafilter  $U$ 
  proof
    fix  $P$ 
    assume eventually  $P\ U$ 
    with proper have frequently  $P\ U$ 
    by (rule eventually-frequently)
    then have frequently  $P\ \text{cofinite}$ 
    using  $\langle U \leq \text{cofinite} \rangle$  by (simp add: le-filter-frequently)
    then show infinite  $\{x. P\ x\}$ 
    by (simp add: frequently-cofinite)
  qed
  then show ?thesis ..
qed

end

```

2 Construction of Star Types Using Ultrafilters

theory *StarDef*

```

imports Free-Ultrafilter
begin

```

2.1 A Free Ultrafilter over the Naturals

```

definition FreeUltrafilterNat :: nat filter (⟨U⟩)
  where U = (SOME U. freeultrafilter U)

```

```

lemma freeultrafilter-FreeUltrafilterNat: freeultrafilter U
  unfolding FreeUltrafilterNat-def
  by (simp add: freeultrafilter-Ex someI-ex)

```

```

interpretation FreeUltrafilterNat: freeultrafilter U
  by (rule freeultrafilter-FreeUltrafilterNat)

```

2.2 Definition of *star* type constructor

```

definition starrel :: ((nat ⇒ 'a) × (nat ⇒ 'a)) set
  where starrel = {(X, Y). eventually (λn. X n = Y n) U}

```

```

definition star = (UNIV :: (nat ⇒ 'a) set) // starrel

```

```

typedef 'a star = star :: (nat ⇒ 'a) set set
  by (auto simp: star-def intro: quotientI)

```

```

definition star-n :: (nat ⇒ 'a) ⇒ 'a star
  where star-n X = Abs-star (starrel “{X}”)

```

```

theorem star-cases [case-names star-n, cases type: star]:
  obtains X where x = star-n X
  by (cases x) (auto simp: star-n-def star-def elim: quotientE)

```

```

lemma all-star-eq: (∀ x. P x) ⟷ (∀ X. P (star-n X))
  by (metis star-cases)

```

```

lemma ex-star-eq: (∃ x. P x) ⟷ (∃ X. P (star-n X))
  by (metis star-cases)

```

Proving that *starrel* is an equivalence relation.

```

lemma starrel-iff [iff]: (X, Y) ∈ starrel ⟷ eventually (λn. X n = Y n) U
  by (simp add: starrel-def)

```

```

lemma equiv-starrel: equiv UNIV starrel
proof (rule equivI)
  show starrel ⊆ UNIV × UNIV by simp
  show refl starrel by (simp add: refl-on-def)
  show sym starrel by (simp add: sym-def eq-commute)
  show trans starrel by (intro transI) (auto elim: eventually-elim2)
qed

```

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

lemma *starrel-in-star*: *starrel*“{*x*} ∈ *star*
by (*simp add: star-def quotientI*)

lemma *star-n-eq-iff*: *star-n X = star-n Y* \longleftrightarrow *eventually* ($\lambda n. X\ n = Y\ n$) \mathcal{U}
by (*simp add: star-n-def Abs-star-inject starrel-in-star equiv-starrel-iff*)

2.3 Transfer principle

This introduction rule starts each transfer proof.

lemma *transfer-start*: *P* \equiv *eventually* ($\lambda n. Q$) $\mathcal{U} \implies \text{Trueprop } P \equiv \text{Trueprop } Q$
by (*simp add: FreeUltrafilterNat.proper*)

Standard principles that play a central role in the transfer tactic.

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

lemma *Ifun-congruent2*: *congruent2 starrel starrel* ($\lambda F\ X. \text{starrel} \text{“}\{\lambda n. F\ n\ (X\ n)\}\text{”}$)
by (*auto simp add: congruent2-def equiv-starrel-iff elim!: eventually-rev-mp*)

lemma *Ifun-star-n*: *star-n F* \star *star-n X* = *star-n* ($\lambda n. F\ n\ (X\ n)$)
by (*simp add: Ifun-def star-n-def Abs-star-inverse starrel-in-star UN-equiv-class2 [OF equiv-starrel equiv-starrel Ifun-congruent2]*)

lemma *transfer-Ifun*: *f* \equiv *star-n F* $\implies x \equiv$ *star-n X* $\implies f \star x \equiv$ *star-n* ($\lambda n. F\ n\ (X\ n)$)
by (*simp only: Ifun-star-n*)

definition *star-of* :: *'a* \Rightarrow *'a star*
where *star-of* *x* \equiv *star-n* ($\lambda n. x$)

Initialize transfer tactic.

ML-file $\langle \text{transfer-principle.ML} \rangle$

method-setup *transfer* =
 $\langle \text{Attrib.thms} \rangle \rangle (fn\ ths \Rightarrow fn\ ctxt \Rightarrow \text{SIMPLE-METHOD}' (\text{Transfer-Principle.transfer-tac } ctxt\ ths))$
transfer principle

Transfer introduction rules.

lemma *transfer-ex* [*transfer-intro*]:
($\bigwedge X. p\ (\text{star-n } X) \equiv \text{eventually } (\lambda n. P\ n\ (X\ n))\ \mathcal{U}$) \implies
 $\exists x :: 'a\ star. p\ x \equiv \text{eventually } (\lambda n. \exists x. P\ n\ x)\ \mathcal{U}$

by (*simp only: ex-star-eq eventually-ex*)

lemma *transfer-all* [*transfer-intro*]:

$(\bigwedge X. p \text{ (star-} n \text{ } X) \equiv \text{eventually } (\lambda n. P \ n \ (X \ n)) \ \mathcal{U}) \implies$
 $\forall x::'a \text{ star. } p \ x \equiv \text{eventually } (\lambda n. \forall x. P \ n \ x) \ \mathcal{U}$

by (*simp only: all-star-eq FreeUltrafilterNat.eventually-all-iff*)

lemma *transfer-not* [*transfer-intro*]: $p \equiv \text{eventually } P \ \mathcal{U} \implies \neg p \equiv \text{eventually } (\lambda n. \neg P \ n) \ \mathcal{U}$

by (*simp only: FreeUltrafilterNat.eventually-not-iff*)

lemma *transfer-conj* [*transfer-intro*]:

$p \equiv \text{eventually } P \ \mathcal{U} \implies q \equiv \text{eventually } Q \ \mathcal{U} \implies p \wedge q \equiv \text{eventually } (\lambda n. P \ n \wedge Q \ n) \ \mathcal{U}$

by (*simp only: eventually-conj-iff*)

lemma *transfer-disj* [*transfer-intro*]:

$p \equiv \text{eventually } P \ \mathcal{U} \implies q \equiv \text{eventually } Q \ \mathcal{U} \implies p \vee q \equiv \text{eventually } (\lambda n. P \ n \vee Q \ n) \ \mathcal{U}$

by (*simp only: FreeUltrafilterNat.eventually-disj-iff*)

lemma *transfer-imp* [*transfer-intro*]:

$p \equiv \text{eventually } P \ \mathcal{U} \implies q \equiv \text{eventually } Q \ \mathcal{U} \implies p \longrightarrow q \equiv \text{eventually } (\lambda n. P \ n \longrightarrow Q \ n) \ \mathcal{U}$

by (*simp only: FreeUltrafilterNat.eventually-imp-iff*)

lemma *transfer-iff* [*transfer-intro*]:

$p \equiv \text{eventually } P \ \mathcal{U} \implies q \equiv \text{eventually } Q \ \mathcal{U} \implies p = q \equiv \text{eventually } (\lambda n. P \ n = Q \ n) \ \mathcal{U}$

by (*simp only: FreeUltrafilterNat.eventually-iff-iff*)

lemma *transfer-if-bool* [*transfer-intro*]:

$p \equiv \text{eventually } P \ \mathcal{U} \implies x \equiv \text{eventually } X \ \mathcal{U} \implies y \equiv \text{eventually } Y \ \mathcal{U} \implies$
 $(\text{if } p \text{ then } x \text{ else } y) \equiv \text{eventually } (\lambda n. \text{if } P \ n \text{ then } X \ n \text{ else } Y \ n) \ \mathcal{U}$

by (*simp only: if-bool-eq-conj transfer-conj transfer-imp transfer-not*)

lemma *transfer-eq* [*transfer-intro*]:

$x \equiv \text{star-} n \ X \implies y \equiv \text{star-} n \ Y \implies x = y \equiv \text{eventually } (\lambda n. X \ n = Y \ n) \ \mathcal{U}$

by (*simp only: star-n-eq-iff*)

lemma *transfer-if* [*transfer-intro*]:

$p \equiv \text{eventually } (\lambda n. P \ n) \ \mathcal{U} \implies x \equiv \text{star-} n \ X \implies y \equiv \text{star-} n \ Y \implies$
 $(\text{if } p \text{ then } x \text{ else } y) \equiv \text{star-} n \ (\lambda n. \text{if } P \ n \text{ then } X \ n \text{ else } Y \ n)$

by (*rule eq-reflection*) (*auto simp: star-n-eq-iff transfer-not elim!: eventually-mono*)

lemma *transfer-fun-eq* [*transfer-intro*]:

$(\bigwedge X. f \text{ (star-} n \text{ } X) = g \text{ (star-} n \text{ } X) \equiv \text{eventually } (\lambda n. F \ n \ (X \ n) = G \ n \ (X \ n)) \ \mathcal{U}) \implies$

$f = g \equiv \text{eventually } (\lambda n. F \ n = G \ n) \ \mathcal{U}$

by (*simp only: fun-eq-iff transfer-all*)

lemma *transfer-star-n* [*transfer-intro*]: $\text{star-n } X \equiv \text{star-n } (\lambda n. X \ n)$
by (*rule reflexive*)

lemma *transfer-bool* [*transfer-intro*]: $p \equiv \text{eventually } (\lambda n. p) \mathcal{U}$
by (*simp add: FreeUltrafilterNat.proper*)

2.4 Standard elements

definition *Standard* :: 'a star set
where *Standard* = range *star-of*

Transfer tactic should remove occurrences of *star-of*.

setup <Transfer-Principle.add-const **const-name** <*star-of*>>

lemma *star-of-inject*: $\text{star-of } x = \text{star-of } y \longleftrightarrow x = y$
by *transfer (rule refl)*

lemma *Standard-star-of* [*simp*]: $\text{star-of } x \in \text{Standard}$
by (*simp add: Standard-def*)

2.5 Internal functions

Transfer tactic should remove occurrences of *Ifun*.

setup <Transfer-Principle.add-const **const-name** <*Ifun*>>

lemma *Ifun-star-of* [*simp*]: $\text{star-of } f \star \text{star-of } x = \text{star-of } (f \ x)$
by *transfer (rule refl)*

lemma *Standard-Ifun* [*simp*]: $f \in \text{Standard} \implies x \in \text{Standard} \implies f \star x \in \text{Standard}$
by (*auto simp add: Standard-def*)

Nonstandard extensions of functions.

definition *starfun* :: ('a \Rightarrow 'b) \Rightarrow 'a star \Rightarrow 'b star
 ($\langle \langle \text{open-block notation} = \langle \text{prefix starfun} \rangle \star f \star - \rangle \rangle$ [80] 80)
where *starfun* $f \equiv \lambda x. \text{star-of } f \star x$

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

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

lemma *starfun-star-n*: $(\star f \star) (\text{star-n } X) = \text{star-n } (\lambda n. f (X \ n))$
by (*simp only: starfun-def star-of-def Ifun-star-n*)

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

by (*simp only: starfun2-def star-of-def Ifun-star-n*)

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

by *transfer (rule refl)*

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

by *transfer (rule refl)*

lemma *Standard-starfun* [*simp*]: $x \in \text{Standard} \implies \text{starfun } f x \in \text{Standard}$

by (*simp add: starfun-def*)

lemma *Standard-starfun2* [*simp*]: $x \in \text{Standard} \implies y \in \text{Standard} \implies \text{starfun2 } f x y \in \text{Standard}$

by (*simp add: starfun2-def*)

lemma *Standard-starfun-iff*:

assumes *inj*: $\bigwedge x y. f x = f y \implies x = y$

shows $\text{starfun } f x \in \text{Standard} \longleftrightarrow x \in \text{Standard}$

proof

assume $x \in \text{Standard}$

then show $\text{starfun } f x \in \text{Standard}$ **by** *simp*

next

from *inj* **have** *inj'*: $\bigwedge x y. \text{starfun } f x = \text{starfun } f y \implies x = y$

by *transfer*

assume $\text{starfun } f x \in \text{Standard}$

then obtain *b* **where** $b: \text{starfun } f x = \text{star-of } b$

unfolding *Standard-def* **..**

then have $\exists x. \text{starfun } f x = \text{star-of } b$ **..**

then have $\exists a. f a = b$ **by** *transfer*

then obtain *a* **where** $f a = b$ **..**

then have $\text{starfun } f (\text{star-of } a) = \text{star-of } b$ **by** *transfer*

with *b* **have** $\text{starfun } f x = \text{starfun } f (\text{star-of } a)$ **by** *simp*

then have $x = \text{star-of } a$ **by** (*rule inj'*)

then show $x \in \text{Standard}$ **by** (*simp add: Standard-def*)

qed

lemma *Standard-starfun2-iff*:

assumes *inj*: $\bigwedge a b a' b'. f a b = f a' b' \implies a = a' \wedge b = b'$

shows $\text{starfun2 } f x y \in \text{Standard} \longleftrightarrow x \in \text{Standard} \wedge y \in \text{Standard}$

proof

assume $x \in \text{Standard} \wedge y \in \text{Standard}$

then show $\text{starfun2 } f x y \in \text{Standard}$ **by** *simp*

next

have *inj'*: $\bigwedge x y z w. \text{starfun2 } f x y = \text{starfun2 } f z w \implies x = z \wedge y = w$

using *inj* **by** *transfer*

assume $\text{starfun2 } f x y \in \text{Standard}$

then obtain *c* **where** $c: \text{starfun2 } f x y = \text{star-of } c$

```

  unfolding Standard-def ..
  then have  $\exists x y. \text{starfun2 } f x y = \text{star-of } c$  by auto
  then have  $\exists a b. f a b = c$  by transfer
  then obtain  $a b$  where  $f a b = c$  by auto
  then have  $\text{starfun2 } f (\text{star-of } a) (\text{star-of } b) = \text{star-of } c$  by transfer
  with  $c$  have  $\text{starfun2 } f x y = \text{starfun2 } f (\text{star-of } a) (\text{star-of } b)$  by simp
  then have  $x = \text{star-of } a \wedge y = \text{star-of } b$  by (rule inj')
  then show  $x \in \text{Standard} \wedge y \in \text{Standard}$  by (simp add: Standard-def)
qed

```

2.6 Internal predicates

definition $\text{unstar} :: \text{bool} \Rightarrow \text{bool}$
 where $\text{unstar } b \longleftrightarrow b = \text{star-of } \text{True}$

lemma unstar-star-n : $\text{unstar } (\text{star-n } P) \longleftrightarrow \text{eventually } P \mathcal{U}$
 by (simp add: unstar-def star-of-def star-n-eq-iff)

lemma unstar-star-of [simp]: $\text{unstar } (\text{star-of } p) = p$
 by (simp add: unstar-def star-of-inject)

Transfer tactic should remove occurrences of *unstar*.

setup $\langle \text{Transfer-Principle.add-const } \textbf{const-name} \langle \text{unstar} \rangle \rangle$

lemma transfer-unstar [transfer-intro]: $p \equiv \text{star-n } P \implies \text{unstar } p \equiv \text{eventually } P \mathcal{U}$
 by (simp only: unstar-star-n)

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

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

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

lemma starP-star-n : $(*p* P) (\text{star-n } X) = \text{eventually } (\lambda n. P (X n)) \mathcal{U}$
 by (simp only: starP-def star-of-def Ifun-star-n unstar-star-n)

lemma starP2-star-n : $(*p2* P) (\text{star-n } X) (\text{star-n } Y) = (\text{eventually } (\lambda n. P (X n) (Y n))) \mathcal{U}$
 by (simp only: starP2-def star-of-def Ifun-star-n unstar-star-n)

lemma starP-star-of [simp]: $(*p* P) (\text{star-of } x) = P x$
 by transfer (rule refl)

lemma *starP2-star-of* [*simp*]: ($*p2* P$) (*star-of* x) = $*p* P x$
by *transfer* (*rule refl*)

2.7 Internal sets

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

lemma *Iset-star-n*: (*star-n* $X \in Iset (star-n A)$) = (*eventually* ($\lambda n. X n \in A n$) \mathcal{U})
by (*simp add: Iset-def starP2-star-n*)

Transfer tactic should remove occurrences of *Iset*.

setup $\langle Transfer-Principle.add-const \textbf{const-name} \langle Iset \rangle \rangle$

lemma *transfer-mem* [*transfer-intro*]:
 $x \equiv star-n X \Longrightarrow a \equiv Iset (star-n A) \Longrightarrow x \in a \equiv eventually (\lambda n. X n \in A n) \mathcal{U}$
by (*simp only: Iset-star-n*)

lemma *transfer-Collect* [*transfer-intro*]:
 $(\bigwedge X. p (star-n X) \equiv eventually (\lambda n. P n (X n)) \mathcal{U}) \Longrightarrow$
 $Collect p \equiv Iset (star-n (\lambda n. Collect (P n)))$
by (*simp add: atomize-eq set-eq-iff all-star-eq Iset-star-n*)

lemma *transfer-set-eq* [*transfer-intro*]:
 $a \equiv Iset (star-n A) \Longrightarrow b \equiv Iset (star-n B) \Longrightarrow a = b \equiv eventually (\lambda n. A n = B n) \mathcal{U}$
by (*simp only: set-eq-iff transfer-all transfer-iff transfer-mem*)

lemma *transfer-ball* [*transfer-intro*]:
 $a \equiv Iset (star-n A) \Longrightarrow (\bigwedge X. p (star-n X) \equiv eventually (\lambda n. P n (X n)) \mathcal{U}) \Longrightarrow$
 $\forall x \in a. p x \equiv eventually (\lambda n. \forall x \in A n. P n x) \mathcal{U}$
by (*simp only: Ball-def transfer-all transfer-imp transfer-mem*)

lemma *transfer-bex* [*transfer-intro*]:
 $a \equiv Iset (star-n A) \Longrightarrow (\bigwedge X. p (star-n X) \equiv eventually (\lambda n. P n (X n)) \mathcal{U}) \Longrightarrow$
 $\exists x \in a. p x \equiv eventually (\lambda n. \exists x \in A n. P n x) \mathcal{U}$
by (*simp only: Bex-def transfer-ex transfer-conj transfer-mem*)

lemma *transfer-Iset* [*transfer-intro*]: $a \equiv star-n A \Longrightarrow Iset a \equiv Iset (star-n (\lambda n. A n))$
by *simp*

Nonstandard extensions of sets.

definition *starset* :: 'a set \Rightarrow 'a star set
 $(\langle \langle open-block notation = \langle prefix starset \rangle *s* - \rangle [80] 80 \rangle$
where *starset* $A = Iset (star-of A)$

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

lemma *starset-mem*: $\text{star-of } x \in \text{**} A \longleftrightarrow x \in A$
by *transfer* (*rule refl*)

lemma *starset-UNIV*: $\text{**} (\text{UNIV}::'a \text{ set}) = (\text{UNIV}::'a \text{ star set})$
by (*transfer UNIV-def*) (*rule refl*)

lemma *starset-empty*: $\text{**} \{\} = \{\}$
by (*transfer empty-def*) (*rule refl*)

lemma *starset-insert*: $\text{**} (\text{insert } x A) = \text{insert } (\text{star-of } x) (\text{**} A)$
by (*transfer insert-def Un-def*) (*rule refl*)

lemma *starset-Un*: $\text{**} (A \cup B) = \text{**} A \cup \text{**} B$
by (*transfer Un-def*) (*rule refl*)

lemma *starset-Int*: $\text{**} (A \cap B) = \text{**} A \cap \text{**} B$
by (*transfer Int-def*) (*rule refl*)

lemma *starset-Compl*: $\text{**} \neg A = \neg (\text{**} A)$
by (*transfer Compl-eq*) (*rule refl*)

lemma *starset-diff*: $\text{**} (A - B) = \text{**} A - \text{**} B$
by (*transfer set-diff-eq*) (*rule refl*)

lemma *starset-image*: $\text{**} (f ` A) = (\text{*f*} f) ` (\text{**} A)$
by (*transfer image-def*) (*rule refl*)

lemma *starset-vimage*: $\text{**} (f - ` A) = (\text{*f*} f) - ` (\text{**} A)$
by (*transfer vimage-def*) (*rule refl*)

lemma *starset-subset*: $(\text{**} A \subseteq \text{**} B) \longleftrightarrow A \subseteq B$
by (*transfer subset-eq*) (*rule refl*)

lemma *starset-eq*: $(\text{**} A = \text{**} B) \longleftrightarrow A = B$
by *transfer* (*rule refl*)

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

2.8 Syntactic classes

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

```

begin
  definition star-zero-def:  $0 \equiv \text{star-of } 0$ 
  instance ..
end

instantiation star :: (one) one
begin
  definition star-one-def:  $1 \equiv \text{star-of } 1$ 
  instance ..
end

instantiation star :: (plus) plus
begin
  definition star-add-def:  $(+) \equiv *f2* (+)$ 
  instance ..
end

instantiation star :: (times) times
begin
  definition star-mult-def:  $((*)) \equiv *f2* ((*))$ 
  instance ..
end

instantiation star :: (uminus) uminus
begin
  definition star-minus-def:  $\text{uminus} \equiv *f* \text{uminus}$ 
  instance ..
end

instantiation star :: (minus) minus
begin
  definition star-diff-def:  $(-) \equiv *f2* (-)$ 
  instance ..
end

instantiation star :: (abs) abs
begin
  definition star-abs-def:  $\text{abs} \equiv *f* \text{abs}$ 
  instance ..
end

instantiation star :: (sgn) sgn
begin
  definition star-sgn-def:  $\text{sgn} \equiv *f* \text{sgn}$ 
  instance ..
end

instantiation star :: (divide) divide
begin

```

```

definition star-divide-def:  $divide \equiv *f2* divide$ 
instance ..
end

```

```

instantiation star :: (inverse) inverse
begin
  definition star-inverse-def:  $inverse \equiv *f* inverse$ 
  instance ..
end

```

```

instance star :: (Rings.dvd) Rings.dvd ..

```

```

instantiation star :: (modulo) modulo
begin
  definition star-mod-def:  $(mod) \equiv *f2* (mod)$ 
  instance ..
end

```

```

instantiation star :: (ord) ord
begin
  definition star-le-def:  $(\leq) \equiv *p2* (\leq)$ 
  definition star-less-def:  $(<) \equiv *p2* (<)$ 
  instance ..
end

```

```

lemmas star-class-defs [transfer-unfold] =
  star-zero-def    star-one-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-mod-def

```

Class operations preserve standard elements.

```

lemma Standard-zero:  $0 \in Standard$ 
  by (simp add: star-zero-def)

```

```

lemma Standard-one:  $1 \in Standard$ 
  by (simp add: star-one-def)

```

```

lemma Standard-add:  $x \in Standard \implies y \in Standard \implies x + y \in Standard$ 
  by (simp add: star-add-def)

```

```

lemma Standard-diff:  $x \in Standard \implies y \in Standard \implies x - y \in Standard$ 
  by (simp add: star-diff-def)

```

```

lemma Standard-minus:  $x \in Standard \implies -x \in Standard$ 
  by (simp add: star-minus-def)

```

```

lemma Standard-mult:  $x \in Standard \implies y \in Standard \implies x * y \in Standard$ 

```

by (*simp add: star-mult-def*)

lemma *Standard-divide*: $x \in \text{Standard} \implies y \in \text{Standard} \implies x / y \in \text{Standard}$
by (*simp add: star-divide-def*)

lemma *Standard-inverse*: $x \in \text{Standard} \implies \text{inverse } x \in \text{Standard}$
by (*simp add: star-inverse-def*)

lemma *Standard-abs*: $x \in \text{Standard} \implies |x| \in \text{Standard}$
by (*simp add: star-abs-def*)

lemma *Standard-mod*: $x \in \text{Standard} \implies y \in \text{Standard} \implies x \bmod y \in \text{Standard}$
by (*simp add: star-mod-def*)

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

star-of preserves class operations.

lemma *star-of-add*: $\text{star-of } (x + y) = \text{star-of } x + \text{star-of } y$
by *transfer (rule refl)*

lemma *star-of-diff*: $\text{star-of } (x - y) = \text{star-of } x - \text{star-of } y$
by *transfer (rule refl)*

lemma *star-of-minus*: $\text{star-of } (-x) = - \text{star-of } x$
by *transfer (rule refl)*

lemma *star-of-mult*: $\text{star-of } (x * y) = \text{star-of } x * \text{star-of } y$
by *transfer (rule refl)*

lemma *star-of-divide*: $\text{star-of } (x / y) = \text{star-of } x / \text{star-of } y$
by *transfer (rule refl)*

lemma *star-of-inverse*: $\text{star-of } (\text{inverse } x) = \text{inverse } (\text{star-of } x)$
by *transfer (rule refl)*

lemma *star-of-mod*: $\text{star-of } (x \bmod y) = \text{star-of } x \bmod \text{star-of } y$
by *transfer (rule refl)*

lemma *star-of-abs*: $\text{star-of } |x| = |\text{star-of } x|$
by *transfer (rule refl)*

star-of preserves numerals.

lemma *star-of-zero*: $\text{star-of } 0 = 0$
by *transfer (rule refl)*

lemma *star-of-one*: $\text{star-of } 1 = 1$
by *transfer* (rule *refl*)

star-of preserves orderings.

lemma *star-of-less*: $(\text{star-of } x < \text{star-of } y) = (x < y)$
by *transfer* (rule *refl*)

lemma *star-of-le*: $(\text{star-of } x \leq \text{star-of } y) = (x \leq y)$
by *transfer* (rule *refl*)

lemma *star-of-eq*: $(\text{star-of } x = \text{star-of } y) = (x = y)$
by *transfer* (rule *refl*)

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*]

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-mod *star-of-abs*
star-of-zero *star-of-one*
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*

2.9 Ordering and lattice classes

instance *star* :: (order) order

proof

show $\bigwedge x y :: 'a \text{ star. } (x < y) = (x \leq y \wedge \neg y \leq x)$
by *transfer* (rule *less-le-not-le*)

show $\bigwedge x :: 'a \text{ star. } x \leq x$

```

    by transfer (rule order-refl)
  show  $\bigwedge x y z :: 'a \text{ star. } \llbracket x \leq y; y \leq z \rrbracket \implies x \leq z$ 
    by transfer (rule order-trans)
  show  $\bigwedge x y :: 'a \text{ star. } \llbracket x \leq y; y \leq x \rrbracket \implies x = y$ 
    by transfer (rule order-antisym)
qed

```

```

instantiation star :: (semilattice-inf) semilattice-inf
begin
  definition star-inf-def [transfer-unfold]:  $\text{inf} \equiv *f2* \text{ inf}$ 
  instance by (standard; transfer) auto
end

```

```

instantiation star :: (semilattice-sup) semilattice-sup
begin
  definition star-sup-def [transfer-unfold]:  $\text{sup} \equiv *f2* \text{ sup}$ 
  instance by (standard; transfer) auto
end

```

```

instance star :: (lattice) lattice ..

```

```

instance star :: (distrib-lattice) distrib-lattice
  by (standard; transfer) (auto simp add: sup-inf-distrib1)

```

```

lemma Standard-inf [simp]:  $x \in \text{Standard} \implies y \in \text{Standard} \implies \text{inf } x y \in \text{Standard}$ 
  by (simp add: star-inf-def)

```

```

lemma Standard-sup [simp]:  $x \in \text{Standard} \implies y \in \text{Standard} \implies \text{sup } x y \in \text{Standard}$ 
  by (simp add: star-sup-def)

```

```

lemma star-of-inf [simp]:  $\text{star-of } (\text{inf } x y) = \text{inf } (\text{star-of } x) (\text{star-of } y)$ 
  by transfer (rule refl)

```

```

lemma star-of-sup [simp]:  $\text{star-of } (\text{sup } x y) = \text{sup } (\text{star-of } x) (\text{star-of } y)$ 
  by transfer (rule refl)

```

```

instance star :: (linorder) linorder
  by (intro-classes, transfer, rule linorder-linear)

```

```

lemma star-max-def [transfer-unfold]:  $\text{max} = *f2* \text{ max}$ 
  unfolding max-def
  by (intro ext, transfer, simp)

```

```

lemma star-min-def [transfer-unfold]:  $\text{min} = *f2* \text{ min}$ 
  unfolding min-def
  by (intro ext, transfer, simp)

```

```

lemma Standard-max [simp]:  $x \in \text{Standard} \implies y \in \text{Standard} \implies \text{max } x y \in$ 

```

Standard

by (*simp add: star-max-def*)

lemma *Standard-min* [*simp*]: $x \in \text{Standard} \implies y \in \text{Standard} \implies \min x y \in \text{Standard}$

by (*simp add: star-min-def*)

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

by *transfer (rule refl)*

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

by *transfer (rule refl)*

2.10 Ordered group classes

instance *star* :: (*semigroup-add*) *semigroup-add*

by (*intro-classes, transfer, rule add.assoc*)

instance *star* :: (*ab-semigroup-add*) *ab-semigroup-add*

by (*intro-classes, transfer, rule add.commute*)

instance *star* :: (*semigroup-mult*) *semigroup-mult*

by (*intro-classes, transfer, rule mult.assoc*)

instance *star* :: (*ab-semigroup-mult*) *ab-semigroup-mult*

by (*intro-classes, transfer, rule mult.commute*)

instance *star* :: (*comm-monoid-add*) *comm-monoid-add*

by (*intro-classes, transfer, rule comm-monoid-add-class.add-0*)

instance *star* :: (*monoid-mult*) *monoid-mult*

apply *intro-classes*

apply (*transfer, rule mult-1-left*)

apply (*transfer, rule mult-1-right*)

done

instance *star* :: (*power*) *power* ..

instance *star* :: (*comm-monoid-mult*) *comm-monoid-mult*

by (*intro-classes, transfer, rule mult-1*)

instance *star* :: (*cancel-semigroup-add*) *cancel-semigroup-add*

apply *intro-classes*

apply (*transfer, erule add-left-imp-eq*)

apply (*transfer, erule add-right-imp-eq*)

done

instance *star* :: (*cancel-ab-semigroup-add*) *cancel-ab-semigroup-add*

by *intro-classes (transfer, simp add: diff-diff-eq)+*


```

instance star :: (cancel-comm-monoid-add) cancel-comm-monoid-add ..

instance star :: (ab-group-add) ab-group-add
  apply intro-classes
  apply (transfer, rule left-minus)
  apply (transfer, rule diff-conv-add-uminus)
  done

instance star :: (ordered-ab-semigroup-add) ordered-ab-semigroup-add
  by (intro-classes, transfer, rule add-left-mono)

instance star :: (ordered-cancel-ab-semigroup-add) ordered-cancel-ab-semigroup-add
  ..

instance star :: (ordered-ab-semigroup-add-imp-le) ordered-ab-semigroup-add-imp-le
  by (intro-classes, transfer, rule add-le-imp-le-left)

instance star :: (ordered-comm-monoid-add) ordered-comm-monoid-add ..
instance star :: (ordered-ab-semigroup-monoid-add-imp-le) ordered-ab-semigroup-monoid-add-imp-le
  ..
instance star :: (ordered-cancel-comm-monoid-add) ordered-cancel-comm-monoid-add
  ..
instance star :: (ordered-ab-group-add) ordered-ab-group-add ..

instance star :: (ordered-ab-group-add-abs) ordered-ab-group-add-abs
  by intro-classes (transfer, simp add: abs-ge-self abs-leI abs-triangle-ineq)+

instance star :: (linordered-cancel-ab-semigroup-add) linordered-cancel-ab-semigroup-add
  ..

```

2.11 Ring and field classes

```

instance star :: (semiring) semiring
  by (intro-classes; transfer) (fact distrib-right distrib-left)+

instance star :: (semiring-0) semiring-0
  by (intro-classes; transfer) simp-all

instance star :: (semiring-0-cancel) semiring-0-cancel ..

instance star :: (comm-semiring) comm-semiring
  by (intro-classes; transfer) (fact distrib-right)

instance star :: (comm-semiring-0) comm-semiring-0 ..
instance star :: (comm-semiring-0-cancel) comm-semiring-0-cancel ..

instance star :: (zero-neq-one) zero-neq-one
  by (intro-classes; transfer) (fact zero-neq-one)

```

```

instance star :: (semiring-1) semiring-1 ..
instance star :: (comm-semiring-1) comm-semiring-1 ..

declare dvd-def [transfer-refold]

instance star :: (comm-semiring-1-cancel) comm-semiring-1-cancel
  by (intro-classes; transfer) (fact right-diff-distrib')

instance star :: (semiring-no-zero-divisors) semiring-no-zero-divisors
  by (intro-classes; transfer) (fact no-zero-divisors)

instance star :: (semiring-1-no-zero-divisors) semiring-1-no-zero-divisors ..

instance star :: (semiring-no-zero-divisors-cancel) semiring-no-zero-divisors-cancel
  by (intro-classes; transfer) simp-all

instance star :: (semiring-1-cancel) semiring-1-cancel ..
instance star :: (ring) ring ..
instance star :: (comm-ring) comm-ring ..
instance star :: (ring-1) ring-1 ..
instance star :: (comm-ring-1) comm-ring-1 ..
instance star :: (semidom) semidom ..

instance star :: (semidom-divide) semidom-divide
  by (intro-classes; transfer) simp-all

instance star :: (ring-no-zero-divisors) ring-no-zero-divisors ..
instance star :: (ring-1-no-zero-divisors) ring-1-no-zero-divisors ..
instance star :: (idom) idom ..
instance star :: (idom-divide) idom-divide ..

instance star :: (divide-trivial) divide-trivial
  by (intro-classes; transfer) simp-all

instance star :: (division-ring) division-ring
  by (intro-classes; transfer) (simp-all add: divide-inverse)

instance star :: (field) field
  by (intro-classes; transfer) (simp-all add: divide-inverse)

instance star :: (ordered-semiring) ordered-semiring
  by (intro-classes; transfer) (fact mult-left-mono mult-right-mono)+

instance star :: (ordered-cancel-semiring) ordered-cancel-semiring ..

instance star :: (linordered-semiring-strict) linordered-semiring-strict
  by (intro-classes; transfer) (fact mult-strict-left-mono mult-strict-right-mono)+

```

```

instance star :: (ordered-comm-semiring) ordered-comm-semiring
  by (intro-classes; transfer) (fact mult-left-mono)

instance star :: (ordered-cancel-comm-semiring) ordered-cancel-comm-semiring ..

instance star :: (linordered-comm-semiring-strict) linordered-comm-semiring-strict
  by (intro-classes; transfer) (fact mult-strict-left-mono)

instance star :: (ordered-ring) ordered-ring ..

instance star :: (ordered-ring-abs) ordered-ring-abs
  by (intro-classes; transfer) (fact abs-eq-mult)

instance star :: (abs-if) abs-if
  by (intro-classes; transfer) (fact abs-if)

instance star :: (linordered-ring-strict) linordered-ring-strict ..
instance star :: (ordered-comm-ring) ordered-comm-ring ..

instance star :: (linordered-semidom) linordered-semidom
  by (intro-classes; transfer) (fact zero-less-one le-add-diff-inverse2)+

instance star :: (linordered-idom) linordered-idom
  by (intro-classes; transfer) (fact sgn-if)

instance star :: (linordered-field) linordered-field ..

instance star :: (algebraic-semidom) algebraic-semidom ..

instantiation star :: (normalization-semidom) normalization-semidom
begin

definition unit-factor-star :: 'a star  $\Rightarrow$  'a star
  where [transfer-unfold]: unit-factor-star = *f* unit-factor

definition normalize-star :: 'a star  $\Rightarrow$  'a star
  where [transfer-unfold]: normalize-star = *f* normalize

instance
  by standard (transfer; simp add: is-unit-unit-factor unit-factor-mult)+

end

instance star :: (semidom-modulo) semidom-modulo
  by standard (transfer; simp)

```

2.12 Power

```

lemma star-power-def [transfer-unfold]: ( $\wedge$ )  $\equiv \lambda x n. ( *f* (\lambda x. x \wedge n) ) x$ 

```

```

proof (rule eq-reflection, rule ext, rule ext)
  show  $x \wedge n = (*f* (\lambda x. x \wedge n)) x$  for  $n :: \text{nat}$  and  $x :: 'a \text{ star}$ 
  proof (induct n arbitrary: x)
    case 0
    have  $\bigwedge x::'a \text{ star}. (*f* (\lambda x. 1)) x = 1$ 
    by transfer simp
    then show ?case by simp
  next
  case (Suc n)
  have  $\bigwedge x::'a \text{ star}. x * (*f* (\lambda x::'a. x \wedge n)) x = (*f* (\lambda x::'a. x * x \wedge n)) x$ 
  by transfer simp
  with Suc show ?case by simp
qed
qed

```

```

lemma Standard-power [simp]:  $x \in \text{Standard} \implies x \wedge n \in \text{Standard}$ 
by (simp add: star-power-def)

```

```

lemma star-of-power [simp]:  $\text{star-of } (x \wedge n) = \text{star-of } x \wedge n$ 
by transfer (rule refl)

```

2.13 Number classes

```

instance star :: (numeral) numeral ..

```

```

lemma star-numeral-def [transfer-unfold]:  $\text{numeral } k = \text{star-of } (\text{numeral } k)$ 
by (induct k) (simp-all only: numeral.simps star-of-one star-of-add)

```

```

lemma Standard-numeral [simp]:  $\text{numeral } k \in \text{Standard}$ 
by (simp add: star-numeral-def)

```

```

lemma star-of-numeral [simp]:  $\text{star-of } (\text{numeral } k) = \text{numeral } k$ 
by transfer (rule refl)

```

```

lemma star-of-nat-def [transfer-unfold]:  $\text{of-nat } n = \text{star-of } (\text{of-nat } n)$ 
by (induct n) simp-all

```

```

lemmas star-of-compare-numeral [simp] =
  star-of-less [of numeral k, simplified star-of-numeral]
  star-of-le [of numeral k, simplified star-of-numeral]
  star-of-eq [of numeral k, simplified star-of-numeral]
  star-of-less [of - numeral k, simplified star-of-numeral]
  star-of-le [of - numeral k, simplified star-of-numeral]
  star-of-eq [of - numeral k, simplified star-of-numeral]
  star-of-less [of - numeral k, simplified star-of-numeral]
  star-of-le [of - numeral k, simplified star-of-numeral]
  star-of-eq [of - numeral k, simplified star-of-numeral]
  star-of-less [of - - numeral k, simplified star-of-numeral]
  star-of-le [of - - numeral k, simplified star-of-numeral]

```

star-of-eq [of - — numeral k , simplified *star-of-numeral*] **for** k

lemma *Standard-of-nat* [*simp*]: *of-nat* $n \in \text{Standard}$
by (*simp add: star-of-nat-def*)

lemma *star-of-of-nat* [*simp*]: *star-of* (*of-nat* n) = *of-nat* n
by *transfer* (*rule refl*)

lemma *star-of-int-def* [*transfer-unfold*]: *of-int* $z = \text{star-of}$ (*of-int* z)
by (*rule int-diff-cases* [*of z*]) *simp*

lemma *Standard-of-int* [*simp*]: *of-int* $z \in \text{Standard}$
by (*simp add: star-of-int-def*)

lemma *star-of-of-int* [*simp*]: *star-of* (*of-int* z) = *of-int* z
by *transfer* (*rule refl*)

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

have *inj* (*star-of* :: ' $a \Rightarrow 'a$ *star*)
by (*rule injI*) *simp*
then have *inj* (*star-of* \circ *of-nat* :: $\text{nat} \Rightarrow 'a$ *star*)
using *inj-of-nat* **by** (*rule inj-compose*)
then show *inj* (*of-nat* :: $\text{nat} \Rightarrow 'a$ *star*)
by (*simp add: comp-def*)

qed

instance *star* :: (*ring-char-0*) *ring-char-0* ..

2.14 Finite class

lemma *starset-finite*: *finite* $A \implies *s*$ $A = \text{star-of}$ ' A
by (*erule finite-induct*) *simp-all*

instance *star* :: (*finite*) *finite*

proof *intro-classes*

show *finite* (*UNIV*::' a *star* *set*)
by (*metis starset-UNIV finite finite-imageI starset-finite*)

qed

end

3 Hypernatural numbers

theory *HyperNat*

imports *StarDef*

begin

type-synonym *hypnat* = *nat star*

abbreviation $\text{hypnat-of-nat} :: \text{nat} \Rightarrow \text{nat star}$

where $\text{hypnat-of-nat} \equiv \text{star-of}$

definition $\text{hSuc} :: \text{hypnat} \Rightarrow \text{hypnat}$

where $\text{hSuc-def} \text{ [transfer-unfold]: } \text{hSuc} = *f* \text{ Suc}$

3.1 Properties Transferred from Naturals

lemma $\text{hSuc-not-zero} \text{ [iff]: } \bigwedge m. \text{hSuc } m \neq 0$

by $\text{transfer (rule Suc-not-Zero)}$

lemma $\text{zero-not-hSuc} \text{ [iff]: } \bigwedge m. 0 \neq \text{hSuc } m$

by $\text{transfer (rule Zero-not-Suc)}$

lemma $\text{hSuc-hSuc-eq} \text{ [iff]: } \bigwedge m n. \text{hSuc } m = \text{hSuc } n \longleftrightarrow m = n$

by $\text{transfer (rule nat.inject)}$

lemma $\text{zero-less-hSuc} \text{ [iff]: } \bigwedge n. 0 < \text{hSuc } n$

by $\text{transfer (rule zero-less-Suc)}$

lemma $\text{hypnat-minus-zero} \text{ [simp]: } \bigwedge z::\text{hypnat}. z - z = 0$

by $\text{transfer (rule diff-self-eq-0)}$

lemma $\text{hypnat-diff-0-eq-0} \text{ [simp]: } \bigwedge n::\text{hypnat}. 0 - n = 0$

by $\text{transfer (rule diff-0-eq-0)}$

lemma $\text{hypnat-add-is-0} \text{ [iff]: } \bigwedge m n::\text{hypnat}. m + n = 0 \longleftrightarrow m = 0 \wedge n = 0$

by $\text{transfer (rule add-is-0)}$

lemma $\text{hypnat-diff-diff-left: } \bigwedge i j k::\text{hypnat}. i - j - k = i - (j + k)$

by $\text{transfer (rule diff-diff-left)}$

lemma $\text{hypnat-diff-commute: } \bigwedge i j k::\text{hypnat}. i - j - k = i - k - j$

by $\text{transfer (rule diff-commute)}$

lemma $\text{hypnat-diff-add-inverse} \text{ [simp]: } \bigwedge m n::\text{hypnat}. n + m - n = m$

by $\text{transfer (rule diff-add-inverse)}$

lemma $\text{hypnat-diff-add-inverse2} \text{ [simp]: } \bigwedge m n::\text{hypnat}. m + n - n = m$

by $\text{transfer (rule diff-add-inverse2)}$

lemma $\text{hypnat-diff-cancel} \text{ [simp]: } \bigwedge k m n::\text{hypnat}. (k + m) - (k + n) = m - n$

by $\text{transfer (rule diff-cancel)}$

lemma $\text{hypnat-diff-cancel2} \text{ [simp]: } \bigwedge k m n::\text{hypnat}. (m + k) - (n + k) = m - n$

by $\text{transfer (rule diff-cancel2)}$

lemma $\text{hypnat-diff-add-0} \text{ [simp]: } \bigwedge m n::\text{hypnat}. n - (n + m) = 0$

by *transfer* (*rule diff-add-0*)

lemma *hypnat-diff-mult-distrib*: $\bigwedge k m n :: \text{hypnat}. (m - n) * k = (m * k) - (n * k)$

by *transfer* (*rule diff-mult-distrib*)

lemma *hypnat-diff-mult-distrib2*: $\bigwedge k m n :: \text{hypnat}. k * (m - n) = (k * m) - (k * n)$

by *transfer* (*rule diff-mult-distrib2*)

lemma *hypnat-le-zero-cancel* [*iff*]: $\bigwedge n :: \text{hypnat}. n \leq 0 \longleftrightarrow n = 0$

by *transfer* (*rule le-0-eq*)

lemma *hypnat-mult-is-0* [*simp*]: $\bigwedge m n :: \text{hypnat}. m * n = 0 \longleftrightarrow m = 0 \vee n = 0$

by *transfer* (*rule mult-is-0*)

lemma *hypnat-diff-is-0-eq* [*simp*]: $\bigwedge m n :: \text{hypnat}. m - n = 0 \longleftrightarrow m \leq n$

by *transfer* (*rule diff-is-0-eq*)

lemma *hypnat-not-less0* [*iff*]: $\bigwedge n :: \text{hypnat}. \neg n < 0$

by *transfer* (*rule not-less0*)

lemma *hypnat-less-one* [*iff*]: $\bigwedge n :: \text{hypnat}. n < 1 \longleftrightarrow n = 0$

by *transfer* (*rule less-one*)

lemma *hypnat-add-diff-inverse*: $\bigwedge m n :: \text{hypnat}. \neg m < n \implies n + (m - n) = m$

by *transfer* (*rule add-diff-inverse*)

lemma *hypnat-le-add-diff-inverse* [*simp*]: $\bigwedge m n :: \text{hypnat}. n \leq m \implies n + (m - n) = m$

by *transfer* (*rule le-add-diff-inverse*)

lemma *hypnat-le-add-diff-inverse2* [*simp*]: $\bigwedge m n :: \text{hypnat}. n \leq m \implies (m - n) + n = m$

by *transfer* (*rule le-add-diff-inverse2*)

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

lemma *hypnat-le0* [*iff*]: $\bigwedge n :: \text{hypnat}. 0 \leq n$

by *transfer* (*rule le0*)

lemma *hypnat-le-add1* [*simp*]: $\bigwedge x n :: \text{hypnat}. x \leq x + n$

by *transfer* (*rule le-add1*)

lemma *hypnat-add-self-le* [*simp*]: $\bigwedge x n :: \text{hypnat}. x \leq n + x$

by *transfer* (*rule le-add2*)

lemma *hypnat-add-one-self-less* [*simp*]: $x < x + 1$ **for** $x :: \text{hypnat}$

by (*fact less-add-one*)

lemma *hypnat-neq0-conv* [iff]: $\bigwedge n :: \text{hypnat}. n \neq 0 \longleftrightarrow 0 < n$
 by *transfer* (rule *neq0-conv*)

lemma *hypnat-gt-zero-iff*: $0 < n \longleftrightarrow 1 \leq n$ **for** $n :: \text{hypnat}$
 by (*auto simp add: linorder-not-less [symmetric]*)

lemma *hypnat-gt-zero-iff2*: $0 < n \longleftrightarrow (\exists m. n = m + 1)$ **for** $n :: \text{hypnat}$
 by (*auto intro!: add-nonneg-pos exI[of - n - 1] simp: hypnat-gt-zero-iff*)

lemma *hypnat-add-self-not-less*: $\neg x + y < x$ **for** $x y :: \text{hypnat}$
 by (*simp add: linorder-not-le [symmetric] add.commute [of x]*)

lemma *hypnat-diff-split*: $P (a - b) \longleftrightarrow (a < b \longrightarrow P 0) \wedge (\forall d. a = b + d \longrightarrow P d)$
for $a b :: \text{hypnat}$
 — elimination of $-$ on *hypnat*
proof (*cases a < b rule: case-split*)
 case *True*
 then show ?thesis
 by (*auto simp add: hypnat-add-self-not-less order-less-imp-le hypnat-diff-is-0-eq [THEN iffD2]*)
 next
 case *False*
 then show ?thesis
 by (*auto simp add: linorder-not-less dest: order-le-less-trans*)
 qed

3.2 Properties of the set of embedded natural numbers

lemma *of-nat-eq-star-of* [simp]: $\text{of-nat} = \text{star-of}$
proof
 show $\text{of-nat } n = \text{star-of } n$ **for** n
 by *transfer simp*
 qed

lemma *Nats-eq-Standard*: $(\text{Nats} :: \text{nat star set}) = \text{Standard}$
 by (*auto simp: Nats-def Standard-def*)

lemma *hypnat-of-nat-mem-Nats* [simp]: $\text{hypnat-of-nat } n \in \text{Nats}$
 by (*simp add: Nats-eq-Standard*)

lemma *hypnat-of-nat-one* [simp]: $\text{hypnat-of-nat } (\text{Suc } 0) = 1$
 by *transfer simp*

lemma *hypnat-of-nat-Suc* [simp]: $\text{hypnat-of-nat } (\text{Suc } n) = \text{hypnat-of-nat } n + 1$
 by *transfer simp*

lemma *of-nat-eq-add*:


```

fixes d::hypnat
shows of-nat m = of-nat n + d  $\implies$  d  $\in$  range of-nat
proof (induct n arbitrary: d)
  case (Suc n)
  then show ?case
    by (metis Nats-def Nats-eq-Standard Standard-simps(4) hypnat-diff-add-inverse
of-nat-in-Nats)
qed auto

```

```

lemma Nats-diff [simp]: a  $\in$  Nats  $\implies$  b  $\in$  Nats  $\implies$  a - b  $\in$  Nats for a b ::
hypnat
by (simp add: Nats-eq-Standard)

```

3.3 Infinite Hypernatural Numbers – *HNatInfinite*

The set of infinite hypernatural numbers.

```

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

```

```

lemma Nats-not-HNatInfinite-iff: x  $\in$  Nats  $\longleftrightarrow$  x  $\notin$  HNatInfinite
by (simp add: HNatInfinite-def)

```

```

lemma HNatInfinite-not-Nats-iff: x  $\in$  HNatInfinite  $\longleftrightarrow$  x  $\notin$  Nats
by (simp add: HNatInfinite-def)

```

```

lemma star-of-neq-HNatInfinite: N  $\in$  HNatInfinite  $\implies$  star-of n  $\neq$  N
by (auto simp add: HNatInfinite-def Nats-eq-Standard)

```

```

lemma star-of-Suc-lessI:  $\bigwedge N. \text{star-of } n < N \implies \text{star-of } (Suc\ n) \neq N \implies \text{star-of}$ 
(Suc n) < N
by transfer (rule Suc-lessI)

```

```

lemma star-of-less-HNatInfinite:
assumes N: N  $\in$  HNatInfinite
shows star-of n < N
proof (induct n)
  case 0
  from N have star-of 0  $\neq$  N
  by (rule star-of-neq-HNatInfinite)
  then show ?case by simp
next
  case (Suc n)
  from N have star-of (Suc n)  $\neq$  N
  by (rule star-of-neq-HNatInfinite)
  with Suc show ?case
  by (rule star-of-Suc-lessI)
qed

```

```

lemma star-of-le-HNatInfinite: N  $\in$  HNatInfinite  $\implies$  star-of n  $\leq$  N

```

by (*rule star-of-less-HNatInfinite [THEN order-less-imp-le]*)

3.3.1 Closure Rules

lemma *Nats-less-HNatInfinite*: $x \in \text{Nats} \implies y \in \text{HNatInfinite} \implies x < y$
by (*auto simp add: Nats-def star-of-less-HNatInfinite*)

lemma *Nats-le-HNatInfinite*: $x \in \text{Nats} \implies y \in \text{HNatInfinite} \implies x \leq y$
by (*rule Nats-less-HNatInfinite [THEN order-less-imp-le]*)

lemma *zero-less-HNatInfinite*: $x \in \text{HNatInfinite} \implies 0 < x$
by (*simp add: Nats-less-HNatInfinite*)

lemma *one-less-HNatInfinite*: $x \in \text{HNatInfinite} \implies 1 < x$
by (*simp add: Nats-less-HNatInfinite*)

lemma *one-le-HNatInfinite*: $x \in \text{HNatInfinite} \implies 1 \leq x$
by (*simp add: Nats-le-HNatInfinite*)

lemma *zero-not-mem-HNatInfinite* [*simp*]: $0 \notin \text{HNatInfinite}$
by (*simp add: HNatInfinite-def*)

lemma *Nats-downward-closed*: $x \in \text{Nats} \implies y \leq x \implies y \in \text{Nats}$ **for** $x\ y :: \text{hypnat}$
using *HNatInfinite-not-Nats-iff Nats-le-HNatInfinite* **by** *fastforce*

lemma *HNatInfinite-upward-closed*: $x \in \text{HNatInfinite} \implies x \leq y \implies y \in \text{HNatInfinite}$
using *HNatInfinite-not-Nats-iff Nats-downward-closed* **by** *blast*

lemma *HNatInfinite-add*: $x \in \text{HNatInfinite} \implies x + y \in \text{HNatInfinite}$
using *HNatInfinite-upward-closed hypnat-le-add1* **by** *blast*

lemma *HNatInfinite-diff*: $\llbracket x \in \text{HNatInfinite}; y \in \text{Nats} \rrbracket \implies x - y \in \text{HNatInfinite}$
by (*metis HNatInfinite-not-Nats-iff Nats-add Nats-le-HNatInfinite le-add-diff-inverse*)

lemma *HNatInfinite-is-Suc*: $x \in \text{HNatInfinite} \implies \exists y. x = y + 1$ **for** $x :: \text{hypnat}$
using *hypnat-gt-zero-iff2 zero-less-HNatInfinite* **by** *blast*

3.4 Existence of an infinite hypernatural number

ω is in fact an infinite hypernatural number = $[<1, 2, 3, \dots>]$

definition *whn* :: *hypnat*

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

lemma *hypnat-of-nat-neq-whn*: $\text{hypnat-of-nat } n \neq \text{whn}$
by (*simp add: FreeUltrafilterNat.singleton' hypnat-omega-def star-of-def star-n-eq-iff*)

lemma *whn-neq-hypnat-of-nat*: $\text{whn} \neq \text{hypnat-of-nat } n$
by (*simp add: FreeUltrafilterNat.singleton hypnat-omega-def star-of-def star-n-eq-iff*)

lemma *whn-not-Nats* [*simp*]: $whn \notin Nats$
by (*simp add: Nats-def image-def whn-neq-hypnat-of-nat*)

lemma *HNatInfinite-whn* [*simp*]: $whn \in HNatInfinite$
by (*simp add: HNatInfinite-def*)

lemma *lemma-unbounded-set* [*simp*]: $eventually (\lambda n::nat. m < n) \mathcal{U}$
by (*rule filter-leD[OF FreeUltrafilterNat.le-cofinite]*)
(auto simp add: cofinite-eq-sequentially eventually-at-top-dense)

lemma *hypnat-of-nat-eq*: $hypnat-of-nat\ m = star-n (\lambda n::nat. m)$
by (*simp add: star-of-def*)

lemma *SHNat-eq*: $Nats = \{n. \exists N. n = hypnat-of-nat\ N\}$
by (*simp add: Nats-def image-def*)

lemma *Nats-less-whn*: $n \in Nats \implies n < whn$
by (*simp add: Nats-less-HNatInfinite*)

lemma *Nats-le-whn*: $n \in Nats \implies n \leq whn$
by (*simp add: Nats-le-HNatInfinite*)

lemma *hypnat-of-nat-less-whn* [*simp*]: $hypnat-of-nat\ n < whn$
by (*simp add: Nats-less-whn*)

lemma *hypnat-of-nat-le-whn* [*simp*]: $hypnat-of-nat\ n \leq whn$
by (*simp add: Nats-le-whn*)

lemma *hypnat-zero-less-hypnat-omega* [*simp*]: $0 < whn$
by (*simp add: Nats-less-whn*)

lemma *hypnat-one-less-hypnat-omega* [*simp*]: $1 < whn$
by (*simp add: Nats-less-whn*)

3.4.1 Alternative characterization of the set of infinite hypernaturals

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

unused, but possibly interesting

lemma *HNatInfinite-FreeUltrafilterNat-eventually*:
assumes $\bigwedge k::nat. eventually (\lambda n. f\ n \neq k) \mathcal{U}$
shows $eventually (\lambda n. m < f\ n) \mathcal{U}$
proof (*induct m*)
case 0
then show ?*case*
using *assms eventually-mono* **by** *fastforce*
next

```

case (Suc m)
then show ?case
  using assms [of Suc m] eventually-elim2 by fastforce
qed

```

lemma *HNatInfinite-iff*: $HNatInfinite = \{N. \forall n \in Nats. n < N\}$
 using *HNatInfinite-def Nats-less-HNatInfinite* by auto

3.4.2 Alternative Characterization of *HNatInfinite* using Free Ultrafilter

lemma *HNatInfinite-FreeUltrafilterNat*:
 $star\text{-}n\ X \in HNatInfinite \implies \forall u. eventually\ (\lambda n. u < X\ n)\ \mathcal{U}$
 by (metis (full-types) starP2-star-of starP-star-n star-less-def star-of-less-HNatInfinite)

lemma *FreeUltrafilterNat-HNatInfinite*:
 $\forall u. eventually\ (\lambda n. u < X\ n)\ \mathcal{U} \implies star\text{-}n\ X \in HNatInfinite$
 by (auto simp add: star-less-def starP2-star-n HNatInfinite-iff SHNat-eq hypnat-of-nat-eq)

lemma *HNatInfinite-FreeUltrafilterNat-iff*:
 $(star\text{-}n\ X \in HNatInfinite) = (\forall u. eventually\ (\lambda n. u < X\ n)\ \mathcal{U})$
 by (rule iffI [OF HNatInfinite-FreeUltrafilterNat FreeUltrafilterNat-HNatInfinite])

3.5 Embedding of the Hypernaturals into other types

definition *of-hypnat* :: *hypnat* \Rightarrow 'a::semiring-1-cancel star
 where *of-hypnat-def* [transfer-unfold]: *of-hypnat* = *f* *of-nat*

lemma *of-hypnat-0* [simp]: *of-hypnat* 0 = 0
 by transfer (rule of-nat-0)

lemma *of-hypnat-1* [simp]: *of-hypnat* 1 = 1
 by transfer (rule of-nat-1)

lemma *of-hypnat-hSuc*: $\bigwedge m. of\text{-hypnat}\ (hSuc\ m) = 1 + of\text{-hypnat}\ m$
 by transfer (rule of-nat-Suc)

lemma *of-hypnat-add* [simp]: $\bigwedge m\ n. of\text{-hypnat}\ (m + n) = of\text{-hypnat}\ m + of\text{-hypnat}\ n$
 by transfer (rule of-nat-add)

lemma *of-hypnat-mult* [simp]: $\bigwedge m\ n. of\text{-hypnat}\ (m * n) = of\text{-hypnat}\ m * of\text{-hypnat}\ n$
 by transfer (rule of-nat-mult)

lemma *of-hypnat-less-iff* [simp]:
 $\bigwedge m\ n. of\text{-hypnat}\ m < (of\text{-hypnat}\ n :: 'a::linordered-semidom\ star) \longleftrightarrow m < n$
 by transfer (rule of-nat-less-iff)

lemma *of-hypnat-0-less-iff* [simp]:
 $\bigwedge n. 0 < (\text{of-hypnat } n :: 'a::\text{linordered-semidom star}) \longleftrightarrow 0 < n$
by transfer (rule of-nat-0-less-iff)

lemma *of-hypnat-less-0-iff* [simp]: $\bigwedge m. \neg (\text{of-hypnat } m :: 'a::\text{linordered-semidom star}) < 0$
by transfer (rule of-nat-less-0-iff)

lemma *of-hypnat-le-iff* [simp]:
 $\bigwedge m n. \text{of-hypnat } m \leq (\text{of-hypnat } n :: 'a::\text{linordered-semidom star}) \longleftrightarrow m \leq n$
by transfer (rule of-nat-le-iff)

lemma *of-hypnat-0-le-iff* [simp]: $\bigwedge n. 0 \leq (\text{of-hypnat } n :: 'a::\text{linordered-semidom star})$
by transfer (rule of-nat-0-le-iff)

lemma *of-hypnat-le-0-iff* [simp]: $\bigwedge m. (\text{of-hypnat } m :: 'a::\text{linordered-semidom star}) \leq 0 \longleftrightarrow m = 0$
by transfer (rule of-nat-le-0-iff)

lemma *of-hypnat-eq-iff* [simp]:
 $\bigwedge m n. \text{of-hypnat } m = (\text{of-hypnat } n :: 'a::\text{linordered-semidom star}) \longleftrightarrow m = n$
by transfer (rule of-nat-eq-iff)

lemma *of-hypnat-eq-0-iff* [simp]: $\bigwedge m. (\text{of-hypnat } m :: 'a::\text{linordered-semidom star}) = 0 \longleftrightarrow m = 0$
by transfer (rule of-nat-eq-0-iff)

lemma *HNatInfinite-of-hypnat-gt-zero*:
 $N \in \text{HNatInfinite} \implies (0 :: 'a::\text{linordered-semidom star}) < \text{of-hypnat } N$
by (rule ccontr) (simp add: linorder-not-less)

end

4 Construction of Hyperreals Using Ultrafilters

theory *HyperDef*
imports *Complex-Main HyperNat*
begin

type-synonym *hypreal* = *real star*

abbreviation *hypreal-of-real* :: *real* \Rightarrow *real star*
where *hypreal-of-real* \equiv *star-of*

abbreviation *hypreal-of-hypnat* :: *hypnat* \Rightarrow *hypreal*
where *hypreal-of-hypnat* \equiv *of-hypnat*

definition *omega* :: *hypreal* ($\langle \omega \rangle$)

where $\omega = \text{star-n } (\lambda n. \text{real } (\text{Suc } n))$
 — an infinite number = $[<1, 2, 3, \dots>]$

definition $\text{epsilon} :: \text{hypreal } (\epsilon)$
where $\epsilon = \text{star-n } (\lambda n. \text{inverse } (\text{real } (\text{Suc } n)))$
 — an infinitesimal number = $[<1, 1/2, 1/3, \dots>]$

4.1 Real vector class instances

instantiation $\text{star} :: (\text{scaleR}) \text{ scaleR}$

begin

definition $\text{star-scaleR-def } [\text{transfer-unfold}]$: $\text{scaleR } r \equiv *f* (\text{scaleR } r)$

instance ..

end

lemma $\text{Standard-scaleR } [\text{simp}]$: $x \in \text{Standard} \implies \text{scaleR } r x \in \text{Standard}$
by $(\text{simp add: star-scaleR-def})$

lemma $\text{star-of-scaleR } [\text{simp}]$: $\text{star-of } (\text{scaleR } r x) = \text{scaleR } r (\text{star-of } x)$
by $\text{transfer } (\text{rule refl})$

instance $\text{star} :: (\text{real-vector}) \text{ real-vector}$

proof

fix $a b :: \text{real}$

show $\bigwedge x y :: 'a \text{ star. } \text{scaleR } a (x + y) = \text{scaleR } a x + \text{scaleR } a y$

by $\text{transfer } (\text{rule scaleR-right-distrib})$

show $\bigwedge x :: 'a \text{ star. } \text{scaleR } (a + b) x = \text{scaleR } a x + \text{scaleR } b x$

by $\text{transfer } (\text{rule scaleR-left-distrib})$

show $\bigwedge x :: 'a \text{ star. } \text{scaleR } a (\text{scaleR } b x) = \text{scaleR } (a * b) x$

by $\text{transfer } (\text{rule scaleR-scaleR})$

show $\bigwedge x :: 'a \text{ star. } \text{scaleR } 1 x = x$

by $\text{transfer } (\text{rule scaleR-one})$

qed

instance $\text{star} :: (\text{real-algebra}) \text{ real-algebra}$

proof

fix $a :: \text{real}$

show $\bigwedge x y :: 'a \text{ star. } \text{scaleR } a x * y = \text{scaleR } a (x * y)$

by $\text{transfer } (\text{rule mult-scaleR-left})$

show $\bigwedge x y :: 'a \text{ star. } x * \text{scaleR } a y = \text{scaleR } a (x * y)$

by $\text{transfer } (\text{rule mult-scaleR-right})$

qed

instance $\text{star} :: (\text{real-algebra-1}) \text{ real-algebra-1} ..$

instance $\text{star} :: (\text{real-div-algebra}) \text{ real-div-algebra} ..$

instance $\text{star} :: (\text{field-char-0}) \text{ field-char-0} ..$

instance *star* :: (*real-field*) *real-field* ..

lemma *star-of-real-def* [*transfer-unfold*]: *of-real* *r* = *star-of* (*of-real* *r*)
by (*unfold of-real-def*, *transfer*, *rule refl*)

lemma *Standard-of-real* [*simp*]: *of-real* *r* ∈ *Standard*
by (*simp add: star-of-real-def*)

lemma *star-of-of-real* [*simp*]: *star-of* (*of-real* *r*) = *of-real* *r*
by *transfer* (*rule refl*)

lemma *of-real-eq-star-of* [*simp*]: *of-real* = *star-of*
proof
show *of-real* *r* = *star-of* *r* **for** *r* :: *real*
by *transfer simp*
qed

lemma *Reals-eq-Standard*: (\mathbb{R} :: *hypreal set*) = *Standard*
by (*simp add: Reals-def Standard-def*)

4.2 Injection from *hypreal*

definition *of-hypreal* :: *hypreal* \Rightarrow '*a::real-algebra-1 star*'
where [*transfer-unfold*]: *of-hypreal* = **f** *of-real*

lemma *Standard-of-hypreal* [*simp*]: *r* ∈ *Standard* \implies *of-hypreal* *r* ∈ *Standard*
by (*simp add: of-hypreal-def*)

lemma *of-hypreal-0* [*simp*]: *of-hypreal* 0 = 0
by *transfer* (*rule of-real-0*)

lemma *of-hypreal-1* [*simp*]: *of-hypreal* 1 = 1
by *transfer* (*rule of-real-1*)

lemma *of-hypreal-add* [*simp*]: $\bigwedge x y. \text{of-hypreal } (x + y) = \text{of-hypreal } x + \text{of-hypreal } y$
by *transfer* (*rule of-real-add*)

lemma *of-hypreal-minus* [*simp*]: $\bigwedge x. \text{of-hypreal } (-x) = - \text{of-hypreal } x$
by *transfer* (*rule of-real-minus*)

lemma *of-hypreal-diff* [*simp*]: $\bigwedge x y. \text{of-hypreal } (x - y) = \text{of-hypreal } x - \text{of-hypreal } y$
by *transfer* (*rule of-real-diff*)

lemma *of-hypreal-mult* [*simp*]: $\bigwedge x y. \text{of-hypreal } (x * y) = \text{of-hypreal } x * \text{of-hypreal } y$
by *transfer* (*rule of-real-mult*)

lemma *of-hypreal-inverse* [simp]:
 $\bigwedge x. \text{of-hypreal } (\text{inverse } x) =$
 $\text{inverse } (\text{of-hypreal } x :: 'a::\{\text{real-div-algebra}, \text{division-ring}\} \text{ star})$
by *transfer (rule of-real-inverse)*

lemma *of-hypreal-divide* [simp]:
 $\bigwedge x y. \text{of-hypreal } (x / y) =$
 $(\text{of-hypreal } x / \text{of-hypreal } y :: 'a::\{\text{real-field}, \text{field}\} \text{ star})$
by *transfer (rule of-real-divide)*

lemma *of-hypreal-eq-iff* [simp]: $\bigwedge x y. (\text{of-hypreal } x = \text{of-hypreal } y) = (x = y)$
by *transfer (rule of-real-eq-iff)*

lemma *of-hypreal-eq-0-iff* [simp]: $\bigwedge x. (\text{of-hypreal } x = 0) = (x = 0)$
by *transfer (rule of-real-eq-0-iff)*

4.3 Properties of *starrel*

lemma *lemma-starrel-refl* [simp]: $x \in \text{starrel} \text{ “ } \{x\}$
by *(simp add: starrel-def)*

lemma *starrel-in-hypreal* [simp]: $\text{starrel} \text{ “ } \{x\} \in \text{star}$
by *(simp add: star-def starrel-def quotient-def, blast)*

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

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

lemma *inj-star-of*: *inj star-of*
by *(rule inj-onI) simp*

lemma *mem-Rep-star-iff*: $X \in \text{Rep-star } x \longleftrightarrow x = \text{star-n } X$
by *(cases x) (simp add: star-n-def)*

lemma *Rep-star-star-n-iff* [simp]: $X \in \text{Rep-star } (\text{star-n } Y) \longleftrightarrow \text{eventually } (\lambda n. Y\ n = X\ n) \mathcal{U}$
by *(simp add: star-n-def)*

lemma *Rep-star-star-n*: $X \in \text{Rep-star } (\text{star-n } X)$
by *simp*

4.5 Properties of *star-n*

lemma *star-n-add*: $\text{star-n } X + \text{star-n } Y = \text{star-n } (\lambda n. X\ n + Y\ n)$
by *(simp only: star-add-def starfun2-star-n)*

lemma *star-n-minus*: $-\text{star-n } X = \text{star-n } (\lambda n. -(X\ n))$
by *(simp only: star-minus-def starfun-star-n)*

lemma *star-n-diff*: $\text{star-n } X - \text{star-n } Y = \text{star-n } (\lambda n. X \ n - Y \ n)$
by (*simp only*: *star-diff-def starfun2-star-n*)

lemma *star-n-mult*: $\text{star-n } X * \text{star-n } Y = \text{star-n } (\lambda n. X \ n * Y \ n)$
by (*simp only*: *star-mult-def starfun2-star-n*)

lemma *star-n-inverse*: $\text{inverse } (\text{star-n } X) = \text{star-n } (\lambda n. \text{inverse } (X \ n))$
by (*simp only*: *star-inverse-def starfun-star-n*)

lemma *star-n-le*: $\text{star-n } X \leq \text{star-n } Y = \text{eventually } (\lambda n. X \ n \leq Y \ n) \ \mathcal{U}$
by (*simp only*: *star-le-def starP2-star-n*)

lemma *star-n-less*: $\text{star-n } X < \text{star-n } Y = \text{eventually } (\lambda n. X \ n < Y \ n) \ \mathcal{U}$
by (*simp only*: *star-less-def starP2-star-n*)

lemma *star-n-zero-num*: $0 = \text{star-n } (\lambda n. 0)$
by (*simp only*: *star-zero-def star-of-def*)

lemma *star-n-one-num*: $1 = \text{star-n } (\lambda n. 1)$
by (*simp only*: *star-one-def star-of-def*)

lemma *star-n-abs*: $|\text{star-n } X| = \text{star-n } (\lambda n. |X \ n|)$
by (*simp only*: *star-abs-def starfun-star-n*)

lemma *hypreal-omega-gt-zero* [*simp*]: $0 < \omega$
by (*simp add*: *omega-def star-n-zero-num star-n-less*)

4.6 Existence of Infinite Hyperreal Number

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

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

proof –

have *False* **if** $\forall_F n \text{ in } \mathcal{U}. x = 1 + \text{real } n$ **for** x

proof –

have *finite* $\{n::\text{nat}. x = 1 + \text{real } n\}$

by (*simp add*: *finite-nat-set-iff-bounded-le*) (*metis add.commute nat-le-linear nat-le-real-less*)

then show *False*

using *FreeUltrafilterNat.finite* **that** **by** *blast*

qed

then show *?thesis*

by (*auto simp add*: *omega-def star-of-def star-n-eq-iff*)

qed

Existence of infinitesimal number also not corresponding to any real number.

lemma *hypreal-of-real-not-eq-epsilon*: $\text{hypreal-of-real } x \neq \varepsilon$

proof –

```

have False if  $\forall_F n \text{ in } \mathcal{U}. x = \text{inverse } (1 + \text{real } n)$  for  $x$ 
proof –
  have finite  $\{n::\text{nat}. x = \text{inverse } (1 + \text{real } n)\}$ 
  by (simp add: finite-nat-set-iff-bounded-le) (metis add.commute inverse-inverse-eq
linear nat-le-real-less of-nat-Suc)
  then show False
  using FreeUltrafilterNat.finite that by blast
qed
then show ?thesis
  by (auto simp: epsilon-def star-of-def star-n-eq-iff)
qed

```

```

lemma epsilon-ge-zero [simp]:  $0 \leq \varepsilon$ 
  by (simp add: epsilon-def star-n-zero-num star-n-le)

```

```

lemma epsilon-not-zero:  $\varepsilon \neq 0$ 
  using hypreal-of-real-not-eq-epsilon by force

```

```

lemma epsilon-inverse-omega:  $\varepsilon = \text{inverse } \omega$ 
  by (simp add: epsilon-def omega-def star-n-inverse)

```

```

lemma epsilon-gt-zero:  $0 < \varepsilon$ 
  by (simp add: epsilon-inverse-omega)

```

4.7 Embedding the Naturals into the Hyperreals

```

abbreviation hypreal-of-nat ::  $\text{nat} \Rightarrow \text{hypreal}$ 
  where hypreal-of-nat  $\equiv \text{of-nat}$ 

```

```

lemma SNat-eq:  $\text{Nats} = \{n. \exists N. n = \text{hypreal-of-nat } N\}$ 
  by (simp add: Nats-def image-def)

```

Naturals embedded in hyperreals: is a hyperreal c.f. NS extension.

```

lemma hypreal-of-nat:  $\text{hypreal-of-nat } m = \text{star-n } (\lambda n. \text{real } m)$ 
  by (simp add: star-of-def [symmetric])

```

```

declaration ⟨
  K (Lin-Arith.add-simps @{thms star-of-zero star-of-one
    star-of-numeral star-of-add
    star-of-minus star-of-diff star-of-mult}
  #> Lin-Arith.add-inj-thms @{thms star-of-le [THEN iffD2]
    star-of-less [THEN iffD2] star-of-eq [THEN iffD2]}}
  #> Lin-Arith.add-inj-const (const-name ⟨StarDef.star-of⟩, typ ⟨real  $\Rightarrow$  hypreal⟩))
  ⟩

```

```

simproc-setup fast-arith-hypreal ((m::hypreal) < n | (m::hypreal) ≤ n | (m::hypreal)
= n) =
  ⟨K Lin-Arith.simproc⟩

```

4.8 Exponentials on the Hyperreals

lemma *hpowr-0* [simp]: $r \wedge 0 = (1::\text{hypreal})$
for $r :: \text{hypreal}$
by (rule power-0)

lemma *hpowr-Suc* [simp]: $r \wedge (\text{Suc } n) = r * (r \wedge n)$
for $r :: \text{hypreal}$
by (rule power-Suc)

lemma *hrealpow*: $\text{star-}n \ X \wedge m = \text{star-}n \ (\lambda n. (X \ n::\text{real}) \wedge m)$
by (induct m) (auto simp: star-n-one-num star-n-mult)

lemma *hrealpow-sum-square-expand*:
 $(x + y) \wedge \text{Suc } (\text{Suc } 0) =$
 $x \wedge \text{Suc } (\text{Suc } 0) + y \wedge \text{Suc } (\text{Suc } 0) + (\text{hypreal-of-nat } (\text{Suc } (\text{Suc } 0))) * x * y$
for $x \ y :: \text{hypreal}$
by (simp add: distrib-left distrib-right)

lemma *power-hypreal-of-real-numeral*:
 $(\text{numeral } v :: \text{hypreal}) \wedge n = \text{hypreal-of-real } ((\text{numeral } v) \wedge n)$
by simp

declare *power-hypreal-of-real-numeral* [of - numeral w, simp] **for** w

lemma *power-hypreal-of-real-neg-numeral*:
 $(- \text{numeral } v :: \text{hypreal}) \wedge n = \text{hypreal-of-real } ((- \text{numeral } v) \wedge n)$
by simp

declare *power-hypreal-of-real-neg-numeral* [of - numeral w, simp] **for** w

4.9 Powers with Hypernatural Exponents

Hypernatural powers of hyperreals.

definition *pow* :: $'a::\text{power star} \Rightarrow \text{nat star} \Rightarrow 'a \text{ star}$ (**infixr** $\langle \text{pow} \rangle$ 80)
where *hyperpow-def* [transfer-unfold]: $R \text{ pow } N = (*f2* \ (\wedge)) \ R \ N$

lemma *Standard-hyperpow* [simp]: $r \in \text{Standard} \Longrightarrow n \in \text{Standard} \Longrightarrow r \text{ pow } n \in \text{Standard}$
by (simp add: hyperpow-def)

lemma *hyperpow*: $\text{star-}n \ X \text{ pow } \text{star-}n \ Y = \text{star-}n \ (\lambda n. X \ n \wedge Y \ n)$
by (simp add: hyperpow-def starfun2-star-n)

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

lemma *hyperpow-not-zero*: $\bigwedge r \ n. r \neq (0::'a::\{\text{field}\} \text{ star}) \Longrightarrow r \text{ pow } n \neq 0$
by transfer (rule power-not-zero)

lemma *hyperpow-inverse*: $\bigwedge r n. r \neq (0::'a::\text{field star}) \implies \text{inverse } (r \text{ pow } n) = (\text{inverse } r) \text{ pow } n$

by *transfer* (*rule power-inverse [symmetric]*)

lemma *hyperpow-hrabs*: $\bigwedge r n. |r::'a::\{\text{linordered-idom}\} \text{ star}| \text{ pow } n = |r \text{ pow } n|$

by *transfer* (*rule power-abs [symmetric]*)

lemma *hyperpow-add*: $\bigwedge r n m. (r::'a::\text{monoid-mult star}) \text{ pow } (n + m) = (r \text{ pow } n) * (r \text{ pow } m)$

by *transfer* (*rule power-add*)

lemma *hyperpow-one [simp]*: $\bigwedge r. (r::'a::\text{monoid-mult star}) \text{ pow } (1::\text{hypnat}) = r$

by *transfer* (*rule power-one-right*)

lemma *hyperpow-two*: $\bigwedge r. (r::'a::\text{monoid-mult star}) \text{ pow } (2::\text{hypnat}) = r * r$

by *transfer* (*rule power2-eq-square*)

lemma *hyperpow-gt-zero*: $\bigwedge r n. (0::'a::\{\text{linordered-semidom}\} \text{ star}) < r \implies 0 < r \text{ pow } n$

by *transfer* (*rule zero-less-power*)

lemma *hyperpow-ge-zero*: $\bigwedge r n. (0::'a::\{\text{linordered-semidom}\} \text{ star}) \leq r \implies 0 \leq r \text{ pow } n$

by *transfer* (*rule zero-le-power*)

lemma *hyperpow-le*: $\bigwedge x y n. (0::'a::\{\text{linordered-semidom}\} \text{ star}) < x \implies x \leq y \implies x \text{ pow } n \leq y \text{ pow } n$

by *transfer* (*rule power-mono [OF - order-less-imp-le]*)

lemma *hyperpow-eq-one [simp]*: $\bigwedge n. 1 \text{ pow } n = (1::'a::\text{monoid-mult star})$

by *transfer* (*rule power-one*)

lemma *hrabs-hyperpow-minus [simp]*: $\bigwedge (a::'a::\text{linordered-idom star}) n. |(-a) \text{ pow } n| = |a \text{ pow } n|$

by *transfer* (*rule abs-power-minus*)

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

by *transfer* (*rule power-mult-distrib*)

lemma *hyperpow-two-le [simp]*: $\bigwedge r. (0::'a::\{\text{monoid-mult, linordered-ring-strict}\} \text{ star}) \leq r \text{ pow } 2$

by (*auto simp add: hyperpow-two zero-le-mult-iff*)

lemma *hyperpow-two-hrabs [simp]*: $|x::'a::\text{linordered-idom star}| \text{ pow } 2 = x \text{ pow } 2$

by (*simp add: hyperpow-hrabs*)

lemma *hyperpow-two-gt-one*: $\bigwedge r::'a::\text{linordered-semidom star}. 1 < r \implies 1 < r \text{ pow } 2$

by *transfer simp*

lemma *hyperpow-two-ge-one*: $\bigwedge r::'a::\text{linordered-semidom star. } 1 \leq r \implies 1 \leq r^{\text{pow } 2}$
by *transfer (rule one-le-power)*

lemma *two-hyperpow-ge-one* [simp]: $(1::\text{hypreal}) \leq 2^{\text{pow } n}$
by (*metis hyperpow-eq-one hyperpow-le one-le-numeral zero-less-one*)

lemma *hyperpow-minus-one2* [simp]: $\bigwedge n. (-1)^{\text{pow } (2 * n)} = (1::\text{hypreal})$
by *transfer (rule power-minus1-even)*

lemma *hyperpow-less-le*: $\bigwedge r n N. (0::\text{hypreal}) \leq r \implies r \leq 1 \implies n < N \implies r^{\text{pow } N} \leq r^{\text{pow } n}$
by *transfer (rule power-decreasing [OF order-less-imp-le])*

lemma *hyperpow-SHNat-le*:
 $0 \leq r \implies r \leq (1::\text{hypreal}) \implies N \in \text{HNatInfinite} \implies \forall n \in \text{Nats. } r^{\text{pow } N} \leq r^{\text{pow } n}$
by (*auto intro!: hyperpow-less-le simp: HNatInfinite-iff*)

lemma *hyperpow-realpow*: $(\text{hypreal-of-real } r)^{\text{pow } (\text{hypnat-of-nat } n)} = \text{hypreal-of-real } (r \wedge n)$
by *transfer (rule refl)*

lemma *hyperpow-SReal* [simp]: $(\text{hypreal-of-real } r)^{\text{pow } (\text{hypnat-of-nat } n)} \in \mathbb{R}$
by (*simp add: Reals-eq-Standard*)

lemma *hyperpow-zero-HNatInfinite* [simp]: $N \in \text{HNatInfinite} \implies (0::\text{hypreal})^{\text{pow } N} = 0$
by (*drule HNatInfinite-is-Suc, auto*)

lemma *hyperpow-le-le*: $(0::\text{hypreal}) \leq r \implies r \leq 1 \implies n \leq N \implies r^{\text{pow } N} \leq r^{\text{pow } n}$
by (*metis hyperpow-less-le le-less*)

lemma *hyperpow-Suc-le-self2*: $(0::\text{hypreal}) \leq r \implies r < 1 \implies r^{\text{pow } (n + (1::\text{hypnat}))} \leq r^{\text{pow } n}$
by (*metis hyperpow-less-le hyperpow-one hypnat-add-self-le le-less*)

lemma *hyperpow-hypnat-of-nat*: $\bigwedge x. x^{\text{pow } \text{hypnat-of-nat } n} = x \wedge n$
by *transfer (rule refl)*

lemma *of-hypreal-hyperpow*:
 $\bigwedge x n. \text{of-hypreal } (x^{\text{pow } n}) = (\text{of-hypreal } x::'a::\{\text{real-algebra-1}\} \text{ star})^{\text{pow } n}$
by *transfer (rule of-real-power)*

end

5 Infinite Numbers, Infinitesimals, Infinitely Close Relation

theory NSA

imports HyperDef HOL-Library.Lub-Glb

begin

definition *hnorm* :: 'a::real-normed-vector star \Rightarrow real star
where [transfer-unfold]: *hnorm* = **f** *norm*

definition *Infinitesimal* :: ('a::real-normed-vector) star set
where *Infinitesimal* = {*x*. $\forall r \in \text{Reals. } 0 < r \longrightarrow \text{hnorm } x < r$ }

definition *HFinite* :: ('a::real-normed-vector) star set
where *HFinite* = {*x*. $\exists r \in \text{Reals. } \text{hnorm } x < r$ }

definition *HInfinite* :: ('a::real-normed-vector) star set
where *HInfinite* = {*x*. $\forall r \in \text{Reals. } r < \text{hnorm } x$ }

definition *approx* :: 'a::real-normed-vector star \Rightarrow 'a star \Rightarrow bool (**infixl** $\langle \approx \rangle$ 50)
where $x \approx y \longleftrightarrow x - y \in \text{Infinitesimal}$
— the “infinitely close” relation

definition *st* :: hypreal \Rightarrow hypreal
where *st* = ($\lambda x. \text{SOME } r. x \in \text{HFinite} \wedge r \in \mathbb{R} \wedge r \approx x$)
— the standard part of a hyperreal

definition *monad* :: 'a::real-normed-vector star \Rightarrow 'a star set
where *monad* *x* = {*y*. $x \approx y$ }

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

lemma *SReal-def*: $\mathbb{R} \equiv \{x. \exists r. x = \text{hypreal-of-real } r\}$
by (*simp add: Reals-def image-def*)

5.1 Nonstandard Extension of the Norm Function

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

lemma *Standard-hnorm* [*simp*]: $x \in \text{Standard} \implies \text{hnorm } x \in \text{Standard}$
by (*simp add: hnorm-def*)

lemma *star-of-norm* [*simp*]: $\text{star-of } (\text{norm } x) = \text{hnorm } (\text{star-of } x)$
by *transfer (rule refl)*

lemma *hnorm-ge-zero* [*simp*]: $\bigwedge x::'a::\text{real-normed-vector star. } 0 \leq \text{hnorm } x$
by *transfer (rule norm-ge-zero)*

lemma *hnorm-eq-zero* [simp]: $\bigwedge x::'a::\text{real-normed-vector star. } \text{hnorm } x = 0 \longleftrightarrow x = 0$

by transfer (rule norm-eq-zero)

lemma *hnorm-triangle-ineq*: $\bigwedge x y::'a::\text{real-normed-vector star. } \text{hnorm } (x + y) \leq \text{hnorm } x + \text{hnorm } y$

by transfer (rule norm-triangle-ineq)

lemma *hnorm-triangle-ineq3*: $\bigwedge x y::'a::\text{real-normed-vector star. } |\text{hnorm } x - \text{hnorm } y| \leq \text{hnorm } (x - y)$

by transfer (rule norm-triangle-ineq3)

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

by transfer (rule norm-scaleR)

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

by transfer (rule norm-scaleR)

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

by transfer (rule norm-mult-ineq)

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

by transfer (rule norm-mult)

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

by transfer (rule norm-power)

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

by transfer (rule norm-one)

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

by transfer (rule norm-zero)

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

by transfer (rule zero-less-norm-iff)

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

by transfer (rule norm-minus-cancel)

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

by *transfer* (*rule norm-minus-commute*)

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

by *transfer* (*rule norm-triangle-ineq2*)

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

by *transfer* (*rule norm-triangle-ineq4*)

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

by *transfer* (*rule abs-norm-cancel*)

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

by *transfer* (*rule norm-of-real*)

lemma *nonzero-hnorm-inverse*:

$\bigwedge a::'a::\text{real-normed-div-algebra star}.$ $a \neq 0 \implies \text{hnorm } (\text{inverse } a) = \text{inverse } (\text{hnorm } a)$

by *transfer* (*rule nonzero-norm-inverse*)

lemma *hnorm-inverse*:

$\bigwedge a::'a::\{\text{real-normed-div-algebra, division-ring}\} \text{ star}.$ $\text{hnorm } (\text{inverse } a) = \text{inverse } (\text{hnorm } a)$

by *transfer* (*rule norm-inverse*)

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

by *transfer* (*rule norm-divide*)

lemma *hypreal-hnorm-def* [*simp*]: $\bigwedge r::\text{hypreal}.$ $\text{hnorm } r = |r|$

by *transfer* (*rule real-norm-def*)

lemma *hnorm-add-less*:

$\bigwedge (x::'a::\text{real-normed-vector star})\ y\ r\ s.$ $\text{hnorm } x < r \implies \text{hnorm } y < s \implies \text{hnorm } (x + y) < r + s$

by *transfer* (*rule norm-add-less*)

lemma *hnorm-mult-less*:

$\bigwedge (x::'a::\text{real-normed-algebra star})\ y\ r\ s.$ $\text{hnorm } x < r \implies \text{hnorm } y < s \implies \text{hnorm } (x * y) < r * s$

by *transfer* (*rule norm-mult-less*)

lemma *hnorm-scaleHR-less*: $|x| < r \implies \text{hnorm } y < s \implies \text{hnorm } (\text{scaleHR } x\ y) < r * s$

by (*simp only: hnorm-scaleHR*) (*simp add: mult-strict-mono'*)

5.2 Closure Laws for the Standard Reals

lemma *Reals-add-cancel*: $x + y \in \mathbb{R} \implies y \in \mathbb{R} \implies x \in \mathbb{R}$
by (*drule* (1) *Reals-diff*) *simp*

lemma *SReal-hrabs*: $x \in \mathbb{R} \implies |x| \in \mathbb{R}$
for $x :: \text{hypreal}$
by (*simp add: Reals-eq-Standard*)

lemma *SReal-hypreal-of-real* [*simp*]: *hypreal-of-real* $x \in \mathbb{R}$
by (*simp add: Reals-eq-Standard*)

lemma *SReal-divide-numeral*: $r \in \mathbb{R} \implies r / (\text{numeral } w :: \text{hypreal}) \in \mathbb{R}$
by *simp*

ε is not in Reals because it is an infinitesimal

lemma *SReal-epsilon-not-mem*: $\varepsilon \notin \mathbb{R}$
by (*auto simp: SReal-def hypreal-of-real-not-eq-epsilon* [*symmetric*])

lemma *SReal-omega-not-mem*: $\omega \notin \mathbb{R}$
by (*auto simp: SReal-def hypreal-of-real-not-eq-omega* [*symmetric*])

lemma *SReal-UNIV-real*: $\{x. \text{hypreal-of-real } x \in \mathbb{R}\} = (\text{UNIV} :: \text{real set})$
by *simp*

lemma *SReal-iff*: $x \in \mathbb{R} \longleftrightarrow (\exists y. x = \text{hypreal-of-real } y)$
by (*simp add: SReal-def*)

lemma *hypreal-of-real-image*: *hypreal-of-real* ‘ $(\text{UNIV} :: \text{real set})$ ’ = \mathbb{R}
by (*simp add: Reals-eq-Standard Standard-def*)

lemma *inv-hypreal-of-real-image*: *inv hypreal-of-real* ‘ \mathbb{R} ’ = *UNIV*
by (*simp add: Reals-eq-Standard Standard-def inj-star-of*)

lemma *SReal-dense*: $x \in \mathbb{R} \implies y \in \mathbb{R} \implies x < y \implies \exists r \in \text{Reals}. x < r \wedge r < y$
for $x y :: \text{hypreal}$
using *dense* **by** (*fastforce simp add: SReal-def*)

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

lemma *HFinite-add*: $x \in \text{HFinite} \implies y \in \text{HFinite} \implies x + y \in \text{HFinite}$
unfolding *HFinite-def* **by** (*blast intro!: Reals-add hnorm-add-less*)

lemma *HFinite-mult*: $x \in \text{HFinite} \implies y \in \text{HFinite} \implies x * y \in \text{HFinite}$
for $x y :: 'a :: \text{real-normed-algebra star}$
unfolding *HFinite-def* **by** (*blast intro!: Reals-mult hnorm-mult-less*)

lemma *HFinite-scaleHR*: $x \in \text{HFinite} \implies y \in \text{HFinite} \implies \text{scaleHR } x y \in \text{HFinite}$
by (*auto simp: HFinite-def intro!: Reals-mult hnorm-scaleHR-less*)

lemma *HFinite-minus-iff*: $- x \in \text{HFinite} \longleftrightarrow x \in \text{HFinite}$
by (*simp add: HFinite-def*)

lemma *HFinite-star-of [simp]*: $\text{star-of } x \in \text{HFinite}$
by (*simp add: HFinite-def*) (*metis SReal-hypreal-of-real gt-ex star-of-less star-of-norm*)

lemma *SReal-subset-HFinite*: $(\mathbb{R}::\text{hypreal set}) \subseteq \text{HFinite}$
by (*auto simp add: SReal-def*)

lemma *HFiniteD*: $x \in \text{HFinite} \implies \exists t \in \text{Reals}. \text{hnorm } x < t$
by (*simp add: HFinite-def*)

lemma *HFinite-hrabs-iff [iff]*: $|x| \in \text{HFinite} \longleftrightarrow x \in \text{HFinite}$
for $x :: \text{hypreal}$
by (*simp add: HFinite-def*)

lemma *HFinite-hnorm-iff [iff]*: $\text{hnorm } x \in \text{HFinite} \longleftrightarrow x \in \text{HFinite}$
for $x :: \text{hypreal}$
by (*simp add: HFinite-def*)

lemma *HFinite-numeral [simp]*: $\text{numeral } w \in \text{HFinite}$
unfolding *star-numeral-def* **by** (*rule HFinite-star-of*)

As always with numerals, 0 and 1 are special cases.

lemma *HFinite-0 [simp]*: $0 \in \text{HFinite}$
unfolding *star-zero-def* **by** (*rule HFinite-star-of*)

lemma *HFinite-1 [simp]*: $1 \in \text{HFinite}$
unfolding *star-one-def* **by** (*rule HFinite-star-of*)

lemma *hrealpow-HFinite*: $x \in \text{HFinite} \implies x \wedge n \in \text{HFinite}$
for $x :: 'a::\{\text{real-normed-algebra}, \text{monoid-mult}\}$ *star*
by (*induct n*) (*auto intro: HFinite-mult*)

lemma *HFinite-bounded*:
fixes $x y :: \text{hypreal}$
assumes $x \in \text{HFinite}$ **and** $y: y \leq x \wedge 0 \leq y$ **shows** $y \in \text{HFinite}$
proof (*cases* $x \leq 0$)
case *True*
then have $y = 0$
using y **by** *auto*
then show *?thesis*
by *simp*
next
case *False*
then show *?thesis*
using *assms le-less-trans* **by** (*auto simp: HFinite-def*)
qed

5.4 Set of Infinitesimals is a Subring of the Hyperreals

lemma *InfinitesimalI*: $(\bigwedge r. r \in \mathbb{R} \implies 0 < r \implies \text{hnorm } x < r) \implies x \in \text{Infinitesimal}$

by (*simp add: Infinitesimal-def*)

lemma *InfinitesimalD*: $x \in \text{Infinitesimal} \implies \forall r \in \text{Reals}. 0 < r \longrightarrow \text{hnorm } x < r$

by (*simp add: Infinitesimal-def*)

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

by (*auto simp add: Infinitesimal-def SReal-def*)

lemma *InfinitesimalD2*: $x \in \text{Infinitesimal} \implies 0 < r \implies \text{hnorm } x < \text{star-of } r$

by (*auto simp add: Infinitesimal-def SReal-def*)

lemma *Infinitesimal-zero [iff]*: $0 \in \text{Infinitesimal}$

by (*simp add: Infinitesimal-def*)

lemma *Infinitesimal-add*:

assumes $x \in \text{Infinitesimal } y \in \text{Infinitesimal}$

shows $x + y \in \text{Infinitesimal}$

proof (*rule InfinitesimalI*)

show $\text{hnorm } (x + y) < r$

if $r \in \mathbb{R}$ **and** $0 < r$ **for** $r :: \text{real star}$

proof –

have $\text{hnorm } x < r/2$ $\text{hnorm } y < r/2$

using *InfinitesimalD SReal-divide-numeral* *assms half-gt-zero* **that** **by** *blast+*

then show *?thesis*

using *hnorm-add-less* **by** *fastforce*

qed

qed

lemma *Infinitesimal-minus-iff [simp]*: $-x \in \text{Infinitesimal} \longleftrightarrow x \in \text{Infinitesimal}$

by (*simp add: Infinitesimal-def*)

lemma *Infinitesimal-hnorm-iff*: $\text{hnorm } x \in \text{Infinitesimal} \longleftrightarrow x \in \text{Infinitesimal}$

by (*simp add: Infinitesimal-def*)

lemma *Infinitesimal-hrabs-iff [iff]*: $|x| \in \text{Infinitesimal} \longleftrightarrow x \in \text{Infinitesimal}$

for $x :: \text{hypreal}$

by (*simp add: abs-if*)

lemma *Infinitesimal-of-hypreal-iff [simp]*:

$(\text{of-hypreal } x :: 'a :: \text{real-normed-algebra-1 star}) \in \text{Infinitesimal} \longleftrightarrow x \in \text{Infinitesimal}$

by (*subst Infinitesimal-hnorm-iff [symmetric]*) *simp*

lemma *Infinitesimal-diff*: $x \in \text{Infinitesimal} \implies y \in \text{Infinitesimal} \implies x - y \in \text{Infinitesimal}$

using *Infinitesimal-add [of x - y]* **by** *simp*

```

lemma Infinitesimal-mult:
  fixes  $x\ y :: 'a::\text{real-normed-algebra}\ \text{star}$ 
  assumes  $x \in \text{Infinitesimal}\ y \in \text{Infinitesimal}$ 
  shows  $x * y \in \text{Infinitesimal}$ 
  proof (rule InfinitesimalI)
  show  $\text{hnorm}\ (x * y) < r$ 
    if  $r \in \mathbb{R}$  and  $0 < r$  for  $r :: \text{real}\ \text{star}$ 
  proof –
    have  $\text{hnorm}\ x < 1\ \text{hnorm}\ y < r$ 
      using assms that by (auto simp add: InfinitesimalD)
    then show ?thesis
      using hnorm-mult-less by fastforce
  qed
qed

lemma Infinitesimal-HFinite-mult:
  fixes  $x\ y :: 'a::\text{real-normed-algebra}\ \text{star}$ 
  assumes  $x \in \text{Infinitesimal}\ y \in \text{HFinite}$ 
  shows  $x * y \in \text{Infinitesimal}$ 
  proof (rule InfinitesimalI)
  obtain  $t$  where  $\text{hnorm}\ y < t\ t \in \text{Reals}$ 
    using HFiniteD  $\langle y \in \text{HFinite} \rangle$  by blast
  then have  $t > 0$ 
    using hnorm-ge-zero le-less-trans by blast
  show  $\text{hnorm}\ (x * y) < r$ 
    if  $r \in \mathbb{R}$  and  $0 < r$  for  $r :: \text{real}\ \text{star}$ 
  proof –
    have  $\text{hnorm}\ x < r/t$ 
      by (meson InfinitesimalD Reals-divide  $\langle \text{hnorm}\ y < t \rangle\ \langle t \in \mathbb{R} \rangle\ \text{assms}(1)$ )
    divide-pos-pos hnorm-ge-zero le-less-trans that
    then have  $\text{hnorm}\ (x * y) < (r / t) * t$ 
      using  $\langle \text{hnorm}\ y < t \rangle\ \text{hnorm-mult-less}$  by blast
    then show ?thesis
      using  $\langle 0 < t \rangle$  by auto
  qed
qed

lemma Infinitesimal-HFinite-scaleHR:
  assumes  $x \in \text{Infinitesimal}\ y \in \text{HFinite}$ 
  shows scaleHR  $x\ y \in \text{Infinitesimal}$ 
  proof (rule InfinitesimalI)
  obtain  $t$  where  $\text{hnorm}\ y < t\ t \in \text{Reals}$ 
    using HFiniteD  $\langle y \in \text{HFinite} \rangle$  by blast
  then have  $t > 0$ 
    using hnorm-ge-zero le-less-trans by blast
  show  $\text{hnorm}\ (\text{scaleHR}\ x\ y) < r$ 
    if  $r \in \mathbb{R}$  and  $0 < r$  for  $r :: \text{real}\ \text{star}$ 
  proof –
    have  $|x| * \text{hnorm}\ y < (r / t) * t$ 

```

by (metis InfinitesimalD Reals-divide $\langle 0 < t \rangle \langle \text{hnorm } y < t \rangle \langle t \in \mathbb{R} \rangle \text{assms}(1)$
 divide-pos-pos hnorm-ge-zero hypreal-hnorm-def mult-strict-mono' that)
 then show ?thesis
 by (simp add: $\langle 0 < t \rangle \text{hnorm-scaleHR less-imp-not-eq2}$)
 qed
 qed

lemma Infinitesimal-HFinite-mult2:

fixes $x \ y :: 'a::\text{real-normed-algebra star}$
 assumes $x \in \text{Infinitesimal } y \in \text{HFinite}$
 shows $y * x \in \text{Infinitesimal}$
proof (rule InfinitesimalI)
 obtain t where $\text{hnorm } y < t \ t \in \text{Reals}$
 using HFiniteD $\langle y \in \text{HFinite} \rangle$ by blast
 then have $t > 0$
 using hnorm-ge-zero le-less-trans by blast
 show $\text{hnorm } (y * x) < r$
 if $r \in \mathbb{R}$ and $0 < r$ for $r :: \text{real star}$
proof –
 have $\text{hnorm } x < r/t$
 by (meson InfinitesimalD Reals-divide $\langle \text{hnorm } y < t \rangle \langle t \in \mathbb{R} \rangle \text{assms}(1)$
 divide-pos-pos hnorm-ge-zero le-less-trans that)
 then have $\text{hnorm } (y * x) < t * (r / t)$
 using $\langle \text{hnorm } y < t \rangle \text{hnorm-mult-less}$ by blast
 then show ?thesis
 using $\langle 0 < t \rangle$ by auto
 qed
 qed

lemma Infinitesimal-scaleR2:

assumes $x \in \text{Infinitesimal}$ shows $a *_{\mathbb{R}} x \in \text{Infinitesimal}$
 by (metis HFinite-star-of Infinitesimal-HFinite-mult2 Infinitesimal-hnorm-iff
 assms hnorm-scaleR hypreal-hnorm-def star-of-norm)

lemma Compl-HFinite: $-\text{HFinite} = \text{HInfinite}$

proof –
 have $r < \text{hnorm } x$ if $*: \bigwedge s. s \in \mathbb{R} \implies s \leq \text{hnorm } x$ and $r \in \mathbb{R}$
 for $x :: 'a \text{ star}$ and $r :: \text{hypreal}$
 using $* \text{ [of } r+1 \text{]} \langle r \in \mathbb{R} \rangle$ by auto
 then show ?thesis
 by (auto simp add: HInfinite-def HFinite-def linorder-not-less)
 qed

lemma HInfinite-inverse-Infinitesimal:

$x \in \text{HInfinite} \implies \text{inverse } x \in \text{Infinitesimal}$
 for $x :: 'a::\text{real-normed-div-algebra star}$
 by (simp add: HInfinite-def InfinitesimalI hnorm-inverse inverse-less-imp-less)

lemma inverse-Infinitesimal-iff-HInfinite:

$x \neq 0 \implies \text{inverse } x \in \text{Infinitesimal} \longleftrightarrow x \in \text{HInfinite}$
for $x :: 'a::\text{real-normed-div-algebra star}$
by (*metis Compl-HFinite Compl-iff HInfinite-inverse-Infinitesimal InfinitesimalD*
Infinitesimal-HFinite-mult Reals-1 hnorm-one left-inverse less-irrefl zero-less-one)

lemma *HInfiniteI*: $(\bigwedge r. r \in \mathbb{R} \implies r < \text{hnorm } x) \implies x \in \text{HInfinite}$
by (*simp add: HInfinite-def*)

lemma *HInfiniteD*: $x \in \text{HInfinite} \implies r \in \mathbb{R} \implies r < \text{hnorm } x$
by (*simp add: HInfinite-def*)

lemma *HInfinite-mult*:
fixes $x \ y :: 'a::\text{real-normed-div-algebra star}$
assumes $x \in \text{HInfinite } y \in \text{HInfinite}$ **shows** $x * y \in \text{HInfinite}$
proof (*rule HInfiniteI, simp only: hnorm-mult*)
have $x \neq 0$
using *Compl-HFinite HFinite-0 assms* **by** *blast*
show $r < \text{hnorm } x * \text{hnorm } y$
if $r \in \mathbb{R}$ **for** $r :: \text{real star}$
proof –
have $r = r * 1$
by *simp*
also have $\dots < \text{hnorm } x * \text{hnorm } y$
by (*meson HInfiniteD Reals-1 $\langle x \neq 0 \rangle$ assms le-numeral-extra(1) mult-strict-mono*
that zero-less-hnorm-iff)
finally show *?thesis* .
qed
qed

lemma *hypreal-add-zero-less-le-mono*: $r < x \implies 0 \leq y \implies r < x + y$
for $r \ x \ y :: \text{hypreal}$
by *simp*

lemma *HInfinite-add-ge-zero*: $x \in \text{HInfinite} \implies 0 \leq y \implies 0 \leq x \implies x + y \in \text{HInfinite}$
for $x \ y :: \text{hypreal}$
by (*auto simp: abs-if add commute HInfinite-def*)

lemma *HInfinite-add-ge-zero2*: $x \in \text{HInfinite} \implies 0 \leq y \implies 0 \leq x \implies y + x \in \text{HInfinite}$
for $x \ y :: \text{hypreal}$
by (*auto intro!: HInfinite-add-ge-zero simp add: add commute*)

lemma *HInfinite-add-gt-zero*: $x \in \text{HInfinite} \implies 0 < y \implies 0 < x \implies x + y \in \text{HInfinite}$
for $x \ y :: \text{hypreal}$
by (*blast intro: HInfinite-add-ge-zero order-less-imp-le*)

lemma *HInfinite-minus-iff*: $-x \in \text{HInfinite} \longleftrightarrow x \in \text{HInfinite}$

by (*simp add: HInfinite-def*)

lemma *HInfinite-add-le-zero*: $x \in \text{HInfinite} \implies y \leq 0 \implies x \leq 0 \implies x + y \in \text{HInfinite}$
for $x\ y :: \text{hypreal}$
by (*metis (no-types, lifting) HInfinite-add-ge-zero2 HInfinite-minus-iff add.inverse-distrib-swap neg-0-le-iff-le*)

lemma *HInfinite-add-lt-zero*: $x \in \text{HInfinite} \implies y < 0 \implies x < 0 \implies x + y \in \text{HInfinite}$
for $x\ y :: \text{hypreal}$
by (*blast intro: HInfinite-add-le-zero order-less-imp-le*)

lemma *not-Infinitesimal-not-zero*: $x \notin \text{Infinitesimal} \implies x \neq 0$
by *auto*

lemma *HFinite-diff-Infinitesimal-hrabs*:
 $x \in \text{HFinite} - \text{Infinitesimal} \implies |x| \in \text{HFinite} - \text{Infinitesimal}$
for $x :: \text{hypreal}$
by *blast*

lemma *hnorm-le-Infinitesimal*: $e \in \text{Infinitesimal} \implies \text{hnorm } x \leq e \implies x \in \text{Infinitesimal}$
by (*auto simp: Infinitesimal-def abs-less-iff*)

lemma *hnorm-less-Infinitesimal*: $e \in \text{Infinitesimal} \implies \text{hnorm } x < e \implies x \in \text{Infinitesimal}$
by (*erule hnorm-le-Infinitesimal, erule order-less-imp-le*)

lemma *hrabs-le-Infinitesimal*: $e \in \text{Infinitesimal} \implies |x| \leq e \implies x \in \text{Infinitesimal}$
for $x :: \text{hypreal}$
by (*erule hnorm-le-Infinitesimal*) *simp*

lemma *hrabs-less-Infinitesimal*: $e \in \text{Infinitesimal} \implies |x| < e \implies x \in \text{Infinitesimal}$
for $x :: \text{hypreal}$
by (*erule hnorm-less-Infinitesimal*) *simp*

lemma *Infinitesimal-interval*:
 $e \in \text{Infinitesimal} \implies e' \in \text{Infinitesimal} \implies e' < x \implies x < e \implies x \in \text{Infinitesimal}$
for $x :: \text{hypreal}$
by (*auto simp add: Infinitesimal-def abs-less-iff*)

lemma *Infinitesimal-interval2*:
 $e \in \text{Infinitesimal} \implies e' \in \text{Infinitesimal} \implies e' \leq x \implies x \leq e \implies x \in \text{Infinitesimal}$
for $x :: \text{hypreal}$
by (*auto intro: Infinitesimal-interval simp add: order-le-less*)

lemma *lemma-Infinitesimal-hyperpow*: $x \in \text{Infinitesimal} \implies 0 < N \implies |x \text{ pow } N| \leq |x|$

for $x :: \text{hypreal}$
apply (*clarsimp simp: Infinitesimal-def*)
by (*metis Reals-1 abs-ge-zero hyperpow-Suc-le-self2 hyperpow-hrabs hypnat-gt-zero-iff2 zero-less-one*)

lemma *Infinitesimal-hyperpow*: $x \in \text{Infinitesimal} \implies 0 < N \implies x \text{ pow } N \in \text{Infinitesimal}$
for $x :: \text{hypreal}$
using *hrabs-le-Infinitesimal lemma-Infinitesimal-hyperpow* **by** *blast*

lemma *hrealpow-hyperpow-Infinitesimal-iff*:
 $(x \wedge n \in \text{Infinitesimal}) \longleftrightarrow x \text{ pow } (\text{hypnat-of-nat } n) \in \text{Infinitesimal}$
by (*simp only: hyperpow-hypnat-of-nat*)

lemma *Infinitesimal-hrealpow*: $x \in \text{Infinitesimal} \implies 0 < n \implies x \wedge n \in \text{Infinitesimal}$
for $x :: \text{hypreal}$
by (*simp add: hrealpow-hyperpow-Infinitesimal-iff Infinitesimal-hyperpow*)

lemma *not-Infinitesimal-mult*:
 $x \notin \text{Infinitesimal} \implies y \notin \text{Infinitesimal} \implies x * y \notin \text{Infinitesimal}$
for $x y :: 'a::\text{real-normed-div-algebra star}$
by (*metis (no-types, lifting) inverse-Infinitesimal-iff-HInfinite ComplI Compl-HFinite Infinitesimal-HFinite-mult divide-inverse eq-divide-imp inverse-inverse-eq mult-zero-right*)

lemma *Infinitesimal-mult-disj*: $x * y \in \text{Infinitesimal} \implies x \in \text{Infinitesimal} \vee y \in \text{Infinitesimal}$
for $x y :: 'a::\text{real-normed-div-algebra star}$
using *not-Infinitesimal-mult* **by** *blast*

lemma *HFinite-Infinitesimal-not-zero*: $x \in \text{HFinite} - \text{Infinitesimal} \implies x \neq 0$
by *blast*

lemma *HFinite-Infinitesimal-diff-mult*:
 $x \in \text{HFinite} - \text{Infinitesimal} \implies y \in \text{HFinite} - \text{Infinitesimal} \implies x * y \in \text{HFinite} - \text{Infinitesimal}$
for $x y :: 'a::\text{real-normed-div-algebra star}$
by (*simp add: HFinite-mult not-Infinitesimal-mult*)

lemma *Infinitesimal-subset-HFinite*: $\text{Infinitesimal} \subseteq \text{HFinite}$
using *HFinite-def InfinitesimalD Reals-1 zero-less-one* **by** *blast*

lemma *Infinitesimal-star-of-mult*: $x \in \text{Infinitesimal} \implies x * \text{star-of } r \in \text{Infinitesimal}$
for $x :: 'a::\text{real-normed-algebra star}$
by (*erule HFinite-star-of [THEN [2] Infinitesimal-HFinite-mult]*)

lemma *Infinitesimal-star-of-mult2*: $x \in \text{Infinitesimal} \implies \text{star-of } r * x \in \text{Infinitesimal}$


```

for x :: 'a::real-normed-algebra star
by (erule HFinite-star-of [THEN [2] Infinitesimal-HFinite-mult2])

```

5.5 The Infinitely Close Relation

```

lemma mem-infmal-iff:  $x \in \text{Infinitesimal} \longleftrightarrow x \approx 0$ 
by (simp add: Infinitesimal-def approx-def)

```

```

lemma approx-minus-iff:  $x \approx y \longleftrightarrow x - y \approx 0$ 
by (simp add: approx-def)

```

```

lemma approx-minus-iff2:  $x \approx y \longleftrightarrow -y + x \approx 0$ 
by (simp add: approx-def add.commute)

```

```

lemma approx-refl [iff]:  $x \approx x$ 
by (simp add: approx-def Infinitesimal-def)

```

```

lemma approx-sym:  $x \approx y \implies y \approx x$ 
by (metis Infinitesimal-minus-iff approx-def minus-diff-eq)

```

```

lemma approx-trans:
  assumes  $x \approx y$   $y \approx z$  shows  $x \approx z$ 
proof -
  have  $x - y \in \text{Infinitesimal}$   $z - y \in \text{Infinitesimal}$ 
  using assms approx-def approx-sym by auto
  then have  $x - z \in \text{Infinitesimal}$ 
  using Infinitesimal-diff by force
  then show ?thesis
  by (simp add: approx-def)
qed

```

```

lemma approx-trans2:  $r \approx x \implies s \approx x \implies r \approx s$ 
by (blast intro: approx-sym approx-trans)

```

```

lemma approx-trans3:  $x \approx r \implies x \approx s \implies r \approx s$ 
by (blast intro: approx-sym approx-trans)

```

```

lemma approx-reorient:  $x \approx y \longleftrightarrow y \approx x$ 
by (blast intro: approx-sym)

```

Reorientation simplification procedure: reorients (polymorphic) $0 = x$, $1 = x$, $nnn = x$ provided x isn't 0 , 1 or a numeral.

```

simproc-setup approx-reorient-simproc
  ( $0 \approx x \mid 1 \approx y \mid \text{numeral } w \approx z \mid -1 \approx y \mid -\text{numeral } w \approx r$ ) =
  <
    let val rule = @{thm approx-reorient} RS eq-reflection
    fun proc ct =
      case Thm.term-of ct of
        - $ t $ u => if can HOLogic.dest-number u then NONE

```

```

      else if can HOLogic.dest-number t then SOME rule else NONE
    | - => NONE
  in K (K proc) end
>

```

lemma *Infinesimal-approx-minus*: $x - y \in \text{Infinesimal} \longleftrightarrow x \approx y$
by (*simp add: approx-minus-iff [symmetric] mem-infmal-iff*)

lemma *approx-monad-iff*: $x \approx y \longleftrightarrow \text{monad } x = \text{monad } y$
apply (*simp add: monad-def set-eq-iff*)
using *approx-reorient approx-trans* **by** *blast*

lemma *Infinesimal-approx*: $x \in \text{Infinesimal} \implies y \in \text{Infinesimal} \implies x \approx y$
by (*simp add: Infinesimal-diff approx-def*)

lemma *approx-add*: $a \approx b \implies c \approx d \implies a + c \approx b + d$
proof (*unfold approx-def*)
assume *inf*: $a - b \in \text{Infinesimal}$ $c - d \in \text{Infinesimal}$
have $a + c - (b + d) = (a - b) + (c - d)$ **by** *simp*
also have $\dots \in \text{Infinesimal}$
using *inf* **by** (*rule Infinesimal-add*)
finally show $a + c - (b + d) \in \text{Infinesimal}$.
qed

lemma *approx-minus*: $a \approx b \implies -a \approx -b$
by (*metis approx-def approx-sym minus-diff-eq minus-diff-minus*)

lemma *approx-minus2*: $-a \approx -b \implies a \approx b$
by (*auto dest: approx-minus*)

lemma *approx-minus-cancel [simp]*: $-a \approx -b \longleftrightarrow a \approx b$
by (*blast intro: approx-minus approx-minus2*)

lemma *approx-add-minus*: $a \approx b \implies c \approx d \implies a + -c \approx b + -d$
by (*blast intro!: approx-add approx-minus*)

lemma *approx-diff*: $a \approx b \implies c \approx d \implies a - c \approx b - d$
using *approx-add [of a b - c - d]* **by** *simp*

lemma *approx-mult1*: $a \approx b \implies c \in \text{HFinite} \implies a * c \approx b * c$
for $a \ b \ c :: 'a::\text{real-normed-algebra}$ *star*
by (*simp add: approx-def Infinesimal-HFinite-mult left-diff-distrib [symmetric]*)

lemma *approx-mult2*: $a \approx b \implies c \in \text{HFinite} \implies c * a \approx c * b$
for $a \ b \ c :: 'a::\text{real-normed-algebra}$ *star*
by (*simp add: approx-def Infinesimal-HFinite-mult2 right-diff-distrib [symmetric]*)

lemma *approx-mult-subst*: $u \approx v * x \implies x \approx y \implies v \in \text{HFinite} \implies u \approx v * y$
for $u \ v \ x \ y :: 'a::\text{real-normed-algebra}$ *star*

by (blast intro: approx-mult2 approx-trans)

lemma approx-mult-subst2: $u \approx x * v \implies x \approx y \implies v \in HFinite \implies u \approx y * v$
 for $u\ v\ x\ y :: 'a::real-normed-algebra\ star$
 by (blast intro: approx-mult1 approx-trans)

lemma approx-mult-subst-star-of: $u \approx x * star-of\ v \implies x \approx y \implies u \approx y * star-of\ v$
 for $u\ x\ y :: 'a::real-normed-algebra\ star$
 by (auto intro: approx-mult-subst2)

lemma approx-eq-imp: $a = b \implies a \approx b$
 by (simp add: approx-def)

lemma Infinitesimal-minus-approx: $x \in Infinitesimal \implies -x \approx x$
 by (blast intro: Infinitesimal-minus-iff [THEN iffD2] mem-infmal-iff [THEN iffD1] approx-trans2)

lemma bex-Infinitesimal-iff: $(\exists y \in Infinitesimal. x - z = y) \longleftrightarrow x \approx z$
 by (simp add: approx-def)

lemma bex-Infinitesimal-iff2: $(\exists y \in Infinitesimal. x = z + y) \longleftrightarrow x \approx z$
 by (force simp add: bex-Infinitesimal-iff [symmetric])

lemma Infinitesimal-add-approx: $y \in Infinitesimal \implies x + y = z \implies x \approx z$
 using approx-sym bex-Infinitesimal-iff2 by blast

lemma Infinitesimal-add-approx-self: $y \in Infinitesimal \implies x \approx x + y$
 by (simp add: Infinitesimal-add-approx)

lemma Infinitesimal-add-approx-self2: $y \in Infinitesimal \implies x \approx y + x$
 by (auto dest: Infinitesimal-add-approx-self simp add: add.commute)

lemma Infinitesimal-add-minus-approx-self: $y \in Infinitesimal \implies x \approx x + -y$
 by (blast intro!: Infinitesimal-add-approx-self Infinitesimal-minus-iff [THEN iffD2])

lemma Infinitesimal-add-cancel: $y \in Infinitesimal \implies x + y \approx z \implies x \approx z$
 using Infinitesimal-add-approx approx-trans by blast

lemma Infinitesimal-add-right-cancel: $y \in Infinitesimal \implies x \approx z + y \implies x \approx z$
 by (metis Infinitesimal-add-approx-self approx-mono-iff)

lemma approx-add-left-cancel: $d + b \approx d + c \implies b \approx c$
 by (metis add-diff-cancel-left bex-Infinitesimal-iff)

lemma approx-add-right-cancel: $b + d \approx c + d \implies b \approx c$
 by (simp add: approx-def)

lemma approx-add-mono1: $b \approx c \implies d + b \approx d + c$

```

by (simp add: approx-add)

lemma approx-add-mono2:  $b \approx c \implies b + a \approx c + a$ 
by (simp add: add commute approx-add-mono1)

lemma approx-add-left-iff [simp]:  $a + b \approx a + c \iff b \approx c$ 
by (fast elim: approx-add-left-cancel approx-add-mono1)

lemma approx-add-right-iff [simp]:  $b + a \approx c + a \iff b \approx c$ 
by (simp add: add commute)

lemma approx-HFinite:  $x \in \text{HFinite} \implies x \approx y \implies y \in \text{HFinite}$ 
by (metis HFinite-add Infinitesimal-subset-HFinite approx-sym subsetD beX-Infinitesimal-iff2)

lemma approx-star-of-HFinite:  $x \approx \text{star-of } D \implies x \in \text{HFinite}$ 
by (rule approx-sym [THEN [2] approx-HFinite], auto)

lemma approx-mult-HFinite:  $a \approx b \implies c \approx d \implies b \in \text{HFinite} \implies d \in \text{HFinite}$ 
 $\implies a * c \approx b * d$ 
for  $a \ b \ c \ d :: 'a::\text{real-normed-algebra star}$ 
by (meson approx-HFinite approx-mult2 approx-mult-subst2 approx-sym)

lemma scaleHR-left-diff-distrib:  $\bigwedge a \ b \ x. \text{scaleHR } (a - b) \ x = \text{scaleHR } a \ x - \text{scaleHR } b \ x$ 
by transfer (rule scaleR-left-diff-distrib)

lemma approx-scaleR1:  $a \approx \text{star-of } b \implies c \in \text{HFinite} \implies \text{scaleHR } a \ c \approx b *_R c$ 
unfolding approx-def
by (metis Infinitesimal-HFinite-scaleHR scaleHR-def scaleHR-left-diff-distrib star-scaleR-def starfun2-star-of)

lemma approx-scaleR2:  $a \approx b \implies c *_R a \approx c *_R b$ 
by (simp add: approx-def Infinitesimal-scaleR2 scaleR-right-diff-distrib [symmetric])

lemma approx-scaleR-HFinite:  $a \approx \text{star-of } b \implies c \approx d \implies d \in \text{HFinite} \implies \text{scaleHR } a \ c \approx b *_R d$ 
by (meson approx-HFinite approx-scaleR1 approx-scaleR2 approx-sym approx-trans)

lemma approx-mult-star-of:  $a \approx \text{star-of } b \implies c \approx \text{star-of } d \implies a * c \approx \text{star-of } b * \text{star-of } d$ 
for  $a \ c :: 'a::\text{real-normed-algebra star}$ 
by (blast intro!: approx-mult-HFinite approx-star-of-HFinite HFinite-star-of)

lemma approx-SReal-mult-cancel-zero:
fixes  $a \ x :: \text{hypreal}$ 
assumes  $a \in \mathbb{R} \ a \neq 0 \ a * x \approx 0$  shows  $x \approx 0$ 
proof –
have  $\text{inverse } a \in \text{HFinite}$ 
using Reals-inverse SReal-subset-HFinite assms(1) by blast

```

then show *?thesis*
using *assms* **by** (*auto dest: approx-mult2 simp add: mult.assoc [symmetric]*)
qed

lemma *approx-mult-SReal1*: $a \in \mathbb{R} \implies x \approx 0 \implies x * a \approx 0$
for $a \, x :: \text{hypreal}$
by (*auto dest: SReal-subset-HFinite [THEN subsetD] approx-mult1*)

lemma *approx-mult-SReal2*: $a \in \mathbb{R} \implies x \approx 0 \implies a * x \approx 0$
for $a \, x :: \text{hypreal}$
by (*auto dest: SReal-subset-HFinite [THEN subsetD] approx-mult2*)

lemma *approx-mult-SReal-zero-cancel-iff* [*simp*]: $a \in \mathbb{R} \implies a \neq 0 \implies a * x \approx 0 \iff x \approx 0$
for $a \, x :: \text{hypreal}$
by (*blast intro: approx-SReal-mult-cancel-zero approx-mult-SReal2*)

lemma *approx-SReal-mult-cancel*:
fixes $a \, w \, z :: \text{hypreal}$
assumes $a \in \mathbb{R} \, a \neq 0 \, a * w \approx a * z$ **shows** $w \approx z$
proof –
have *inverse* $a \in \text{HFinite}$
using *Reals-inverse SReal-subset-HFinite assms(1)* **by** *blast*
then show *?thesis*
using *assms* **by** (*auto dest: approx-mult2 simp add: mult.assoc [symmetric]*)
qed

lemma *approx-SReal-mult-cancel-iff1* [*simp*]: $a \in \mathbb{R} \implies a \neq 0 \implies a * w \approx a * z \iff w \approx z$
for $a \, w \, z :: \text{hypreal}$
by (*meson SReal-subset-HFinite approx-SReal-mult-cancel approx-mult2 subsetD*)

lemma *approx-le-bound*:
fixes $z :: \text{hypreal}$
assumes $z \leq f \, f \approx g \, g \leq z$ **shows** $f \approx z$
proof –
obtain y **where** $z \leq g + y$ **and** $y \in \text{Infinitesimal}$ $f = g + y$
using *assms bex-Infinitesimal-iff2* **by** *auto*
then have $z - g \in \text{Infinitesimal}$
using *assms(3) hrabs-le-Infinitesimal* **by** *auto*
then show *?thesis*
by (*metis approx-def approx-trans2 assms(2)*)
qed

lemma *approx-hnorm*: $x \approx y \implies \text{hnorm } x \approx \text{hnorm } y$
for $x \, y :: 'a::\text{real-normed-vector star}$
proof (*unfold approx-def*)
assume $x - y \in \text{Infinitesimal}$
then have $\text{hnorm } (x - y) \in \text{Infinitesimal}$

```

  by (simp only: Infinitesimal-hnorm-iff)
moreover have  $(0::\text{real star}) \in \text{Infinitesimal}$ 
  by (rule Infinitesimal-zero)
moreover have  $0 \leq |\text{hnorm } x - \text{hnorm } y|$ 
  by (rule abs-ge-zero)
moreover have  $|\text{hnorm } x - \text{hnorm } y| \leq \text{hnorm } (x - y)$ 
  by (rule hnorm-triangle-ineq3)
ultimately have  $|\text{hnorm } x - \text{hnorm } y| \in \text{Infinitesimal}$ 
  by (rule Infinitesimal-interval2)
then show  $\text{hnorm } x - \text{hnorm } y \in \text{Infinitesimal}$ 
  by (simp only: Infinitesimal-hrabs-iff)
qed

```

5.6 Zero is the Only Infinitesimal that is also a Real

```

lemma Infinitesimal-less-SReal:  $x \in \mathbb{R} \implies y \in \text{Infinitesimal} \implies 0 < x \implies y < x$ 
  for  $x \ y :: \text{hypreal}$ 
  using InfinitesimalD by fastforce

```

```

lemma Infinitesimal-less-SReal2:  $y \in \text{Infinitesimal} \implies \forall r \in \text{Reals}. 0 < r \longrightarrow y < r$ 
  for  $y :: \text{hypreal}$ 
  by (blast intro: Infinitesimal-less-SReal)

```

```

lemma SReal-not-Infinitesimal:  $0 < y \implies y \in \mathbb{R} \implies y \notin \text{Infinitesimal}$ 
  for  $y :: \text{hypreal}$ 
  by (auto simp add: Infinitesimal-def abs-if)

```

```

lemma SReal-minus-not-Infinitesimal:  $y < 0 \implies y \in \mathbb{R} \implies y \notin \text{Infinitesimal}$ 
  for  $y :: \text{hypreal}$ 
  using Infinitesimal-minus-iff Reals-minus SReal-not-Infinitesimal neg-0-less-iff-less
  by blast

```

```

lemma SReal-Int-Infinitesimal-zero:  $\mathbb{R} \cap \text{Infinitesimal} = \{0::\text{hypreal}\}$ 
  proof -
    have  $x = 0$  if  $x \in \mathbb{R} \ x \in \text{Infinitesimal}$  for  $x :: \text{real star}$ 
      using that SReal-minus-not-Infinitesimal SReal-not-Infinitesimal not-less-iff-gr-or-eq
    by blast
    then show ?thesis
      by auto
  qed

```

```

lemma SReal-Infinitesimal-zero:  $x \in \mathbb{R} \implies x \in \text{Infinitesimal} \implies x = 0$ 
  for  $x :: \text{hypreal}$ 
  using SReal-Int-Infinitesimal-zero by blast

```

```

lemma SReal-HFinite-diff-Infinitesimal:  $x \in \mathbb{R} \implies x \neq 0 \implies x \in \text{HFinite} - \text{Infinitesimal}$ 

```

for $x :: \text{hypreal}$
by (*auto dest: SReal-Infinesimal-zero SReal-subset-HFinite [THEN subsetD]*)

lemma *hypreal-of-real-HFinite-diff-Infinesimal*:
 $\text{hypreal-of-real } x \neq 0 \implies \text{hypreal-of-real } x \in \text{HFinite} - \text{Infinesimal}$
by (*rule SReal-HFinite-diff-Infinesimal*) **auto**

lemma *star-of-Infinesimal-iff-0 [iff]*: $\text{star-of } x \in \text{Infinesimal} \longleftrightarrow x = 0$

proof

show $x = 0$ **if** $\text{star-of } x \in \text{Infinesimal}$

proof –

have $\text{hnorm } (\text{star-n } (\lambda n. x)) \in \text{Standard}$

by (*metis Reals-eq-Standard SReal-iff star-of-def star-of-norm*)

then show *?thesis*

by (*metis InfinesimalD2 less-irrefl star-of-norm that zero-less-norm-iff*)

qed

qed *auto*

lemma *star-of-HFinite-diff-Infinesimal*: $x \neq 0 \implies \text{star-of } x \in \text{HFinite} - \text{Infinesimal}$
by *simp*

lemma *numeral-not-Infinesimal [simp]*:
 $\text{numeral } w \neq (0 :: \text{hypreal}) \implies (\text{numeral } w :: \text{hypreal}) \notin \text{Infinesimal}$
by (*fast dest: Reals-numeral [THEN SReal-Infinesimal-zero]*)

Again: 1 is a special case, but not 0 this time.

lemma *one-not-Infinesimal [simp]*:
 $(1 :: 'a :: \{\text{real-normed-vector}, \text{zero-neq-one}\} \text{star}) \notin \text{Infinesimal}$
by (*metis star-of-Infinesimal-iff-0 star-one-def zero-neq-one*)

lemma *approx-SReal-not-zero*: $y \in \mathbb{R} \implies x \approx y \implies y \neq 0 \implies x \neq 0$
for $x y :: \text{hypreal}$
using *SReal-Infinesimal-zero approx-sym mem-infmal-iff* **by** *auto*

lemma *HFinite-diff-Infinesimal-approx*:
 $x \approx y \implies y \in \text{HFinite} - \text{Infinesimal} \implies x \in \text{HFinite} - \text{Infinesimal}$
by (*meson Diff-iff approx-HFinite approx-sym approx-trans3 mem-infmal-iff*)

The premise $y \neq 0$ is essential; otherwise $x / y = 0$ and we lose the *HFinite* premise.

lemma *Infinesimal-ratio*:
 $y \neq 0 \implies y \in \text{Infinesimal} \implies x / y \in \text{HFinite} \implies x \in \text{Infinesimal}$
for $x y :: 'a :: \{\text{real-normed-div-algebra}, \text{field}\} \text{star}$
using *Infinesimal-HFinite-mult* **by** *fastforce*

lemma *Infinesimal-SReal-divide*: $x \in \text{Infinesimal} \implies y \in \mathbb{R} \implies x / y \in \text{Infinesimal}$
for $x y :: \text{hypreal}$

by (metis *HFinite-star-of Infinitesimal-HFinite-mult Reals-inverse SReal-iff divide-inverse*)

6 Standard Part Theorem

Every finite $x \in R^*$ is infinitely close to a unique real number (i.e. a member of *Reals*).

6.1 Uniqueness: Two Infinitely Close Reals are Equal

lemma *star-of-approx-iff* [simp]: $\text{star-of } x \approx \text{star-of } y \longleftrightarrow x = y$

by (metis *approx-def right-minus-eq star-of-Infinitesimal-iff-0 star-of-simps(2)*)

lemma *SReal-approx-iff*: $x \in \mathbb{R} \implies y \in \mathbb{R} \implies x \approx y \longleftrightarrow x = y$

for $x \ y :: \text{hypreal}$

by (meson *Reals-diff SReal-Infinitesimal-zero approx-def approx-refl right-minus-eq*)

lemma *numeral-approx-iff* [simp]:

$(\text{numeral } v \approx (\text{numeral } w :: 'a::\{\text{numeral}, \text{real-normed-vector}\} \text{ star})) = (\text{numeral } v = (\text{numeral } w :: 'a))$

by (metis *star-of-approx-iff star-of-numeral*)

And also for $0 \approx \#nn$ and $1 \approx \#nn$, $\#nn \approx 0$ and $\#nn \approx 1$.

lemma [simp]:

$(\text{numeral } w \approx (0::'a::\{\text{numeral}, \text{real-normed-vector}\} \text{ star})) = (\text{numeral } w = (0::'a))$

$((0::'a::\{\text{numeral}, \text{real-normed-vector}\} \text{ star}) \approx \text{numeral } w) = (\text{numeral } w = (0::'a))$

$(\text{numeral } w \approx (1::'b::\{\text{numeral}, \text{one}, \text{real-normed-vector}\} \text{ star})) = (\text{numeral } w = (1::'b))$

$((1::'b::\{\text{numeral}, \text{one}, \text{real-normed-vector}\} \text{ star}) \approx \text{numeral } w) = (\text{numeral } w = (1::'b))$

$\neg (0 \approx (1::'c::\{\text{zero-neq-one}, \text{real-normed-vector}\} \text{ star}))$

$\neg (1 \approx (0::'c::\{\text{zero-neq-one}, \text{real-normed-vector}\} \text{ star}))$

unfolding *star-numeral-def star-zero-def star-one-def star-of-approx-iff*

by (auto intro: sym)

lemma *star-of-approx-numeral-iff* [simp]: $\text{star-of } k \approx \text{numeral } w \longleftrightarrow k = \text{numeral } w$

by (subst *star-of-approx-iff* [symmetric]) auto

lemma *star-of-approx-zero-iff* [simp]: $\text{star-of } k \approx 0 \longleftrightarrow k = 0$

by (simp-all add: *star-of-approx-iff* [symmetric])

lemma *star-of-approx-one-iff* [simp]: $\text{star-of } k \approx 1 \longleftrightarrow k = 1$

by (simp-all add: *star-of-approx-iff* [symmetric])

lemma *approx-unique-real*: $r \in \mathbb{R} \implies s \in \mathbb{R} \implies r \approx x \implies s \approx x \implies r = s$

for $r \ s :: \text{hypreal}$

by (blast intro: *SReal-approx-iff* [THEN *iffD1*] *approx-trans2*)

6.2 Existence of Unique Real Infinitely Close

6.2.1 Lifting of the Ub and Lub Properties

lemma *hypreal-of-real-isUb-iff*: $isUb \mathbb{R} (hypreal-of-real \text{ ‘ } Q) (hypreal-of-real Y) = isUb UNIV Q Y$

for $Q :: real \text{ set}$ **and** $Y :: real$
by (*simp add: isUb-def settle-def*)

lemma *hypreal-of-real-isLub-iff*:

$isLub \mathbb{R} (hypreal-of-real \text{ ‘ } Q) (hypreal-of-real Y) = isLub (UNIV :: real \text{ set}) Q Y$
(is ?lhs = ?rhs)

for $Q :: real \text{ set}$ **and** $Y :: real$

proof

assume *?lhs*

then show *?rhs*

by (*simp add: isLub-def leastP-def*) (*metis hypreal-of-real-isUb-iff mem-Collect-eq setge-def star-of-le*)

next

assume *?rhs*

then show *?lhs*

apply (*simp add: isLub-def leastP-def hypreal-of-real-isUb-iff setge-def*)

by (*metis SReal-iff hypreal-of-real-isUb-iff isUb-def star-of-le*)

qed

lemma *lemma-isUb-hypreal-of-real*: $isUb \mathbb{R} P Y \implies \exists Yo. isUb \mathbb{R} P (hypreal-of-real Yo)$

by (*auto simp add: SReal-iff isUb-def*)

lemma *lemma-isLub-hypreal-of-real*: $isLub \mathbb{R} P Y \implies \exists Yo. isLub \mathbb{R} P (hypreal-of-real Yo)$

by (*auto simp add: isLub-def leastP-def isUb-def SReal-iff*)

lemma *SReal-complete*:

fixes $P :: hypreal \text{ set}$

assumes $isUb \mathbb{R} P Y P \subseteq \mathbb{R} P \neq \{\}$

shows $\exists t. isLub \mathbb{R} P t$

proof –

obtain Q **where** $P = hypreal-of-real \text{ ‘ } Q$

by (*metis $\langle P \subseteq \mathbb{R} \rangle hypreal-of-real-image subset-imageE$*)

then show *?thesis*

by (*metis assms(1) $\langle P \neq \{\} \rangle equals0I hypreal-of-real-isLub-iff hypreal-of-real-isUb-iff image-empty lemma-isUb-hypreal-of-real reals-complete$*)

qed

Lemmas about lubs.

lemma *lemma-st-part-lub*:

fixes $x :: hypreal$

assumes $x \in HFinite$

shows $\exists t. isLub \mathbb{R} \{s. s \in \mathbb{R} \wedge s < x\} t$

proof –

obtain t **where** $t: t \in \mathbb{R} \text{ hnorm } x < t$
using $HFiniteD$ $assms$ **by** $blast$
then have $isUb \mathbb{R} \{s. s \in \mathbb{R} \wedge s < x\} t$
by ($simp$ $add: abs-less-iff$ $isUbI$ $le-less-linear$ $less-imp-not-less$ $settleI$)
moreover have $\exists y. y \in \mathbb{R} \wedge y < x$
using t **by** ($rule-tac$ $x = -t$ **in** exI) ($auto$ $simp$ $add: abs-less-iff$)
ultimately show $?thesis$
using $SReal-complete$ **by** $fastforce$
qed

lemma $hypreal-settle-less-trans$: $S * \leq x \implies x < y \implies S * \leq y$
for $x y :: hypreal$
by ($meson$ $le-less-trans$ $less-imp-le$ $settle-def$)

lemma $hypreal-gt-isUb$: $isUb R S x \implies x < y \implies y \in R \implies isUb R S y$
for $x y :: hypreal$
using $hypreal-settle-less-trans$ $isUb-def$ **by** $blast$

lemma $lemma-SReal-ub$: $x \in \mathbb{R} \implies isUb \mathbb{R} \{s. s \in \mathbb{R} \wedge s < x\} x$
for $x :: hypreal$
by ($auto$ $intro: isUbI$ $settleI$ $order-less-imp-le$)

lemma $lemma-SReal-lub$:
fixes $x :: hypreal$
assumes $x \in \mathbb{R}$ **shows** $isLub \mathbb{R} \{s. s \in \mathbb{R} \wedge s < x\} x$

proof –

have $x \leq y$ **if** $isUb \mathbb{R} \{s \in \mathbb{R}. s < x\} y$ **for** y

proof –

have $y \in \mathbb{R}$

using $isUbD2a$ **that** **by** $blast$

show $?thesis$

proof ($cases$ $x y$ $rule: linorder-cases$)

case $greater$

then obtain r **where** $y < r$ $r < x$

using $dense$ **by** $blast$

then show $?thesis$

using $isUbD$ [OF $that$]

by $simp$ ($meson$ $SReal-dense$ $\langle y \in \mathbb{R} \rangle$ $assms$ $greater$ $not-le$)

qed $auto$

qed

with $assms$ **show** $?thesis$

by ($simp$ $add: isLubI2$ $isUbI$ $setgeI$ $settleI$)

qed

lemma $lemma-st-part-major$:

fixes $x r t :: hypreal$

assumes $x: x \in HFinite$ **and** $r: r \in \mathbb{R}$ $0 < r$ **and** $t: isLub \mathbb{R} \{s. s \in \mathbb{R} \wedge s < x\} t$

```

shows  $|x - t| < r$ 
proof -
  have  $t \in \mathbb{R}$ 
    using isLubD1a t by blast
  have lemma-st-part-gt-ub:  $x < r \implies r \in \mathbb{R} \implies \text{isUb } \mathbb{R} \{s. s \in \mathbb{R} \wedge s < x\} r$ 
    for  $r :: \text{hypreal}$ 
    by (auto dest: order-less-trans intro: order-less-imp-le intro!: isUbI settleI)

  have isUb  $\mathbb{R} \{s \in \mathbb{R}. s < x\} t$ 
    by (simp add: t isLub-isUb)
  then have  $\neg r + t < x$ 
    by (metis (mono-tags, lifting) Reals-add <t ∈ ℝ> add-le-same-cancel2 isUbD leD mem-Collect-eq r)
  then have  $x - t \leq r$ 
    by simp
  moreover have  $\neg x < t - r$ 
    using lemma-st-part-gt-ub isLub-le-isUb <t ∈ ℝ> r t x by fastforce
  then have  $-(x - t) \leq r$ 
    by linarith
  moreover have False if  $x - t = r \vee -(x - t) = r$ 
  proof -
    have  $x \in \mathbb{R}$ 
      by (metis <t ∈ ℝ> <r ∈ ℝ> that Reals-add-cancel Reals-minus-iff add-uminus-conv-diff)
    then have isLub  $\mathbb{R} \{s \in \mathbb{R}. s < x\} x$ 
      by (rule lemma-SReal-lub)
    then show False
      using r t that x isLub-unique by force
  qed
  ultimately show ?thesis
    using abs-less-iff dual-order.order-iff-strict by blast
qed

```

lemma *lemma-st-part-major2*:

```

 $x \in \text{HFinite} \implies \text{isLub } \mathbb{R} \{s. s \in \mathbb{R} \wedge s < x\} t \implies \forall r \in \text{Reals}. 0 < r \longrightarrow |x - t| < r$ 
  for  $x t :: \text{hypreal}$ 
  by (blast dest!: lemma-st-part-major)

```

Existence of real and Standard Part Theorem.

```

lemma lemma-st-part-Ex:  $x \in \text{HFinite} \implies \exists t \in \text{Reals}. \forall r \in \text{Reals}. 0 < r \longrightarrow |x - t| < r$ 
  for  $x :: \text{hypreal}$ 
  by (meson isLubD1a lemma-st-part-lub lemma-st-part-major2)

```

lemma *st-part-Ex*: $x \in \text{HFinite} \implies \exists t \in \text{Reals}. x \approx t$

```

  for  $x :: \text{hypreal}$ 
  by (metis InfinitesimalI approx-def hypreal-hnorm-def lemma-st-part-Ex)

```

There is a unique real infinitely close.

lemma *st-part-Ex1*: $x \in \text{HFinite} \implies \exists !t::\text{hypreal}. t \in \mathbb{R} \wedge x \approx t$
by (*meson SReal-approx-iff approx-trans2 st-part-Ex*)

6.3 Finite, Infinite and Infinitesimal

lemma *HFinite-Int-HInfinite-empty* [*simp*]: $\text{HFinite Int HInfinite} = \{\}$
using *Compl-HFinite* **by** *blast*

lemma *HFinite-not-HInfinite*:
assumes $x: x \in \text{HFinite}$ **shows** $x \notin \text{HInfinite}$
using *Compl-HFinite* x **by** *blast*

lemma *not-HFinite-HInfinite*: $x \notin \text{HFinite} \implies x \in \text{HInfinite}$
using *Compl-HFinite* **by** *blast*

lemma *HInfinite-HFinite-disj*: $x \in \text{HInfinite} \vee x \in \text{HFinite}$
by (*blast intro: not-HFinite-HInfinite*)

lemma *HInfinite-HFinite-iff*: $x \in \text{HInfinite} \longleftrightarrow x \notin \text{HFinite}$
by (*blast dest: HFinite-not-HInfinite not-HFinite-HInfinite*)

lemma *HFinite-HInfinite-iff*: $x \in \text{HFinite} \longleftrightarrow x \notin \text{HInfinite}$
by (*simp add: HInfinite-HFinite-iff*)

lemma *HInfinite-diff-HFinite-Infinitesimal-disj*:
 $x \notin \text{Infinitesimal} \implies x \in \text{HInfinite} \vee x \in \text{HFinite} - \text{Infinitesimal}$
by (*fast intro: not-HFinite-HInfinite*)

lemma *HFinite-inverse*: $x \in \text{HFinite} \implies x \notin \text{Infinitesimal} \implies \text{inverse } x \in \text{HFinite}$
for $x :: 'a::\text{real-normed-div-algebra}$ **star**
using *HInfinite-inverse-Infinitesimal not-HFinite-HInfinite* **by** *force*

lemma *HFinite-inverse2*: $x \in \text{HFinite} - \text{Infinitesimal} \implies \text{inverse } x \in \text{HFinite}$
for $x :: 'a::\text{real-normed-div-algebra}$ **star**
by (*blast intro: HFinite-inverse*)

Stronger statement possible in fact.

lemma *Infinitesimal-inverse-HFinite*: $x \notin \text{Infinitesimal} \implies \text{inverse } x \in \text{HFinite}$
for $x :: 'a::\text{real-normed-div-algebra}$ **star**
using *HFinite-HInfinite-iff HInfinite-inverse-Infinitesimal* **by** *fastforce*

lemma *HFinite-not-Infinitesimal-inverse*:
 $x \in \text{HFinite} - \text{Infinitesimal} \implies \text{inverse } x \in \text{HFinite} - \text{Infinitesimal}$
for $x :: 'a::\text{real-normed-div-algebra}$ **star**
using *HFinite-Infinitesimal-not-zero HFinite-inverse2 Infinitesimal-HFinite-mult2*
by *fastforce*

lemma *approx-inverse*:

```

fixes  $x\ y :: 'a::\text{real-normed-div-algebra star}$ 
assumes  $x \approx y$  and  $y: y \in \text{HFinite} - \text{Infinitesimal}$  shows  $\text{inverse } x \approx \text{inverse } y$ 
proof –
  have  $x: x \in \text{HFinite} - \text{Infinitesimal}$ 
    using  $\text{HFinite-diff-Infinitesimal-approx assms}(1)$   $y$  by blast
  with  $y$   $\text{HFinite-inverse2}$  have  $\text{inverse } x \in \text{HFinite}$   $\text{inverse } y \in \text{HFinite}$ 
    by blast+
  then have  $\text{inverse } y * x \approx 1$ 
    by (metis Diff-iff approx-mult2 assms(1) left-inverse not-Infinitesimal-not-zero
 $y$ )
  then show ?thesis
    by (metis (no-types, lifting) DiffD2 HFinite-Infinitesimal-not-zero Infinitesimal-mult-disj x approx-def approx-sym left-diff-distrib left-inverse)
qed

```

```

lemmas  $\text{star-of-approx-inverse} = \text{star-of-HFinite-diff-Infinitesimal} \ [ \text{THEN } [2] \ \text{approx-inverse} ]$ 
lemmas  $\text{hypreal-of-real-approx-inverse} = \text{hypreal-of-real-HFinite-diff-Infinitesimal} \ [ \text{THEN } [2] \ \text{approx-inverse} ]$ 

```

```

lemma  $\text{inverse-add-Infinitesimal-approx}$ :
 $x \in \text{HFinite} - \text{Infinitesimal} \implies h \in \text{Infinitesimal} \implies \text{inverse } (x + h) \approx \text{inverse } x$ 
for  $x\ h :: 'a::\text{real-normed-div-algebra star}$ 
by (auto intro: approx-inverse approx-sym Infinitesimal-add-approx-self)

```

```

lemma  $\text{inverse-add-Infinitesimal-approx2}$ :
 $x \in \text{HFinite} - \text{Infinitesimal} \implies h \in \text{Infinitesimal} \implies \text{inverse } (h + x) \approx \text{inverse } x$ 
for  $x\ h :: 'a::\text{real-normed-div-algebra star}$ 
by (metis add.commute inverse-add-Infinitesimal-approx)

```

```

lemma  $\text{inverse-add-Infinitesimal-approx-Infinitesimal}$ :
 $x \in \text{HFinite} - \text{Infinitesimal} \implies h \in \text{Infinitesimal} \implies \text{inverse } (x + h) - \text{inverse } x \approx h$ 
for  $x\ h :: 'a::\text{real-normed-div-algebra star}$ 
by (meson Infinitesimal-approx bex-Infinitesimal-iff inverse-add-Infinitesimal-approx)

```

```

lemma  $\text{Infinitesimal-square-iff}$ :  $x \in \text{Infinitesimal} \longleftrightarrow x * x \in \text{Infinitesimal}$ 
for  $x :: 'a::\text{real-normed-div-algebra star}$ 
using  $\text{Infinitesimal-mult Infinitesimal-mult-disj}$  by auto
declare  $\text{Infinitesimal-square-iff} \ [ \text{symmetric, simp} ]$ 

```

```

lemma  $\text{HFinite-square-iff} \ [ \text{simp} ]$ :  $x * x \in \text{HFinite} \longleftrightarrow x \in \text{HFinite}$ 
for  $x :: 'a::\text{real-normed-div-algebra star}$ 
using  $\text{HFinite-HInfinitesimal-iff HFinite-mult HInfinitesimal-mult}$  by blast

```

lemma *HInfinite-square-iff* [simp]: $x * x \in HInfinite \longleftrightarrow x \in HInfinite$
for $x :: 'a::real-normed-div-algebra$ *star*
by (auto simp add: *HInfinite-HFinite-iff*)

lemma *approx-HFinite-mult-cancel*: $a \in HFinite - Infinitesimal \implies a * w \approx a * z \implies w \approx z$
for $a w z :: 'a::real-normed-div-algebra$ *star*
by (metis *DiffD2 Infinitesimal-mult-disj bex-Infinitesimal-iff right-diff-distrib*)

lemma *approx-HFinite-mult-cancel-iff1*: $a \in HFinite - Infinitesimal \implies a * w \approx a * z \longleftrightarrow w \approx z$
for $a w z :: 'a::real-normed-div-algebra$ *star*
by (auto intro: *approx-mult2 approx-HFinite-mult-cancel*)

lemma *HInfinite-HFinite-add-cancel*: $x + y \in HInfinite \implies y \in HFinite \implies x \in HInfinite$
using *HFinite-add HInfinite-HFinite-iff* **by** blast

lemma *HInfinite-HFinite-add*: $x \in HInfinite \implies y \in HFinite \implies x + y \in HInfinite$
by (metis (no-types, opaque-lifting) *HFinite-HInfinite-iff HFinite-add HFinite-minus-iff add commute add-minus-cancel*)

lemma *HInfinite-ge-HInfinite*: $x \in HInfinite \implies x \leq y \implies 0 \leq x \implies y \in HInfinite$
for $x y :: hypreal$
by (auto intro: *HFinite-bounded simp add: HInfinite-HFinite-iff*)

lemma *Infinitesimal-inverse-HInfinite*: $x \in Infinitesimal \implies x \neq 0 \implies \text{inverse } x \in HInfinite$
for $x :: 'a::real-normed-div-algebra$ *star*
by (metis *Infinitesimal-HFinite-mult not-HFinite-HInfinite one-not-Infinitesimal right-inverse*)

lemma *HInfinite-HFinite-not-Infinitesimal-mult*:
 $x \in HInfinite \implies y \in HFinite - Infinitesimal \implies x * y \in HInfinite$
for $x y :: 'a::real-normed-div-algebra$ *star*
by (metis (no-types, opaque-lifting) *HFinite-HInfinite-iff HFinite-Infinitesimal-not-zero HFinite-inverse2 HFinite-mult mult.assoc mult.right-neutral right-inverse*)

lemma *HInfinite-HFinite-not-Infinitesimal-mult2*:
 $x \in HInfinite \implies y \in HFinite - Infinitesimal \implies y * x \in HInfinite$
for $x y :: 'a::real-normed-div-algebra$ *star*
by (metis *Diff-iff HInfinite-HFinite-iff HInfinite-inverse-Infinitesimal Infinitesimal-HFinite-mult2 divide-inverse mult-zero-right nonzero-eq-divide-eq*)

lemma *HInfinite-gt-SReal*: $x \in HInfinite \implies 0 < x \implies y \in \mathbb{R} \implies y < x$
for $x y :: hypreal$
by (auto dest!: *bspec simp add: HInfinite-def abs-if order-less-imp-le*)

lemma *HInfinite-gt-zero-gt-one*: $x \in HInfinite \implies 0 < x \implies 1 < x$
for $x :: hypreal$
by (*auto intro: HInfinite-gt-SReal*)

lemma *not-HInfinite-one* [*simp*]: $1 \notin HInfinite$
by (*simp add: HInfinite-HFinite-iff*)

lemma *approx-hrabs-disj*: $|x| \approx x \vee |x| \approx -x$
for $x :: hypreal$
by (*simp add: abs-if*)

6.4 Theorems about Monads

lemma *monad-hrabs-Un-subset*: $monad\ |x| \leq monad\ x \cup monad\ (-x)$
for $x :: hypreal$
by (*simp add: abs-if*)

lemma *Infinitesimal-monad-eq*: $e \in Infinitesimal \implies monad\ (x + e) = monad\ x$
by (*fast intro!: Infinitesimal-add-approx-self [THEN approx-sym] approx-monad-iff [THEN iffD1]*)

lemma *mem-monad-iff*: $u \in monad\ x \longleftrightarrow -u \in monad\ (-x)$
by (*simp add: monad-def*)

lemma *Infinitesimal-monad-zero-iff*: $x \in Infinitesimal \longleftrightarrow x \in monad\ 0$
by (*auto intro: approx-sym simp add: monad-def mem-infmal-iff*)

lemma *monad-zero-minus-iff*: $x \in monad\ 0 \longleftrightarrow -x \in monad\ 0$
by (*simp add: Infinitesimal-monad-zero-iff [symmetric]*)

lemma *monad-zero-hrabs-iff*: $x \in monad\ 0 \longleftrightarrow |x| \in monad\ 0$
for $x :: hypreal$
using *Infinitesimal-monad-zero-iff* **by** *blast*

lemma *mem-monad-self* [*simp*]: $x \in monad\ x$
by (*simp add: monad-def*)

6.5 Proof that $x \approx y$ implies $|x| \approx |y|$

lemma *approx-subset-monad*: $x \approx y \implies \{x, y\} \leq monad\ x$
by (*simp (no-asm) (simp add: approx-monad-iff)*)

lemma *approx-subset-monad2*: $x \approx y \implies \{x, y\} \leq monad\ y$
using *approx-subset-monad approx-sym* **by** *auto*

lemma *mem-monad-approx*: $u \in monad\ x \implies x \approx u$
by (*simp add: monad-def*)

lemma *approx-mem-monad*: $x \approx u \implies u \in monad\ x$

by (*simp add: monad-def*)

lemma *approx-mem-monad2*: $x \approx u \implies x \in \text{monad } u$
using *approx-mem-monad approx-sym* **by** *blast*

lemma *approx-mem-monad-zero*: $x \approx y \implies x \in \text{monad } 0 \implies y \in \text{monad } 0$
using *approx-trans monad-def* **by** *blast*

lemma *Infinitesimal-approx-hrabs*: $x \approx y \implies x \in \text{Infinitesimal} \implies |x| \approx |y|$
for $x y :: \text{hypreal}$
using *approx-hnorm* **by** *fastforce*

lemma *less-Infinitesimal-less*: $0 < x \implies x \notin \text{Infinitesimal} \implies e \in \text{Infinitesimal} \implies e < x$
for $x :: \text{hypreal}$
using *Infinitesimal-interval less-linear* **by** *blast*

lemma *Ball-mem-monad-gt-zero*: $0 < x \implies x \notin \text{Infinitesimal} \implies u \in \text{monad } x \implies 0 < u$
for $u x :: \text{hypreal}$
by (*metis* *bex-Infinitesimal-iff2 less-Infinitesimal-less less-add-same-cancel2 mem-monad-approx*)

lemma *Ball-mem-monad-less-zero*: $x < 0 \implies x \notin \text{Infinitesimal} \implies u \in \text{monad } x \implies u < 0$
for $u x :: \text{hypreal}$
by (*metis* *Ball-mem-monad-gt-zero approx-monad-iff less-asym linorder-neqE-linordered-idom mem-infmal-iff mem-monad-approx mem-monad-self*)

lemma *lemma-approx-gt-zero*: $0 < x \implies x \notin \text{Infinitesimal} \implies x \approx y \implies 0 < y$
for $x y :: \text{hypreal}$
by (*blast dest: Ball-mem-monad-gt-zero approx-subset-monad*)

lemma *lemma-approx-less-zero*: $x < 0 \implies x \notin \text{Infinitesimal} \implies x \approx y \implies y < 0$
for $x y :: \text{hypreal}$
by (*blast dest: Ball-mem-monad-less-zero approx-subset-monad*)

lemma *approx-hrabs*: $x \approx y \implies |x| \approx |y|$
for $x y :: \text{hypreal}$
by (*drule approx-hnorm*) *simp*

lemma *approx-hrabs-zero-cancel*: $|x| \approx 0 \implies x \approx 0$
for $x :: \text{hypreal}$
using *mem-infmal-iff* **by** *blast*

lemma *approx-hrabs-add-Infinitesimal*: $e \in \text{Infinitesimal} \implies |x| \approx |x + e|$
for $e x :: \text{hypreal}$
by (*fast intro: approx-hrabs Infinitesimal-add-approx-self*)

lemma *approx-hrabs-add-minus-Infinitesimal*: $e \in \text{Infinitesimal} \implies |x| \approx |x + -e|$

for $e \ x :: \text{hypreal}$

by (*fast intro: approx-hrabs Infinitesimal-add-minus-approx-self*)

lemma *hrabs-add-Infinitesimal-cancel*:

$e \in \text{Infinitesimal} \implies e' \in \text{Infinitesimal} \implies |x + e| = |y + e'| \implies |x| \approx |y|$

for $e \ e' \ x \ y :: \text{hypreal}$

by (*metis approx-hrabs-add-Infinitesimal approx-trans2*)

lemma *hrabs-add-minus-Infinitesimal-cancel*:

$e \in \text{Infinitesimal} \implies e' \in \text{Infinitesimal} \implies |x + -e| = |y + -e'| \implies |x| \approx |y|$

for $e \ e' \ x \ y :: \text{hypreal}$

by (*meson Infinitesimal-minus-iff hrabs-add-Infinitesimal-cancel*)

6.6 More HFinite and Infinitesimal Theorems

Interesting slightly counterintuitive theorem: necessary for proving that an open interval is an NS open set.

lemma *Infinitesimal-add-hypreal-of-real-less*:

assumes $x < y$ **and** $u: u \in \text{Infinitesimal}$

shows *hypreal-of-real* $x + u < \text{hypreal-of-real } y$

proof –

have $|u| < \text{hypreal-of-real } y - \text{hypreal-of-real } x$

using *InfinitesimalD* $\langle x < y \rangle \ u$ **by** *fastforce*

then show *?thesis*

by (*simp add: abs-less-iff*)

qed

lemma *Infinitesimal-add-hrabs-hypreal-of-real-less*:

$x \in \text{Infinitesimal} \implies |\text{hypreal-of-real } r| < \text{hypreal-of-real } y \implies$

$|\text{hypreal-of-real } r + x| < \text{hypreal-of-real } y$

by (*metis Infinitesimal-add-hypreal-of-real-less approx-hrabs-add-Infinitesimal approx-sym beX-Infinitesimal-iff2 star-of-abs star-of-less*)

lemma *Infinitesimal-add-hrabs-hypreal-of-real-less2*:

$x \in \text{Infinitesimal} \implies |\text{hypreal-of-real } r| < \text{hypreal-of-real } y \implies$

$|x + \text{hypreal-of-real } r| < \text{hypreal-of-real } y$

using *Infinitesimal-add-hrabs-hypreal-of-real-less* **by** *fastforce*

lemma *hypreal-of-real-le-add-Infininitesimal-cancel*:

assumes *le*: *hypreal-of-real* $x + u \leq \text{hypreal-of-real } y + v$

and $u: u \in \text{Infinitesimal}$ **and** $v: v \in \text{Infinitesimal}$

shows *hypreal-of-real* $x \leq \text{hypreal-of-real } y$

proof (*rule ccontr*)

assume $\neg \text{hypreal-of-real } x \leq \text{hypreal-of-real } y$

then have *hypreal-of-real* $y + (v - u) < \text{hypreal-of-real } x$

by (*simp add: Infinitesimal-add-hypreal-of-real-less Infinitesimal-diff u v*)

then show *False*

by (*simp add: add-diff-eq add-le-imp-le-diff le leD*)
qed

lemma *hypreal-of-real-le-add-Infininitesimal-cancel2*:

$u \in \text{Infininitesimal} \implies v \in \text{Infininitesimal} \implies$

$\text{hypreal-of-real } x + u \leq \text{hypreal-of-real } y + v \implies x \leq y$

by (*blast intro: star-of-le [THEN iffD1] intro!: hypreal-of-real-le-add-Infininitesimal-cancel*)

lemma *hypreal-of-real-less-Infininitesimal-le-zero*:

$\text{hypreal-of-real } x < e \implies e \in \text{Infininitesimal} \implies \text{hypreal-of-real } x \leq 0$

by (*metis Infininitesimal-interval eq-iff le-less-linear star-of-Infininitesimal-iff-0 star-of-eq-0*)

lemma *Infininitesimal-add-not-zero*: $h \in \text{Infininitesimal} \implies x \neq 0 \implies \text{star-of } x + h \neq 0$

by (*metis Infininitesimal-add-approx-self star-of-approx-zero-iff*)

lemma *monad-hrabs-less*: $y \in \text{monad } x \implies 0 < \text{hypreal-of-real } e \implies |y - x| < \text{hypreal-of-real } e$

by (*simp add: Infininitesimal-approx-minus approx-sym less-Infininitesimal-less mem-monad-approx*)

lemma *mem-monad-SReal-HFfinite*: $x \in \text{monad } (\text{hypreal-of-real } a) \implies x \in \text{HFfinite}$

using *HFfinite-star-of approx-HFfinite mem-monad-approx* **by** *blast*

6.7 Theorems about Standard Part

lemma *st-approx-self*: $x \in \text{HFfinite} \implies \text{st } x \approx x$

by (*metis (no-types, lifting) approx-refl approx-trans3 someI-ex st-def st-part-Ex st-part-Ex1*)

lemma *st-SReal*: $x \in \text{HFfinite} \implies \text{st } x \in \mathbb{R}$

by (*metis (mono-tags, lifting) approx-sym someI-ex st-def st-part-Ex*)

lemma *st-HFfinite*: $x \in \text{HFfinite} \implies \text{st } x \in \text{HFfinite}$

by (*erule st-SReal [THEN SReal-subset-HFfinite [THEN subsetD]]*)

lemma *st-unique*: $r \in \mathbb{R} \implies r \approx x \implies \text{st } x = r$

by (*meson SReal-subset-HFfinite approx-HFfinite approx-unique-real st-SReal st-approx-self subsetD*)

lemma *st-SReal-eq*: $x \in \mathbb{R} \implies \text{st } x = x$

by (*metis approx-refl st-unique*)

lemma *st-hypreal-of-real [simp]*: $\text{st } (\text{hypreal-of-real } x) = \text{hypreal-of-real } x$

by (*rule SReal-hypreal-of-real [THEN st-SReal-eq]*)

lemma *st-eq-approx*: $x \in \text{HFfinite} \implies y \in \text{HFfinite} \implies \text{st } x = \text{st } y \implies x \approx y$

by (*auto dest!: st-approx-self elim!: approx-trans3*)

lemma *approx-st-eq*:

assumes $x: x \in \text{HFinite}$ **and** $y: y \in \text{HFinite}$ **and** $xy: x \approx y$

shows $st\ x = st\ y$

proof –

have $st\ x \approx x\ st\ y \approx y\ st\ x \in \mathbb{R}\ st\ y \in \mathbb{R}$

by (*simp-all add: st-approx-self st-SReal x y*)

with xy **show** *?thesis*

by (*fast elim: approx-trans approx-trans2 SReal-approx-iff [THEN iffD1]*)

qed

lemma *st-eq-approx-iff*: $x \in \text{HFinite} \implies y \in \text{HFinite} \implies x \approx y \longleftrightarrow st\ x = st\ y$

by (*blast intro: approx-st-eq st-eq-approx*)

lemma *st-Infinitesimal-add-SReal*: $x \in \mathbb{R} \implies e \in \text{Infinitesimal} \implies st\ (x + e) = x$

by (*simp add: Infinitesimal-add-approx-self st-unique*)

lemma *st-Infinitesimal-add-SReal2*: $x \in \mathbb{R} \implies e \in \text{Infinitesimal} \implies st\ (e + x) = x$

by (*metis add.commute st-Infinitesimal-add-SReal*)

lemma *HFinite-st-Infinitesimal-add*: $x \in \text{HFinite} \implies \exists e \in \text{Infinitesimal}. x = st(x) + e$

by (*blast dest!: st-approx-self [THEN approx-sym] be-Infinitesimal-iff2 [THEN iffD2]*)

lemma *st-add*: $x \in \text{HFinite} \implies y \in \text{HFinite} \implies st\ (x + y) = st\ x + st\ y$

by (*simp add: st-unique st-SReal st-approx-self approx-add*)

lemma *st-numeral* [*simp*]: $st\ (\text{numeral } w) = \text{numeral } w$

by (*rule Reals-numeral [THEN st-SReal-eq]*)

lemma *st-neg-numeral* [*simp*]: $st\ (-\ \text{numeral } w) = -\ \text{numeral } w$

using *st-unique* **by** *auto*

lemma *st-0* [*simp*]: $st\ 0 = 0$

by (*simp add: st-SReal-eq*)

lemma *st-1* [*simp*]: $st\ 1 = 1$

by (*simp add: st-SReal-eq*)

lemma *st-neg-1* [*simp*]: $st\ (-\ 1) = -\ 1$

by (*simp add: st-SReal-eq*)

lemma *st-minus*: $x \in \text{HFinite} \implies st\ (-\ x) = -\ st\ x$

by (*simp add: st-unique st-SReal st-approx-self approx-minus*)

lemma *st-diff*: $\llbracket x \in \text{HFinite}; y \in \text{HFinite} \rrbracket \implies st\ (x - y) = st\ x - st\ y$

by (*simp add: st-unique st-SReal st-approx-self approx-diff*)

lemma *st-mult*: $\llbracket x \in \text{HFinite}; y \in \text{HFinite} \rrbracket \implies \text{st } (x * y) = \text{st } x * \text{st } y$
by (*simp add: st-unique st-SReal st-approx-self approx-mult-HFinite*)

lemma *st-Infinitesimal*: $x \in \text{Infinitesimal} \implies \text{st } x = 0$
by (*simp add: st-unique mem-infmal-iff*)

lemma *st-not-Infinitesimal*: $\text{st}(x) \neq 0 \implies x \notin \text{Infinitesimal}$
by (*fast intro: st-Infinitesimal*)

lemma *st-inverse*: $x \in \text{HFinite} \implies \text{st } x \neq 0 \implies \text{st } (\text{inverse } x) = \text{inverse } (\text{st } x)$
by (*simp add: approx-inverse st-SReal st-approx-self st-not-Infinitesimal st-unique*)

lemma *st-divide* [*simp*]: $x \in \text{HFinite} \implies y \in \text{HFinite} \implies \text{st } y \neq 0 \implies \text{st } (x / y) = \text{st } x / \text{st } y$
by (*simp add: divide-inverse st-mult st-not-Infinitesimal HFinite-inverse st-inverse*)

lemma *st-idempotent* [*simp*]: $x \in \text{HFinite} \implies \text{st } (\text{st } x) = \text{st } x$
by (*blast intro: st-HFinite st-approx-self approx-st-eq*)

lemma *Infinitesimal-add-st-less*:
 $x \in \text{HFinite} \implies y \in \text{HFinite} \implies u \in \text{Infinitesimal} \implies \text{st } x < \text{st } y \implies \text{st } x + u < \text{st } y$
by (*metis Infinitesimal-add-hypreal-of-real-less SReal-iff st-SReal star-of-less*)

lemma *Infinitesimal-add-st-le-cancel*:
 $x \in \text{HFinite} \implies y \in \text{HFinite} \implies u \in \text{Infinitesimal} \implies \text{st } x \leq \text{st } y + u \implies \text{st } x \leq \text{st } y$
by (*meson Infinitesimal-add-st-less leD le-less-linear*)

lemma *st-le*: $x \in \text{HFinite} \implies y \in \text{HFinite} \implies x \leq y \implies \text{st } x \leq \text{st } y$
by (*metis approx-le-bound approx-sym linear st-SReal st-approx-self st-part-Ex1*)

lemma *st-zero-le*: $0 \leq x \implies x \in \text{HFinite} \implies 0 \leq \text{st } x$
by (*metis HFinite-0 st-0 st-le*)

lemma *st-zero-ge*: $x \leq 0 \implies x \in \text{HFinite} \implies \text{st } x \leq 0$
by (*metis HFinite-0 st-0 st-le*)

lemma *st-hrabs*: $x \in \text{HFinite} \implies |\text{st } x| = \text{st } |x|$
by (*simp add: order-class.order.antisym st-zero-ge linorder-not-le st-zero-le abs-if st-minus linorder-not-less*)

6.8 Alternative Definitions using Free Ultrafilter

6.8.1 *HFinite*

lemma *HFinite-FreeUltrafilterNat*:
assumes *star-n* $X \in \text{HFinite}$
shows $\exists u. \text{eventually } (\lambda n. \text{norm } (X \ n) < u) \mathcal{U}$

proof –

obtain r **where** $hnorm\ (star\text{-}n\ X) < hypreal\text{-}of\text{-}real\ r$
using $HFiniteD\ SReal\text{-}iff\ assms$ **by** $fastforce$
then have $\forall_F n\ in\ \mathcal{U}. norm\ (X\ n) < r$
by $(simp\ add: hnorm\text{-}def\ star\text{-}n\text{-}less\ star\text{-}of\text{-}def\ starfun\text{-}star\text{-}n)$
then show $?thesis ..$

qed

lemma $FreeUltrafilterNat\text{-}HFinite$:

assumes $eventually\ (\lambda n. norm\ (X\ n) < u)\ \mathcal{U}$
shows $star\text{-}n\ X \in HFinite$

proof –

have $hnorm\ (star\text{-}n\ X) < hypreal\text{-}of\text{-}real\ u$
by $(simp\ add: assms\ hnorm\text{-}def\ star\text{-}n\text{-}less\ star\text{-}of\text{-}def\ starfun\text{-}star\text{-}n)$
then show $?thesis$
by $(meson\ HInfiniteD\ SReal\text{-}hypreal\text{-}of\text{-}real\ less\text{-}asym\ not\text{-}HFinite\ HInfinite)$

qed

lemma $HFinite\text{-}FreeUltrafilterNat\text{-}iff$:

$star\text{-}n\ X \in HFinite \longleftrightarrow (\exists u. eventually\ (\lambda n. norm\ (X\ n) < u)\ \mathcal{U})$
using $FreeUltrafilterNat\text{-}HFinite\ HFinite\text{-}FreeUltrafilterNat$ **by** $blast$

6.8.2 $HInfinite$

Exclude this type of sets from free ultrafilter for Infinite numbers!

lemma $FreeUltrafilterNat\text{-}const\text{-}Finite$:

$eventually\ (\lambda n. norm\ (X\ n) = u)\ \mathcal{U} \implies star\text{-}n\ X \in HFinite$
by $(simp\ add: FreeUltrafilterNat\text{-}HFinite\ [where\ u = u+1]\ eventually\text{-}mono)$

lemma $HInfinite\text{-}FreeUltrafilterNat$:

assumes $star\text{-}n\ X \in HInfinite$ **shows** $\forall_F n\ in\ \mathcal{U}. u < norm\ (X\ n)$

proof –

have $\neg (\forall_F n\ in\ \mathcal{U}. norm\ (X\ n) < u + 1)$
using $FreeUltrafilterNat\text{-}HFinite\ HFinite\text{-}HInfinite\text{-}iff\ assms$ **by** $auto$
then show $?thesis$
by $(auto\ simp\ flip: FreeUltrafilterNat.\text{eventually}\text{-}not\text{-}iff\ elim: eventually\text{-}mono)$

qed

lemma $FreeUltrafilterNat\text{-}HInfinite$:

assumes $\bigwedge u. eventually\ (\lambda n. u < norm\ (X\ n))\ \mathcal{U}$
shows $star\text{-}n\ X \in HInfinite$

proof –

{ fix u
assume $\forall_F n\ in\ \mathcal{U}. norm\ (X\ n) < u\ \forall_F n\ in\ \mathcal{U}. u < norm\ (X\ n)$
then have $\forall_F x\ in\ \mathcal{U}. False$
by $eventually\text{-}elim\ auto$
then have $False$
by $(simp\ add: eventually\text{-}False\ FreeUltrafilterNat.\text{proper})$ **}**
then show $?thesis$

using *HFinite-FreeUltrafilterNat HInfinite-HFinite-iff* **assms** **by** *blast*
qed

lemma *HInfinite-FreeUltrafilterNat-iff*:
 $\text{star-}n\ X \in \text{HInfinite} \longleftrightarrow (\forall u. \text{eventually } (\lambda n. u < \text{norm } (X\ n))\ \mathcal{U})$
using *HInfinite-FreeUltrafilterNat FreeUltrafilterNat-HInfinite* **by** *blast*

6.8.3 Infinitesimal

lemma *ball-SReal-eq*: $(\forall x::\text{hypreal} \in \text{Reals}. P\ x) \longleftrightarrow (\forall x::\text{real}. P\ (\text{star-of } x))$
by *(auto simp: SReal-def)*

lemma *Infinitesimal-FreeUltrafilterNat-iff*:
 $(\text{star-}n\ X \in \text{Infinitesimal}) = (\forall u>0. \text{eventually } (\lambda n. \text{norm } (X\ n) < u)\ \mathcal{U})$ **(is**
 $?lhs = ?rhs)$
proof –
have $?lhs \longleftrightarrow (\forall r>0. \text{hnorm } (\text{star-}n\ X) < \text{hypreal-of-real } r)$
by *(simp add: Infinitesimal-def ball-SReal-eq)*
also have $\dots \longleftrightarrow ?rhs$
by *(simp add: hnrm-def starfun-star-n star-of-def star-less-def starP2-star-n)*
finally show $?thesis$.
qed

Infinitesimals as smaller than $1/n$ for all $n::\text{nat } (> 0)$.

lemma *lemma-Infinitesimal*: $(\forall r. 0 < r \longrightarrow x < r) \longleftrightarrow (\forall n. x < \text{inverse } (\text{real } (\text{Suc } n)))$
by *(meson inverse-positive-iff-positive less-trans of-nat-0-less-iff reals-Archimedean zero-less-Suc)*

lemma *lemma-Infinitesimal2*:
 $(\forall r \in \text{Reals}. 0 < r \longrightarrow x < r) \longleftrightarrow (\forall n. x < \text{inverse}(\text{hypreal-of-nat } (\text{Suc } n)))$
(is - = ?rhs)
proof *(intro iffI strip)*
assume $R: ?rhs$
fix $r::\text{hypreal}$
assume $r \in \mathbb{R}\ 0 < r$
then obtain $n\ y$ **where** $\text{inverse } (\text{real } (\text{Suc } n)) < y$ **and** $r: r = \text{hypreal-of-real } y$
by *(metis SReal-iff reals-Archimedean star-of-0-less)*
then have $\text{inverse } (1 + \text{hypreal-of-nat } n) < \text{hypreal-of-real } y$
by *(metis of-nat-Suc star-of-inverse star-of-less star-of-nat-def)*
then show $x < r$
by *(metis R r le-less-trans less-imp-le of-nat-Suc)*
qed *(meson Reals-inverse Reals-of-nat of-nat-0-less-iff positive-imp-inverse-positive zero-less-Suc)*

lemma *Infinitesimal-hypreal-of-nat-iff*:
 $\text{Infinitesimal} = \{x. \forall n. \text{hnorm } x < \text{inverse } (\text{hypreal-of-nat } (\text{Suc } n))\}$

using *Infinitesimal-def lemma-Infinitesimal2* **by** *auto*

6.9 Proof that ω is an infinite number

It will follow that ε is an infinitesimal number.

lemma *Suc-Un-eq*: $\{n. n < \text{Suc } m\} = \{n. n < m\} \cup \{n. n = m\}$
by (*auto simp add: less-Suc-eq*)

Prove that any segment is finite and hence cannot belong to \mathcal{U} .

lemma *finite-real-of-nat-segment*: $\text{finite } \{n::\text{nat}. \text{real } n < \text{real } (m::\text{nat})\}$
by *auto*

lemma *finite-real-of-nat-less-real*: $\text{finite } \{n::\text{nat}. \text{real } n < u\}$

proof –

obtain m **where** $u < \text{real } m$

using *reals-Archimedean2* **by** *blast*

then have $\{n. \text{real } n < u\} \subseteq \{.. < m\}$

by *force*

then show *?thesis*

using *finite-nat-iff-bounded* **by** *force*

qed

lemma *finite-real-of-nat-le-real*: $\text{finite } \{n::\text{nat}. \text{real } n \leq u\}$

by (*metis infinite-nat-iff-unbounded leD le-nat-floor mem-Collect-eq*)

lemma *finite-rabs-real-of-nat-le-real*: $\text{finite } \{n::\text{nat}. |\text{real } n| \leq u\}$

by (*simp add: finite-real-of-nat-le-real*)

lemma *rabs-real-of-nat-le-real-FreeUltrafilterNat*:

$\neg \text{eventually } (\lambda n. |\text{real } n| \leq u) \mathcal{U}$

by (*blast intro!: FreeUltrafilterNat.finite finite-rabs-real-of-nat-le-real*)

lemma *FreeUltrafilterNat-nat-gt-real*: $\text{eventually } (\lambda n. u < \text{real } n) \mathcal{U}$

proof –

have $\{n::\text{nat}. \neg u < \text{real } n\} = \{n. \text{real } n \leq u\}$

by *auto*

then show *?thesis*

by (*auto simp add: FreeUltrafilterNat.finite' finite-real-of-nat-le-real*)

qed

The complement of $\{n. |\text{real } n| \leq u\} = \{n. u < |\text{real } n|\}$ is in \mathcal{U} by property of (free) ultrafilters.

ω is a member of *HInfinite*.

theorem *HInfinite-omega* [*simp*]: $\omega \in \text{HInfinite}$

proof –

have $\forall_F n \text{ in } \mathcal{U}. u < \text{norm } (1 + \text{real } n)$ **for** u

using *FreeUltrafilterNat-nat-gt-real* [*of u-1*] *eventually-mono* **by** *fastforce*

then show *?thesis*
by (*simp add: omega-def FreeUltrafilterNat-HInfinite*)
qed

Epsilon is a member of Infinitesimal.

lemma *Infinitesimal-epsilon* [*simp*]: $\varepsilon \in \text{Infinitesimal}$
by (*auto intro!: HInfinite-inverse-Infinitesimal HInfinite-omega simp add: epsilon-inverse-omega*)

lemma *HFinite-epsilon* [*simp*]: $\varepsilon \in \text{HFinite}$
by (*auto intro: Infinitesimal-subset-HFinite [THEN subsetD]*)

lemma *epsilon-approx-zero* [*simp*]: $\varepsilon \approx 0$
by (*simp add: mem-infmal-iff [symmetric]*)

Needed for proof that we define a hyperreal $[<X(n)] \approx \text{hypreal-of-real } a$ given that $\forall n. |X\ n - a| < 1/n$. Used in proof of *NSLIM* \Rightarrow *LIM*.

lemma *real-of-nat-less-inverse-iff*: $0 < u \implies u < \text{inverse}(\text{real}(\text{Suc } n)) \longleftrightarrow \text{real}(\text{Suc } n) < \text{inverse } u$
using *less-imp-inverse-less* **by** *force*

lemma *finite-inverse-real-of-posnat-gt-real*: $0 < u \implies \text{finite } \{n. u < \text{inverse}(\text{real}(\text{Suc } n))\}$

proof (*simp only: real-of-nat-less-inverse-iff*)
have $\{n. 1 + \text{real } n < \text{inverse } u\} = \{n. \text{real } n < \text{inverse } u - 1\}$
by *fastforce*
then show $\text{finite } \{n. \text{real}(\text{Suc } n) < \text{inverse } u\}$
using *finite-real-of-nat-less-real [of inverse u - 1]*
by *auto*
qed

lemma *finite-inverse-real-of-posnat-ge-real*:

assumes $0 < u$
shows $\text{finite } \{n. u \leq \text{inverse}(\text{real}(\text{Suc } n))\}$

proof –

have $\forall na. u \leq \text{inverse}(1 + \text{real } na) \longrightarrow na \leq \text{ceiling}(\text{inverse } u)$
by (*smt (verit, best) assms ceiling-less-cancel ceiling-of-nat inverse-inverse-eq inverse-le-iff-le*)
then show *?thesis*
apply (*auto simp add: finite-nat-set-iff-bounded-le*)
by (*meson assms inverse-positive-iff-positive le-nat-iff less-imp-le zero-less-ceiling*)
qed

lemma *inverse-real-of-posnat-ge-real-FreeUltrafilterNat*:

$0 < u \implies \neg \text{eventually } (\lambda n. u \leq \text{inverse}(\text{real}(\text{Suc } n))) \mathcal{U}$
by (*blast intro!: FreeUltrafilterNat.finite finite-inverse-real-of-posnat-ge-real*)

lemma *FreeUltrafilterNat-inverse-real-of-posnat*:

$0 < u \implies \text{eventually } (\lambda n. \text{inverse}(\text{real}(\text{Suc } n)) < u) \mathcal{U}$

by (*drule inverse-real-of-posnat-ge-real-FreeUltrafilterNat*)
(simp add: FreeUltrafilterNat.eventually-not-iff not-le[symmetric]))

Example of an hypersequence (i.e. an extended standard sequence) whose term with an hypernatural suffix is an infinitesimal i.e. the n th term of the hypersequence is a member of Infinitesimal

lemma *SEQ-Infinitesimal*: ($*f*$ ($\lambda n::nat. inverse(real(Suc n))$)) $whn \in Infinitesimal$

by (*simp add: hypnat-omega-def starfun-star-n star-n-inverse Infinitesimal-FreeUltrafilterNat-iff*
FreeUltrafilterNat-inverse-real-of-posnat del: of-nat-Suc)

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 and standard definitions of pointwise limit.

$|X(n) - x| < 1/n \implies [X n] - hypreal-of-real x \in Infinitesimal$

lemma *real-seq-to-hypreal-Infinitesimal*:

$\forall n. norm(X n - x) < inverse(real(Suc n)) \implies star-n X - star-of x \in Infinitesimal$

unfolding *star-n-diff star-of-def Infinitesimal-FreeUltrafilterNat-iff star-n-inverse*

by (*auto dest!: FreeUltrafilterNat-inverse-real-of-posnat*
intro: order-less-trans elim!: eventually-mono)

lemma *real-seq-to-hypreal-approx*:

$\forall n. norm(X n - x) < inverse(real(Suc n)) \implies star-n X \approx star-of x$

by (*metis bex-Infinitesimal-iff real-seq-to-hypreal-Infinitesimal*)

lemma *real-seq-to-hypreal-approx2*:

$\forall n. norm(x - X n) < inverse(real(Suc n)) \implies star-n X \approx star-of x$

by (*metis norm-minus-commute real-seq-to-hypreal-approx*)

lemma *real-seq-to-hypreal-Infinitesimal2*:

$\forall n. norm(X n - Y n) < inverse(real(Suc n)) \implies star-n X - star-n Y \in Infinitesimal$

unfolding *Infinitesimal-FreeUltrafilterNat-iff star-n-diff*

by (*auto dest!: FreeUltrafilterNat-inverse-real-of-posnat*
intro: order-less-trans elim!: eventually-mono)

end

7 Nonstandard Complex Numbers

theory *NSComplex*

imports *NSA*

begin

type-synonym *hcomplex* = *complex star*

abbreviation $hcomplex\text{-}of\text{-}complex :: complex \Rightarrow complex\ star$
where $hcomplex\text{-}of\text{-}complex \equiv star\text{-}of$

abbreviation $hcm\text{-}of :: complex\ star \Rightarrow real\ star$
where $hcm\text{-}of \equiv hnorm$

7.0.1 Real and Imaginary parts

definition $hRe :: hcomplex \Rightarrow hypreal$
where $hRe = *f* Re$

definition $hIm :: hcomplex \Rightarrow hypreal$
where $hIm = *f* Im$

7.0.2 Imaginary unit

definition $iii :: hcomplex$
where $iii = star\text{-}of\ i$

7.0.3 Complex conjugate

definition $hcnj :: hcomplex \Rightarrow hcomplex$
where $hcnj = *f* cnj$

7.0.4 Argand

definition $hsgn :: hcomplex \Rightarrow hcomplex$
where $hsgn = *f* sgn$

definition $harg :: hcomplex \Rightarrow hypreal$
where $harg = *f* Arg$

definition — abbreviation for $\cos a + i \sin a$
 $hcis :: hypreal \Rightarrow hcomplex$
where $hcis = *f* cis$

7.0.5 Injection from hyperreals

abbreviation $hcomplex\text{-}of\text{-}hypreal :: hypreal \Rightarrow hcomplex$
where $hcomplex\text{-}of\text{-}hypreal \equiv of\text{-}hypreal$

definition — abbreviation for $r * (\cos a + i \sin a)$
 $hrcis :: hypreal \Rightarrow hypreal \Rightarrow hcomplex$
where $hrcis = *f2* rcis$

7.0.6 $e^{\wedge}(x + iy)$

definition $hExp :: hcomplex \Rightarrow hcomplex$
where $hExp = *f* exp$

definition $HComplex :: hypreal \Rightarrow hypreal \Rightarrow hcomplex$
where $HComplex = *f2* Complex$

lemmas $hcomplex-defs [transfer-unfold] =$
 $hRe-def hIm-def iii-def hcnj-def hsgn-def harg-def hcis-def$
 $hrcis-def hExp-def HComplex-def$

lemma $Standard-hRe [simp]: x \in Standard \implies hRe x \in Standard$
by ($simp add: hcomplex-defs$)

lemma $Standard-hIm [simp]: x \in Standard \implies hIm x \in Standard$
by ($simp add: hcomplex-defs$)

lemma $Standard-iii [simp]: iii \in Standard$
by ($simp add: hcomplex-defs$)

lemma $Standard-hcnj [simp]: x \in Standard \implies hcnj x \in Standard$
by ($simp add: hcomplex-defs$)

lemma $Standard-hsgn [simp]: x \in Standard \implies hsgn x \in Standard$
by ($simp add: hcomplex-defs$)

lemma $Standard-harg [simp]: x \in Standard \implies harg x \in Standard$
by ($simp add: hcomplex-defs$)

lemma $Standard-hcis [simp]: r \in Standard \implies hcis r \in Standard$
by ($simp add: hcomplex-defs$)

lemma $Standard-hExp [simp]: x \in Standard \implies hExp x \in Standard$
by ($simp add: hcomplex-defs$)

lemma $Standard-hrcis [simp]: r \in Standard \implies s \in Standard \implies hrcis r s \in Standard$
by ($simp add: hcomplex-defs$)

lemma $Standard-HComplex [simp]: r \in Standard \implies s \in Standard \implies HComplex r s \in Standard$
by ($simp add: hcomplex-defs$)

lemma $hcmmod-def: hcmmod = *f* cmod$
by ($rule hnrm-def$)

7.1 Properties of Nonstandard Real and Imaginary Parts

lemma $hcomplex-hRe-hIm-cancel-iff: \bigwedge w z. w = z \longleftrightarrow hRe w = hRe z \wedge hIm w = hIm z$
by $transfer (rule complex-eq-iff)$

lemma *hcomplex-equality* [intro?]: $\bigwedge z w. hRe\ z = hRe\ w \implies hIm\ z = hIm\ w \implies z = w$

by transfer (rule complex-eqI)

lemma *hcomplex-hRe-zero* [simp]: $hRe\ 0 = 0$

by transfer simp

lemma *hcomplex-hIm-zero* [simp]: $hIm\ 0 = 0$

by transfer simp

lemma *hcomplex-hRe-one* [simp]: $hRe\ 1 = 1$

by transfer simp

lemma *hcomplex-hIm-one* [simp]: $hIm\ 1 = 0$

by transfer simp

7.2 Addition for Nonstandard Complex Numbers

lemma *hRe-add*: $\bigwedge x y. hRe\ (x + y) = hRe\ x + hRe\ y$

by transfer simp

lemma *hIm-add*: $\bigwedge x y. hIm\ (x + y) = hIm\ x + hIm\ y$

by transfer simp

7.3 More Minus Laws

lemma *hRe-minus*: $\bigwedge z. hRe\ (-\ z) = -\ hRe\ z$

by transfer (rule uminus-complex.sel)

lemma *hIm-minus*: $\bigwedge z. hIm\ (-\ z) = -\ hIm\ z$

by transfer (rule uminus-complex.sel)

lemma *hcomplex-add-minus-eq-minus*: $x + y = 0 \implies x = -\ y$

for $x\ y :: hcomplex$

apply (drule minus-unique)

apply (simp add: minus-equation-iff [of $x\ y$])

done

lemma *hcomplex-i-mult-eq* [simp]: $iii * iii = -\ 1$

by transfer (rule i-squared)

lemma *hcomplex-i-mult-left* [simp]: $\bigwedge z. iii * (iii * z) = -\ z$

by transfer (rule complex-i-mult-minus)

lemma *hcomplex-i-not-zero* [simp]: $iii \neq 0$

by transfer (rule complex-i-not-zero)

7.4 More Multiplication Laws

lemma *hcomplex-mult-minus-one*: $-\ 1 * z = -\ z$

for $z :: hcomplex$
by *simp*

lemma *hcomplex-mult-minus-one-right*: $z * - 1 = - z$
for $z :: hcomplex$
by *simp*

lemma *hcomplex-mult-left-cancel*: $c \neq 0 \implies c * a = c * b \longleftrightarrow a = b$
for $a b c :: hcomplex$
by *simp*

lemma *hcomplex-mult-right-cancel*: $c \neq 0 \implies a * c = b * c \longleftrightarrow a = b$
for $a b c :: hcomplex$
by *simp*

7.5 Subtraction and Division

lemma *hcomplex-diff-eq-eq* [*simp*]: $x - y = z \longleftrightarrow x = z + y$
for $x y z :: hcomplex$
by (*rule diff-eq-eq*)

7.6 Embedding Properties for *hcomplex-of-hypreal* Map

lemma *hRe-hcomplex-of-hypreal* [*simp*]: $\bigwedge z. hRe (hcomplex-of-hypreal z) = z$
by *transfer (rule Re-complex-of-real)*

lemma *hIm-hcomplex-of-hypreal* [*simp*]: $\bigwedge z. hIm (hcomplex-of-hypreal z) = 0$
by *transfer (rule Im-complex-of-real)*

lemma *hcomplex-of-epsilon-not-zero* [*simp*]: $hcomplex-of-hypreal \varepsilon \neq 0$
by (*simp add: epsilon-not-zero*)

7.7 *HComplex* theorems

lemma *hRe-HComplex* [*simp*]: $\bigwedge x y. hRe (HComplex x y) = x$
by *transfer simp*

lemma *hIm-HComplex* [*simp*]: $\bigwedge x y. hIm (HComplex x y) = y$
by *transfer simp*

lemma *hcomplex-surj* [*simp*]: $\bigwedge z. HComplex (hRe z) (hIm z) = z$
by *transfer (rule complex-surj)*

lemma *hcomplex-induct* [*case-names rect*]:
 $(\bigwedge x y. P (HComplex x y)) \implies P z$
by (*rule hcomplex-surj [THEN subst]*) *blast*

7.8 Modulus (Absolute Value) of Nonstandard Complex Number

lemma *hcomplex-of-hypreal-abs*:

$hcomplex\text{-of-hypreal } |x| = hcomplex\text{-of-hypreal } (hmod (hcomplex\text{-of-hypreal } x))$
by *simp*

lemma *HComplex-inject* [*simp*]: $\bigwedge x y x' y'. HComplex x y = HComplex x' y' \longleftrightarrow x = x' \wedge y = y'$

by *transfer (rule complex.inject)*

lemma *HComplex-add* [*simp*]:

$\bigwedge x1 y1 x2 y2. HComplex x1 y1 + HComplex x2 y2 = HComplex (x1 + x2) (y1 + y2)$
by *transfer (rule complex-add)*

lemma *HComplex-minus* [*simp*]: $\bigwedge x y. - HComplex x y = HComplex (- x) (- y)$

by *transfer (rule complex-minus)*

lemma *HComplex-diff* [*simp*]:

$\bigwedge x1 y1 x2 y2. HComplex x1 y1 - HComplex x2 y2 = HComplex (x1 - x2) (y1 - y2)$
by *transfer (rule complex-diff)*

lemma *HComplex-mult* [*simp*]:

$\bigwedge x1 y1 x2 y2. HComplex x1 y1 * HComplex x2 y2 = HComplex (x1*x2 - y1*y2) (x1*y2 + y1*x2)$
by *transfer (rule complex-mult)*

HComplex-inverse is proved below.

lemma *hcomplex-of-hypreal-eq*: $\bigwedge r. hcomplex\text{-of-hypreal } r = HComplex r 0$

by *transfer (rule complex-of-real-def)*

lemma *HComplex-add-hcomplex-of-hypreal* [*simp*]:

$\bigwedge x y r. HComplex x y + hcomplex\text{-of-hypreal } r = HComplex (x + r) y$
by *transfer (rule Complex-add-complex-of-real)*

lemma *hcomplex-of-hypreal-add-HComplex* [*simp*]:

$\bigwedge r x y. hcomplex\text{-of-hypreal } r + HComplex x y = HComplex (r + x) y$
by *transfer (rule complex-of-real-add-Complex)*

lemma *HComplex-mult-hcomplex-of-hypreal*:

$\bigwedge x y r. HComplex x y * hcomplex\text{-of-hypreal } r = HComplex (x * r) (y * r)$
by *transfer (rule Complex-mult-complex-of-real)*

lemma *hcomplex-of-hypreal-mult-HComplex*:

$\bigwedge r x y. hcomplex\text{-of-hypreal } r * HComplex x y = HComplex (r * x) (r * y)$
by *transfer (rule complex-of-real-mult-Complex)*

lemma *i-hcomplex-of-hypreal* [simp]: $\bigwedge r. \text{iii} * \text{hcomplex-of-hypreal } r = \text{HComplex } 0 \ r$

by *transfer* (rule *i-complex-of-real*)

lemma *hcomplex-of-hypreal-i* [simp]: $\bigwedge r. \text{hcomplex-of-hypreal } r * \text{iii} = \text{HComplex } 0 \ r$

by *transfer* (rule *complex-of-real-i*)

7.9 Conjugation

lemma *hcomplex-hcnj-cancel-iff* [iff]: $\bigwedge x \ y. \text{hcnj } x = \text{hcnj } y \longleftrightarrow x = y$

by *transfer* (rule *complex-cnj-cancel-iff*)

lemma *hcomplex-hcnj-hcnj* [simp]: $\bigwedge z. \text{hcnj } (\text{hcnj } z) = z$

by *transfer* (rule *complex-cnj-cnj*)

lemma *hcomplex-hcnj-hcomplex-of-hypreal* [simp]:
 $\bigwedge x. \text{hcnj } (\text{hcomplex-of-hypreal } x) = \text{hcomplex-of-hypreal } x$

by *transfer* (rule *complex-cnj-complex-of-real*)

lemma *hcomplex-hmod-hcnj* [simp]: $\bigwedge z. \text{hcm}od (\text{hcnj } z) = \text{hcm}od \ z$

by *transfer* (rule *complex-mod-cnj*)

lemma *hcomplex-hcnj-minus*: $\bigwedge z. \text{hcnj } (- \ z) = - \ \text{hcnj } z$

by *transfer* (rule *complex-cnj-minus*)

lemma *hcomplex-hcnj-inverse*: $\bigwedge z. \text{hcnj } (\text{inverse } z) = \text{inverse } (\text{hcnj } z)$

by *transfer* (rule *complex-cnj-inverse*)

lemma *hcomplex-hcnj-add*: $\bigwedge w \ z. \text{hcnj } (w + z) = \text{hcnj } w + \text{hcnj } z$

by *transfer* (rule *complex-cnj-add*)

lemma *hcomplex-hcnj-diff*: $\bigwedge w \ z. \text{hcnj } (w - z) = \text{hcnj } w - \text{hcnj } z$

by *transfer* (rule *complex-cnj-diff*)

lemma *hcomplex-hcnj-mult*: $\bigwedge w \ z. \text{hcnj } (w * z) = \text{hcnj } w * \text{hcnj } z$

by *transfer* (rule *complex-cnj-mult*)

lemma *hcomplex-hcnj-divide*: $\bigwedge w \ z. \text{hcnj } (w / z) = \text{hcnj } w / \text{hcnj } z$

by *transfer* (rule *complex-cnj-divide*)

lemma *hcnj-one* [simp]: $\text{hcnj } 1 = 1$

by *transfer* (rule *complex-cnj-one*)

lemma *hcomplex-hcnj-zero* [simp]: $\text{hcnj } 0 = 0$

by *transfer* (rule *complex-cnj-zero*)

lemma *hcomplex-hcnj-zero-iff* [iff]: $\bigwedge z. \text{hcnj } z = 0 \longleftrightarrow z = 0$

by *transfer* (rule *complex-cnj-zero-iff*)

lemma *hcomplex-mult-hcnj*: $\bigwedge z. z * \text{hcnj } z = \text{hcomplex-of-hypreal } ((\text{hRe } z)^2 + (\text{hIm } z)^2)$
by *transfer (rule complex-mult-cnj)*

7.10 More Theorems about the Function *hcm*

lemma *hcm-hcomplex-of-hypreal-of-nat [simp]*:
 $\text{hcm} (\text{hcomplex-of-hypreal } (\text{hypreal-of-nat } n)) = \text{hypreal-of-nat } n$
by *simp*

lemma *hcm-hcomplex-of-hypreal-of-hypnat [simp]*:
 $\text{hcm} (\text{hcomplex-of-hypreal}(\text{hypreal-of-hypnat } n)) = \text{hypreal-of-hypnat } n$
by *simp*

lemma *hcm-mult-hcnj*: $\bigwedge z. \text{hcm} (z * \text{hcnj } z) = (\text{hcm } z)^2$
by *transfer (rule complex-mod-mult-cnj)*

lemma *hcm-triangle-ineq2 [simp]*: $\bigwedge a b. \text{hcm} (b + a) - \text{hcm } b \leq \text{hcm } a$
by *transfer (rule complex-mod-triangle-ineq2)*

lemma *hcm-diff-ineq [simp]*: $\bigwedge a b. \text{hcm } a - \text{hcm } b \leq \text{hcm} (a + b)$
by *transfer (rule norm-diff-ineq)*

7.11 Exponentiation

lemma *hcomplexpow-0 [simp]*: $z \wedge 0 = 1$
for $z :: \text{hcomplex}$
by *(rule power-0)*

lemma *hcomplexpow-Suc [simp]*: $z \wedge (\text{Suc } n) = z * (z \wedge n)$
for $z :: \text{hcomplex}$
by *(rule power-Suc)*

lemma *hcomplexpow-i-squared [simp]*: $i^2 = -1$
by *transfer (rule power2-i)*

lemma *hcomplex-of-hypreal-pow*: $\bigwedge x. \text{hcomplex-of-hypreal } (x \wedge n) = \text{hcomplex-of-hypreal } x \wedge n$
by *transfer (rule of-real-power)*

lemma *hcomplex-hcnj-pow*: $\bigwedge z. \text{hcnj } (z \wedge n) = \text{hcnj } z \wedge n$
by *transfer (rule complex-cnj-power)*

lemma *hcm-hcomplexpow*: $\bigwedge x. \text{hcm} (x \wedge n) = \text{hcm } x \wedge n$
by *transfer (rule norm-power)*

lemma *hcpow-minus*:
 $\bigwedge x n. (- x :: \text{hcomplex}) \text{ pow } n = (\text{if } (*p* \text{ even}) \text{ then } (x \text{ pow } n) \text{ else } - (x \text{ pow } n))$

by *transfer simp*

lemma *hcpow-mult*: $(r * s) \text{ pow } n = (r \text{ pow } n) * (s \text{ pow } n)$
 for $r s :: \text{hcomplex}$
 by (*fact hyperpow-mult*)

lemma *hcpow-zero2* [*simp*]: $\bigwedge n. 0 \text{ pow } (\text{hSuc } n) = (0 :: 'a :: \text{semiring-1 star})$
 by *transfer (rule power-0-Suc)*

lemma *hcpow-not-zero* [*simp,intro*]: $\bigwedge r n. r \neq 0 \implies r \text{ pow } n \neq (0 :: \text{hcomplex})$
 by (*fact hyperpow-not-zero*)

lemma *hcpow-zero-zero*: $r \text{ pow } n = 0 \implies r = 0$
 for $r :: \text{hcomplex}$
 by (*blast intro: ccontr dest: hcpow-not-zero*)

7.12 The Function *hsgn*

lemma *hsgn-zero* [*simp*]: $\text{hsgn } 0 = 0$
 by *transfer (rule sgn-zero)*

lemma *hsgn-one* [*simp*]: $\text{hsgn } 1 = 1$
 by *transfer (rule sgn-one)*

lemma *hsgn-minus*: $\bigwedge z. \text{hsgn } (- z) = - \text{hsgn } z$
 by *transfer (rule sgn-minus)*

lemma *hsgn-eq*: $\bigwedge z. \text{hsgn } z = z / \text{hcomplex-of-hypreal } (\text{hcm} \text{ mod } z)$
 by *transfer (rule sgn-eq)*

lemma *hcmmod-i*: $\bigwedge x y. \text{hcm} \text{ mod } (\text{HComplex } x y) = (*f* \text{ sqrt}) (x^2 + y^2)$
 by *transfer (rule complex-norm)*

lemma *hcomplex-eq-cancel-iff1* [*simp*]:
 $\text{hcomplex-of-hypreal } xa = \text{HComplex } x y \longleftrightarrow xa = x \wedge y = 0$
 by (*simp add: hcomplex-of-hypreal-eq*)

lemma *hcomplex-eq-cancel-iff2* [*simp*]:
 $\text{HComplex } x y = \text{hcomplex-of-hypreal } xa \longleftrightarrow x = xa \wedge y = 0$
 by (*simp add: hcomplex-of-hypreal-eq*)

lemma *HComplex-eq-0* [*simp*]: $\bigwedge x y. \text{HComplex } x y = 0 \longleftrightarrow x = 0 \wedge y = 0$
 by *transfer (rule Complex-eq-0)*

lemma *HComplex-eq-1* [*simp*]: $\bigwedge x y. \text{HComplex } x y = 1 \longleftrightarrow x = 1 \wedge y = 0$
 by *transfer (rule Complex-eq-1)*

lemma *i-eq-HComplex-0-1*: $iii = \text{HComplex } 0 1$
 by *transfer (simp add: complex-eq-iff)*

lemma *HComplex-eq-i* [simp]: $\bigwedge x y. HComplex\ x\ y = iii \longleftrightarrow x = 0 \wedge y = 1$
by transfer (rule *Complex-eq-i*)

lemma *hRe-hsgn* [simp]: $\bigwedge z. hRe\ (hsgn\ z) = hRe\ z\ /\ hmod\ z$
by transfer (rule *Re-sgn*)

lemma *hIm-hsgn* [simp]: $\bigwedge z. hIm\ (hsgn\ z) = hIm\ z\ /\ hmod\ z$
by transfer (rule *Im-sgn*)

lemma *HComplex-inverse*: $\bigwedge x y. inverse\ (HComplex\ x\ y) = HComplex\ (x\ /\ (x^2 + y^2))\ (-\ y\ /\ (x^2 + y^2))$
by transfer (rule *complex-inverse*)

lemma *hRe-mult-i-eq* [simp]: $\bigwedge y. hRe\ (iii * hcomplex-of-hypreal\ y) = 0$
by transfer simp

lemma *hIm-mult-i-eq* [simp]: $\bigwedge y. hIm\ (iii * hcomplex-of-hypreal\ y) = y$
by transfer simp

lemma *hmod-mult-i* [simp]: $\bigwedge y. hmod\ (iii * hcomplex-of-hypreal\ y) = |y|$
by transfer (simp add: *norm-complex-def*)

lemma *hmod-mult-i2* [simp]: $\bigwedge y. hmod\ (hcomplex-of-hypreal\ y * iii) = |y|$
by transfer (simp add: *norm-complex-def*)

7.12.1 harg

lemma *cos-harg-i-mult-zero* [simp]: $\bigwedge y. y \neq 0 \implies (*f* cos)\ (harg\ (HComplex\ 0\ y)) = 0$
by transfer (simp add: *Complex-eq*)

7.13 Polar Form for Nonstandard Complex Numbers

lemma *complex-split-polar2*: $\forall n. \exists r a. (z\ n) = complex-of-real\ r * Complex\ (cos\ a)\ (sin\ a)$
unfolding *Complex-eq* **by** (auto intro: *complex-split-polar*)

lemma *hcomplex-split-polar*:
 $\bigwedge z. \exists r a. z = hcomplex-of-hypreal\ r * (HComplex\ ((*f* cos)\ a)\ ((*f* sin)\ a))$
by transfer (simp add: *Complex-eq* *complex-split-polar*)

lemma *hcis-eq*:
 $\bigwedge a. hcis\ a = hcomplex-of-hypreal\ ((*f* cos)\ a) + iii * hcomplex-of-hypreal\ ((*f* sin)\ a)$
by transfer (simp add: *complex-eq-iff*)

lemma *hrcis-Ex*: $\bigwedge z. \exists r a. z = hrcis\ r\ a$
by transfer (rule *rcis-Ex*)

lemma *hRe-hcomplex-polar [simp]*:

$\bigwedge r a. hRe (hcomplex-of-hypreal r * HComplex ((*f* cos) a) ((*f* sin) a)) = r$
 $* (*f* cos) a$
by *transfer simp*

lemma *hRe-hrcis [simp]*: $\bigwedge r a. hRe (hrcis r a) = r * (*f* cos) a$
by *transfer (rule Re-rcis)*

lemma *hIm-hcomplex-polar [simp]*:

$\bigwedge r a. hIm (hcomplex-of-hypreal r * HComplex ((*f* cos) a) ((*f* sin) a)) = r$
 $* (*f* sin) a$
by *transfer simp*

lemma *hIm-hrcis [simp]*: $\bigwedge r a. hIm (hrcis r a) = r * (*f* sin) a$
by *transfer (rule Im-rcis)*

lemma *hcmmod-unit-one [simp]*: $\bigwedge a. hcmmod (HComplex ((*f* cos) a) ((*f* sin) a)) = 1$
by *transfer (simp add: cmod-unit-one)*

lemma *hcmmod-complex-polar [simp]*:

$\bigwedge r a. hcmmod (hcomplex-of-hypreal r * HComplex ((*f* cos) a) ((*f* sin) a))$
 $= |r|$
by *transfer (simp add: Complex-eq cmod-complex-polar)*

lemma *hcmmod-hrcis [simp]*: $\bigwedge r a. hcmmod(hrcis r a) = |r|$
by *transfer (rule complex-mod-rcis)*

$(r1 * hrcis a) * (r2 * hrcis b) = r1 * r2 * hrcis (a + b)$

lemma *hcis-hrcis-eq*: $\bigwedge a. hcis a = hrcis 1 a$
by *transfer (rule cis-rcis-eq)*

declare *hcis-hrcis-eq [symmetric, simp]*

lemma *hrcis-mult*: $\bigwedge a b r1 r2. hrcis r1 a * hrcis r2 b = hrcis (r1 * r2) (a + b)$
by *transfer (rule rcis-mult)*

lemma *hcis-mult*: $\bigwedge a b. hcis a * hcis b = hcis (a + b)$
by *transfer (rule cis-mult)*

lemma *hcis-zero [simp]*: $hcis 0 = 1$
by *transfer (rule cis-zero)*

lemma *hrcis-zero-mod [simp]*: $\bigwedge a. hrcis 0 a = 0$
by *transfer (rule rcis-zero-mod)*

lemma *hrcis-zero-arg [simp]*: $\bigwedge r. hrcis r 0 = hcomplex-of-hypreal r$
by *transfer (rule rcis-zero-arg)*

lemma *hcomplex-i-mult-minus [simp]*: $\bigwedge x. iii * (iii * x) = - x$

by *transfer (rule complex-i-mult-minus)*

lemma *hcomplex-i-mult-minus2 [simp]: $iii * iii * x = - x$*
by *simp*

lemma *hcis-hypreal-of-nat-Suc-mult:*
 $\bigwedge a. hcis (hypreal-of-nat (Suc n) * a) = hcis a * hcis (hypreal-of-nat n * a)$
by *transfer (simp add: distrib-right cis-mult)*

lemma *NSDeMoivre:* $\bigwedge a. (hcis a) ^ n = hcis (hypreal-of-nat n * a)$
by *transfer (rule DeMoivre)*

lemma *hcis-hypreal-of-hypnat-Suc-mult:*
 $\bigwedge a n. hcis (hypreal-of-hypnat (n + 1) * a) = hcis a * hcis (hypreal-of-hypnat n * a)$
by *transfer (simp add: distrib-right cis-mult)*

lemma *NSDeMoivre-ext:* $\bigwedge a n. (hcis a) pow n = hcis (hypreal-of-hypnat n * a)$
by *transfer (rule DeMoivre)*

lemma *NSDeMoivre2:* $\bigwedge a r. (hrcis r a) ^ n = hrcis (r ^ n) (hypreal-of-nat n * a)$
by *transfer (rule DeMoivre2)*

lemma *DeMoivre2-ext:* $\bigwedge a r n. (hrcis r a) pow n = hrcis (r pow n) (hypreal-of-hypnat n * a)$
by *transfer (rule DeMoivre2)*

lemma *hcis-inverse [simp]: $\bigwedge a. inverse (hcis a) = hcis (- a)$*
by *transfer (rule cis-inverse)*

lemma *hrcis-inverse:* $\bigwedge a r. inverse (hrcis r a) = hrcis (inverse r) (- a)$
by *transfer (simp add: rcis-inverse inverse-eq-divide [symmetric])*

lemma *hRe-hcis [simp]: $\bigwedge a. hRe (hcis a) = (*f* cos) a$*
by *transfer simp*

lemma *hIm-hcis [simp]: $\bigwedge a. hIm (hcis a) = (*f* sin) a$*
by *transfer simp*

lemma *cos-n-hRe-hcis-pow-n:* $(*f* cos) (hypreal-of-nat n * a) = hRe (hcis a ^ n)$
by *(simp add: NSDeMoivre)*

lemma *sin-n-hIm-hcis-pow-n:* $(*f* sin) (hypreal-of-nat n * a) = hIm (hcis a ^ n)$
by *(simp add: NSDeMoivre)*

lemma *cos-n-hRe-hcis-hcpow-n:* $(*f* cos) (hypreal-of-hypnat n * a) = hRe (hcis a pow n)$
by *(simp add: NSDeMoivre-ext)*

lemma *sin-n-hIm-hcis-hcpow-n*: ($*f*$ *sin*) (*hypreal-of-hypnat* $n * a$) = *hIm* (*hcis* $a \text{ pow } n$)

by (*simp add: NSDeMoivre-ext*)

lemma *hExp-add*: $\bigwedge a b. \text{hExp } (a + b) = \text{hExp } a * \text{hExp } b$

by *transfer (rule exp-add)*

7.14 *hcomplex-of-complex*: the Injection from type *complex* to *hcomplex*

lemma *hcomplex-of-complex-i*: $iii = \text{hcomplex-of-complex } i$

by (*rule iii-def*)

lemma *hRe-hcomplex-of-complex*: $\text{hRe } (\text{hcomplex-of-complex } z) = \text{hypreal-of-real } (\text{Re } z)$

by *transfer (rule refl)*

lemma *hIm-hcomplex-of-complex*: $\text{hIm } (\text{hcomplex-of-complex } z) = \text{hypreal-of-real } (\text{Im } z)$

by *transfer (rule refl)*

lemma *hcmmod-hcomplex-of-complex*: $\text{hcmmod } (\text{hcomplex-of-complex } x) = \text{hypreal-of-real } (\text{cmmod } x)$

by *transfer (rule refl)*

7.15 Numerals and Arithmetic

lemma *hcomplex-of-hypreal-eq-hcomplex-of-complex*:

$\text{hcomplex-of-hypreal } (\text{hypreal-of-real } x) = \text{hcomplex-of-complex } (\text{complex-of-real } x)$

by *transfer (rule refl)*

lemma *hcomplex-hypreal-numeral*:

$\text{hcomplex-of-complex } (\text{numeral } w) = \text{hcomplex-of-hypreal}(\text{numeral } w)$

by *transfer (rule of-real-numeral [symmetric])*

lemma *hcomplex-hypreal-neg-numeral*:

$\text{hcomplex-of-complex } (- \text{numeral } w) = \text{hcomplex-of-hypreal}(- \text{numeral } w)$

by *transfer (rule of-real-neg-numeral [symmetric])*

lemma *hcomplex-numeral-hcnj* [*simp*]: $\text{hcnj } (\text{numeral } v :: \text{hcomplex}) = \text{numeral } v$

by *transfer (rule complex-cnj-numeral)*

lemma *hcomplex-numeral-hcmmod* [*simp*]: $\text{hcmmod } (\text{numeral } v :: \text{hcomplex}) = (\text{numeral } v :: \text{hypreal})$

by *transfer (rule norm-numeral)*

lemma *hcomplex-neg-numeral-hcmmod* [*simp*]: $\text{hcmmod } (- \text{numeral } v :: \text{hcomplex}) = (\text{numeral } v :: \text{hypreal})$

by *transfer (rule norm-neg-numeral)*

```

lemma hcomplex-numeral-hRe [simp]: hRe (numeral v :: hcomplex) = numeral v
  by transfer (rule complex-Re-numeral)

lemma hcomplex-numeral-hIm [simp]: hIm (numeral v :: hcomplex) = 0
  by transfer (rule complex-Im-numeral)

end

```

8 Star-Transforms in Non-Standard Analysis

```

theory Star
  imports NSA
begin

```

```

definition — internal sets
  starset-n :: (nat  $\Rightarrow$  'a set)  $\Rightarrow$  'a star set
    ( $\langle \langle \langle \text{open-block notation} = \langle \text{prefix starset-}n \rangle \rangle *sn* - \rangle \rangle$  [80] 80)
  where  $*sn* As = Iset (star-n As)$ 

```

```

definition InternalSets :: 'a star set set
  where InternalSets = {X.  $\exists As. X = *sn* As$ }

```

```

definition — nonstandard extension of function
  is-starext :: ('a star  $\Rightarrow$  'a star)  $\Rightarrow$  ('a  $\Rightarrow$  'a)  $\Rightarrow$  bool
  where is-starext F  $\longleftrightarrow$ 
    ( $\forall x y. \exists X \in Rep\text{-}star\ x. \exists Y \in Rep\text{-}star\ y. y = F\ x \longleftrightarrow eventually (\lambda n. Y\ n = f(X\ n))\ \mathcal{U}$ )

```

```

definition — internal functions
  starfun-n :: (nat  $\Rightarrow$  'a  $\Rightarrow$  'b)  $\Rightarrow$  'a star  $\Rightarrow$  'b star
    ( $\langle \langle \langle \text{open-block notation} = \langle \text{prefix starfun-}n \rangle \rangle *fn* - \rangle \rangle$  [80] 80)
  where  $*fn* F = Ifun (star-n F)$ 

```

```

definition InternalFuns :: ('a star  $\Rightarrow$  'b star) set
  where InternalFuns = {X.  $\exists F. X = *fn* F$ }

```

8.1 Preamble - Pulling \exists over \forall

This proof does not need AC and was suggested by the referee for the JCM Paper: let $f\ x$ be least y such that $Q\ x\ y$.

```

lemma no-choice:  $\forall x. \exists y. Q\ x\ y \implies \exists f :: 'a \Rightarrow nat. \forall x. Q\ x\ (f\ x)$ 
  by (rule exI [where  $x = \lambda x. LEAST\ y. Q\ x\ y$ ]) (blast intro: LeastI)

```

8.2 Properties of the Star-transform Applied to Sets of Reals

```

lemma STAR-star-of-image-subset: star-of 'A  $\subseteq$   $*s*$  A
  by auto

```

lemma *STAR-hypreal-of-real-Int*: $*s* X \cap \mathbb{R} = \text{hypreal-of-real } ' X$
by (*auto simp add: SReal-def*)

lemma *STAR-star-of-Int*: $*s* X \cap \text{Standard} = \text{star-of } ' X$
by (*auto simp add: Standard-def*)

lemma *lemma-not-hyprealA*: $x \notin \text{hypreal-of-real } ' A \implies \forall y \in A. x \neq \text{hypreal-of-real } y$
by *auto*

lemma *lemma-not-starA*: $x \notin \text{star-of } ' A \implies \forall y \in A. x \neq \text{star-of } y$
by *auto*

lemma *STAR-real-seq-to-hypreal*: $\forall n. (X n) \notin M \implies \text{star-n } X \notin *s* M$
by (*simp add: starset-def star-of-def Iset-star-n FreeUltrafilterNat.proper*)

lemma *STAR-singleton*: $*s* \{x\} = \{\text{star-of } x\}$
by *simp*

lemma *STAR-not-mem*: $x \notin F \implies \text{star-of } x \notin *s* F$
by *transfer*

lemma *STAR-subset-closed*: $x \in *s* A \implies A \subseteq B \implies x \in *s* B$
by (*erule rev-subsetD*) *simp*

Nonstandard extension of a set (defined using a constant sequence) as a special case of an internal set.

lemma *starset-n-starset*: $\forall n. As n = A \implies *sn* As = *s* A$
by (*drule fun-eq-iff [THEN iffD2]*) (*simp add: starset-n-def starset-def star-of-def*)

8.3 Theorems about nonstandard extensions of functions

Nonstandard extension of a function (defined using a constant sequence) as a special case of an internal function.

lemma *starfun-n-starfun*: $F = (\lambda n. f) \implies *fn* F = *f* f$
by (*simp add: starfun-n-def starfun-def star-of-def*)

Prove that *abs* for hypreal is a nonstandard extension of *abs* for real w/o use of congruence property (proved after this for general nonstandard extensions of real valued functions).

Proof now Uses the ultrafilter tactic!

lemma *hrabs-is-starext-rabs*: *is-starext abs abs*

proof –

have $\exists f \in \text{Rep-star } (\text{star-n } h). \exists g \in \text{Rep-star } (\text{star-n } k). (\text{star-n } k = |\text{star-n } h|) =$
 $(\forall_F n \text{ in } \mathcal{U}. (g n :: 'a) = |f n|)$
for $x y :: 'a \text{ star}$ **and** $h k$

by (metis (full-types) Rep-star-star-n star-n-abs star-n-eq-iff)
 then show ?thesis
 unfolding is-starext-def by (metis star-cases)
 qed

Nonstandard extension of functions.

lemma starfun: ($*f*$ f) ($star\text{-}n$ X) = $star\text{-}n$ ($\lambda n.$ f (X n))
 by (rule starfun-star-n)

lemma starfun-if-eq: $\bigwedge w. w \neq star\text{-}of\ x \implies (*f* (\lambda z. \text{if } z = x \text{ then } a \text{ else } g\ z))$
 $w = (*f* g)\ w$
 by transfer simp

Multiplication: ($*f$) x ($*g$) = $*(f\ x\ g)$

lemma starfun-mult: $\bigwedge x. (*f* f)\ x * (*f* g)\ x = (*f* (\lambda x. f\ x * g\ x))\ x$
 by transfer (rule refl)
declare starfun-mult [symmetric, simp]

Addition: ($*f$) + ($*g$) = $*(f + g)$

lemma starfun-add: $\bigwedge x. (*f* f)\ x + (*f* g)\ x = (*f* (\lambda x. f\ x + g\ x))\ x$
 by transfer (rule refl)
declare starfun-add [symmetric, simp]

Subtraction: ($*f$) + $-(*g)$ = $*(f + -g)$

lemma starfun-minus: $\bigwedge x. - (*f* f)\ x = (*f* (\lambda x. - f\ x))\ x$
 by transfer (rule refl)
declare starfun-minus [symmetric, simp]

lemma starfun-add-minus: $\bigwedge x. (*f* f)\ x + - (*f* g)\ x = (*f* (\lambda x. f\ x + -g\ x))\ x$
 by transfer (rule refl)
declare starfun-add-minus [symmetric, simp]

lemma starfun-diff: $\bigwedge x. (*f* f)\ x - (*f* g)\ x = (*f* (\lambda x. f\ x - g\ x))\ x$
 by transfer (rule refl)
declare starfun-diff [symmetric, simp]

Composition: ($*f$) \circ ($*g$) = $*(f \circ g)$

lemma starfun-o2: $(\lambda x. (*f* f)\ ((*f* g)\ x)) = *f* (\lambda x. f\ (g\ x))$
 by transfer (rule refl)

lemma starfun-o: ($*f* f$) \circ ($*f* g$) = ($*f* (f \circ g)$)
 by (transfer o-def) (rule refl)

NS extension of constant function.

lemma starfun-const-fun [simp]: $\bigwedge x. (*f* (\lambda x. k))\ x = star\text{-}of\ k$
 by transfer (rule refl)

The NS extension of the identity function.

lemma *starfun-Id* [*simp*]: $\bigwedge x. (*f* (\lambda x. x)) x = x$
by *transfer (rule refl)*

The Star-function is a (nonstandard) extension of the function.

lemma *is-starext-starfun*: *is-starext* ($*f* f$) *f*
proof –
have $\exists X \in \text{Rep-star } x. \exists Y \in \text{Rep-star } y. (y = (*f* f) x) = (\forall_F n \text{ in } \mathcal{U}. Y n = f (X n))$
for $x y$
by (*metis (mono-tags) Rep-star-star-n star-cases star-n-eq-iff starfun-star-n*)
then show *?thesis*
by (*auto simp: is-starext-def*)
qed

Any nonstandard extension is in fact the Star-function.

lemma *is-starfun-starext*:
assumes *is-starext* $F f$
shows $F = *f* f$
proof –
have $F x = (*f* f) x$
if $\forall x y. \exists X \in \text{Rep-star } x. \exists Y \in \text{Rep-star } y. (y = F x) = (\forall_F n \text{ in } \mathcal{U}. Y n = f (X n))$ **for** x
by (*metis that mem-Rep-star-iff star-n-eq-iff starfun-star-n*)
with *assms* **show** *?thesis*
by (*force simp add: is-starext-def*)
qed

lemma *is-starext-starfun-iff*: *is-starext* $F f \longleftrightarrow F = *f* f$
by (*blast intro: is-starfun-starext is-starext-starfun*)

Extended function has same solution as its standard version for real arguments. i.e they are the same for all real arguments.

lemma *starfun-eq*: $(*f* f) (\text{star-of } a) = \text{star-of } (f a)$
by (*rule starfun-star-of*)

lemma *starfun-approx*: $(*f* f) (\text{star-of } a) \approx \text{star-of } (f a)$
by *simp*

Useful for NS definition of derivatives.

lemma *starfun-lambda-cancel*: $\bigwedge x'. (*f* (\lambda h. f (x + h))) x' = (*f* f) (\text{star-of } x + x')$
by *transfer (rule refl)*

lemma *starfun-lambda-cancel2*: $(*f* (\lambda h. f (g (x + h)))) x' = (*f* (f \circ g)) (\text{star-of } x + x')$
unfolding *o-def* **by** (*rule starfun-lambda-cancel*)

lemma *starfun-mult-HFinite-approx*:

$(*f* f) x \approx l \implies (*f* g) x \approx m \implies l \in HFinite \implies m \in HFinite \implies$
 $(*f* (\lambda x. f x * g x)) x \approx l * m$
for $l m :: 'a :: \text{real-normed-algebra star}$
using *approx-mult-HFinite* **by** *auto*

lemma *starfun-add-approx*: $(*f* f) x \approx l \implies (*f* g) x \approx m \implies (*f* (\%x. f x + g x)) x \approx l + m$
by (*auto intro: approx-add*)

Examples: *hrabs* is nonstandard extension of *rabs*, *inverse* is nonstandard extension of *inverse*.

Can be proved easily using theorem *starfun* and properties of ultrafilter as for *inverse* below we use the theorem we proved above instead.

lemma *starfun-rabs-hrabs*: $*f* abs = abs$
by (*simp only: star-abs-def*)

lemma *starfun-inverse-inverse* [*simp*]: $(*f* inverse) x = inverse x$
by (*simp only: star-inverse-def*)

lemma *starfun-inverse*: $\bigwedge x. inverse ((*f* f) x) = (*f* (\lambda x. inverse (f x))) x$
by *transfer (rule refl)*
declare *starfun-inverse* [*symmetric, simp*]

lemma *starfun-divide*: $\bigwedge x. (*f* f) x / (*f* g) x = (*f* (\lambda x. f x / g x)) x$
by *transfer (rule refl)*
declare *starfun-divide* [*symmetric, simp*]

lemma *starfun-inverse2*: $\bigwedge x. inverse ((*f* f) x) = (*f* (\lambda x. inverse (f x))) x$
by *transfer (rule refl)*

General lemma/theorem needed for proofs in elementary topology of the reals.

lemma *starfun-mem-starset*: $\bigwedge x. (*f* f) x \in *s* A \implies x \in *s* \{x. f x \in A\}$
by *transfer simp*

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*: $|star-n X| = star-n (\lambda n. |X n|)$
by (*simp only: starfun-rabs-hrabs [symmetric] starfun*)

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 extension. See second NS set extension below.

lemma *STAR-rabs-add-minus*: $*s* \{x. |x + - y| < r\} = \{x. |x + -star-of y| < star-of r\}$

by *transfer* (*rule refl*)

lemma *STAR-starfun-rabs-add-minus*:

$*s* \{x. |f x + - y| < r\} = \{x. |(*f* f) x + -star-of y| < star-of r\}$

by *transfer* (*rule refl*)

Another characterization of Infinitesimal and one of \approx relation. In this theory since *hypreal-hrabs* proved here. Maybe move both theorems??

lemma *Infinitesimal-FreeUltrafilterNat-iff2*:

$star-n X \in Infinitesimal \longleftrightarrow (\forall m. eventually (\lambda n. norm (X n) < inverse (real (Suc m))) \mathcal{U})$

by (*simp add: Infinitesimal-hypreal-of-nat-iff star-of-def hnrm-def star-of-nat-def starfun-star-n star-n-inverse star-n-less*)

lemma *HNatInfinite-inverse-Infinitesimal* [*simp*]:

assumes $n \in HNatInfinite$

shows $inverse (hypreal-of-hypnat n) \in Infinitesimal$

proof (*cases n*)

case (*star-n X*)

then have $*$: $\bigwedge k. \forall_F n \text{ in } \mathcal{U}. k < X n$

using *HNatInfinite-FreeUltrafilterNat assms* **by** *blast*

have $\forall_F n \text{ in } \mathcal{U}. inverse (real (X n)) < inverse (1 + real m)$ **for** m

using $*$ [*of Suc m*] **by** (*auto elim!: eventually-mono*)

then show *?thesis*

using *star-n* **by** (*auto simp: of-hypnat-def starfun-star-n star-n-inverse Infinitesimal-FreeUltrafilterNat-iff2*)

qed

lemma *approx-FreeUltrafilterNat-iff*:

$star-n X \approx star-n Y \longleftrightarrow (\forall r > 0. eventually (\lambda n. norm (X n - Y n) < r) \mathcal{U})$

(*is ?lhs = ?rhs*)

proof –

have *?lhs* = (*star-n X* – *star-n Y* ≈ 0)

using *approx-minus-iff* **by** *blast*

also have ... = *?rhs*

by (*metis (full-types) Infinitesimal-FreeUltrafilterNat-iff mem-infmal-iff star-n-diff*)

finally show *?thesis* .

qed

lemma *approx-FreeUltrafilterNat-iff2*:

$star-n X \approx star-n Y \longleftrightarrow (\forall m. eventually (\lambda n. norm (X n - Y n) < inverse (real (Suc m))) \mathcal{U})$

(*is ?lhs = ?rhs*)

proof –

have *?lhs* = (*star-n X* – *star-n Y* ≈ 0)

using *approx-minus-iff* **by** *blast*

also have ... = *?rhs*

by (*metis* (*full-types*) *Infinitesimal-FreeUltrafilterNat-iff2 mem-infmal-iff star-n-diff*)
 finally show ?thesis .
 qed

lemma *inj-starfun*: *inj starfun*
 proof (rule *inj-onI*)
 show $\varphi = \psi$ if eq: $*f* \varphi = *f* \psi$ for $\varphi \psi :: 'a \Rightarrow 'b$
 proof (rule *ext*, rule *ccontr*)
 show *False*
 if $\varphi x \neq \psi x$ for x
 by (*metis* eq that *star-of-inject starfun-eq*)
 qed
 qed
 end

9 Star-transforms for the Hypernaturals

theory *NatStar*
 imports *Star*
 begin

lemma *star-n-eq-starfun-whn*: $\text{star-}n\ X = (*f*\ X)$ whn
 by (*simp* add: *hypnat-omega-def starfun-def star-of-def Ifun-star-n*)

lemma *starset-n-Un*: $*sn* (\lambda n. (A\ n) \cup (B\ n)) = *sn* A \cup *sn* B$
 proof –
 have $\bigwedge N. \text{Iset} ((*f* (\lambda n. \{x. x \in A\ n \vee x \in B\ n\}))\ N) =$
 $\{x. x \in \text{Iset} ((*f* A)\ N) \vee x \in \text{Iset} ((*f* B)\ N)\}$
 by *transfer simp*
 then show ?thesis
 by (*simp* add: *starset-n-def star-n-eq-starfun-whn Un-def*)
 qed

lemma *InternalSets-Un*: $X \in \text{InternalSets} \implies Y \in \text{InternalSets} \implies X \cup Y \in \text{InternalSets}$
 by (*auto simp* add: *InternalSets-def starset-n-Un [symmetric]*)

lemma *starset-n-Int*: $*sn* (\lambda n. A\ n \cap B\ n) = *sn* A \cap *sn* B$
 proof –
 have $\bigwedge N. \text{Iset} ((*f* (\lambda n. \{x. x \in A\ n \wedge x \in B\ n\}))\ N) =$
 $\{x. x \in \text{Iset} ((*f* A)\ N) \wedge x \in \text{Iset} ((*f* B)\ N)\}$
 by *transfer simp*
 then show ?thesis
 by (*simp* add: *starset-n-def star-n-eq-starfun-whn Int-def*)
 qed

lemma *InternalSets-Int*: $X \in \text{InternalSets} \implies Y \in \text{InternalSets} \implies X \cap Y \in \text{InternalSets}$

by (auto simp add: InternalSets-def starset-n-Int [symmetric])

lemma starset-n-Compl: $*sn* ((\lambda n. - A\ n)) = - (*sn* A)$

proof –

have $\bigwedge N. \text{Iset } ((*f* (\lambda n. \{x. x \notin A\ n\}))\ N) =$
 $\{x. x \notin \text{Iset } ((*f* A)\ N)\}$

by transfer simp

then show ?thesis

by (simp add: starset-n-def star-n-eq-starfun-whn Compl-eq)

qed

lemma InternalSets-Compl: $X \in \text{InternalSets} \implies - X \in \text{InternalSets}$

by (auto simp add: InternalSets-def starset-n-Compl [symmetric])

lemma starset-n-diff: $*sn* (\lambda n. (A\ n) - (B\ n)) = *sn* A - *sn* B$

proof –

have $\bigwedge N. \text{Iset } ((*f* (\lambda n. \{x. x \in A\ n \wedge x \notin B\ n\}))\ N) =$
 $\{x. x \in \text{Iset } ((*f* A)\ N) \wedge x \notin \text{Iset } ((*f* B)\ N)\}$

by transfer simp

then show ?thesis

by (simp add: starset-n-def star-n-eq-starfun-whn set-diff-eq)

qed

lemma InternalSets-diff: $X \in \text{InternalSets} \implies Y \in \text{InternalSets} \implies X - Y \in \text{InternalSets}$

by (auto simp add: InternalSets-def starset-n-diff [symmetric])

lemma NatStar-SHNat-subset: $\text{Nats} \leq *s* (\text{UNIV}:: \text{nat set})$

by simp

lemma NatStar-hypreal-of-real-Int: $*s* X\ \text{Int}\ \text{Nats} = \text{hypnat-of-nat } X$

by (auto simp add: SHNat-eq)

lemma starset-starset-n-eq: $*s* X = *sn* (\lambda n. X)$

by (simp add: starset-n-starset)

lemma InternalSets-starset-n [simp]: $(*s* X) \in \text{InternalSets}$

by (auto simp add: InternalSets-def starset-starset-n-eq)

lemma InternalSets-UNIV-diff: $X \in \text{InternalSets} \implies \text{UNIV} - X \in \text{InternalSets}$

by (simp add: InternalSets-Compl diff-eq)

9.1 Nonstandard Extensions of Functions

Example of transfer of a property from reals to hyperreals — used for limit comparison of sequences.

lemma starfun-le-mono: $\forall n. N \leq n \longrightarrow f\ n \leq g\ n \implies$

$\forall n. \text{hypnat-of-nat } N \leq n \longrightarrow (*f* f)\ n \leq (*f* g)\ n$

by transfer

And another:

lemma *starfun-less-mono*:

$\forall n. N \leq n \longrightarrow f\ n < g\ n \implies \forall n. \text{hypnat-of-nat } N \leq n \longrightarrow (*f* f)\ n < (*f* g)\ n$
by *transfer*

Nonstandard extension when we increment the argument by one.

lemma *starfun-shift-one*: $\bigwedge N. (*f* (\lambda n. f\ (Suc\ n)))\ N = (*f* f)\ (N + (1::\text{hypnat}))$
by *transfer simp*

Nonstandard extension with absolute value.

lemma *starfun-abs*: $\bigwedge N. (*f* (\lambda n. |f\ n|))\ N = |(*f* f)\ N|$
by *transfer (rule refl)*

The *hyperpow* function as a nonstandard extension of *realpow*.

lemma *starfun-pow*: $\bigwedge N. (*f* (\lambda r. r \wedge n))\ N = \text{hypreal-of-real } r\ \text{pow } N$
by *transfer (rule refl)*

lemma *starfun-pow2*: $\bigwedge N. (*f* (\lambda n. X\ n \wedge m))\ N = (*f* X)\ N\ \text{pow } \text{hypnat-of-nat } m$
by *transfer (rule refl)*

lemma *starfun-pow3*: $\bigwedge R. (*f* (\lambda r. r \wedge n))\ R = R\ \text{pow } \text{hypnat-of-nat } n$
by *transfer (rule refl)*

The *hypreal-of-hypnat* function as a nonstandard extension of *real*.

lemma *starfunNat-real-of-nat*: $(*f* \text{real}) = \text{hypreal-of-hypnat}$
by *transfer (simp add: fun-eq-iff)*

lemma *starfun-inverse-real-of-nat-eq*:

$N \in \text{HNatInfinite} \implies (*f* (\lambda x::\text{nat}. \text{inverse } (\text{real } x)))\ N = \text{inverse } (\text{hypreal-of-hypnat } N)$
by *(metis of-hypnat-def starfun-inverse2)*

Internal functions – some redundancy with **f** now.

lemma *starfun-n*: $(*fn* f)\ (\text{star-n } X) = \text{star-n } (\lambda n. f\ n\ (X\ n))$
by *(simp add: starfun-n-def Ifun-star-n)*

Multiplication: $(*fn)\ x\ (*gn) = *(fn\ x\ gn)$

lemma *starfun-n-mult*: $(*fn* f)\ z\ (*fn* g)\ z = (*fn* (\lambda i\ x. f\ i\ x\ * g\ i\ x))\ z$
by *(cases z) (simp add: starfun-n star-n-mult)*

Addition: $(*fn) + (*gn) = *(fn + gn)$

lemma *starfun-n-add*: $(*fn* f)\ z + (*fn* g)\ z = (*fn* (\lambda i\ x. f\ i\ x + g\ i\ x))\ z$
by *(cases z) (simp add: starfun-n star-n-add)*

Subtraction: $(*fn) - (*gn) = *(fn + -\ gn)$

lemma *starfun-n-add-minus*: $(*fn* f) z + - (*fn* g) z = (*fn* (\lambda i x. f i x + -g i x)) z$

by (cases z) (simp add: starfun-n star-n-minus star-n-add)

Composition: $(*fn) \circ (*gn) = *(fn \circ gn)$

lemma *starfun-n-const-fun* [simp]: $(*fn* (\lambda i x. k)) z = \text{star-of } k$

by (cases z) (simp add: starfun-n star-of-def)

lemma *starfun-n-minus*: $- (*fn* f) x = (*fn* (\lambda i x. - (f i) x)) x$

by (cases x) (simp add: starfun-n star-n-minus)

lemma *starfun-n-eq* [simp]: $(*fn* f) (\text{star-of } n) = \text{star-n } (\lambda i. f i n)$

by (simp add: starfun-n star-of-def)

lemma *starfun-eq-iff*: $((*f* f) = (*f* g)) \longleftrightarrow f = g$

by transfer (rule refl)

lemma *starfunNat-inverse-real-of-nat-Infinitesimal* [simp]:

$N \in HNatInfinite \implies (*f* (\lambda x. \text{inverse } (\text{real } x))) N \in Infinitesimal$

using starfun-inverse-real-of-nat-eq **by** auto

9.2 Nonstandard Characterization of Induction

lemma *hypnat-induct-obj*:

$\bigwedge n. ((*p* P) (0::hypnat) \wedge (\forall n. (*p* P) n \longrightarrow (*p* P) (n + 1))) \longrightarrow (*p* P) n$

by transfer (induct-tac n, auto)

lemma *hypnat-induct*:

$\bigwedge n. (*p* P) (0::hypnat) \implies (\bigwedge n. (*p* P) n \implies (*p* P) (n + 1)) \implies (*p* P) n$

by transfer (induct-tac n, auto)

lemma *starP2-eq-iff*: $(*p2* (=)) = (=)$

by transfer (rule refl)

lemma *starP2-eq-iff2*: $(*p2* (\lambda x y. x = y)) X Y \longleftrightarrow X = Y$

by (simp add: starP2-eq-iff)

lemma *nonempty-set-star-has-least-lemma*:

$\exists n \in S. \forall m \in S. n \leq m$ **if** $S \neq \{\}$ **for** $S :: \text{nat set}$

proof

show $\forall m \in S. (\text{LEAST } n. n \in S) \leq m$

by (simp add: Least-le)

show $(\text{LEAST } n. n \in S) \in S$

by (meson that LeastI-ex equals0I)

qed

lemma *nonempty-set-star-has-least*:

$\bigwedge S :: \text{nat set star. } \text{Iset } S \neq \{\} \implies \exists n \in \text{Iset } S. \forall m \in \text{Iset } S. n \leq m$
using *nonempty-set-star-has-least-lemma* **by** (*transfer empty-def*)

lemma *nonempty-InternalNatSet-has-least*: $S \in \text{InternalSets} \implies S \neq \{\} \implies \exists n \in S. \forall m \in S. n \leq m$
for $S :: \text{hypnat set}$
by (*force simp add: InternalSets-def starset-n-def dest!: nonempty-set-star-has-least*)

Goldblatt, page 129 Thm 11.3.2.

lemma *internal-induct-lemma*:
 $\bigwedge X :: \text{nat set star.}$
 $(0 :: \text{hypnat}) \in \text{Iset } X \implies \forall n. n \in \text{Iset } X \longrightarrow n + 1 \in \text{Iset } X \implies \text{Iset } X =$
 $(\text{UNIV} :: \text{hypnat set})$
apply (*transfer UNIV-def*)
apply (*rule equalityI [OF subset-UNIV subsetI]*)
apply (*induct-tac x, auto*)
done

lemma *internal-induct*:
 $X \in \text{InternalSets} \implies (0 :: \text{hypnat}) \in X \implies \forall n. n \in X \longrightarrow n + 1 \in X \implies X =$
 $(\text{UNIV} :: \text{hypnat set})$
apply (*clarsimp simp add: InternalSets-def starset-n-def*)
apply (*erule (1) internal-induct-lemma*)
done

end

10 Sequences and Convergence (Nonstandard)

theory *HSEQ*
imports *Complex-Main NatStar*
abbrevs $---> = \longrightarrow_{NS}$
begin

definition *NSLIMSEQ* :: $(\text{nat} \Rightarrow 'a :: \text{real-normed-vector}) \Rightarrow 'a \Rightarrow \text{bool}$
 $(\langle \langle \text{notation} = \langle \text{mixfix NSLIMSEQ} \rangle \rangle (-) / \longrightarrow_{NS} (-) \rangle [60, 60] 60)$ **where**
 — Nonstandard definition of convergence of sequence
 $X \longrightarrow_{NS} L \longleftrightarrow (\forall N \in \text{HNatInfinite. } (*f* X) N \approx \text{star-of } L)$

definition *nslim* :: $(\text{nat} \Rightarrow 'a :: \text{real-normed-vector}) \Rightarrow 'a$
where $\text{nslim } X = (\text{THE } L. X \longrightarrow_{NS} L)$
 — Nonstandard definition of limit using choice operator

definition *NSconvergent* :: $(\text{nat} \Rightarrow 'a :: \text{real-normed-vector}) \Rightarrow \text{bool}$
where $\text{NSconvergent } X \longleftrightarrow (\exists L. X \longrightarrow_{NS} L)$
 — Nonstandard definition of convergence

definition *NSBseq* :: $(\text{nat} \Rightarrow 'a :: \text{real-normed-vector}) \Rightarrow \text{bool}$

where $NSBseq\ X \longleftrightarrow (\forall N \in HNatInfinite. (*f* X)\ N \in HFinite)$
 — Nonstandard definition for bounded sequence

definition $NSCauchy :: (nat \Rightarrow 'a::real-normed-vector) \Rightarrow bool$
where $NSCauchy\ X \longleftrightarrow (\forall M \in HNatInfinite. \forall N \in HNatInfinite. (*f* X)\ M \approx (*f* X)\ N)$
 — Nonstandard definition

10.1 Limits of Sequences

lemma $NSLIMSEQ-I: (\bigwedge N. N \in HNatInfinite \implies starfun\ X\ N \approx star-of\ L) \implies X \longrightarrow_{NS} L$
by (*simp add: NSLIMSEQ-def*)

lemma $NSLIMSEQ-D: X \longrightarrow_{NS} L \implies N \in HNatInfinite \implies starfun\ X\ N \approx star-of\ L$
by (*simp add: NSLIMSEQ-def*)

lemma $NSLIMSEQ-const: (\lambda n. k) \longrightarrow_{NS} k$
by (*simp add: NSLIMSEQ-def*)

lemma $NSLIMSEQ-add: X \longrightarrow_{NS} a \implies Y \longrightarrow_{NS} b \implies (\lambda n. X\ n + Y\ n) \longrightarrow_{NS} a + b$
by (*auto intro: approx-add simp add: NSLIMSEQ-def*)

lemma $NSLIMSEQ-add-const: f \longrightarrow_{NS} a \implies (\lambda n. f\ n + b) \longrightarrow_{NS} a + b$
by (*simp only: NSLIMSEQ-add NSLIMSEQ-const*)

lemma $NSLIMSEQ-mult: X \longrightarrow_{NS} a \implies Y \longrightarrow_{NS} b \implies (\lambda n. X\ n * Y\ n) \longrightarrow_{NS} a * b$
for $a\ b :: 'a::real-normed-algebra$
by (*auto intro!: approx-mult-HFinite simp add: NSLIMSEQ-def*)

lemma $NSLIMSEQ-minus: X \longrightarrow_{NS} a \implies (\lambda n. - X\ n) \longrightarrow_{NS} - a$
by (*auto simp add: NSLIMSEQ-def*)

lemma $NSLIMSEQ-minus-cancel: (\lambda n. - X\ n) \longrightarrow_{NS} - a \implies X \longrightarrow_{NS} a$
by (*drule NSLIMSEQ-minus simp*)

lemma $NSLIMSEQ-diff: X \longrightarrow_{NS} a \implies Y \longrightarrow_{NS} b \implies (\lambda n. X\ n - Y\ n) \longrightarrow_{NS} a - b$
using $NSLIMSEQ-add$ [of $X\ a - Y - b$] **by** (*simp add: NSLIMSEQ-minus fun-Compl-def*)

lemma $NSLIMSEQ-diff-const: f \longrightarrow_{NS} a \implies (\lambda n. f\ n - b) \longrightarrow_{NS} a - b$
by (*simp add: NSLIMSEQ-diff NSLIMSEQ-const*)

lemma $NSLIMSEQ-inverse: X \longrightarrow_{NS} a \implies a \neq 0 \implies (\lambda n. inverse\ (X\ n))$

$\longrightarrow_{NS} \text{inverse } a$

for $a :: 'a::\text{real-normed-div-algebra}$

by (*simp add: NSLIMSEQ-def star-of-approx-inverse*)

lemma *NSLIMSEQ-mult-inverse*: $X \longrightarrow_{NS} a \implies Y \longrightarrow_{NS} b \implies b \neq 0 \implies (\lambda n. X\ n / Y\ n) \longrightarrow_{NS} a / b$

for $a\ b :: 'a::\text{real-normed-field}$

by (*simp add: NSLIMSEQ-mult NSLIMSEQ-inverse divide-inverse*)

lemma *starfun-hnorm*: $\bigwedge x. \text{hnorm } ((\ast f \ast f) x) = (\ast f \ast (\lambda x. \text{norm } (f x))) x$

by *transfer simp*

lemma *NSLIMSEQ-norm*: $X \longrightarrow_{NS} a \implies (\lambda n. \text{norm } (X\ n)) \longrightarrow_{NS} \text{norm } a$

by (*simp add: NSLIMSEQ-def starfun-hnorm [symmetric] approx-hnorm*)

Uniqueness of limit.

lemma *NSLIMSEQ-unique*: $X \longrightarrow_{NS} a \implies X \longrightarrow_{NS} b \implies a = b$

unfolding *NSLIMSEQ-def*

using *HNatInfinite-wn approx-trans3 star-of-approx-iff* **by** *blast*

lemma *NSLIMSEQ-pow [rule-format]*: $(X \longrightarrow_{NS} a) \longrightarrow ((\lambda n. (X\ n) ^ m) \longrightarrow_{NS} a ^ m)$

for $a :: 'a::\{\text{real-normed-algebra}, \text{power}\}$

by (*induct m*) (*auto intro: NSLIMSEQ-mult NSLIMSEQ-const*)

We can now try and derive a few properties of sequences, starting with the limit comparison property for sequences.

lemma *NSLIMSEQ-le*: $f \longrightarrow_{NS} l \implies g \longrightarrow_{NS} m \implies \exists N. \forall n \geq N. f\ n \leq g\ n \implies l \leq m$

for $l\ m :: \text{real}$

unfolding *NSLIMSEQ-def*

by (*metis HNatInfinite-wn bex-Infinitesimal-iff2 hypnat-of-nat-le-wn hypreal-of-real-le-add-Infinitesimal-c starfun-le-mono*)

lemma *NSLIMSEQ-le-const*: $X \longrightarrow_{NS} r \implies \forall n. a \leq X\ n \implies a \leq r$

for $a\ r :: \text{real}$

by (*erule NSLIMSEQ-le [OF NSLIMSEQ-const]*) *auto*

lemma *NSLIMSEQ-le-const2*: $X \longrightarrow_{NS} r \implies \forall n. X\ n \leq a \implies r \leq a$

for $a\ r :: \text{real}$

by (*erule NSLIMSEQ-le [OF - NSLIMSEQ-const]*) *auto*

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-iff*: $((\lambda n. f\ (Suc\ n)) \longrightarrow_{NS} l) \longleftrightarrow (f \longrightarrow_{NS} l)$

proof

```

assume *:  $f \longrightarrow_{NS} l$ 
show  $(\lambda n. f(Suc\ n)) \longrightarrow_{NS} l$ 
proof (rule NSLIMSEQ-I)
  fix  $N$ 
  assume  $N \in HNatInfinite$ 
  then have  $(*f* f) (N + 1) \approx star-of\ l$ 
    by (simp add: HNatInfinite-add NSLIMSEQ-D *)
  then show  $(*f* (\lambda n. f(Suc\ n)))\ N \approx star-of\ l$ 
    by (simp add: starfun-shift-one)
qed
next
assume *:  $(\lambda n. f(Suc\ n)) \longrightarrow_{NS} l$ 
show  $f \longrightarrow_{NS} l$ 
proof (rule NSLIMSEQ-I)
  fix  $N$ 
  assume  $N \in HNatInfinite$ 
  then have  $(*f* (\lambda n. f(Suc\ n))) (N - 1) \approx star-of\ l$ 
    using * by (simp add: HNatInfinite-diff NSLIMSEQ-D)
  then show  $(*f* f)\ N \approx star-of\ l$ 
    by (simp add:  $\langle N \in HNatInfinite \rangle one-le-HNatInfinite\ starfun-shift-one$ )
qed
qed

```

10.1.1 Equivalence of *LIMSEQ* and *NSLIMSEQ*

lemma *LIMSEQ-NSLIMSEQ*:

```

assumes  $X: X \longrightarrow L$ 
shows  $X \longrightarrow_{NS} L$ 
proof (rule NSLIMSEQ-I)
  fix  $N$ 
  assume  $N: N \in HNatInfinite$ 
  have  $starfun\ X\ N - star-of\ L \in Infinitesimal$ 
  proof (rule InfinitesimalI2)
    fix  $r :: real$ 
    assume  $r: 0 < r$ 
    from LIMSEQ-D [OF  $X\ r$ ] obtain  $no\ where\ \forall n \geq no. norm\ (X\ n - L) < r ..$ 
    then have  $\forall n \geq star-of\ no. hnorm\ (starfun\ X\ n - star-of\ L) < star-of\ r$ 
      by transfer
    then show  $hnorm\ (starfun\ X\ N - star-of\ L) < star-of\ r$ 
      using  $N$  by (simp add: star-of-le-HNatInfinite)
  qed
  then show  $starfun\ X\ N \approx star-of\ L$ 
    by (simp only: approx-def)
qed

```

lemma *NSLIMSEQ-LIMSEQ*:

```

assumes  $X: X \longrightarrow_{NS} L$ 
shows  $X \longrightarrow L$ 
proof (rule LIMSEQ-I)

```

```

fix  $r :: \text{real}$ 
assume  $r: 0 < r$ 
have  $\exists no. \forall n \geq no. \text{hnorm} (\text{starfun } X \ n - \text{star-of } L) < \text{star-of } r$ 
proof (intro exI allI impI)
  fix  $n$ 
  assume  $whn \leq n$ 
  with  $\text{HNatInfinite-wn}$  have  $n \in \text{HNatInfinite}$ 
  by (rule HNatInfinite-upward-closed)
  with  $X$  have  $\text{starfun } X \ n \approx \text{star-of } L$ 
  by (rule NSLIMSEQ-D)
  then have  $\text{starfun } X \ n - \text{star-of } L \in \text{Infinitesimal}$ 
  by (simp only: approx-def)
  then show  $\text{hnorm} (\text{starfun } X \ n - \text{star-of } L) < \text{star-of } r$ 
  using  $r$  by (rule InfinitesimalD2)
qed
then show  $\exists no. \forall n \geq no. \text{norm} (X \ n - L) < r$ 
by transfer
qed

```

theorem *LIMSEQ-NSLIMSEQ-iff*: $f \longrightarrow L \longleftrightarrow f \longrightarrow_{NS} L$
by (*blast intro: LIMSEQ-NSLIMSEQ NSLIMSEQ-LIMSEQ*)

10.1.2 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*: $(\lambda n. \text{norm} (X \ n)) \longrightarrow_{NS} 0 \longleftrightarrow X \longrightarrow_{NS} 0$
by (*simp add: LIMSEQ-NSLIMSEQ-iff [symmetric] tendsto-norm-zero-iff*)

lemma *NSLIMSEQ-rabs-zero*: $(\lambda n. |f \ n|) \longrightarrow_{NS} 0 \longleftrightarrow f \longrightarrow_{NS} (0 :: \text{real})$
by (*simp add: LIMSEQ-NSLIMSEQ-iff [symmetric] tendsto-rabs-zero-iff*)

Generalization to other limits.

lemma *NSLIMSEQ-imp-rabs*: $f \longrightarrow_{NS} l \implies (\lambda n. |f \ n|) \longrightarrow_{NS} |l|$
for $l :: \text{real}$
by (*simp add: NSLIMSEQ-def (auto intro: approx-hrabs simp add: starfun-abs)*)

lemma *NSLIMSEQ-inverse-zero*: $\forall y :: \text{real}. \exists N. \forall n \geq N. y < f \ n \implies (\lambda n. \text{inverse} (f \ n)) \longrightarrow_{NS} 0$
by (*simp add: LIMSEQ-NSLIMSEQ-iff [symmetric] LIMSEQ-inverse-zero*)

lemma *NSLIMSEQ-inverse-real-of-nat*: $(\lambda n. \text{inverse} (\text{real} (\text{Suc } n))) \longrightarrow_{NS} 0$
by (*simp add: LIMSEQ-NSLIMSEQ-iff [symmetric] LIMSEQ-inverse-real-of-nat del: of-nat-Suc*)

lemma *NSLIMSEQ-inverse-real-of-nat-add*: $(\lambda n. r + \text{inverse} (\text{real} (\text{Suc } n))) \longrightarrow_{NS} r$

by (*simp add: LIMSEQ-NSLIMSEQ-iff* [*symmetric*] *LIMSEQ-inverse-real-of-nat-add*
del: of-nat-Suc)

lemma *NSLIMSEQ-inverse-real-of-nat-add-minus*: $(\lambda n. r + - \text{inverse} (\text{real} (\text{Suc } n))) \longrightarrow_{NS} r$
using *LIMSEQ-inverse-real-of-nat-add-minus* **by** (*simp add: LIMSEQ-NSLIMSEQ-iff*
[*symmetric*])

lemma *NSLIMSEQ-inverse-real-of-nat-add-minus-mult*:
 $(\lambda n. r * (1 + - \text{inverse} (\text{real} (\text{Suc } n)))) \longrightarrow_{NS} r$
using *LIMSEQ-inverse-real-of-nat-add-minus-mult*
by (*simp add: LIMSEQ-NSLIMSEQ-iff* [*symmetric*])

10.2 Convergence

lemma *nslimI*: $X \longrightarrow_{NS} L \implies \text{nslim } X = L$
by (*simp add: nslim-def*) (*blast intro: NSLIMSEQ-unique*)

lemma *lim-nslim-iff*: $\text{lim } X = \text{nslim } X$
by (*simp add: lim-def nslim-def LIMSEQ-NSLIMSEQ-iff*)

lemma *NSconvergentD*: $\text{NSconvergent } X \implies \exists L. X \longrightarrow_{NS} L$
by (*simp add: NSconvergent-def*)

lemma *NSconvergentI*: $X \longrightarrow_{NS} L \implies \text{NSconvergent } X$
by (*auto simp add: NSconvergent-def*)

lemma *convergent-NSconvergent-iff*: $\text{convergent } X = \text{NSconvergent } X$
by (*simp add: convergent-def NSconvergent-def LIMSEQ-NSLIMSEQ-iff*)

lemma *NSconvergent-NSLIMSEQ-iff*: $\text{NSconvergent } X \longleftrightarrow X \longrightarrow_{NS} \text{nslim } X$
by (*auto intro: theI NSLIMSEQ-unique simp add: NSconvergent-def nslim-def*)

10.3 Bounded Monotonic Sequences

lemma *NSBseqD*: $\text{NSBseq } X \implies N \in \text{HNatInfinite} \implies (*f* X) N \in \text{HFinite}$
by (*simp add: NSBseq-def*)

lemma *Standard-subset-HFinite*: $\text{Standard} \subseteq \text{HFinite}$
by (*auto simp: Standard-def*)

lemma *NSBseqD2*: $\text{NSBseq } X \implies (*f* X) N \in \text{HFinite}$
using *HNatInfinite-def NSBseq-def Nats-eq-Standard Standard-starfun Standard-subset-HFinite*
by *blast*

lemma *NSBseqI*: $\forall N \in \text{HNatInfinite}. (*f* X) N \in \text{HFinite} \implies \text{NSBseq } X$
by (*simp add: NSBseq-def*)

The standard definition implies the nonstandard definition.

lemma *Bseq-NSBseq*: $\text{Bseq } X \implies \text{NSBseq } X$

```

unfolding NSBseq-def
proof safe
  assume  $X: Bseq\ X$ 
  fix  $N$ 
  assume  $N: N \in HNatInfinite$ 
  from  $BseqD\ [OF\ X]$  obtain  $K$  where  $\forall n. norm\ (X\ n) \leq K$ 
  by fast
  then have  $\forall N. hnorm\ (starfun\ X\ N) \leq star-of\ K$ 
  by transfer
  then have  $hnorm\ (starfun\ X\ N) \leq star-of\ K$ 
  by simp
  also have  $star-of\ K < star-of\ (K + 1)$ 
  by simp
  finally have  $\exists x \in Reals. hnorm\ (starfun\ X\ N) < x$ 
  by (rule bexI) simp
  then show  $starfun\ X\ N \in HFinite$ 
  by (simp add: HFinite-def)
qed

```

The nonstandard definition implies the standard definition.

```

lemma SReal-less-omega:  $r \in \mathbb{R} \implies r < \omega$ 
  using HInfinite-omega
  by (simp add: HInfinite-def) (simp add: order-less-imp-le)

```

```

lemma NSBseq-Bseq:  $NSBseq\ X \implies Bseq\ X$ 
proof (rule ccontr)
  let  $?n = \lambda K. LEAST\ n. K < norm\ (X\ n)$ 
  assume  $NSBseq\ X$ 
  then have  $finite: (*f* X) ((*f* ?n)\ \omega) \in HFinite$ 
  by (rule NSBseqD2)
  assume  $\neg Bseq\ X$ 
  then have  $\forall K > 0. \exists n. K < norm\ (X\ n)$ 
  by (simp add: Bseq-def linorder-not-le)
  then have  $\forall K > 0. K < norm\ (X\ (?n\ K))$ 
  by (auto intro: LeastI-ex)
  then have  $\forall K > 0. K < hnorm\ ((*f* X) ((*f* ?n)\ K))$ 
  by transfer
  then have  $\omega < hnorm\ ((*f* X) ((*f* ?n)\ \omega))$ 
  by simp
  then have  $\forall r \in \mathbb{R}. r < hnorm\ ((*f* X) ((*f* ?n)\ \omega))$ 
  by (simp add: order-less-trans [OF SReal-less-omega])
  then have  $(*f* X) ((*f* ?n)\ \omega) \in HInfinite$ 
  by (simp add: HInfinite-def)
  with finite show False
  by (simp add: HFinite-HInfinite-iff)
qed

```

Equivalence of nonstandard and standard definitions for a bounded sequence.

lemma *Bseq-NSBseq-iff*: $Bseq\ X = NSBseq\ X$
by (*blast intro!*: *NSBseq-Bseq Bseq-NSBseq*)

A convergent sequence is bounded: Boundedness as a necessary condition for convergence. The nonstandard version has no existential, as usual.

lemma *NSconvergent-NSBseq*: $NSconvergent\ X \implies NSBseq\ X$
by (*simp add*: *NSconvergent-def NSBseq-def NSLIMSEQ-def*)
(*blast intro*: *HFinite-star-of approx-sym approx-HFinite*)

Standard Version: easily now proved using equivalence of NS and standard definitions.

lemma *convergent-Bseq*: $convergent\ X \implies Bseq\ X$
for $X :: nat \Rightarrow 'b::real-normed-vector$
by (*simp add*: *NSconvergent-NSBseq convergent-NSconvergent-iff Bseq-NSBseq-iff*)

10.3.1 Upper Bounds and Lubs of Bounded Sequences

lemma *NSBseq-isUb*: $NSBseq\ X \implies \exists U::real. isUb\ UNIV\ \{x. \exists n. X\ n = x\}\ U$
by (*simp add*: *Bseq-NSBseq-iff [symmetric] Bseq-isUb*)

lemma *NSBseq-isLub*: $NSBseq\ X \implies \exists U::real. isLub\ UNIV\ \{x. \exists n. X\ n = x\}\ U$
by (*simp add*: *Bseq-NSBseq-iff [symmetric] Bseq-isLub*)

10.3.2 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*: $\forall_F\ k\ in\ sequentially. X\ k = X\ m \implies X \longrightarrow_{NS} X\ m$
unfolding *LIMSEQ-NSLIMSEQ-iff [symmetric]*
by (*simp add*: *eventually-mono eventually-nhds-x-imp-x filterlim-iff*)

lemma *NSBseq-mono-NSconvergent*: $NSBseq\ X \implies \forall m. \forall n \geq m. X\ m \leq X\ n \implies NSconvergent\ X$
for $X :: nat \Rightarrow real$
by (*auto intro*: *Bseq-mono-convergent*
simp: *convergent-NSconvergent-iff [symmetric] Bseq-NSBseq-iff [symmetric]*)

10.4 Cauchy Sequences

lemma *NSCauchyI*:
 $(\bigwedge M\ N. M \in HNatInfinite \implies N \in HNatInfinite \implies starfun\ X\ M \approx starfun\ X\ N) \implies NSCauchy\ X$
by (*simp add*: *NSCauchy-def*)

lemma *NSCauchyD*:

$NSCauchy\ X \implies M \in HNatInfinite \implies N \in HNatInfinite \implies starfun\ X\ M \approx starfun\ X\ N$
by (*simp add: NSCauchy-def*)

10.4.1 Equivalence Between NS and Standard

lemma *Cauchy-NSCauchy:*

assumes $X: Cauchy\ X$

shows $NSCauchy\ X$

proof (*rule NSCauchyI*)

fix M

assume $M: M \in HNatInfinite$

fix N

assume $N: N \in HNatInfinite$

have $starfun\ X\ M - starfun\ X\ N \in Infinitesimal$

proof (*rule InfinitesimalI2*)

fix $r :: real$

assume $r: 0 < r$

from $CauchyD\ [OF\ X\ r]$ **obtain** k **where** $\forall m \geq k. \forall n \geq k. norm\ (X\ m - X\ n) < r ..$

then have $\forall m \geq star\text{-}of\ k. \forall n \geq star\text{-}of\ k. hnorm\ (starfun\ X\ m - starfun\ X\ n) < star\text{-}of\ r$

by *transfer*

then show $hnorm\ (starfun\ X\ M - starfun\ X\ N) < star\text{-}of\ r$

using $M\ N$ **by** (*simp add: star-of-le-HNatInfinite*)

qed

then show $starfun\ X\ M \approx starfun\ X\ N$

by (*simp only: approx-def*)

qed

lemma *NSCauchy-Cauchy:*

assumes $X: NSCauchy\ X$

shows $Cauchy\ X$

proof (*rule CauchyI*)

fix $r :: real$

assume $r: 0 < r$

have $\exists k. \forall m \geq k. \forall n \geq k. hnorm\ (starfun\ X\ m - starfun\ X\ n) < star\text{-}of\ r$

proof (*intro exI allI impI*)

fix M

assume $whn \leq M$

with $HNatInfinite\text{-}whn$ **have** $M: M \in HNatInfinite$

by (*rule HNatInfinite-upward-closed*)

fix N

assume $whn \leq N$

with $HNatInfinite\text{-}whn$ **have** $N: N \in HNatInfinite$

by (*rule HNatInfinite-upward-closed*)

from $X\ M\ N$ **have** $starfun\ X\ M \approx starfun\ X\ N$

by (*rule NSCauchyD*)

then have $starfun\ X\ M - starfun\ X\ N \in Infinitesimal$


```

    by (simp only: approx-def)
  then show  $hnorm (starfun X M - starfun X N) < star-of r$ 
    using  $r$  by (rule InfinitesimalD2)
qed
then show  $\exists k. \forall m \geq k. \forall n \geq k. norm (X m - X n) < r$ 
  by transfer
qed

```

theorem *NSCauchy-Cauchy-iff*: $NSCauchy X = Cauchy X$
 by (blast intro!: NSCauchy-Cauchy Cauchy-NSCauchy)

10.4.2 Cauchy Sequences are Bounded

A Cauchy sequence is bounded – nonstandard version.

lemma *NSCauchy-NSBseq*: $NSCauchy X \implies NSBseq X$
 by (simp add: Cauchy-Bseq Bseq-NSBseq-iff [symmetric] NSCauchy-Cauchy-iff)

10.4.3 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 ‘epsilon-delta’ proof(s) in this case since the NS formulations do not involve existential quantifiers.

lemma *NSconvergent-NSCauchy*: $NSconvergent X \implies NSCauchy X$
 by (simp add: NSconvergent-def NSLIMSEQ-def NSCauchy-def) (auto intro: approx-trans2)

lemma *real-NSCauchy-NSconvergent*:

```

  fixes  $X :: nat \Rightarrow real$ 
  assumes  $NSCauchy X$  shows  $NSconvergent X$ 
  unfolding NSconvergent-def NSLIMSEQ-def
proof -
  have  $(\ast f \ast X) whn \in HFinite$ 
    by (simp add: NSBseqD2 NSCauchy-NSBseq assms)
  moreover have  $\forall N \in HNatInfinite. (\ast f \ast X) whn \approx (\ast f \ast X) N$ 
    using  $HNatInfinite-wnn NSCauchy-def assms$  by blast
  ultimately show  $\exists L. \forall N \in HNatInfinite. (\ast f \ast X) N \approx hypreal-of-real L$ 
    by (force dest!: st-part-Ex simp add: SReal-iff intro: approx-trans3)
qed

```

lemma *NSCauchy-NSconvergent*: $NSCauchy X \implies NSconvergent X$
 for $X :: nat \Rightarrow 'a::banach$
 using Cauchy-convergent NSCauchy-Cauchy convergent-NSconvergent-iff by auto

lemma *NSCauchy-NSconvergent-iff*: $NSCauchy\ X = NSconvergent\ X$
for $X :: nat \Rightarrow 'a::banach$
by (*fast intro: NSCauchy-NSconvergent NSconvergent-NSCauchy*)

10.5 Power Sequences

The sequence x^n tends to 0 if $0 \leq x$ and $x < 1$. 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*:
fixes $x :: real$
assumes $0 \leq x < 1$ **shows** $(\lambda n. x \wedge n) \longrightarrow_{NS} 0$
proof –
have $(\ast f \ast (\wedge) x) N \approx 0$
if $N: N \in HNatInfinite$ **and** $x: NSconvergent\ ((\wedge) x)$ **for** N
proof –
have $hypreal\text{-of}\text{-real}\ x\ pow\ N \approx hypreal\text{-of}\text{-real}\ x\ pow\ (N + 1)$
by (*metis HNatInfinite-add N NSCauchy-NSconvergent-iff NSCauchy-def starfun-pow x*)
moreover obtain L **where** $L: hypreal\text{-of}\text{-real}\ x\ pow\ N \approx hypreal\text{-of}\text{-real}\ L$
using *NSconvergentD [OF x] N* **by** (*auto simp add: NSLIMSEQ-def starfun-pow*)
ultimately have $hypreal\text{-of}\text{-real}\ x\ pow\ N \approx hypreal\text{-of}\text{-real}\ L \ast hypreal\text{-of}\text{-real}\ x$
by (*simp add: approx-mult-subst-star-of hyperpow-add*)
then have $hypreal\text{-of}\text{-real}\ L \approx hypreal\text{-of}\text{-real}\ L \ast hypreal\text{-of}\text{-real}\ x$
using L *approx-trans3* **by** *blast*
then show *?thesis*
by (*metis L $\langle x < 1 \rangle$ hyperpow-def less-irrefl mult.right-neutral mult-left-cancel star-of-approx-iff star-of-mult star-of-simps(9) starfun2-star-of*)
qed
with *assms* **show** *?thesis*
by (*force dest!: convergent-realpow simp add: NSLIMSEQ-def convergent-NSconvergent-iff*)
qed

lemma *NSLIMSEQ-abs-realpow-zero*: $|c| < 1 \implies (\lambda n. |c| \wedge n) \longrightarrow_{NS} 0$
for $c :: real$
by (*simp add: LIMSEQ-abs-realpow-zero LIMSEQ-NSLIMSEQ-iff [symmetric]*)

lemma *NSLIMSEQ-abs-realpow-zero2*: $|c| < 1 \implies (\lambda n. c \wedge n) \longrightarrow_{NS} 0$
for $c :: real$
by (*simp add: LIMSEQ-abs-realpow-zero2 LIMSEQ-NSLIMSEQ-iff [symmetric]*)

end

11 Finite Summation and Infinite Series for Hyperreals

```
theory HSeries
  imports HSEQ
begin
```

```
definition sumhr :: hypnat × hypnat × (nat ⇒ real) ⇒ hypreal
  where sumhr = (λ(M,N,f). starfun2 (λm n. sum f {m.. $n$ }) M N)
```

```
definition NSsums :: (nat ⇒ real) ⇒ real ⇒ bool (infixr <NSsums> 80)
  where f NSsums s = (λn. sum f {.. $n$ })  $\longrightarrow_{NS}$  s
```

```
definition NSsummable :: (nat ⇒ real) ⇒ bool
  where NSsummable f  $\longleftrightarrow$  (∃ 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* (λm n. sum f {m.. $n$ })) M N
  by (simp add: sumhr-def)
```

Base case in definition of *sumr*.

```
lemma sumhr-zero [simp]: ∧m. sumhr (m, 0, f) = 0
  unfolding sumhr-app by transfer simp
```

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)
  unfolding sumhr-app by transfer simp
```

```
lemma sumhr-Suc-zero [simp]: ∧n. sumhr (n + 1, n, f) = 0
  unfolding sumhr-app by transfer simp
```

```
lemma sumhr-eq-bounds [simp]: ∧n. sumhr (n, n, f) = 0
  unfolding sumhr-app by transfer simp
```

```
lemma sumhr-Suc [simp]: ∧m. sumhr (m, m + 1, f) = (*f* f) m
  unfolding sumhr-app by transfer simp
```

```
lemma sumhr-add-lbound-zero [simp]: ∧k m. sumhr (m + k, k, f) = 0
  unfolding sumhr-app by transfer simp
```

```
lemma sumhr-add: ∧m n. sumhr (m, n, f) + sumhr (m, n, g) = sumhr (m, n,
  λi. f i + g i)
  unfolding sumhr-app by transfer (rule sum.distrib [symmetric])
```

lemma *sumhr-mult*: $\bigwedge m\ n. \text{hypreal-of-real } r * \text{sumhr } (m, n, f) = \text{sumhr } (m, n, \lambda n. r * f\ n)$

unfolding *sumhr-app* **by** *transfer* (rule *sum-distrib-left*)

lemma *sumhr-split-add*: $\bigwedge n\ p. n < p \implies \text{sumhr } (0, n, f) + \text{sumhr } (n, p, f) = \text{sumhr } (0, p, f)$

unfolding *sumhr-app* **by** *transfer* (*simp add: sum.atLeastLessThan-concat*)

lemma *sumhr-split-diff*: $n < p \implies \text{sumhr } (0, p, f) - \text{sumhr } (0, n, f) = \text{sumhr } (n, p, f)$

by (*drule sumhr-split-add [symmetric, where f = f]*) *simp*

lemma *sumhr-hrabs*: $\bigwedge m\ n. |\text{sumhr } (m, n, f)| \leq \text{sumhr } (m, n, \lambda i. |f\ i|)$

unfolding *sumhr-app* **by** *transfer* (rule *sum-abs*)

Other general version also needed.

lemma *sumhr-fun-hypnat-eq*:

$(\forall r. m \leq r \wedge r < n \longrightarrow f\ r = g\ r) \longrightarrow$
 $\text{sumhr } (\text{hypnat-of-nat } m, \text{hypnat-of-nat } n, f) =$
 $\text{sumhr } (\text{hypnat-of-nat } m, \text{hypnat-of-nat } n, g)$

unfolding *sumhr-app* **by** *transfer simp*

lemma *sumhr-const*: $\bigwedge n. \text{sumhr } (0, n, \lambda i. r) = \text{hypreal-of-hypnat } n * \text{hypreal-of-real } r$

unfolding *sumhr-app* **by** *transfer simp*

lemma *sumhr-less-bounds-zero* [*simp*]: $\bigwedge m\ n. n < m \implies \text{sumhr } (m, n, f) = 0$

unfolding *sumhr-app* **by** *transfer simp*

lemma *sumhr-minus*: $\bigwedge m\ n. \text{sumhr } (m, n, \lambda i. -f\ i) = - \text{sumhr } (m, n, f)$

unfolding *sumhr-app* **by** *transfer* (rule *sum-negf*)

lemma *sumhr-shift-bounds*:

$\bigwedge m\ n. \text{sumhr } (m + \text{hypnat-of-nat } k, n + \text{hypnat-of-nat } k, f) =$
 $\text{sumhr } (m, n, \lambda i. f\ (i + k))$

unfolding *sumhr-app* **by** *transfer* (rule *sum.shift-bounds-nat-ivl*)

11.1 Nonstandard Sums

Infinite sums are obtained by summing to some infinite hypernatural (such as *whn*).

lemma *sumhr-hypreal-of-hypnat-omega*: $\text{sumhr } (0, \text{whn}, \lambda i. 1) = \text{hypreal-of-hypnat } \text{whn}$

by (*simp add: sumhr-const*)

lemma *whn-eq- ω m1*: $\text{hypreal-of-hypnat } \text{whn} = \omega - 1$

unfolding *star-class-defs omega-def hypnat-omega-def of-hypnat-def star-of-def*

by (*simp add: starfun-star-n starfun2-star-n*)

lemma *sumhr-hypreal-omega-minus-one*: $\text{sumhr}(0, \text{whn}, \lambda i. 1) = \omega - 1$
by (*simp add: sumhr-const whn-eq- $\omega m 1$*)

lemma *sumhr-minus-one-realpow-zero* [*simp*]: $\bigwedge N. \text{sumhr}(0, N + N, \lambda i. (-1)^\wedge (i + 1)) = 0$
unfolding *sumhr-app*
by *transfer (induct-tac N, auto)*

lemma *sumhr-interval-const*:
 $(\forall n. m \leq \text{Suc } n \longrightarrow f\ n = r) \wedge m \leq na \implies$
 $\text{sumhr}(\text{hypnat-of-nat } m, \text{hypnat-of-nat } na, f) = \text{hypreal-of-nat } (na - m) * \text{hypreal-of-real } r$
unfolding *sumhr-app* **by** *transfer simp*

lemma *starfunNat-sumr*: $\bigwedge N. (*f* (\lambda n. \text{sum } f \{0..<n\}))\ N = \text{sumhr}(0, N, f)$
unfolding *sumhr-app* **by** *transfer (rule refl)*

lemma *sumhr-hrabs-approx* [*simp*]: $\text{sumhr}(0, M, f) \approx \text{sumhr}(0, N, f) \implies |\text{sumhr}(M, N, f)| \approx 0$
using *linorder-less-linear* [**where** $x = M$ **and** $y = N$]
by (*metis (no-types, lifting) abs-zero approx-hrabs approx-minus-iff approx-refl approx-sym sumhr-eq-bounds sumhr-less-bounds-zero sumhr-split-diff*)

11.2 Infinite sums: Standard and NS theorems

lemma *sums-NSsums-iff*: $f \text{ sums } l \longleftrightarrow f \text{ NSsums } l$
by (*simp add: sums-def NSsums-def LIMSEQ-NSLIMSEQ-iff*)

lemma *summable-NSsummable-iff*: $\text{summable } f \longleftrightarrow \text{NSsummable } f$
by (*simp add: summable-def NSsummable-def sums-NSsums-iff*)

lemma *suminf-NSsuminf-iff*: $\text{suminf } f = \text{NSsuminf } f$
by (*simp add: suminf-def NSsuminf-def sums-NSsums-iff*)

lemma *NSsums-NSsummable*: $f \text{ NSsums } l \implies \text{NSsummable } f$
unfolding *NSsums-def NSsummable-def* **by** *blast*

lemma *NSsummable-NSsums*: $\text{NSsummable } f \implies f \text{ NSsums } (\text{NSsuminf } f)$
unfolding *NSsummable-def NSsuminf-def NSsums-def*
by (*blast intro: theI NSLIMSEQ-unique*)

lemma *NSsums-unique*: $f \text{ NSsums } s \implies s = \text{NSsuminf } f$
by (*simp add: suminf-NSsuminf-iff [symmetric] sums-NSsums-iff sums-unique*)

lemma *NSseries-zero*: $\forall m. n \leq \text{Suc } m \longrightarrow f\ m = 0 \implies f \text{ NSsums } (\text{sum } f \{..<n\})$
by (*auto simp add: sums-NSsums-iff [symmetric] not-le[symmetric] intro!: sums-finite*)

lemma *NSsummable-NSCauchy*:

$NSummable\ f \longleftrightarrow (\forall M \in HNatInfinite. \forall N \in HNatInfinite. |sumhr\ (M, N, f)| \approx 0)$ (is ?L=?R)
proof –
have ?L = $(\forall M \in HNatInfinite. \forall N \in HNatInfinite. sumhr\ (0, M, f) \approx sumhr\ (0, N, f))$
by (auto simp add: summable-iff-convergent convergent-NSconvergent-iff NSCauchy-def starfunNat-sumr
simp flip: NSCauchy-NSconvergent-iff summable-NSsummable-iff atLeast0LessThan)
also have ... \longleftrightarrow ?R
by (metis approx-hrabs-zero-cancel approx-minus-iff approx-refl approx-sym
linorder-less-linear sumhr-hrabs-approx sumhr-split-diff)
finally show ?thesis .
qed

Terms of a convergent series tend to zero.

lemma *NSummable-NSLIMSEQ-zero*: $NSummable\ f \implies f \longrightarrow_{NS} 0$
by (metis HNatInfinite-add NSLIMSEQ-def NSummable-NSCauchy approx-hrabs-zero-cancel
star-of-zero sumhr-Suc)

Nonstandard comparison test.

lemma *NSummable-comparison-test*: $\exists N. \forall n. N \leq n \longrightarrow |f\ n| \leq g\ n \implies NSummable\ g \implies NSummable\ f$
by (metis real-norm-def summable-NSsummable-iff summable-comparison-test)

lemma *NSummable-rabs-comparison-test*:
 $\exists N. \forall n. N \leq n \longrightarrow |f\ n| \leq g\ n \implies NSummable\ g \implies NSummable\ (\lambda k. |f\ k|)$
by (rule NSummable-comparison-test) auto

end

12 Limits and Continuity (Nonstandard)

theory *HLim*
imports *Star*
abbrevs $---> = -\square\rightarrow_{NS}$
begin

Nonstandard Definitions.

definition *NSLIM* :: $('a::real-normed-vector \Rightarrow 'b::real-normed-vector) \Rightarrow 'a \Rightarrow 'b \Rightarrow bool$
 $(\langle \langle notation = \langle mixfix\ NSLIM \rangle \rangle (-) / -(-) / \rightarrow_{NS} (-) \rangle [60, 0, 60]\ 60)$
where $f -a \rightarrow_{NS} L \longleftrightarrow (\forall x. x \neq star-of\ a \wedge x \approx star-of\ a \longrightarrow (*f* f)\ x \approx star-of\ L)$

definition *isNSCont* :: $('a::real-normed-vector \Rightarrow 'b::real-normed-vector) \Rightarrow 'a \Rightarrow bool$
where — NS definition dispenses with limit notions
 $isNSCont\ f\ a \longleftrightarrow (\forall y. y \approx star-of\ a \longrightarrow (*f* f)\ y \approx star-of\ (f\ a))$

definition *isNSUCont* :: ('a::real-normed-vector \Rightarrow 'b::real-normed-vector) \Rightarrow bool
where *isNSUCont* *f* $\longleftrightarrow (\forall x y. x \approx y \longrightarrow (*f* f) x \approx (*f* f) y)$

12.1 Limits of Functions

lemma *NSLIM-I*: $(\bigwedge x. x \neq \text{star-of } a \implies x \approx \text{star-of } a \implies \text{starfun } f x \approx \text{star-of } L) \implies f -a \rightarrow_{NS} L$
by (*simp add: NSLIM-def*)

lemma *NSLIM-D*: $f -a \rightarrow_{NS} L \implies x \neq \text{star-of } a \implies x \approx \text{star-of } a \implies \text{starfun } f x \approx \text{star-of } L$
by (*simp add: NSLIM-def*)

Proving properties of limits using nonstandard definition. The properties hold for standard limits as well!

lemma *NSLIM-mult*: $f -x \rightarrow_{NS} l \implies g -x \rightarrow_{NS} m \implies (\lambda x. f x * g x) -x \rightarrow_{NS} (l * m)$
for *l m* :: 'a::real-normed-algebra
by (*auto simp add: NSLIM-def intro!: approx-mult-HFinite*)

lemma *starfun-scaleR* [*simp*]: $\text{starfun } (\lambda x. f x *_R g x) = (\lambda x. \text{scaleHR } (\text{starfun } f x) (\text{starfun } g x))$
by *transfer (rule refl)*

lemma *NSLIM-scaleR*: $f -x \rightarrow_{NS} l \implies g -x \rightarrow_{NS} m \implies (\lambda x. f x *_R g x) -x \rightarrow_{NS} (l *_R m)$
by (*auto simp add: NSLIM-def intro!: approx-scaleR-HFinite*)

lemma *NSLIM-add*: $f -x \rightarrow_{NS} l \implies g -x \rightarrow_{NS} m \implies (\lambda x. f x + g x) -x \rightarrow_{NS} (l + m)$
by (*auto simp add: NSLIM-def intro!: approx-add*)

lemma *NSLIM-const* [*simp*]: $(\lambda x. k) -x \rightarrow_{NS} k$
by (*simp add: NSLIM-def*)

lemma *NSLIM-minus*: $f -a \rightarrow_{NS} L \implies (\lambda x. - f x) -a \rightarrow_{NS} -L$
by (*simp add: NSLIM-def*)

lemma *NSLIM-diff*: $f -x \rightarrow_{NS} l \implies g -x \rightarrow_{NS} m \implies (\lambda x. f x - g x) -x \rightarrow_{NS} (l - m)$
by (*simp only: NSLIM-add NSLIM-minus diff-conv-add-uminus*)

lemma *NSLIM-add-minus*: $f -x \rightarrow_{NS} l \implies g -x \rightarrow_{NS} m \implies (\lambda x. f x + - g x) -x \rightarrow_{NS} (l + -m)$
by (*simp only: NSLIM-add NSLIM-minus*)

lemma *NSLIM-inverse*: $f -a \rightarrow_{NS} L \implies L \neq 0 \implies (\lambda x. \text{inverse } (f x)) -a \rightarrow_{NS} (\text{inverse } L)$

for $L :: 'a::\text{real-normed-div-algebra}$
 unfolding $NSLIM\text{-def}$ by (metis (no-types) star-of-approx-inverse star-of-simps(6)
 starfun-inverse)

lemma $NSLIM\text{-zero}$:
 assumes $f: f - a \rightarrow_{NS} l$
 shows $(\lambda x. f(x) - l) - a \rightarrow_{NS} 0$
 proof -
 have $(\lambda x. f x - l) - a \rightarrow_{NS} l - l$
 by (rule $NSLIM\text{-diff}$ [OF f $NSLIM\text{-const}$])
 then show ?thesis by simp
 qed

lemma $NSLIM\text{-zero-cancel}$:
 assumes $(\lambda x. f x - l) - x \rightarrow_{NS} 0$
 shows $f - x \rightarrow_{NS} l$
 proof -
 have $(\lambda x. f x - l + l) - x \rightarrow_{NS} 0 + l$
 by (fast intro: assms $NSLIM\text{-const}$ $NSLIM\text{-add}$)
 then show ?thesis
 by simp
 qed

lemma $NSLIM\text{-const-eq}$:
 fixes $a :: 'a::\text{real-normed-algebra-1}$
 assumes $(\lambda x. k) - a \rightarrow_{NS} l$
 shows $k = l$
 proof -
 have $\neg (\lambda x. k) - a \rightarrow_{NS} l$ if $k \neq l$
 proof -
 have $\text{star-of } a + \text{of-hypreal } \varepsilon \approx \text{star-of } a$
 by (simp add: approx-def)
 then show ?thesis
 using $\text{epsilon-not-zero that}$ by (force simp add: $NSLIM\text{-def}$)
 qed
 with assms show ?thesis by metis
 qed

lemma $NSLIM\text{-unique}$: $f - a \rightarrow_{NS} l \implies f - a \rightarrow_{NS} M \implies l = M$
 for $a :: 'a::\text{real-normed-algebra-1}$
 by (drule (1) $NSLIM\text{-diff}$) (auto dest!: $NSLIM\text{-const-eq}$)

lemma $NSLIM\text{-mult-zero}$: $f - x \rightarrow_{NS} 0 \implies g - x \rightarrow_{NS} 0 \implies (\lambda x. f x * g x)$
 $- x \rightarrow_{NS} 0$
 for $f g :: 'a::\text{real-normed-vector} \Rightarrow 'b::\text{real-normed-algebra}$
 by (drule $NSLIM\text{-mult}$) auto

lemma $NSLIM\text{-self}$: $(\lambda x. x) - a \rightarrow_{NS} a$
 by (simp add: $NSLIM\text{-def}$)

12.1.1.1 Equivalence of *filterlim* and *NSLIM*

lemma *LIM-NSLIM*:

assumes $f: f - a \rightarrow L$

shows $f - a \rightarrow_{NS} L$

proof (rule *NSLIM-I*)

fix x

assume *neg*: $x \neq \text{star-of } a$

assume *approx*: $x \approx \text{star-of } a$

have $\text{starfun } f \ x - \text{star-of } L \in \text{Infinitesimal}$

proof (rule *InfinitesimalI2*)

fix $r :: \text{real}$

assume $r: 0 < r$

from *LIM-D* [*OF* $f \ r$] **obtain** s

where $s: 0 < s$ and *less-r*: $\bigwedge x. x \neq a \implies \text{norm } (x - a) < s \implies \text{norm } (f \ x - L) < r$

by *fast*

from *less-r* **have** *less-r'*:

$\bigwedge x. x \neq \text{star-of } a \implies \text{hnorm } (x - \text{star-of } a) < \text{star-of } s \implies$
 $\text{hnorm } (\text{starfun } f \ x - \text{star-of } L) < \text{star-of } r$

by *transfer*

from *approx* **have** $x - \text{star-of } a \in \text{Infinitesimal}$

by (*simp only*: *approx-def*)

then have $\text{hnorm } (x - \text{star-of } a) < \text{star-of } s$

using s **by** (rule *InfinitesimalD2*)

with *neg* **show** $\text{hnorm } (\text{starfun } f \ x - \text{star-of } L) < \text{star-of } r$

by (rule *less-r'*)

qed

then show $\text{starfun } f \ x \approx \text{star-of } L$

by (*unfold approx-def*)

qed

lemma *NSLIM-LIM*:

assumes $f: f - a \rightarrow_{NS} L$

shows $f - a \rightarrow L$

proof (rule *LIM-I*)

fix $r :: \text{real}$

assume $r: 0 < r$

have $\exists s > 0. \forall x. x \neq \text{star-of } a \wedge \text{hnorm } (x - \text{star-of } a) < s \longrightarrow$
 $\text{hnorm } (\text{starfun } f \ x - \text{star-of } L) < \text{star-of } r$

proof (rule *exI*, *safe*)

show $0 < \varepsilon$

by (rule *epsilon-gt-zero*)

next

fix x

assume *neg*: $x \neq \text{star-of } a$

assume $\text{hnorm } (x - \text{star-of } a) < \varepsilon$

with *Infinitesimal-epsilon* **have** $x - \text{star-of } a \in \text{Infinitesimal}$

by (rule *hnorm-less-Infinitesimal*)

then have $x \approx \text{star-of } a$

by (unfold approx-def)
 with $f \text{ neq}$ have starfun $f \ x \approx \text{star-of } L$
 by (rule NSLIM-D)
 then have starfun $f \ x - \text{star-of } L \in \text{Infinitesimal}$
 by (unfold approx-def)
 then show $\text{hnorm } (\text{starfun } f \ x - \text{star-of } L) < \text{star-of } r$
 using r by (rule InfinitesimalD2)
 qed
 then show $\exists s > 0. \forall x. x \neq a \wedge \text{norm } (x - a) < s \longrightarrow \text{norm } (f \ x - L) < r$
 by transfer
 qed

theorem *LIM-NSLIM-iff*: $f -x \rightarrow L \longleftrightarrow f -x \rightarrow_{NS} L$
 by (blast intro: LIM-NSLIM NSLIM-LIM)

12.2 Continuity

lemma *isNSContD*: $\text{isNSCont } f \ a \Longrightarrow y \approx \text{star-of } a \Longrightarrow (*f*) \ y \approx \text{star-of } (f \ a)$
 by (simp add: isNSCont-def)

lemma *isNSCont-NSLIM*: $\text{isNSCont } f \ a \Longrightarrow f -a \rightarrow_{NS} (f \ a)$
 by (simp add: isNSCont-def NSLIM-def)

lemma *NSLIM-isNSCont*: $f -a \rightarrow_{NS} (f \ a) \Longrightarrow \text{isNSCont } f \ a$
 by (force simp add: isNSCont-def NSLIM-def)

NS continuity can be defined using NS Limit in similar fashion to standard definition of continuity.

lemma *isNSCont-NSLIM-iff*: $\text{isNSCont } f \ a \longleftrightarrow f -a \rightarrow_{NS} (f \ a)$
 by (blast intro: isNSCont-NSLIM NSLIM-isNSCont)

Hence, NS continuity can be given in terms of standard limit.

lemma *isNSCont-LIM-iff*: $(\text{isNSCont } f \ a) = (f -a \rightarrow (f \ a))$
 by (simp add: LIM-NSLIM-iff isNSCont-NSLIM-iff)

Moreover, it's trivial now that NS continuity is equivalent to standard continuity.

lemma *isNSCont-isCont-iff*: $\text{isNSCont } f \ a \longleftrightarrow \text{isCont } f \ a$
 by (simp add: isCont-def) (rule isNSCont-LIM-iff)

Standard continuity \Longrightarrow NS continuity.

lemma *isCont-isNSCont*: $\text{isCont } f \ a \Longrightarrow \text{isNSCont } f \ a$
 by (erule isNSCont-isCont-iff [THEN iffD2])

NS continuity \Longrightarrow Standard continuity.

lemma *isNSCont-isCont*: $\text{isNSCont } f \ a \Longrightarrow \text{isCont } f \ a$
 by (erule isNSCont-isCont-iff [THEN iffD1])

Alternative definition of continuity.

Prove equivalence between NS limits – seems easier than using standard definition.

```

lemma NSLIM-at0-iff:  $f -a \rightarrow_{NS} L \longleftrightarrow (\lambda h. f (a + h)) -0 \rightarrow_{NS} L$ 
proof
  assume  $f -a \rightarrow_{NS} L$ 
  then show  $(\lambda h. f (a + h)) -0 \rightarrow_{NS} L$ 
    by (simp add: NSLIM-def) (metis (no-types) add-cancel-left-right approx-add-left-iff
starfun-lambda-cancel)
  next
    assume  $*$ :  $(\lambda h. f (a + h)) -0 \rightarrow_{NS} L$ 
    show  $f -a \rightarrow_{NS} L$ 
    proof (clarsimp simp: NSLIM-def)
      fix  $x$ 
      assume  $x \neq \text{star-of } a \approx \text{star-of } a$ 
      then have  $(*\!*\! (\lambda h. f (a + h))) (- \text{star-of } a + x) \approx \text{star-of } L$ 
        by (metis (no-types, lifting) * NSLIM-D add.right-neutral add-minus-cancel
approx-minus-iff2 star-zero-def)
      then show  $(*\!*\! f) x \approx \text{star-of } L$ 
        by (simp add: starfun-lambda-cancel)
    qed
  qed

```

```

lemma isNSCont-minus:  $\text{isNSCont } f \ a \implies \text{isNSCont } (\lambda x. - f x) \ a$ 
  by (simp add: isNSCont-def)

```

```

lemma isNSCont-inverse:  $\text{isNSCont } f \ x \implies f \ x \neq 0 \implies \text{isNSCont } (\lambda x. \text{inverse}$ 
 $(f \ x)) \ x$ 
  for  $f :: 'a::\text{real-normed-vector} \Rightarrow 'b::\text{real-normed-div-algebra}$ 
  using NSLIM-inverse NSLIM-isNSCont isNSCont-NSLIM by blast

```

```

lemma isNSCont-const [simp]:  $\text{isNSCont } (\lambda x. k) \ a$ 
  by (simp add: isNSCont-def)

```

```

lemma isNSCont-abs [simp]:  $\text{isNSCont } \text{abs } a$ 
  for  $a :: \text{real}$ 
  by (auto simp: isNSCont-def intro: approx-hrabs simp: starfun-rabs-hrabs)

```

12.3 Uniform Continuity

```

lemma isNSUContD:  $\text{isNSUCont } f \implies x \approx y \implies (*\!*\! f) \ x \approx (*\!*\! f) \ y$ 
  by (simp add: isNSUCont-def)

```

```

lemma isUCont-isNSUCont:
  fixes  $f :: 'a::\text{real-normed-vector} \Rightarrow 'b::\text{real-normed-vector}$ 
  assumes  $f$ : isUCont  $f$ 
  shows isNSUCont  $f$ 
  unfolding isNSUCont-def

```

```

proof safe
  fix x y :: 'a star
  assume approx: x ≈ y
  have starfun f x - starfun f y ∈ Infinitesimal
  proof (rule InfinitesimalI2)
    fix r :: real
    assume r: 0 < r
    with f obtain s where s: 0 < s
    and less-r:  $\bigwedge x y. \text{norm } (x - y) < s \implies \text{norm } (f x - f y) < r$ 
    by (auto simp add: isUCont-def dist-norm)
    from less-r have less-r':
       $\bigwedge x y. \text{hnorm } (x - y) < \text{star-of } s \implies \text{hnorm } (\text{starfun } f x - \text{starfun } f y) <$ 
star-of r
    by transfer
    from approx have x - y ∈ Infinitesimal
    by (unfold approx-def)
    then have hnorm (x - y) < star-of s
    using s by (rule InfinitesimalD2)
    then show hnorm (starfun f x - starfun f y) < star-of r
    by (rule less-r')
  qed
then show starfun f x ≈ starfun f y
by (unfold approx-def)
qed

lemma isNSUCont-isUCont:
  fixes f :: 'a::real-normed-vector  $\Rightarrow$  'b::real-normed-vector
  assumes f: isNSUCont f
  shows isUCont f
  unfolding isUCont-def dist-norm
proof safe
  fix r :: real
  assume r: 0 < r
  have  $\exists s > 0. \forall x y. \text{hnorm } (x - y) < s \longrightarrow \text{hnorm } (\text{starfun } f x - \text{starfun } f y) <$ 
star-of r
  proof (rule exI, safe)
    show 0 < ε
    by (rule epsilon-gt-zero)
  next
    fix x y :: 'a star
    assume hnorm (x - y) < ε
    with Infinitesimal-epsilon have x - y ∈ Infinitesimal
    by (rule hnorm-less-Infinitesimal)
    then have x ≈ y
    by (unfold approx-def)
    with f have starfun f x ≈ starfun f y
    by (simp add: isNSUCont-def)
    then have starfun f x - starfun f y ∈ Infinitesimal
    by (unfold approx-def)
  
```

```

    then show hnorm (starfun f x - starfun f y) < star-of r
    using r by (rule InfinitesimalD2)
qed
then show  $\exists s > 0. \forall x y. \text{norm } (x - y) < s \longrightarrow \text{norm } (f x - f y) < r$ 
    by transfer
qed
end

```

13 Differentiation (Nonstandard)

```

theory HDeriv
  imports HLim
begin

```

Nonstandard Definitions.

```

definition nsderiv :: [a::real-normed-field  $\Rightarrow$  'a, 'a, 'a]  $\Rightarrow$  bool
  ( $\langle \langle \text{notation} = \langle \text{mixfix NSDERIV} \rangle \rangle \text{NSDERIV } (-) / (-) / :> (-) \rangle [1000, 1000, 60]$ 
  60)
  where NSDERIV f x :> D  $\longleftrightarrow$ 
    ( $\forall h \in \text{Infinitesimal} - \{0\}. (( *f* f)(\text{star-of } x + h) - \text{star-of } (f x)) / h \approx$ 
    star-of D)

```

```

definition NSdifferentiable :: [a::real-normed-field  $\Rightarrow$  'a, 'a]  $\Rightarrow$  bool
  (infixl  $\langle \text{NSdifferentiable} \rangle$  60)
  where f NSdifferentiable x  $\longleftrightarrow (\exists D. \text{NSDERIV } f x :> D)$ 

```

```

definition increment :: (real  $\Rightarrow$  real)  $\Rightarrow$  real  $\Rightarrow$  hypreal  $\Rightarrow$  hypreal
  where increment f x h =
    (SOME inc. f NSdifferentiable x  $\wedge$  inc = ( *f* f) (hypreal-of-real x + h) -
    hypreal-of-real (f x))

```

13.1 Derivatives

```

lemma DERIV-NS-iff: (DERIV f x :> D)  $\longleftrightarrow (\lambda h. (f (x + h) - f x) / h) - 0 \rightarrow_{NS} D$ 

```

```

  by (simp add: DERIV-def LIM-NSLIM-iff)

```

```

lemma NS-DERIV-D: DERIV f x :> D  $\implies (\lambda h. (f (x + h) - f x) / h) - 0 \rightarrow_{NS} D$ 

```

```

  by (simp add: DERIV-def LIM-NSLIM-iff)

```

```

lemma Infinitesimal-of-hypreal:

```

```

  x  $\in$  Infinitesimal  $\implies (( *f* \text{ of-real } x :: 'a :: \text{real-normed-div-algebra star}) \in \text{Infinitesimal}$ 

```

```

  by (metis Infinitesimal-of-hypreal-iff of-hypreal-def)

```

```

lemma of-hypreal-eq-0-iff:  $\bigwedge x. (( *f* \text{ of-real } x = (0 :: 'a :: \text{real-algebra-1 star})) =$ 
  (x = 0)

```

by *transfer (rule of-real-eq-0-iff)*

lemma *NSDeriv-unique:*

assumes $NSDERIV\ f\ x\ :\>\ D\ NSDERIV\ f\ x\ :\>\ E$

shows $NSDERIV\ f\ x\ :\>\ D \implies NSDERIV\ f\ x\ :\>\ E \implies D = E$

proof –

have $\exists s. (s::'a\ star) \in Infinitesimal - \{0\}$

by (*metis Diff-iff HDeriv.of-hypreal-eq-0-iff Infinitesimal-epsilon Infinitesimal-of-hypreal epsilon-not-zero singletonD*)

with *assms show ?thesis*

by (*meson approx-trans3 nsderiv-def star-of-approx-iff*)

qed

First *NSDERIV* in terms of *NSLIM*.

First equivalence.

lemma *NSDERIV-NSLIM-iff:* $(NSDERIV\ f\ x\ :\>\ D) \longleftrightarrow (\lambda h. (f\ (x + h) - f\ x) / h) - 0 \rightarrow_{NS} D$

by (*auto simp add: nsderiv-def NSLIM-def starfun-lambda-cancel mem-infmal-iff*)

Second equivalence.

lemma *NSDERIV-NSLIM-iff2:* $(NSDERIV\ f\ x\ :\>\ D) \longleftrightarrow (\lambda z. (f\ z - f\ x) / (z - x)) - x \rightarrow_{NS} D$

by (*simp add: NSDERIV-NSLIM-iff DERIV-LIM-iff LIM-NSLIM-iff [symmetric]*)

While we’re at it!

lemma *NSDERIV-iff2:*

$(NSDERIV\ f\ x\ :\>\ D) \longleftrightarrow$

$(\forall w. w \neq star\text{-}of\ x \wedge w \approx star\text{-}of\ x \longrightarrow (*f* (\lambda z. (f\ z - f\ x) / (z - x))) w \approx star\text{-}of\ D)$

by (*simp add: NSDERIV-NSLIM-iff2 NSLIM-def*)

lemma *NSDERIVD5:*

$\llbracket NSDERIV\ f\ x\ :\>\ D; u \approx hypreal\text{-}of\text{-}real\ x \rrbracket \implies$

$(*f* (\lambda z. f\ z - f\ x))\ u \approx hypreal\text{-}of\text{-}real\ D * (u - hypreal\text{-}of\text{-}real\ x)$

unfolding *NSDERIV-iff2*

apply (*case-tac u = hypreal-of-real x, auto*)

by (*metis (mono-tags, lifting) HFinite-star-of Infinitesimal-ratio approx-def approx-minus-iff approx-mult-subst approx-star-of-HFinite approx-sym mult-zero-right right-minus-eq*)

lemma *NSDERIVD4:*

$\llbracket NSDERIV\ f\ x\ :\>\ D; h \in Infinitesimal \rrbracket$

$\implies (*f* f)(hypreal\text{-}of\text{-}real\ x + h) - hypreal\text{-}of\text{-}real\ (f\ x) \approx hypreal\text{-}of\text{-}real\ D * h$

apply (*clarsimp simp add: nsderiv-def*)

apply (*case-tac h = 0, simp*)

by (*meson DiffI Infinitesimal-approx Infinitesimal-ratio Infinitesimal-star-of-mult2 approx-star-of-HFinite singletonD*)

Differentiability implies continuity nice and simple "algebraic" proof.

lemma *NSDERIV-isNSCont*:

assumes *NSDERIV* $f\ x :> D$ **shows** *isNSCont* $f\ x$

unfolding *isNSCont-NSLIM-iff* *NSLIM-def*

proof *clarify*

fix x'

assume $x' \neq \text{star-of } x \approx \text{star-of } x$

then have $m0: x' - \text{star-of } x \in \text{Infinitesimal} - \{0\}$

using *bex-Infinitesimal-iff* **by** *auto*

then have $((\ast f \ast f)\ x' - \text{star-of } (f\ x)) / (x' - \text{star-of } x) \approx \text{star-of } D$

by (*metis* $\langle x' \approx \text{star-of } x \rangle$ *add-diff-cancel-left'* *assms* *bex-Infinitesimal-iff2* *ns-deriv-def*)

then have $((\ast f \ast f)\ x' - \text{star-of } (f\ x)) / (x' - \text{star-of } x) \in \text{HFinite}$

by (*metis* *approx-star-of-HFinite*)

then show $(\ast f \ast f)\ x' \approx \text{star-of } (f\ x)$

by (*metis* (*no-types*) *Diff-iff* *Infinitesimal-ratio* $m0$ *bex-Infinitesimal-iff* *insert-iff*)

qed

Differentiation rules for combinations of functions follow from clear, straightforward, algebraic manipulations.

Constant function.

lemma *NSDERIV-const* [*simp*]: *NSDERIV* $(\lambda x. k)\ x :> 0$

by (*simp* *add: NSDERIV-NSLIM-iff*)

Sum of functions- proved easily.

lemma *NSDERIV-add*:

assumes *NSDERIV* $f\ x :> Da$ *NSDERIV* $g\ x :> Db$

shows *NSDERIV* $(\lambda x. f\ x + g\ x)\ x :> Da + Db$

proof –

have $((\lambda x. f\ x + g\ x)\ \text{has-field-derivative } Da + Db)\ (at\ x)$

using *assms* *DERIV-NS-iff* *NSDERIV-NSLIM-iff* *field-differentiable-add* **by**

blast

then show *?thesis*

by (*simp* *add: DERIV-NS-iff* *NSDERIV-NSLIM-iff*)

qed

Product of functions - Proof is simple.

lemma *NSDERIV-mult*:

assumes *NSDERIV* $g\ x :> Db$ *NSDERIV* $f\ x :> Da$

shows *NSDERIV* $(\lambda x. f\ x \ast g\ x)\ x :> (Da \ast g\ x) + (Db \ast f\ x)$

proof –

have $(f\ \text{has-field-derivative } Da)\ (at\ x)\ (g\ \text{has-field-derivative } Db)\ (at\ x)$

using *assms* **by** (*simp-all* *add: DERIV-NS-iff* *NSDERIV-NSLIM-iff*)

then have $((\lambda a. f\ a \ast g\ a)\ \text{has-field-derivative } Da \ast g\ x + Db \ast f\ x)\ (at\ x)$

using *DERIV-mult* **by** *blast*

then show *?thesis*

by (*simp add: DERIV-NS-iff NSDERIV-NSLIM-iff*)
qed

Multiplying by a constant.

lemma *NSDERIV-cmult*: $NSDERIV\ f\ x\ :\>\ D \implies NSDERIV\ (\lambda x. c * f\ x)\ x\ :\>\ c * D$

unfolding *times-divide-eq-right* [*symmetric*] *NSDERIV-NSLIM-iff*
minus-mult-right right-diff-distrib [*symmetric*]
by (*erule NSLIM-const* [*THEN NSLIM-mult*])

Negation of function.

lemma *NSDERIV-minus*: $NSDERIV\ f\ x\ :\>\ D \implies NSDERIV\ (\lambda x. - f\ x)\ x\ :\>\ - D$

proof (*simp add: NSDERIV-NSLIM-iff*)

assume $(\lambda h. (f\ (x + h) - f\ x) / h) - 0 \rightarrow_{NS} D$

then have *deriv*: $(\lambda h. - ((f\ (x + h) - f\ x) / h)) - 0 \rightarrow_{NS} - D$

by (*rule NSLIM-minus*)

have $\forall h. - ((f\ (x + h) - f\ x) / h) = (- f\ (x + h) + f\ x) / h$

by (*simp add: minus-divide-left*)

with *deriv* **have** $(\lambda h. (- f\ (x + h) + f\ x) / h) - 0 \rightarrow_{NS} - D$

by *simp*

then show $(\lambda h. (f\ (x + h) - f\ x) / h) - 0 \rightarrow_{NS} D \implies (\lambda h. (f\ x - f\ (x + h)) / h) - 0 \rightarrow_{NS} - D$

by *simp*

qed

Subtraction.

lemma *NSDERIV-add-minus*:

$NSDERIV\ f\ x\ :\>\ Da \implies NSDERIV\ g\ x\ :\>\ Db \implies NSDERIV\ (\lambda x. f\ x + - g\ x)\ x\ :\>\ Da + - Db$

by (*blast dest: NSDERIV-add NSDERIV-minus*)

lemma *NSDERIV-diff*:

$NSDERIV\ f\ x\ :\>\ Da \implies NSDERIV\ g\ x\ :\>\ Db \implies NSDERIV\ (\lambda x. f\ x - g\ x)\ x\ :\>\ Da - Db$

using *NSDERIV-add-minus* [*of f x Da g Db*] **by** *simp*

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.

13.2 Lemmas

lemma *NSDERIV-zero*:

$\llbracket NSDERIV\ g\ x\ :\>\ D; (*f* g)\ (star-of\ x + y) = star-of\ (g\ x); y \in Infinitesimal; y \neq 0 \rrbracket$

$\implies D = 0$
by (*force simp add: nsderiv-def*)

Can be proved differently using *NSLIM-isCont-iff*.

lemma *NSDERIV-approx*:
 $NSDERIV f x :> D \implies h \in Infinitesimal \implies h \neq 0 \implies$
 $(\ast f \ast f) (star-of x + h) - star-of (f x) \approx 0$
by (*meson DiffI Infinitesimal-ratio approx-star-of-HFinite mem-infmal-iff ns-deriv-def singletonD*)

From one version of differentiability

$f x - f a \text{ ----- } \approx Db x - a$

lemma *NSDERIVD1*:
 $\llbracket NSDERIV f (g x) :> Da;$
 $(\ast f \ast g) (star-of x + y) \neq star-of (g x);$
 $(\ast f \ast g) (star-of x + y) \approx star-of (g x) \rrbracket$
 $\implies ((\ast f \ast f) ((\ast f \ast g) (star-of x + y)) -$
 $star-of (f (g x))) / ((\ast f \ast g) (star-of x + y) - star-of (g x)) \approx$
 $star-of Da$
by (*auto simp add: NSDERIV-NSLIM-iff2 NSLIM-def*)

From other version of differentiability

$f (x + h) - f x \text{ ----- } \approx Db h$

lemma *NSDERIVD2*: $\llbracket NSDERIV g x :> Db; y \in Infinitesimal; y \neq 0 \rrbracket$
 $\implies ((\ast f \ast g) (star-of(x) + y) - star-of(g x)) / y$
 $\approx star-of(Db)$
by (*auto simp add: NSDERIV-NSLIM-iff NSLIM-def mem-infmal-iff starfun-lambda-cancel*)

This proof uses both definitions of differentiability.

lemma *NSDERIV-chain*:
 $NSDERIV f (g x) :> Da \implies NSDERIV g x :> Db \implies NSDERIV (f \circ g) x :>$
 $Da \ast Db$
using *DERIV-NS-iff DERIV-chain NSDERIV-NSLIM-iff* **by** *blast*

Differentiation of natural number powers.

lemma *NSDERIV-Id* [*simp*]: $NSDERIV (\lambda x. x) x :> 1$
by (*simp add: NSDERIV-NSLIM-iff NSLIM-def del: divide-self-if*)

lemma *NSDERIV-cmult-Id* [*simp*]: $NSDERIV ((\ast) c) x :> c$
using *NSDERIV-Id [THEN NSDERIV-cmult]* **by** *simp*

lemma *NSDERIV-inverse*:
fixes $x :: 'a::real-normed-field$
assumes $x \neq 0$ — can’t get rid of $x \neq 0$ because it isn’t continuous at zero
shows $NSDERIV (\lambda x. inverse x) x :> - (inverse x \wedge Suc (Suc 0))$
proof —
{

```

fix h :: 'a star
assume h-Inf: h ∈ Infinitesimal
from this assms have not-0: star-of x + h ≠ 0
  by (rule Infinitesimal-add-not-zero)
assume h ≠ 0
from h-Inf have h * star-of x ∈ Infinitesimal
  by (rule Infinitesimal-HFinite-mult) simp
with assms have inverse (− (h * star-of x) + − (star-of x * star-of x)) ≈
  inverse (− (star-of x * star-of x))
proof −
  have − (h * star-of x) + − (star-of x * star-of x) ≈ − (star-of x * star-of x)
    using ⟨h * star-of x ∈ Infinitesimal⟩ assms bex-Infinitesimal-iff by fastforce
  then show ?thesis
    by (metis assms mult-eq-0-iff neg-equal-0-iff-equal star-of-approx-inverse
star-of-minus star-of-mult)
qed
moreover from not-0 ⟨h ≠ 0⟩ assms
have inverse (− (h * star-of x) + − (star-of x * star-of x))
  = (inverse (star-of x + h) − inverse (star-of x)) / h
  by (simp add: division-ring-inverse-diff inverse-mult-distrib [symmetric]
inverse-minus-eq [symmetric] algebra-simps)
ultimately have (inverse (star-of x + h) − inverse (star-of x)) / h ≈
  − (inverse (star-of x) * inverse (star-of x))
  using assms by simp
}
then show ?thesis by (simp add: nsderiv-def)
qed

```

13.2.1 Equivalence of NS and Standard definitions

lemma *divideR-eq-divide*: $x /_R y = x / y$
 by (simp add: divide-inverse mult.commute)

Now equivalence between *NSDERIV* and *DERIV*.

lemma *NSDERIV-DERIV-iff*: $NSDERIV f x :> D \longleftrightarrow DERIV f x :> D$
 by (simp add: DERIV-def NSDERIV-NSLIM-iff LIM-NSLIM-iff)

NS version.

lemma *NSDERIV-pow*: $NSDERIV (\lambda x. x ^ n) x :> real\ n * (x ^ (n - Suc\ 0))$
 by (simp add: NSDERIV-DERIV-iff DERIV-pow)

Derivative of inverse.

lemma *NSDERIV-inverse-fun*:
 $NSDERIV f x :> d \implies f x \neq 0 \implies$
 $NSDERIV (\lambda x. inverse (f x)) x :> (− (d * inverse (f x ^ Suc (Suc 0))))$
 for $x :: 'a :: \{real-normed-field\}$
 by (simp add: NSDERIV-DERIV-iff DERIV-inverse-fun del: power-Suc)

Derivative of quotient.

lemma *NSDERIV-quotient*:

fixes $x :: 'a::\text{real-normed-field}$

shows $\text{NSDERIV } f \, x :> d \implies \text{NSDERIV } g \, x :> e \implies g \, x \neq 0 \implies$
 $\text{NSDERIV } (\lambda y. f \, y / g \, y) \, x :> (d * g \, x - (e * f \, x)) / (g \, x \wedge \text{Suc } (\text{Suc } 0))$
by (*simp add: NSDERIV-DERIV-iff DERIV-quotient del: power-Suc*)

lemma *CARAT-NSDERIV*:

$\text{NSDERIV } f \, x :> l \implies \exists g. (\forall z. f \, z - f \, x = g \, z * (z - x)) \wedge \text{isNSCont } g \, x \wedge g \, x = l$

by (*simp add: CARAT-DERIV NSDERIV-DERIV-iff isNSCont-isCont-iff*)

lemma *hypreal-eq-minus-iff3*: $x = y + z \longleftrightarrow x + -z = y$

for $x \, y \, z :: \text{hypreal}$

by *auto*

lemma *CARAT-DERIVD*:

assumes *all*: $\forall z. f \, z - f \, x = g \, z * (z - x)$

and *nsc*: $\text{isNSCont } g \, x$

shows $\text{NSDERIV } f \, x :> g \, x$

proof –

from *nsc* **have** $\forall w. w \neq \text{star-of } x \wedge w \approx \text{star-of } x \longrightarrow$

$(*f* g) \, w * (w - \text{star-of } x) / (w - \text{star-of } x) \approx \text{star-of } (g \, x)$

by (*simp add: isNSCont-def*)

with *all* **show** *?thesis*

by (*simp add: NSDERIV-iff2 starfun-if-eq cong: if-cong*)

qed

13.2.2 Differentiability predicate

lemma *NSdifferentiableD*: $f \, \text{NSdifferentiable } x \implies \exists D. \text{NSDERIV } f \, x :> D$

by (*simp add: NSdifferentiable-def*)

lemma *NSdifferentiableI*: $\text{NSDERIV } f \, x :> D \implies f \, \text{NSdifferentiable } x$

by (*force simp add: NSdifferentiable-def*)

13.3 (NS) Increment

lemma *incrementI*:

$f \, \text{NSdifferentiable } x \implies$

$\text{increment } f \, x \, h = (*f* f) (\text{hypreal-of-real } x + h) - \text{hypreal-of-real } (f \, x)$

by (*simp add: increment-def*)

lemma *incrementI2*:

$\text{NSDERIV } f \, x :> D \implies$

$\text{increment } f \, x \, h = (*f* f) (\text{hypreal-of-real } x + h) - \text{hypreal-of-real } (f \, x)$

by (*erule NSdifferentiableI [THEN incrementI]*)

The Increment theorem – Keisler p. 65.

lemma *increment-thm*:

```

assumes NSDERIV f x :> D h ∈ Infinitesimal h ≠ 0
shows ∃ e ∈ Infinitesimal. increment f x h = hypreal-of-real D * h + e * h
proof –
  have inc: increment f x h = ( *f* f ) (hypreal-of-real x + h) – hypreal-of-real (f
x)
    using assms(1) incrementI2 by auto
  have (( *f* f ) (hypreal-of-real x + h) – hypreal-of-real (f x)) / h ≈ hypreal-of-real
D
    by (simp add: NSDERIVD2 assms)
  then obtain y where y ∈ Infinitesimal
    (( *f* f ) (hypreal-of-real x + h) – hypreal-of-real (f x)) / h = hypreal-of-real D
+ y
    by (metis bex-Infinitesimal-iff2)
  then have increment f x h / h = hypreal-of-real D + y
    by (metis inc)
  then show ?thesis
    by (metis (no-types) ⟨y ∈ Infinitesimal⟩ ⟨h ≠ 0⟩ distrib-right mult.commute
nonzero-mult-div-cancel-left times-divide-eq-right)
qed

```

```

lemma increment-approx-zero: NSDERIV f x :> D ⇒ h ≈ 0 ⇒ h ≠ 0 ⇒
increment f x h ≈ 0

```

```

  by (simp add: NSDERIV-approx incrementI2 mem-infmal-iff)

```

end

14 Nonstandard Extensions of Transcendental Functions

```

theory HTranscendental
imports Complex-Main HSeries HDeriv
begin

```

definition

```

  exphr :: real ⇒ hypreal where
    — define exponential function using standard part
    exphr x ≡ st(sumhr (0, whn, λn. inverse (fact n) * (x ^ n)))

```

definition

```

  sinhr :: real ⇒ hypreal where
    sinhr x ≡ st(sumhr (0, whn, λn. sin-coeff n * x ^ n))

```

definition

```

  coshr :: real ⇒ hypreal where
    coshr x ≡ st(sumhr (0, whn, λn. cos-coeff n * x ^ n))

```

14.1 Nonstandard Extension of Square Root Function

lemma *STAR-sqrt-zero* [simp]: $(\text{*f* sqrt})\ 0 = 0$
by (simp add: starfun star-n-zero-num)

lemma *STAR-sqrt-one* [simp]: $(\text{*f* sqrt})\ 1 = 1$
by (simp add: starfun star-n-one-num)

lemma *hypreal-sqrt-pow2-iff*: $((\text{*f* sqrt})(x))^2 = x = (0 \leq x)$

proof (cases x)

case (star-n X)

then show ?thesis

by (simp add: star-n-le star-n-zero-num starfun hrealpow star-n-eq-iff del:
 hpowr-Suc power-Suc)

qed

lemma *hypreal-sqrt-gt-zero-pow2*: $\bigwedge x. 0 < x \implies (\text{*f* sqrt})(x)^2 = x$
by transfer simp

lemma *hypreal-sqrt-pow2-gt-zero*: $0 < x \implies 0 < (\text{*f* sqrt})(x)^2$
by (frule hypreal-sqrt-gt-zero-pow2, auto)

lemma *hypreal-sqrt-not-zero*: $0 < x \implies (\text{*f* sqrt})(x) \neq 0$
using hypreal-sqrt-gt-zero-pow2 **by** fastforce

lemma *hypreal-inverse-sqrt-pow2*:

$0 < x \implies \text{inverse}((\text{*f* sqrt})(x))^2 = \text{inverse } x$

by (simp add: hypreal-sqrt-gt-zero-pow2 power-inverse)

lemma *hypreal-sqrt-mult-distrib*:

$\bigwedge x\ y. [0 < x; 0 < y] \implies$

$(\text{*f* sqrt})(x*y) = (\text{*f* sqrt})(x) * (\text{*f* sqrt})(y)$

by transfer (auto intro: real-sqrt-mult)

lemma *hypreal-sqrt-mult-distrib2*:

$[0 \leq x; 0 \leq y] \implies (\text{*f* sqrt})(x*y) = (\text{*f* sqrt})(x) * (\text{*f* sqrt})(y)$

by (auto intro: hypreal-sqrt-mult-distrib simp add: order-le-less)

lemma *hypreal-sqrt-approx-zero* [simp]:

assumes $0 < x$

shows $((\text{*f* sqrt})\ x \approx 0) \longleftrightarrow (x \approx 0)$

proof –

have $(\text{*f* sqrt})\ x \in \text{Infinitesimal} \longleftrightarrow ((\text{*f* sqrt})\ x)^2 \in \text{Infinitesimal}$

by (metis Infinitesimal-hrealpow pos2 power2-eq-square Infinitesimal-square-iff)

also have $\dots \longleftrightarrow x \in \text{Infinitesimal}$

by (simp add: assms hypreal-sqrt-gt-zero-pow2)

finally show ?thesis

using mem-infmal-iff **by** blast

qed

lemma *hypreal-sqrt-approx-zero2* [simp]:
 $0 \leq x \implies ((*f* \text{ sqrt})(x) \approx 0) = (x \approx 0)$
 by (auto simp add: order-le-less)

lemma *hypreal-sqrt-gt-zero*: $\bigwedge x. 0 < x \implies 0 < (*f* \text{ sqrt})(x)$
 by transfer (simp add: real-sqrt-gt-zero)

lemma *hypreal-sqrt-ge-zero*: $0 \leq x \implies 0 \leq (*f* \text{ sqrt})(x)$
 by (auto intro: hypreal-sqrt-gt-zero simp add: order-le-less)

lemma *hypreal-sqrt-lessI*:
 $\bigwedge x u. \llbracket 0 < u; x < u^2 \rrbracket \implies (*f* \text{ sqrt}) x < u$
proof transfer
 show $\bigwedge x u. \llbracket 0 < u; x < u^2 \rrbracket \implies \text{sqrt } x < u$
 by (metis less-le real-sqrt-less-iff real-sqrt-pow2 real-sqrt-power)
qed

lemma *hypreal-sqrt-hrabs* [simp]: $\bigwedge x. (*f* \text{ sqrt})(x^2) = |x|$
 by transfer simp

lemma *hypreal-sqrt-hrabs2* [simp]: $\bigwedge x. (*f* \text{ sqrt})(x*x) = |x|$
 by transfer simp

lemma *hypreal-sqrt-hyperpow-hrabs* [simp]:
 $\bigwedge x. (*f* \text{ sqrt})(x \text{ pow } (\text{hypnat-of-nat } 2)) = |x|$
 by transfer simp

lemma *star-sqrt-HFinite*: $\llbracket x \in \text{HFinite}; 0 \leq x \rrbracket \implies (*f* \text{ sqrt}) x \in \text{HFinite}$
 by (metis HFinite-square-iff hypreal-sqrt-pow2-iff power2-eq-square)

lemma *st-hypreal-sqrt*:
 assumes $x \in \text{HFinite}$ $0 \leq x$
 shows $st((*f* \text{ sqrt}) x) = (*f* \text{ sqrt})(st x)$
proof (rule power-inject-base)
 show $st((*f* \text{ sqrt}) x) \wedge \text{Suc } 1 = (*f* \text{ sqrt})(st x) \wedge \text{Suc } 1$
 using assms hypreal-sqrt-pow2-iff [of x]
 by (metis HFinite-square-iff hypreal-sqrt-hrabs2 power2-eq-square st-hrabs st-mult)
 show $0 \leq st((*f* \text{ sqrt}) x)$
 by (simp add: assms hypreal-sqrt-ge-zero st-zero-le star-sqrt-HFinite)
 show $0 \leq (*f* \text{ sqrt})(st x)$
 by (simp add: assms hypreal-sqrt-ge-zero st-zero-le)
qed

lemma *hypreal-sqrt-sum-squares-ge1* [simp]: $\bigwedge x y. x \leq (*f* \text{ sqrt})(x^2 + y^2)$
 by transfer (rule real-sqrt-sum-squares-ge1)

lemma *HFinite-hypreal-sqrt-imp-HFinite*:
 $\llbracket 0 \leq x; (*f* \text{ sqrt}) x \in \text{HFinite} \rrbracket \implies x \in \text{HFinite}$
 by (metis HFinite-mult hypreal-sqrt-pow2-iff power2-eq-square)

lemma *HFinite-hypreal-sqrt-iff* [simp]:

$0 \leq x \implies ((*f* \text{ sqrt}) x \in \text{HFinite}) = (x \in \text{HFinite})$

by (blast intro: star-sqrt-HFinite HFinite-hypreal-sqrt-imp-HFinite)

lemma *Infinesimal-hypreal-sqrt*:

$\llbracket 0 \leq x; x \in \text{Infinesimal} \rrbracket \implies (*f* \text{ sqrt}) x \in \text{Infinesimal}$

by (simp add: mem-infmal-iff)

lemma *Infinesimal-hypreal-sqrt-imp-Infinesimal*:

$\llbracket 0 \leq x; (*f* \text{ sqrt}) x \in \text{Infinesimal} \rrbracket \implies x \in \text{Infinesimal}$

using hypreal-sqrt-approx-zero2 mem-infmal-iff **by** blast

lemma *Infinesimal-hypreal-sqrt-iff* [simp]:

$0 \leq x \implies ((*f* \text{ sqrt}) x \in \text{Infinesimal}) = (x \in \text{Infinesimal})$

by (blast intro: Infinesimal-hypreal-sqrt-imp-Infinesimal Infinesimal-hypreal-sqrt)

lemma *HInfinite-hypreal-sqrt*:

$\llbracket 0 \leq x; x \in \text{HInfinite} \rrbracket \implies (*f* \text{ sqrt}) x \in \text{HInfinite}$

by (simp add: HInfinite-HFinite-iff)

lemma *HInfinite-hypreal-sqrt-imp-HInfinite*:

$\llbracket 0 \leq x; (*f* \text{ sqrt}) x \in \text{HInfinite} \rrbracket \implies x \in \text{HInfinite}$

using HFinite-hypreal-sqrt-iff HInfinite-HFinite-iff **by** blast

lemma *HInfinite-hypreal-sqrt-iff* [simp]:

$0 \leq x \implies ((*f* \text{ sqrt}) x \in \text{HInfinite}) = (x \in \text{HInfinite})$

by (blast intro: HInfinite-hypreal-sqrt HInfinite-hypreal-sqrt-imp-HInfinite)

lemma *HFinite-exp* [simp]:

$\text{sumhr } (0, \text{whn}, \lambda n. \text{inverse } (\text{fact } n) * x \wedge n) \in \text{HFinite}$

unfolding sumhr-app star-zero-def starfun2-star-of atLeast0LessThan

by (metis NSBseqD2 NSconvergent-NSBseq convergent-NSconvergent-iff summable-iff-convergent summable-exp)

lemma *exp-hr-zero* [simp]: $\text{exp-hr } 0 = 1$

proof –

have $\forall x > 1. 1 = \text{sumhr } (0, 1, \lambda n. \text{inverse } (\text{fact } n) * 0 \wedge n) + \text{sumhr } (1, x, \lambda n. \text{inverse } (\text{fact } n) * 0 \wedge n)$

unfolding sumhr-app **by** transfer (simp add: power-0-left)

then have $\text{sumhr } (0, 1, \lambda n. \text{inverse } (\text{fact } n) * 0 \wedge n) + \text{sumhr } (1, \text{whn}, \lambda n. \text{inverse } (\text{fact } n) * 0 \wedge n) \approx 1$

by auto

then show ?thesis

unfolding exp-hr-def

using sumhr-split-add [OF hypnat-one-less-hypnat-omega] st-unique **by** auto

qed

lemma *cosh-hr-zero* [simp]: $\text{cosh-hr } 0 = 1$

```

proof –
  have  $\forall x > 1. 1 = \text{sumhr } (0, 1, \lambda n. \text{cos-coeff } n * 0 \wedge n) + \text{sumhr } (1, x, \lambda n. \text{cos-coeff } n * 0 \wedge n)$ 
  unfolding sumhr-app by transfer (simp add: power-0-left)
  then have  $\text{sumhr } (0, 1, \lambda n. \text{cos-coeff } n * 0 \wedge n) + \text{sumhr } (1, \text{whn}, \lambda n. \text{cos-coeff } n * 0 \wedge n) \approx 1$ 
  by auto
  then show ?thesis
  unfolding cosh-def
  using sumhr-split-add [OF hypnat-one-less-hypnat-omega] st-unique by auto
qed

```

lemma *STAR-exp-zero-approx-one [simp]*: $(*f* \text{ exp}) (0::\text{hypreal}) \approx 1$

```

proof –
  have  $( *f* \text{ exp} ) (0::\text{real star}) = 1$ 
  by transfer simp
  then show ?thesis
  by auto
qed

```

lemma *STAR-exp-Infinitesimal*:

```

  assumes  $x \in \text{Infinitesimal}$  shows  $( *f* \text{ exp} ) (x::\text{hypreal}) \approx 1$ 
proof (cases  $x = 0$ )
  case False
  have NSDERIV exp 0 :> 1
  by (metis DERIV-exp NSDERIV-DERIV-iff exp-zero)
  then have  $(( *f* \text{ exp} ) x - 1) / x \approx 1$ 
  using nsderiv-def False NSDERIVD2 assms by fastforce
  then have  $( *f* \text{ exp} ) x - 1 \approx x$ 
  using NSDERIVD4 ⟨NSDERIV exp 0 :> 1⟩ assms by fastforce
  then show ?thesis
  by (meson Infinitesimal-approx approx-minus-iff approx-trans2 assms not-Infinitesimal-not-zero)
qed auto

```

lemma *STAR-exp-epsilon [simp]*: $(*f* \text{ exp}) \varepsilon \approx 1$

by (*auto intro: STAR-exp-Infinitesimal*)

lemma *STAR-exp-add*:

```

 $\bigwedge (x::'a:: \{\text{banach, real-normed-field}\} \text{ star}) y. ( *f* \text{ exp} )(x + y) = ( *f* \text{ exp} ) x * ( *f* \text{ exp} ) y$ 
by transfer (rule exp-add)

```

lemma *exp-hypreal-of-real-exp-eq*: $\text{exp-hr } x = \text{hypreal-of-real } (\text{exp } x)$

```

proof –
  have  $(\lambda n. \text{inverse } (\text{fact } n) * x \wedge n) \text{ sums } \text{exp } x$ 
  using exp-converges [of x] by simp
  then have  $(\lambda n. \sum n < n. \text{inverse } (\text{fact } n) * x \wedge n) \longrightarrow_{NS} \text{exp } x$ 
  using NSsums-def sums-NSsums-iff by blast
  then have  $\text{hypreal-of-real } (\text{exp } x) \approx \text{sumhr } (0, \text{whn}, \lambda n. \text{inverse } (\text{fact } n) * x \wedge n)$ 

```


n)
unfolding *starfunNat-sumr* [*symmetric*] *atLeast0LessThan*
using *HNatInfinite-wnn NSLIMSEQ-def approx-sym* **by** *blast*
then show *?thesis*
unfolding *exphr-def* **using** *st-eq-approx-iff* **by** *auto*
qed

lemma *starfun-exp-ge-add-one-self* [*simp*]: $\bigwedge x::\text{hypreal}. 0 \leq x \implies (1 + x) \leq (*f* \exp) x$
by *transfer (rule exp-ge-add-one-self-aux)*

exp maps infinities to infinities

lemma *starfun-exp-HInfinite*:
fixes $x::\text{hypreal}$
assumes $x \in HInfinite$ $0 \leq x$
shows $(*f* \exp) x \in HInfinite$
proof –
have $x \leq 1 + x$
by *simp*
also have $\dots \leq (*f* \exp) x$
by (*simp add: <0 ≤ x>*)
finally show *?thesis*
using *HInfinite-ge-HInfinite assms* **by** *blast*
qed

lemma *starfun-exp-minus*:
 $\bigwedge x::'a::\{\text{banach, real-normed-field}\} \text{star}. (*f* \exp) (-x) = \text{inverse}((*f* \exp) x)$
by *transfer (rule exp-minus)*

exp maps infinitesimals to infinitesimals

lemma *starfun-exp-Infinitesimal*:
fixes $x::\text{hypreal}$
assumes $x \in HInfinite$ $x \leq 0$
shows $(*f* \exp) x \in Infinitesimal$
proof –
obtain y **where** $x = -y$ $y \geq 0$
by (*metis abs-of-nonpos assms(2) eq-abs-iff'*)
then have $(*f* \exp) y \in HInfinite$
using *HInfinite-minus-iff assms(1) starfun-exp-HInfinite* **by** *blast*
then show *?thesis*
by (*simp add: HInfinite-inverse-Infinitesimal <x = - y> starfun-exp-minus*)
qed

lemma *starfun-exp-gt-one* [*simp*]: $\bigwedge x::\text{hypreal}. 0 < x \implies 1 < (*f* \exp) x$
by *transfer (rule exp-gt-one)*

abbreviation *real-ln* :: *real* \Rightarrow *real* **where**
real-ln $\equiv \ln$

lemma *starfun-ln-exp* [simp]: $\bigwedge x. (\text{*f* real-ln}) ((\text{*f* exp}) x) = x$
 by transfer (rule ln-exp)

lemma *starfun-exp-ln-iff* [simp]: $\bigwedge x. ((\text{*f* exp}) ((\text{*f* real-ln}) x) = x) = (0 < x)$
 by transfer (rule exp-ln-iff)

lemma *starfun-exp-ln-eq*: $\bigwedge u x. (\text{*f* exp}) u = x \implies (\text{*f* real-ln}) x = u$
 by transfer (rule ln-unique)

lemma *starfun-ln-less-self* [simp]: $\bigwedge x. 0 < x \implies (\text{*f* real-ln}) x < x$
 by transfer (rule ln-less-self)

lemma *starfun-ln-ge-zero* [simp]: $\bigwedge x. 1 \leq x \implies 0 \leq (\text{*f* real-ln}) x$
 by transfer (rule ln-ge-zero)

lemma *starfun-ln-gt-zero* [simp]: $\bigwedge x. 1 < x \implies 0 < (\text{*f* real-ln}) x$
 by transfer (rule ln-gt-zero)

lemma *starfun-ln-not-eq-zero* [simp]: $\bigwedge x. \llbracket 0 < x; x \neq 1 \rrbracket \implies (\text{*f* real-ln}) x \neq 0$
 by transfer simp

lemma *starfun-ln-HFinite*: $\llbracket x \in \text{HFinite}; 1 \leq x \rrbracket \implies (\text{*f* real-ln}) x \in \text{HFinite}$
 by (metis HFinite-HInfinite-iff less-le-trans starfun-exp-HInfinite starfun-exp-ln-iff
 starfun-ln-ge-zero zero-less-one)

lemma *starfun-ln-inverse*: $\bigwedge x. 0 < x \implies (\text{*f* real-ln}) (\text{inverse } x) = - (\text{*f* ln}) x$
 by transfer (rule ln-inverse)

lemma *starfun-abs-exp-cancel*: $\bigwedge x. | (\text{*f* exp}) (x :: \text{hypreal}) | = (\text{*f* exp}) x$
 by transfer (rule abs-exp-cancel)

lemma *starfun-exp-less-mono*: $\bigwedge x y :: \text{hypreal}. x < y \implies (\text{*f* exp}) x < (\text{*f* exp}) y$
 by transfer (rule exp-less-mono)

lemma *starfun-exp-HFinite*:
 fixes $x :: \text{hypreal}$
 assumes $x \in \text{HFinite}$
 shows $(\text{*f* exp}) x \in \text{HFinite}$
proof –
 obtain u where $u \in \mathbb{R} \mid x < u$
 using HFiniteD assms by force
 with assms have $| (\text{*f* exp}) x | < (\text{*f* exp}) u$
 using starfun-abs-exp-cancel starfun-exp-less-mono by auto
 with $\langle u \in \mathbb{R} \rangle$ show ?thesis
 by (force simp: HFinite-def Reals-eq-Standard)
qed

lemma *starfun-exp-add-HFinite-Infinesimal-approx*:

fixes $x :: \text{hypreal}$
shows $\llbracket x \in \text{Infinesimal}; z \in \text{HFinite} \rrbracket \implies (*f* \text{ exp}) (z + x::\text{hypreal}) \approx (*f* \text{ exp}) z$
using *STAR-exp-Infinesimal approx-mult2 starfun-exp-HFinite* **by** (*fastforce simp add: STAR-exp-add*)

lemma *starfun-ln-HInfinite*:

$\llbracket x \in \text{HInfinite}; 0 < x \rrbracket \implies (*f* \text{ real-ln}) x \in \text{HInfinite}$
by (*metis HInfinite-HFinite-iff starfun-exp-HFinite starfun-exp-ln-iff*)

lemma *starfun-exp-HInfinite-Infinesimal-disj*:

fixes $x :: \text{hypreal}$
shows $x \in \text{HInfinite} \implies (*f* \text{ exp}) x \in \text{HInfinite} \vee (*f* \text{ exp}) (x::\text{hypreal}) \in \text{Infinesimal}$
by (*meson linear starfun-exp-HInfinite starfun-exp-Infinesimal*)

lemma *starfun-ln-HFinite-not-Infinesimal*:

$\llbracket x \in \text{HFinite} - \text{Infinesimal}; 0 < x \rrbracket \implies (*f* \text{ real-ln}) x \in \text{HFinite}$
by (*metis DiffD1 DiffD2 HInfinite-HFinite-iff starfun-exp-HInfinite-Infinesimal-disj starfun-exp-ln-iff*)

lemma *starfun-ln-Infinesimal-HInfinite*:

assumes $x \in \text{Infinesimal}$ $0 < x$
shows $(*f* \text{ real-ln}) x \in \text{HInfinite}$
proof –
have *inverse* $x \in \text{HInfinite}$
using *Infinesimal-inverse-HInfinite assms* **by** *blast*
then show *?thesis*
using *HInfinite-minus-iff assms(2) starfun-ln-HInfinite starfun-ln-inverse* **by** *fastforce*
qed

lemma *starfun-ln-less-zero*: $\bigwedge x. \llbracket 0 < x; x < 1 \rrbracket \implies (*f* \text{ real-ln}) x < 0$

by *transfer (rule ln-less-zero)*

lemma *starfun-ln-Infinesimal-less-zero*:

$\llbracket x \in \text{Infinesimal}; 0 < x \rrbracket \implies (*f* \text{ real-ln}) x < 0$
by (*auto intro!: starfun-ln-less-zero simp add: Infinesimal-def*)

lemma *starfun-ln-HInfinite-gt-zero*:

$\llbracket x \in \text{HInfinite}; 0 < x \rrbracket \implies 0 < (*f* \text{ real-ln}) x$
by (*auto intro!: starfun-ln-gt-zero simp add: HInfinite-def*)

lemma *HFinite-sin [simp]*: $\text{sumhr } (0, \text{whn}, \lambda n. \text{sin-coeff } n * x \wedge n) \in \text{HFinite}$

proof –

have *summable* $(\lambda i. \text{sin-coeff } i * x \wedge i)$

```

    using summable-norm-sin [of x] by (simp add: summable-rabs-cancel)
  then have (*f* ( $\lambda n. \sum n < n. \text{sin-coeff } n * x ^ n$ )) whn  $\in \text{HFinite}$ 
    unfolding summable-sums-iff sums-NSsums-iff NSsums-def NSLIMSEQ-def
    using HFinite-star-of HNatInfinite-whn approx-HFinite approx-sym by blast
  then show ?thesis
    unfolding sumhr-app
    by (simp only: star-zero-def starfun2-star-of atLeast0LessThan)
qed

```

```

lemma STAR-sin-zero [simp]: (*f* sin) 0 = 0
  by transfer (rule sin-zero)

```

```

lemma STAR-sin-Infinitesimal [simp]:
  fixes x :: 'a::{real-normed-field,banach} star
  assumes x  $\in \text{Infinitesimal}$ 
  shows (*f* sin) x  $\approx x$ 
proof (cases x = 0)
  case False
  have NSDERIV sin 0  $> 1$ 
    by (metis DERIV-sin NSDERIV-DERIV-iff cos-zero)
  then have (*f* sin) x / x  $\approx 1$ 
    using False NSDERIVD2 assms by fastforce
  with assms show ?thesis
    unfolding star-one-def
    by (metis False Infinitesimal-approx Infinitesimal-ratio approx-star-of-HFinite)
qed auto

```

```

lemma HFinite-cos [simp]: sumhr (0, whn,  $\lambda n. \text{cos-coeff } n * x ^ n$ )  $\in \text{HFinite}$ 
proof -
  have summable ( $\lambda i. \text{cos-coeff } i * x ^ i$ )
    using summable-norm-cos [of x] by (simp add: summable-rabs-cancel)
  then have (*f* ( $\lambda n. \sum n < n. \text{cos-coeff } n * x ^ n$ )) whn  $\in \text{HFinite}$ 
    unfolding summable-sums-iff sums-NSsums-iff NSsums-def NSLIMSEQ-def
    using HFinite-star-of HNatInfinite-whn approx-HFinite approx-sym by blast
  then show ?thesis
    unfolding sumhr-app
    by (simp only: star-zero-def starfun2-star-of atLeast0LessThan)
qed

```

```

lemma STAR-cos-zero [simp]: (*f* cos) 0 = 1
  by transfer (rule cos-zero)

```

```

lemma STAR-cos-Infinitesimal [simp]:
  fixes x :: 'a::{real-normed-field,banach} star
  assumes x  $\in \text{Infinitesimal}$ 
  shows (*f* cos) x  $\approx 1$ 
proof (cases x = 0)
  case False
  have NSDERIV cos 0  $> 0$ 

```

by (metis DERIV-cos NSDERIV-DERIV-iff add.inverse-neutral sin-zero)
 then have $(\ast f \ast \cos) x - 1 \approx 0$
 using NSDERIV-approx assms by fastforce
 with assms show ?thesis
 using approx-minus-iff by blast
 qed auto

lemma STAR-tan-zero [simp]: $(\ast f \ast \tan) 0 = 0$
 by transfer (rule tan-zero)

lemma STAR-tan-Infinitesimal [simp]:
 assumes $x \in \text{Infinitesimal}$
 shows $(\ast f \ast \tan) x \approx x$
 proof (cases $x = 0$)
 case False
 have NSDERIV tan 0 :> 1
 using DERIV-tan [of 0] by (simp add: NSDERIV-DERIV-iff)
 then have $(\ast f \ast \tan) x / x \approx 1$
 using False NSDERIVD2 assms by fastforce
 with assms show ?thesis
 unfolding star-one-def
 by (metis False Infinitesimal-approx Infinitesimal-ratio approx-star-of-HFinite)
 qed auto

lemma STAR-sin-cos-Infinitesimal-mult:
 fixes $x :: 'a :: \{\text{real-normed-field}, \text{banach}\}$ star
 shows $x \in \text{Infinitesimal} \implies (\ast f \ast \sin) x \ast (\ast f \ast \cos) x \approx x$
 using approx-mult-HFinite [of $(\ast f \ast \sin) x - (\ast f \ast \cos) x 1$]
 by (simp add: Infinitesimal-subset-HFinite [THEN subsetD])

lemma HFinite-pi: $\text{hypreal-of-real } \pi \in \text{HFinite}$
 by simp

lemma STAR-sin-Infinitesimal-divide:
 fixes $x :: 'a :: \{\text{real-normed-field}, \text{banach}\}$ star
 shows $\llbracket x \in \text{Infinitesimal}; x \neq 0 \rrbracket \implies (\ast f \ast \sin) x / x \approx 1$
 using DERIV-sin [of 0::'a]
 by (simp add: NSDERIV-DERIV-iff [symmetric] nsderiv-def)

14.2 Proving $\sin \ast (1/n) \times 1/(1/n) \approx 1$ for $n = \infty$

lemma lemma-sin-pi:
 $n \in \text{HNatInfinite}$
 $\implies (\ast f \ast \sin) (\text{inverse } (\text{hypreal-of-hypnat } n)) / (\text{inverse } (\text{hypreal-of-hypnat } n))$
 ≈ 1
 by (simp add: STAR-sin-Infinitesimal-divide zero-less-HNatInfinite)

lemma STAR-sin-inverse-HNatInfinite:

$n \in \text{HNatInfinite}$
 $\implies (*f* \sin) (\text{inverse} (\text{hypreal-of-hypnat } n)) * \text{hypreal-of-hypnat } n \approx 1$
by (metis field-class.field-divide-inverse inverse-inverse-eq lemma-sin-pi)

lemma *Infinitesimal-pi-divide-HNatInfinite*:

$N \in \text{HNatInfinite}$
 $\implies \text{hypreal-of-real } \pi / (\text{hypreal-of-hypnat } N) \in \text{Infinitesimal}$
by (simp add: Infinitesimal-HFinite-mult2 field-class.field-divide-inverse)

lemma *pi-divide-HNatInfinite-not-zero* [simp]:

$N \in \text{HNatInfinite} \implies \text{hypreal-of-real } \pi / (\text{hypreal-of-hypnat } N) \neq 0$
by (simp add: zero-less-HNatInfinite)

lemma *STAR-sin-pi-divide-HNatInfinite-approx-pi*:

assumes $n \in \text{HNatInfinite}$
shows $(*f* \sin) (\text{hypreal-of-real } \pi / \text{hypreal-of-hypnat } n) * \text{hypreal-of-hypnat } n \approx$
 $\text{hypreal-of-real } \pi$
proof –
have $(*f* \sin) (\text{hypreal-of-real } \pi / \text{hypreal-of-hypnat } n) / (\text{hypreal-of-real } \pi / \text{hypreal-of-hypnat } n) \approx 1$
using *Infinitesimal-pi-divide-HNatInfinite STAR-sin-Infinitesimal-divide assms pi-divide-HNatInfinite-not-zero* **by** blast
then have $\text{hypreal-of-hypnat } n * \text{star-of } \sin \star (\text{hypreal-of-real } \pi / \text{hypreal-of-hypnat } n) / \text{hypreal-of-real } \pi \approx 1$
by (simp add: mult.commute starfun-def)
then show ?thesis
apply (simp add: starfun-def field-simps)
by (metis (no-types, lifting) approx-mult-subst-star-of approx-refl mult-cancel-right1 nonzero-eq-divide-eq pi-neq-zero star-of-eq-0)
qed

lemma *STAR-sin-pi-divide-HNatInfinite-approx-pi2*:

$n \in \text{HNatInfinite}$
 $\implies \text{hypreal-of-hypnat } n * (*f* \sin) (\text{hypreal-of-real } \pi / (\text{hypreal-of-hypnat } n)) \approx \text{hypreal-of-real } \pi$
by (metis STAR-sin-pi-divide-HNatInfinite-approx-pi mult.commute)

lemma *starfunNat-pi-divide-n-Infinitesimal*:

$N \in \text{HNatInfinite} \implies (*f* (\lambda x. \pi / \text{real } x)) N \in \text{Infinitesimal}$
by (simp add: Infinitesimal-HFinite-mult2 divide-inverse starfunNat-real-of-nat)

lemma *STAR-sin-pi-divide-n-approx*:

assumes $N \in \text{HNatInfinite}$
shows $(*f* \sin) ((*f* (\lambda x. \pi / \text{real } x)) N) \approx \text{hypreal-of-real } \pi / (\text{hypreal-of-hypnat } N)$
proof –
have $\exists s. (*f* \sin) ((*f* (\lambda n. \pi / \text{real } n)) N) \approx s \wedge \text{hypreal-of-real } \pi / \text{hypreal-of-hypnat } N \approx s$

by (metis (lifting) Infinitesimal-approx Infinitesimal-pi-divide-HNatInfinite STAR-sin-Infinitesimal
 assms starfunNat-pi-divide-n-Infinitesimal)
 then show ?thesis
 by (meson approx-trans2)
 qed

lemma NSLIMSEQ-sin-pi: $(\lambda n. \text{real } n * \sin (\pi / \text{real } n)) \longrightarrow_{NS} \pi$
 proof –
 have *: hypreal-of-hypnat $N * (*f* \sin) ((*f* (\lambda x. \pi / \text{real } x)) N) \approx \text{hypreal-of-real } \pi$
 if $N \in \text{HNatInfinite}$
 for $N :: \text{nat star}$
 using that
 by simp (metis STAR-sin-pi-divide-HNatInfinite-approx-pi2 starfunNat-real-of-nat)
 show ?thesis
 by (simp add: NSLIMSEQ-def starfunNat-real-of-nat) (metis * starfun-o2)
 qed

lemma NSLIMSEQ-cos-one: $(\lambda n. \cos (\pi / \text{real } n)) \longrightarrow_{NS} 1$
 proof –
 have $(*f* \cos) ((*f* (\lambda x. \pi / \text{real } x)) N) \approx 1$
 if $N \in \text{HNatInfinite}$ for N
 using that STAR-cos-Infinitesimal starfunNat-pi-divide-n-Infinitesimal by blast
 then show ?thesis
 by (simp add: NSLIMSEQ-def) (metis STAR-cos-Infinitesimal starfunNat-pi-divide-n-Infinitesimal
 starfun-o2)
 qed

lemma NSLIMSEQ-sin-cos-pi:
 $(\lambda n. \text{real } n * \sin (\pi / \text{real } n) * \cos (\pi / \text{real } n)) \longrightarrow_{NS} \pi$
 using NSLIMSEQ-cos-one NSLIMSEQ-mult NSLIMSEQ-sin-pi by force

A familiar approximation to $\cos x$ when x is small

lemma STAR-cos-Infinitesimal-approx:
 fixes $x :: 'a :: \{\text{real-normed-field}, \text{banach}\} \text{ star}$
 shows $x \in \text{Infinitesimal} \implies (*f* \cos) x \approx 1 - x^2$
 by (metis Infinitesimal-square-iff STAR-cos-Infinitesimal approx-diff approx-sym
 diff-zero mem-infmal-iff power2-eq-square)

lemma STAR-cos-Infinitesimal-approx2:
 fixes $x :: \text{hypreal}$
 assumes $x \in \text{Infinitesimal}$
 shows $(*f* \cos) x \approx 1 - (x^2)/2$
 proof –
 have $1 \approx 1 - x^2 / 2$
 using assms
 by (auto intro: Infinitesimal-SReal-divide simp add: Infinitesimal-approx-minus
 [symmetric] numeral-2-eq-2)
 then show ?thesis

```

    using STAR-cos-Infinitesimal approx-trans assms by blast
qed

end

```

15 Non-Standard Complex Analysis

```

theory NSCA
imports NSComplex HTranscendental
begin

```

abbreviation

```

    SComplex :: hcomplex set where
    SComplex  $\equiv$  Standard

```

definition — standard part map

```

    stc :: hcomplex => hcomplex where
    stc x = (SOME r. x  $\in$  HFinite  $\wedge$  r  $\in$  SComplex  $\wedge$  r  $\approx$  x)

```

15.1 Closure Laws for SComplex, the Standard Complex Numbers

lemma *SComplex-minus-iff* [simp]: $(-x \in SComplex) = (x \in SComplex)$
 using *Standard-minus* by fastforce

lemma *SComplex-add-cancel*:

```

     $\llbracket x + y \in SComplex; y \in SComplex \rrbracket \implies x \in SComplex$ 
    using Standard-diff by fastforce

```

lemma *SReal-hcmod-hcomplex-of-complex* [simp]:

```

    hcmod (hcomplex-of-complex r)  $\in$   $\mathbb{R}$ 
    by (simp add: Reals-eq-Standard)

```

lemma *SReal-hcmod-numeral*: $hcmod \text{ (numeral } w :: hcomplex) \in \mathbb{R}$
 by simp

lemma *SReal-hcmod-SComplex*: $x \in SComplex \implies hcmod x \in \mathbb{R}$
 by (simp add: Reals-eq-Standard)

lemma *SComplex-divide-numeral*:

```

     $r \in SComplex \implies r / (\text{numeral } w :: hcomplex) \in SComplex$ 
    by simp

```

lemma *SComplex-UNIV-complex*:

```

     $\{x. \text{hcomplex-of-complex } x \in SComplex\} = (\text{UNIV} :: \text{complex set})$ 
    by simp

```

lemma *SComplex-iff*: $(x \in SComplex) = (\exists y. x = \text{hcomplex-of-complex } y)$

by (*simp add: Standard-def image-def*)

lemma *hcomplex-of-complex-image*:
 $\text{range } h\text{complex-of-complex} = S\text{Complex}$
by (*simp add: Standard-def*)

lemma *inv-hcomplex-of-complex-image*: $\text{inv } h\text{complex-of-complex } 'S\text{Complex} = \text{UNIV}$
by (*auto simp add: Standard-def image-def*) (*metis inj-star-of inv-f-f*)

lemma *SComplex-hcomplex-of-complex-image*:
 $\llbracket \exists x. x \in P; P \leq S\text{Complex} \rrbracket \implies \exists Q. P = h\text{complex-of-complex } 'Q$
by (*metis Standard-def subset-imageE*)

lemma *SComplex-SReal-dense*:
 $\llbracket x \in S\text{Complex}; y \in S\text{Complex}; h\text{cmod } x < h\text{cmod } y \rrbracket \implies \exists r \in \text{Reals}. h\text{cmod } x < r \wedge r < h\text{cmod } y$
by (*simp add: SReal-dense SReal-hcmod-SComplex*)

15.2 The Finite Elements form a Subring

lemma *HFinite-hcmod-hcomplex-of-complex [simp]*:
 $h\text{cmod } (h\text{complex-of-complex } r) \in H\text{Finite}$
by (*auto intro!: SReal-subset-HFinite [THEN subsetD]*)

lemma *HFinite-hcmod-iff [simp]*: $h\text{cmod } x \in H\text{Finite} \longleftrightarrow x \in H\text{Finite}$
by (*simp add: HFinite-def*)

lemma *HFinite-bounded-hcmod*:
 $\llbracket x \in H\text{Finite}; y \leq h\text{cmod } x; 0 \leq y \rrbracket \implies y \in H\text{Finite}$
using *HFinite-bounded HFinite-hcmod-iff* **by** *blast*

15.3 The Complex Infinitesimals form a Subring

lemma *Infinitesimal-hcmod-iff*:
 $(z \in \text{Infinitesimal}) = (h\text{cmod } z \in \text{Infinitesimal})$
by (*simp add: Infinitesimal-def*)

lemma *HInfinite-hcmod-iff*: $(z \in H\text{Infinite}) = (h\text{cmod } z \in H\text{Infinite})$
by (*simp add: HInfinite-def*)

lemma *HFinite-diff-Infinitesimal-hcmod*:
 $x \in H\text{Finite} - \text{Infinitesimal} \implies h\text{cmod } x \in H\text{Finite} - \text{Infinitesimal}$
by (*simp add: Infinitesimal-hcmod-iff*)

lemma *hcmod-less-Infinitesimal*:
 $\llbracket e \in \text{Infinitesimal}; h\text{cmod } x < h\text{cmod } e \rrbracket \implies x \in \text{Infinitesimal}$
by (*auto elim: hrabs-less-Infinitesimal simp add: Infinitesimal-hcmod-iff*)

lemma *hcmod-le-Infinitesimal*:
 $\llbracket e \in \text{Infinitesimal}; h\text{cmod } x \leq h\text{cmod } e \rrbracket \implies x \in \text{Infinitesimal}$

by (auto elim: hrabs-le-Infinesimal simp add: Infinitesimal-hcmod-iff)

15.4 The “Infinitely Close” Relation

lemma *approx-SComplex-mult-cancel-zero*:

$\llbracket a \in SComplex; a \neq 0; a*x \approx 0 \rrbracket \implies x \approx 0$

by (metis Infinitesimal-mult-disj SComplex-iff mem-infmal-iff star-of-Infinesimal-iff-0 star-zero-def)

lemma *approx-mult-SComplex1*: $\llbracket a \in SComplex; x \approx 0 \rrbracket \implies x*a \approx 0$

using SComplex-iff approx-mult-subst-star-of by fastforce

lemma *approx-mult-SComplex2*: $\llbracket a \in SComplex; x \approx 0 \rrbracket \implies a*x \approx 0$

by (metis approx-mult-SComplex1 mult.commute)

lemma *approx-mult-SComplex-zero-cancel-iff* [simp]:

$\llbracket a \in SComplex; a \neq 0 \rrbracket \implies (a*x \approx 0) = (x \approx 0)$

using approx-SComplex-mult-cancel-zero approx-mult-SComplex2 by blast

lemma *approx-SComplex-mult-cancel*:

$\llbracket a \in SComplex; a \neq 0; a*w \approx a*z \rrbracket \implies w \approx z$

by (metis approx-SComplex-mult-cancel-zero approx-minus-iff right-diff-distrib)

lemma *approx-SComplex-mult-cancel-iff1* [simp]:

$\llbracket a \in SComplex; a \neq 0 \rrbracket \implies (a*w \approx a*z) = (w \approx z)$

by (metis HFinite-star-of SComplex-iff approx-SComplex-mult-cancel approx-mult2)

lemma *approx-hcmod-approx-zero*: $(x \approx y) = (hcmod (y - x) \approx 0)$

by (simp add: Infinitesimal-hcmod-iff approx-def hnrm-minus-commute)

lemma *approx-approx-zero-iff*: $(x \approx 0) = (hcmod x \approx 0)$

by (simp add: approx-hcmod-approx-zero)

lemma *approx-minus-zero-cancel-iff* [simp]: $(-x \approx 0) = (x \approx 0)$

by (simp add: approx-def)

lemma *Infinitesimal-hcmod-add-diff*:

$u \approx 0 \implies hcmod(x + u) - hcmod x \in Infinitesimal$

by (metis add.commute add.left-neutral approx-add-right-iff approx-def approx-hnorm)

lemma *approx-hcmod-add-hcmod*: $u \approx 0 \implies hcmod(x + u) \approx hcmod x$

using Infinitesimal-hcmod-add-diff approx-def by blast

15.5 Zero is the Only Infinitesimal Complex Number

lemma *Infinitesimal-less-SComplex*:

$\llbracket x \in SComplex; y \in Infinitesimal; 0 < hcmod x \rrbracket \implies hcmod y < hcmod x$

by (auto intro: Infinitesimal-less-SReal SReal-hcmod-SComplex simp add: Infinitesimal-hcmod-iff)

lemma *SComplex-Int-Infinitesimal-zero*: $SComplex\ Int\ Infinitesimal = \{0\}$
by (auto simp add: Standard-def Infinitesimal-hcmod-iff)

lemma *SComplex-Infinitesimal-zero*:
 $\llbracket x \in SComplex; x \in Infinitesimal \rrbracket \implies x = 0$
using *SComplex-iff* **by** auto

lemma *SComplex-HFinite-diff-Infinitesimal*:
 $\llbracket x \in SComplex; x \neq 0 \rrbracket \implies x \in HFinite - Infinitesimal$
using *SComplex-iff* **by** auto

lemma *numeral-not-Infinitesimal* [simp]:
 $numeral\ w \neq (0::hcomplex) \implies (numeral\ w::hcomplex) \notin Infinitesimal$
by (fast dest: Standard-numeral [THEN *SComplex-Infinitesimal-zero*])

lemma *approx-SComplex-not-zero*:
 $\llbracket y \in SComplex; x \approx y; y \neq 0 \rrbracket \implies x \neq 0$
by (auto dest: *SComplex-Infinitesimal-zero* approx-sym [THEN mem-infmal-iff [THEN iffD2]])

lemma *SComplex-approx-iff*:
 $\llbracket x \in SComplex; y \in SComplex \rrbracket \implies (x \approx y) = (x = y)$
by (auto simp add: Standard-def)

lemma *approx-unique-complex*:
 $\llbracket r \in SComplex; s \in SComplex; r \approx x; s \approx x \rrbracket \implies r = s$
by (blast intro: *SComplex-approx-iff* [THEN iffD1] approx-trans2)

15.6 Properties of *hRe*, *hIm* and *HComplex*

lemma *abs-hRe-le-hcmod*: $\bigwedge x. |hRe\ x| \leq hcmod\ x$
by transfer (rule abs-Re-le-cmod)

lemma *abs-hIm-le-hcmod*: $\bigwedge x. |hIm\ x| \leq hcmod\ x$
by transfer (rule abs-Im-le-cmod)

lemma *Infinitesimal-hRe*: $x \in Infinitesimal \implies hRe\ x \in Infinitesimal$
using *Infinitesimal-hcmod-iff* *abs-hRe-le-hcmod* *hrabs-le-Infinitesimal* **by** blast

lemma *Infinitesimal-hIm*: $x \in Infinitesimal \implies hIm\ x \in Infinitesimal$
using *Infinitesimal-hcmod-iff* *abs-hIm-le-hcmod* *hrabs-le-Infinitesimal* **by** blast

lemma *Infinitesimal-HComplex*:
assumes $x: x \in Infinitesimal$ **and** $y: y \in Infinitesimal$
shows $HComplex\ x\ y \in Infinitesimal$
proof –

have $hmod (HComplex\ 0\ y) \in Infinitesimal$
by (*simp add: hmod-i y*)
moreover have $hmod (hcomplex-of-hypreal\ x) \in Infinitesimal$
using *Infinitesimal-hmod-iff Infinitesimal-of-hypreal-iff x* **by** *blast*
ultimately have $hmod (HComplex\ x\ y) \in Infinitesimal$
by (*metis Infinitesimal-add Infinitesimal-hmod-iff add.right-neutral hcomplex-of-hypreal-add-HComplex*)
then show *?thesis*
by (*simp add: Infinitesimal-hnorm-iff*)
qed

lemma *hcomplex-Infinitesimal-iff*:
 $(x \in Infinitesimal) \longleftrightarrow (hRe\ x \in Infinitesimal \wedge hIm\ x \in Infinitesimal)$
using *Infinitesimal-HComplex Infinitesimal-hIm Infinitesimal-hRe* **by** *fastforce*

lemma *hRe-diff [simp]*: $\bigwedge x\ y. hRe\ (x - y) = hRe\ x - hRe\ y$
by *transfer simp*

lemma *hIm-diff [simp]*: $\bigwedge x\ y. hIm\ (x - y) = hIm\ x - hIm\ y$
by *transfer simp*

lemma *approx-hRe*: $x \approx y \implies hRe\ x \approx hRe\ y$
unfolding *approx-def* **by** (*drule Infinitesimal-hRe*) *simp*

lemma *approx-hIm*: $x \approx y \implies hIm\ x \approx hIm\ y$
unfolding *approx-def* **by** (*drule Infinitesimal-hIm*) *simp*

lemma *approx-HComplex*:
 $\llbracket a \approx b; c \approx d \rrbracket \implies HComplex\ a\ c \approx HComplex\ b\ d$
unfolding *approx-def* **by** (*simp add: Infinitesimal-HComplex*)

lemma *hcomplex-approx-iff*:
 $(x \approx y) = (hRe\ x \approx hRe\ y \wedge hIm\ x \approx hIm\ y)$
unfolding *approx-def* **by** (*simp add: hcomplex-Infinitesimal-iff*)

lemma *HFinite-hRe*: $x \in HFinite \implies hRe\ x \in HFinite$
using *HFinite-bounded-hmod abs-ge-zero abs-hRe-le-hmod* **by** *blast*

lemma *HFinite-hIm*: $x \in HFinite \implies hIm\ x \in HFinite$
using *HFinite-bounded-hmod abs-ge-zero abs-hIm-le-hmod* **by** *blast*

lemma *HFinite-HComplex*:
assumes $x \in HFinite\ y \in HFinite$
shows $HComplex\ x\ y \in HFinite$

proof –
have $HComplex\ x\ 0 \in HFinite\ HComplex\ 0\ y \in HFinite$
using *HFinite-hmod-iff assms hmod-i* **by** *fastforce+*
then have $HComplex\ x\ 0 + HComplex\ 0\ y \in HFinite$
using *HFinite-add* **by** *blast*

then show *?thesis*
by *simp*
qed

lemma *hcomplex-HFinite-iff*:
 $(x \in HFinite) = (hRe\ x \in HFinite \wedge hIm\ x \in HFinite)$
using *HFinite-HComplex HFinite-hIm HFinite-hRe* **by** *fastforce*

lemma *hcomplex-HInfinite-iff*:
 $(x \in HInfinite) = (hRe\ x \in HInfinite \vee hIm\ x \in HInfinite)$
by (*simp add: HInfinite-HFinite-iff hcomplex-HFinite-iff*)

lemma *hcomplex-of-hypreal-approx-iff* [*simp*]:
 $(hcomplex\ of\ hypreal\ x \approx hcomplex\ of\ hypreal\ z) = (x \approx z)$
by (*simp add: hcomplex-approx-iff*)

lemma *stc-part-Ex*:
assumes $x \in HFinite$
shows $\exists t \in SComplex. x \approx t$
proof –
let $?t = HComplex\ (st\ (hRe\ x))\ (st\ (hIm\ x))$
have $?t \in SComplex$
using *HFinite-hIm HFinite-hRe Reals-eq-Standard assms st-SReal* **by** *auto*
moreover have $x \approx ?t$
by (*simp add: HFinite-hIm HFinite-hRe assms hcomplex-approx-iff st-HFinite st-eq-approx*)
ultimately show *?thesis ..*
qed

lemma *stc-part-Ex1*: $x \in HFinite \implies \exists! t. t \in SComplex \wedge x \approx t$
using *approx-sym approx-unique-complex stc-part-Ex* **by** *blast*

15.7 Theorems About Monads

lemma *monad-zero-hcmod-iff*: $(x \in monad\ 0) = (hcmod\ x \in monad\ 0)$
by (*simp add: Infinitesimal-monad-zero-iff [symmetric] Infinitesimal-hcmod-iff*)

15.8 Theorems About Standard Part

lemma *stc-approx-self*: $x \in HFinite \implies stc\ x \approx x$
unfolding *stc-def*
by (*metis (no-types, lifting) approx-reorient someI-ex stc-part-Ex1*)

lemma *stc-SComplex*: $x \in HFinite \implies stc\ x \in SComplex$
unfolding *stc-def*
by (*metis (no-types, lifting) SComplex-iff approx-sym someI-ex stc-part-Ex*)

lemma *stc-HFinite*: $x \in HFinite \implies stc\ x \in HFinite$
by (*erule stc-SComplex [THEN Standard-subset-HFinite [THEN subsetD]]*)

lemma *stc-unique*: $\llbracket y \in SComplex; y \approx x \rrbracket \implies stc\ x = y$
by (*metis* *SComplex-approx-iff* *SComplex-iff* *approx-monad-iff* *approx-star-of-HFinite* *stc-SComplex* *stc-approx-self*)

lemma *stc-SComplex-eq* [*simp*]: $x \in SComplex \implies stc\ x = x$
by (*simp* *add*: *stc-unique*)

lemma *stc-eq-approx*:
 $\llbracket x \in HFinite; y \in HFinite; stc\ x = stc\ y \rrbracket \implies x \approx y$
by (*auto* *dest*!: *stc-approx-self* *elim*!: *approx-trans3*)

lemma *approx-stc-eq*:
 $\llbracket x \in HFinite; y \in HFinite; x \approx y \rrbracket \implies stc\ x = stc\ y$
by (*metis* *approx-sym* *approx-trans3* *stc-part-Ex1* *stc-unique*)

lemma *stc-eq-approx-iff*:
 $\llbracket x \in HFinite; y \in HFinite \rrbracket \implies (x \approx y) = (stc\ x = stc\ y)$
by (*blast* *intro*: *approx-stc-eq* *stc-eq-approx*)

lemma *stc-Infinitesimal-add-SComplex*:
 $\llbracket x \in SComplex; e \in Infinitesimal \rrbracket \implies stc(x + e) = x$
using *Infinitesimal-add-approx-self* *stc-unique* **by** *blast*

lemma *stc-Infinitesimal-add-SComplex2*:
 $\llbracket x \in SComplex; e \in Infinitesimal \rrbracket \implies stc(e + x) = x$
using *Infinitesimal-add-approx-self2* *stc-unique* **by** *blast*

lemma *HFinite-stc-Infinitesimal-add*:
 $x \in HFinite \implies \exists e \in Infinitesimal. x = stc(x) + e$
by (*blast* *dest*!: *stc-approx-self* [*THEN* *approx-sym*] *bex-Infinitesimal-iff2* [*THEN* *iffD2*])

lemma *stc-add*:
 $\llbracket x \in HFinite; y \in HFinite \rrbracket \implies stc\ (x + y) = stc(x) + stc(y)$
by (*simp* *add*: *stc-unique* *stc-SComplex* *stc-approx-self* *approx-add*)

lemma *stc-zero*: $stc\ 0 = 0$
by *simp*

lemma *stc-one*: $stc\ 1 = 1$
by *simp*

lemma *stc-minus*: $y \in HFinite \implies stc(-y) = -stc(y)$
by (*simp* *add*: *stc-unique* *stc-SComplex* *stc-approx-self* *approx-minus*)

lemma *stc-diff*:
 $\llbracket x \in HFinite; y \in HFinite \rrbracket \implies stc\ (x - y) = stc(x) - stc(y)$
by (*simp* *add*: *stc-unique* *stc-SComplex* *stc-approx-self* *approx-diff*)

lemma *stc-mult*:

$\llbracket x \in \text{HFinite}; y \in \text{HFinite} \rrbracket$
 $\implies \text{stc } (x * y) = \text{stc}(x) * \text{stc}(y)$
by (*simp add: stc-unique stc-SComplex stc-approx-self approx-mult-HFinite*)

lemma *stc-Infinitesimal*: $x \in \text{Infinitesimal} \implies \text{stc } x = 0$

by (*simp add: stc-unique mem-infmal-iff*)

lemma *stc-not-Infinitesimal*: $\text{stc}(x) \neq 0 \implies x \notin \text{Infinitesimal}$

by (*fast intro: stc-Infinitesimal*)

lemma *stc-inverse*:

$\llbracket x \in \text{HFinite}; \text{stc } x \neq 0 \rrbracket \implies \text{stc}(\text{inverse } x) = \text{inverse } (\text{stc } x)$
by (*simp add: stc-unique stc-SComplex stc-approx-self approx-inverse stc-not-Infinitesimal*)

lemma *stc-divide* [*simp*]:

$\llbracket x \in \text{HFinite}; y \in \text{HFinite}; \text{stc } y \neq 0 \rrbracket$
 $\implies \text{stc}(x/y) = (\text{stc } x) / (\text{stc } y)$
by (*simp add: divide-inverse stc-mult stc-not-Infinitesimal HFinite-inverse stc-inverse*)

lemma *stc-idempotent* [*simp*]: $x \in \text{HFinite} \implies \text{stc}(\text{stc}(x)) = \text{stc}(x)$

by (*blast intro: stc-HFinite stc-approx-self approx-stc-eq*)

lemma *HFinite-HFinite-hcomplex-of-hypreal*:

$z \in \text{HFinite} \implies \text{hcomplex-of-hypreal } z \in \text{HFinite}$
by (*simp add: hcomplex-HFinite-iff*)

lemma *SComplex-SReal-hcomplex-of-hypreal*:

$x \in \mathbb{R} \implies \text{hcomplex-of-hypreal } x \in \text{SComplex}$
by (*simp add: Reals-eq-Standard*)

lemma *stc-hcomplex-of-hypreal*:

$z \in \text{HFinite} \implies \text{stc}(\text{hcomplex-of-hypreal } z) = \text{hcomplex-of-hypreal } (\text{st } z)$
by (*simp add: SComplex-SReal-hcomplex-of-hypreal st-SReal st-approx-self stc-unique*)

lemma *hmod-stc-eq*:

assumes $x \in \text{HFinite}$
shows $\text{hmod}(\text{stc } x) = \text{st}(\text{hmod } x)$
by (*metis SReal-hmod-SComplex approx-HFinite approx-hnorm assms st-unique stc-SComplex-eq stc-eq-approx-iff stc-part-Ex*)

lemma *Infinitesimal-hcnj-iff* [*simp*]:

$(\text{hcnj } z \in \text{Infinitesimal}) \longleftrightarrow (z \in \text{Infinitesimal})$
by (*simp add: Infinitesimal-hcmmod-iff*)

end

16 Star-transforms in NSA, Extending Sets of Complex Numbers and Complex Functions

theory *CStar*
imports *NSCA*
begin

16.1 Properties of the *-Transform Applied to Sets of Reals

lemma *STARC-hcomplex-of-complex-Int*: $*s* X \cap SComplex = hcomplex-of-complex \text{ ‘ } X$
by (*auto simp: Standard-def*)

lemma *lemma-not-hcomplexA*: $x \notin hcomplex-of-complex \text{ ‘ } A \implies \forall y \in A. x \neq hcomplex-of-complex y$
by *auto*

16.2 Theorems about Nonstandard Extensions of Functions

lemma *starfunC-hcpow*: $\bigwedge Z. (*f* (\lambda z. z \hat{^} n)) Z = Z \text{ pow hypnat-of-nat } n$
by *transfer (rule refl)*

lemma *starfunCR-cmod*: $*f* cmod = hcmod$
by *transfer (rule refl)*

16.3 Internal Functions - Some Redundancy With *f* Now

lemma *starfun-Re*: $(*f* (\lambda x. Re (f x))) = (\lambda x. hRe ((*f* f) x))$
by *transfer (rule refl)*

lemma *starfun-Im*: $(*f* (\lambda x. Im (f x))) = (\lambda x. hIm ((*f* f) x))$
by *transfer (rule refl)*

lemma *starfunC-eq-Re-Im-iff*:
 $(*f* f) x = z \iff (*f* (\lambda x. Re (f x))) x = hRe z \wedge (*f* (\lambda x. Im (f x))) x = hIm z$
by (*simp add: hcomplex-hRe-hIm-cancel-iff starfun-Re starfun-Im*)

lemma *starfunC-approx-Re-Im-iff*:
 $(*f* f) x \approx z \iff (*f* (\lambda x. Re (f x))) x \approx hRe z \wedge (*f* (\lambda x. Im (f x))) x \approx hIm z$
by (*simp add: hcomplex-approx-iff starfun-Re starfun-Im*)

end

17 Limits, Continuity and Differentiation for Complex Functions

theory *CLim*


```

imports CStar
begin

```

```

declare epsilon-not-zero [simp]

```

```

lemma lemma-complex-mult-inverse-squared [simp]:  $x \neq 0 \implies x * (\text{inverse } x)^2 =$ 
   $\text{inverse } x$ 
for  $x :: \text{complex}$ 
by (simp add: numeral-2-eq-2)

```

Changing the quantified variable. Install earlier?

```

lemma all-shift:  $(\forall x::'a::\text{comm-ring-1}. P\ x) \longleftrightarrow (\forall x. P\ (x - a))$ 
by (metis add-diff-cancel)

```

17.1 Limit of Complex to Complex Function

```

lemma NSLIM-Re:  $f -a \rightarrow_{NS} L \implies (\lambda x. \text{Re } (f\ x)) -a \rightarrow_{NS} \text{Re } L$ 
by (simp add: NSLIM-def starfunC-approx-Re-Im-iff hRe-hcomplex-of-complex)

```

```

lemma NSLIM-Im:  $f -a \rightarrow_{NS} L \implies (\lambda x. \text{Im } (f\ x)) -a \rightarrow_{NS} \text{Im } L$ 
by (simp add: NSLIM-def starfunC-approx-Re-Im-iff hIm-hcomplex-of-complex)

```

```

lemma LIM-Re:  $f -a \rightarrow L \implies (\lambda x. \text{Re } (f\ x)) -a \rightarrow \text{Re } L$ 
for  $f :: 'a::\text{real-normed-vector} \Rightarrow \text{complex}$ 
by (simp add: LIM-NSLIM-iff NSLIM-Re)

```

```

lemma LIM-Im:  $f -a \rightarrow L \implies (\lambda x. \text{Im } (f\ x)) -a \rightarrow \text{Im } L$ 
for  $f :: 'a::\text{real-normed-vector} \Rightarrow \text{complex}$ 
by (simp add: LIM-NSLIM-iff NSLIM-Im)

```

```

lemma LIM-cn timer:  $f -a \rightarrow L \implies (\lambda x. \text{cnj } (f\ x)) -a \rightarrow \text{cnj } L$ 
for  $f :: 'a::\text{real-normed-vector} \Rightarrow \text{complex}$ 
by (simp add: LIM-eq complex-cn timer-diff [symmetric] del: complex-cn timer-diff)

```

```

lemma LIM-cn timer-iff:  $((\lambda x. \text{cnj } (f\ x)) -a \rightarrow \text{cnj } L) \longleftrightarrow f -a \rightarrow L$ 
for  $f :: 'a::\text{real-normed-vector} \Rightarrow \text{complex}$ 
by (simp add: LIM-eq complex-cn timer-diff [symmetric] del: complex-cn timer-diff)

```

```

lemma starfun-norm:  $( *f* (\lambda x. \text{norm } (f\ x))) = (\lambda x. \text{hnorm } (( *f* f)\ x))$ 
by transfer (rule refl)

```

```

lemma star-of-Re [simp]:  $\text{star-of } (\text{Re } x) = \text{hRe } (\text{star-of } x)$ 
by transfer (rule refl)

```

```

lemma star-of-Im [simp]:  $\text{star-of } (\text{Im } x) = \text{hIm } (\text{star-of } x)$ 
by transfer (rule refl)

```

Another equivalence result.

lemma *NSCLIM-NSCRLIM-iff*: $f - x \rightarrow_{NS} L \longleftrightarrow (\lambda y. \text{cmod } (f y - L)) - x \rightarrow_{NS} 0$
by (*simp add: NSLIM-def starfun-norm approx-approx-zero-iff [symmetric] approx-minus-iff [symmetric]*)

Much, much easier standard proof.

lemma *CLIM-CRLIM-iff*: $f - x \rightarrow L \longleftrightarrow (\lambda y. \text{cmod } (f y - L)) - x \rightarrow 0$
for $f :: 'a::\text{real-normed-vector} \Rightarrow \text{complex}$
by (*simp add: LIM-eq*)

So this is nicer nonstandard proof.

lemma *NSCLIM-NSCRLIM-iff2*: $f - x \rightarrow_{NS} L \longleftrightarrow (\lambda y. \text{cmod } (f y - L)) - x \rightarrow_{NS} 0$
by (*simp add: LIM-NSLIM-iff [symmetric] CLIM-CRLIM-iff*)

lemma *NSLIM-NSCRLIM-Re-Im-iff*:
 $f - a \rightarrow_{NS} L \longleftrightarrow (\lambda x. \text{Re } (f x)) - a \rightarrow_{NS} \text{Re } L \wedge (\lambda x. \text{Im } (f x)) - a \rightarrow_{NS} \text{Im } L$
apply (*auto intro: NSLIM-Re NSLIM-Im*)
apply (*auto simp add: NSLIM-def starfun-Re starfun-Im*)
apply (*auto dest!: spec*)
apply (*simp add: hcomplex-approx-iff*)
done

lemma *LIM-CRLIM-Re-Im-iff*: $f - a \rightarrow L \longleftrightarrow (\lambda x. \text{Re } (f x)) - a \rightarrow \text{Re } L \wedge (\lambda x. \text{Im } (f x)) - a \rightarrow \text{Im } L$
for $f :: 'a::\text{real-normed-vector} \Rightarrow \text{complex}$
by (*simp add: LIM-NSLIM-iff NSLIM-NSCRLIM-Re-Im-iff*)

17.2 Continuity

lemma *NSLIM-isContc-iff*: $f - a \rightarrow_{NS} f a \longleftrightarrow (\lambda h. f (a + h)) - 0 \rightarrow_{NS} f a$
by (*rule NSLIM-at0-iff*)

17.3 Functions from Complex to Reals

lemma *isNSContCR-cmod [simp]*: *isNSCont cmod a*
by (*auto intro: approx-hnorm simp: starfunCR-cmod hmod-hcomplex-of-complex [symmetric] isNSCont-def*)

lemma *isContCR-cmod [simp]*: *isCont cmod a*
by (*simp add: isNSCont-isCont-iff [symmetric]*)

lemma *isCont-Re*: *isCont f a \implies isCont $(\lambda x. \text{Re } (f x)) a$*
for $f :: 'a::\text{real-normed-vector} \Rightarrow \text{complex}$
by (*simp add: isCont-def LIM-Re*)

lemma *isCont-Im*: *isCont f a \implies isCont $(\lambda x. \text{Im } (f x)) a$*
for $f :: 'a::\text{real-normed-vector} \Rightarrow \text{complex}$
by (*simp add: isCont-def LIM-Im*)

17.4 Differentiation of Natural Number Powers

lemma *CDERIV-pow* [simp]: $DERIV (\lambda x. x \wedge n) x :> \text{complex-of-real } (\text{real } n) * (x \wedge (n - \text{Suc } 0))$
apply (induct n)
apply (drule-tac [2] *DERIV-ident* [THEN *DERIV-mult*])
apply (auto simp add: distrib-right of-nat-Suc)
apply (case-tac n)
apply (auto simp add: ac-simps)
done

Nonstandard version.

lemma *NSCDERIV-pow*: $NSDERIV (\lambda x. x \wedge n) x :> \text{complex-of-real } (\text{real } n) * (x \wedge (n - 1))$
by (metis *CDERIV-pow NSDERIV-DERIV-iff One-nat-def*)

Can't relax the premise $x \neq 0$: it isn't continuous at zero.

lemma *NSCDERIV-inverse*: $x \neq 0 \implies NSDERIV (\lambda x. \text{inverse } x) x :> -(\text{inverse } x)^2$
for $x :: \text{complex}$
unfolding numeral-2-eq-2 **by** (rule *NSDERIV-inverse*)

lemma *CDERIV-inverse*: $x \neq 0 \implies DERIV (\lambda x. \text{inverse } x) x :> -(\text{inverse } x)^2$
for $x :: \text{complex}$
unfolding numeral-2-eq-2 **by** (rule *DERIV-inverse*)

17.5 Derivative of Reciprocals (Function *inverse*)

lemma *CDERIV-inverse-fun*:
 $DERIV f x :> d \implies f x \neq 0 \implies DERIV (\lambda x. \text{inverse } (f x)) x :> -(d * \text{inverse } ((f x)^2))$
for $x :: \text{complex}$
unfolding numeral-2-eq-2 **by** (rule *DERIV-inverse-fun*)

lemma *NSCDERIV-inverse-fun*:
 $NSDERIV f x :> d \implies f x \neq 0 \implies NSDERIV (\lambda x. \text{inverse } (f x)) x :> -(d * \text{inverse } ((f x)^2))$
for $x :: \text{complex}$
unfolding numeral-2-eq-2 **by** (rule *NSDERIV-inverse-fun*)

17.6 Derivative of Quotient

lemma *CDERIV-quotient*:
 $DERIV f x :> d \implies DERIV g x :> e \implies g(x) \neq 0 \implies$
 $DERIV (\lambda y. f y / g y) x :> (d * g x - (e * f x)) / (g x)^2$
for $x :: \text{complex}$
unfolding numeral-2-eq-2 **by** (rule *DERIV-quotient*)

lemma *NSCDERIV-quotient*:
 $NSDERIV f x :> d \implies NSDERIV g x :> e \implies g x \neq (0 :: \text{complex}) \implies$

NSDERIV ($\lambda y. f y / g y$) $x :> (d * g x - (e * f x)) / (g x)^2$
unfolding *numeral-2-eq-2* **by** (*rule NSDERIV-quotient*)

17.7 Caratheodory Formulation of Derivative at a Point: Standard Proof

lemma *CARAT-CDERIVD*:

($\forall z. f z - f x = g z * (z - x)$) \wedge *isNSCont* $g x \wedge g x = l \implies$ *NSDERIV* $f x :> l$
by *clarify* (*rule CARAT-DERIVD*)

end

18 Logarithms: Non-Standard Version

theory *HLog*

imports *HTranscendental*

begin

definition *powhr* :: *hypreal* \Rightarrow *hypreal* \Rightarrow *hypreal* (**infixr** \langle *powhr* \rangle 80)
where [*transfer-unfold*]: $x \text{ powhr } a = \text{starfun2 } (\text{powr}) x a$

definition *hlog* :: *hypreal* \Rightarrow *hypreal* \Rightarrow *hypreal*
where [*transfer-unfold*]: $\text{hlog } a x = \text{starfun2 } \text{log } a x$

lemma *powhr*: ($\text{star-n } X$) *powhr* ($\text{star-n } Y$) = $\text{star-n } (\lambda n. (X n) \text{ powr } (Y n))$
by (*simp add: powhr-def starfun2-star-n*)

lemma *powhr-one-eq-one* [*simp*]: $\bigwedge a. 1 \text{ powhr } a = 1$
by *transfer simp*

lemma *powhr-mult*: $\bigwedge a x y. 0 < x \implies 0 < y \implies (x * y) \text{ powhr } a = (x \text{ powhr } a) * (y \text{ powhr } a)$
by *transfer (simp add: powr-mult)*

lemma *powhr-gt-zero* [*simp*]: $\bigwedge a x. 0 < x \text{ powhr } a \longleftrightarrow x \neq 0$
by *transfer simp*

lemma *powhr-not-zero* [*simp*]: $\bigwedge a x. x \text{ powhr } a \neq 0 \longleftrightarrow x \neq 0$
by *transfer simp*

lemma *powhr-divide*: $\bigwedge a x y. 0 \leq x \implies 0 \leq y \implies (x / y) \text{ powhr } a = (x \text{ powhr } a) / (y \text{ powhr } a)$
by *transfer (rule powr-divide)*

lemma *powhr-add*: $\bigwedge a b x. x \text{ powhr } (a + b) = (x \text{ powhr } a) * (x \text{ powhr } b)$
by *transfer (rule powr-add)*

lemma *powhr-powhr*: $\bigwedge a b x. (x \text{ powhr } a) \text{ powhr } b = x \text{ powhr } (a * b)$
by *transfer (rule powr-powr)*

lemma *powhr-powhr-swap*: $\bigwedge a \ b \ x. (x \text{ powhr } a) \text{ powhr } b = (x \text{ powhr } b) \text{ powhr } a$
by *transfer (rule powr-powr-swap)*

lemma *powhr-minus*: $\bigwedge a \ x. x \text{ powhr } (- a) = \text{inverse } (x \text{ powhr } a)$
by *transfer (rule powr-minus)*

lemma *powhr-minus-divide*: $x \text{ powhr } (- a) = 1 / (x \text{ powhr } a)$
by *(simp add: divide-inverse powhr-minus)*

lemma *powhr-less-mono*: $\bigwedge a \ b \ x. a < b \implies 1 < x \implies x \text{ powhr } a < x \text{ powhr } b$
by *transfer simp*

lemma *powhr-less-cancel*: $\bigwedge a \ b \ x. x \text{ powhr } a < x \text{ powhr } b \implies 1 < x \implies a < b$
by *transfer simp*

lemma *powhr-less-cancel-iff* [simp]: $1 < x \implies x \text{ powhr } a < x \text{ powhr } b \longleftrightarrow a < b$
by *(blast intro: powhr-less-cancel powhr-less-mono)*

lemma *powhr-le-cancel-iff* [simp]: $1 < x \implies x \text{ powhr } a \leq x \text{ powhr } b \longleftrightarrow a \leq b$
by *(simp add: linorder-not-less [symmetric])*

lemma *hlog*: $\text{hlog } (\text{star-}n \ X) (\text{star-}n \ Y) = \text{star-}n \ (\lambda n. \log (X \ n) (Y \ n))$
by *(simp add: hlog-def starfun2-star-n)*

lemma *hlog-starfun-ln*: $\bigwedge x. (*f* \ \ln) \ x = \text{hlog } ((*f* \ \exp) \ 1) \ x$
by *transfer (rule log-ln)*

lemma *powhr-hlog-cancel* [simp]: $\bigwedge a \ x. 0 < a \implies a \neq 1 \implies 0 < x \implies a \text{ powhr } (\text{hlog } a \ x) = x$
by *transfer simp*

lemma *hlog-powhr-cancel* [simp]: $\bigwedge a \ y. 0 < a \implies a \neq 1 \implies \text{hlog } a \ (a \text{ powhr } y) = y$
by *transfer simp*

lemma *hlog-mult*:
 $\bigwedge a \ x \ y. \text{hlog } a \ (x * y) = (\text{if } x \neq 0 \ \wedge \ y \neq 0 \text{ then } \text{hlog } a \ x + \text{hlog } a \ y \text{ else } 0)$
by *transfer (rule log-mult)*

lemma *hlog-as-starfun*: $\bigwedge a \ x. 0 < a \implies a \neq 1 \implies \text{hlog } a \ x = (*f* \ \ln) \ x / (*f* \ \ln) \ a$
by *transfer (simp add: log-def)*

lemma *hlog-eq-div-starfun-ln-mult-hlog*:
 $\bigwedge a \ b \ x. 0 < a \implies a \neq 1 \implies 0 < b \implies b \neq 1 \implies 0 < x \implies$
 $\text{hlog } a \ x = ((*f* \ \ln) \ b / (*f* \ \ln) \ a) * \text{hlog } b \ x$
by *transfer (rule log-eq-div-ln-mult-log)*

lemma *powhr-as-starfun*: $\bigwedge a x. x \text{ powhr } a = (\text{if } x = 0 \text{ then } 0 \text{ else } (*f* \text{ exp}) (a * (*f* \text{ real-ln}) x))$

by *transfer (simp add: powr-def)*

lemma *HInfinite-powhr*:

$x \in HInfinite \implies 0 < x \implies a \in HFinite - Infinitesimal \implies 0 < a \implies x \text{ powhr } a \in HInfinite$

by (*auto intro!*: *starfun-ln-ge-zero starfun-ln-HInfinite*

HInfinite-HFinite-not-Infinitesimal-mult2 starfun-exp-HInfinite

simp add: order-less-imp-le HInfinite-gt-zero-gt-one powhr-as-starfun zero-le-mult-iff)

lemma *hlog-hrabs-HInfinite-Infinitesimal*:

$x \in HFinite - Infinitesimal \implies a \in HInfinite \implies 0 < a \implies hlog a |x| \in Infinitesimal$

apply (*frule HInfinite-gt-zero-gt-one*)

apply (*auto intro!*: *starfun-ln-HFinite-not-Infinitesimal*

HInfinite-inverse-Infinitesimal Infinitesimal-HFinite-mult2

simp add: starfun-ln-HInfinite not-Infinitesimal-not-zero

hlog-as-starfun divide-inverse)

done

lemma *hlog-HInfinite-as-starfun*: $a \in HInfinite \implies 0 < a \implies hlog a x = (*f* \text{ ln}) x / (*f* \text{ ln}) a$

by (*rule hlog-as-starfun auto*)

lemma *hlog-one [simp]*: $\bigwedge a. hlog a 1 = 0$

by *transfer simp*

lemma *hlog-eq-one [simp]*: $\bigwedge a. 0 < a \implies a \neq 1 \implies hlog a a = 1$

by *transfer (rule log-eq-one)*

lemma *hlog-inverse*: $\bigwedge a x. hlog a (\text{inverse } x) = - hlog a x$

by *transfer (simp add: log-inverse)*

lemma *hlog-divide*: $hlog a (x / y) = (\text{if } x \neq 0 \wedge y \neq 0 \text{ then } hlog a x - hlog a y \text{ else } 0)$

by (*simp add: hlog-mult hlog-inverse divide-inverse*)

lemma *hlog-less-cancel-iff [simp]*:

$\bigwedge a x y. 1 < a \implies 0 < x \implies 0 < y \implies hlog a x < hlog a y \longleftrightarrow x < y$

by *transfer simp*

lemma *hlog-le-cancel-iff [simp]*: $1 < a \implies 0 < x \implies 0 < y \implies hlog a x \leq hlog a y \longleftrightarrow x \leq y$

by (*simp add: linorder-not-less [symmetric]*)

end

```
theory Hyperreal  
imports HLog  
begin
```

```
end  
theory Hypercomplex  
imports CLim Hyperreal  
begin
```

```
end
```

```
theory Nonstandard-Analysis  
imports Hypercomplex  
begin
```

```
end
```