

# Euler’s Exponential Series as an Infinite Polynomial

Jacques D. Fleuriot  
University of Edinburgh

June 10, 2026

## Abstract

In this formalisation, we reconstruct Euler’s derivation of the power series for the exponential function as expounded in his famous *Introductio in analysin infinitorum*, first published in 1748. Using nonstandard analysis, we mechanize his mixture of infinitesimal and infinite ‘algebraic’ reasoning in the proof assistant Isabelle. In so doing, we demonstrate that the gist of his arguments can be reconstructed formally, with Isabelle and nonstandard analysis shoring up crucial aspects of his reasoning that some historians have qualified as being “more a matter of faith than science”.

## Contents

<b>1</b>	<b>Generalized falling factorial</b>	<b>2</b>
1.1	Falling factorial expressed recursively . . . . .	2
1.2	Falling factorial and binomial coefficients . . . . .	3
<b>2</b>	<b>Various supporting lemmas</b>	<b>4</b>
<b>3</b>	<b>Hyperinteger numbers</b>	<b>14</b>
3.1	Properties of Hyperintegers . . . . .	15
3.2	Embedding of the hyperintegers into other types . . . . .	16
3.3	Embedding the naturals into the hyperintegers . . . . .	18
3.4	Magnitude of a hyperinteger . . . . .	19
3.5	The divides relation . . . . .	21
3.6	Division on <i>hypint</i> . . . . .	22
3.7	Properties of the set of embedded integers . . . . .	23
3.8	Infinite hyperintegers . . . . .	23
3.9	Embedding of hypertegers in hyperreals . . . . .	26
<b>4</b>	<b>Internal sets and functions</b>	<b>27</b>
4.1	Overspill theorem . . . . .	31
4.2	The Sequential Theorem . . . . .	32

<b>5</b>	<b>Hyperfinite sets</b>	<b>33</b>
5.1	Properties of hyperfinite sets . . . . .	33
5.2	Hyperfold function . . . . .	36
<b>6</b>	<b>Hyperfinite summation</b>	<b>36</b>
6.1	Hypersum properties . . . . .	37
6.2	The sumhr function as a hypersum. . . . .	40
6.3	Hyper-convergence . . . . .	40
<b>7</b>	<b>The floor and ceiling functions extended to hypernumbers</b>	<b>47</b>
7.1	The nonstandard floor function . . . . .	47
7.2	The nonstandard ceiling function . . . . .	50
7.3	The nonstandard fractional part . . . . .	53
7.4	Miscellaneous properties . . . . .	54
<b>8</b>	<b>The hyperfactorial and hyperbinomial coefficient functions</b>	<b>56</b>
8.1	The hyperbinomial coefficients . . . . .	56
8.2	The hyperfactorial function . . . . .	57
8.3	The hyperbinomial theorem . . . . .	58
8.4	Nonstandard falling factorial . . . . .	58
8.5	Nonstandard version of Pochhammer's symbol i.e. the rising factorial . . . . .	62
<b>9</b>	<b>Hyperreal powers</b>	<b>66</b>
9.1	Tranferred properties . . . . .	66
9.2	Relating pow and hpow . . . . .	70
9.3	Some nonstandardard theorems . . . . .	70
<b>10</b>	<b>The hyper logarithm</b>	<b>72</b>
<b>11</b>	<b>Deriving the exponential series Euler-style</b>	<b>75</b>
11.1	Euler's witness: Introductio, paragraph 114 . . . . .	75
11.2	Defining Euleur's finite witness . . . . .	76
11.3	Euler's Introductio, Paragraph 115: first expansion . . . . .	78
11.4	Euler's Introductio, Paragraph 116: exponential function. . .	85
11.5	Euler's Introductio, Paragraph 122: exponential series expan- sion for the base $\epsilon$ . . . . .	87
11.6	Hypernatural logarithm . . . . .	88
11.7	Linking the Euler and Isabelle library exponential functions. .	90
11.8	Derivative of Euler's exponential function. . . . .	91

# 1 Generalized falling factorial

```
theory FallFactorial
imports "HOL.Binomial" "HOL.Rat"
begin
```

## definition

```
"fallfactpow (a::'a::comm_ring_1) n = (if n = 0 then 1 else prod (λn.
a - of_nat n) {0 .. n - 1})"
```

```
lemma fallfactpow_0[simp]: "fallfactpow a 0 = 1"
  <proof>
```

```
lemma fallfactpow_1[simp]: "fallfactpow a 1 = a" <proof>
```

```
lemma fallfactpow_Suc0[simp]: "fallfactpow a (Suc 0) = a" <proof>
```

```
lemma fallfactpow_Suc_setprod: "fallfactpow a (Suc n) = prod (λn. a -
of_nat n) {0 .. n}"
  <proof>
```

## 1.1 Falling factorial expressed recursively

```
lemma fallfactpow_Suc: "fallfactpow a (Suc n) = fallfactpow a n * (a
- of_nat n)"
  <proof>
```

```
lemma fallfactpow_rec: "fallfactpow a (Suc n) = a * fallfactpow (a -
1) n"
  <proof>
```

```
lemma fallfactpow_base_zero[simp]: "fallfactpow 0 (Suc n) = 0"
  <proof>
```

```
lemma fallfactpow_fact: "of_nat (fact n) = fallfactpow (of_nat n) n"
  <proof>
```

```
lemma fallfactpow_altdef:
  "fallfactpow a k = (∏ i<k. a - of_nat i)"
  <proof>
```

The falling factorial and Pochhammer functions

```
lemma pochhammer_fallfactpow_eq:
  "pochhammer a n = fallfactpow (a + of_nat n - 1) n"
  <proof>
```

```
lemma fallfactpow_of_nat_eq_0_lemma [simp]:
  assumes kn: "k > n"
  shows "fallfactpow (of_nat n :: 'a:: idom) k = 0"
  <proof>
```

```

lemma setprod_zero_of_nat_diff_lemma:
  "a < k  $\implies$  ( $\prod_{i < k}$ . of_nat (a - i)) = (0 :: 'a :: comm_semiring_1)"
<proof>

lemma fallfactpow_of_nat_altdef:
  "fallfactpow (of_nat a :: 'a :: idom) k = (( $\prod_{i < k}$ . of_nat (a - i)) ::
'a :: idom)"
<proof>

lemma fallfactpow_ge_zero:
  "of_nat k < j  $\implies$  0  $\leq$  fallfactpow (j :: 'a :: {linordered_idom}) k"
<proof>

lemma fallfactpow_le_power_self [simp]:
  "fallfactpow (of_nat k :: 'a :: {linordered_idom}) k  $\leq$  of_nat k ^ k"
<proof>

lemma fallfactpow_le_power:
  assumes "of_nat k  $\leq$  j"
  shows "fallfactpow (j :: 'a :: {linordered_idom}) k  $\leq$  j ^ k"
<proof>

lemma fallfactpow_ge_zero' [simp]:
  "fallfactpow (of_nat n) k  $\geq$  (0 :: 'a :: linordered_idom)"
<proof>

```

## 1.2 Falling factorial and binomial coefficients

```

lemma binomial_fallfactpow_altdef_of_nat:
  "of_nat (n choose k) =
    (fallfactpow (of_nat n) k :: 'a :: {field_char_0}) / fact k"
<proof>

lemma binomial_fallfactpow_ring:
  "fallfactpow (a + b :: 'a :: {comm_ring_1}) n =
    ( $\sum_{k=0..n}$ . of_nat (n choose k) * fallfactpow a (n - k) * fallfactpow
b k)"
<proof>

lemma binomial_vandermonde:
  "(r + s) ^ k / fact k =
    ( $\sum_{i=0..k}$ . (r :: 'a :: field_char_0) ^ i / fact i * s ^ (k - i) /
fact (k - i))"
<proof>

lemma natsum_reverse_index:

```

```

fixes m::nat
shows " $(\bigwedge k. m \leq k \implies k \leq n \implies g\ k = f\ (m + n - k)) \implies (\sum_{k=m..n}. f\ k) = (\sum_{k=m..n}. g\ k)$ "
  <proof>

lemma binomial_fallfactpow_vandermonde:
  "fallfactpow (r + s) k / fact k =
    ( $\sum_{i=0..k}. fallfactpow\ (r::'a::field\_char\_0)\ i / fact\ i * fallfactpow\ s\ (k - i) / fact\ (k - i)$ )"
  <proof>

lemma gbinomial_fallfactpow: "a gchoose n = fallfactpow a n / fact n"
  <proof>

lemma binomial_fallfactpow_vandermonde_alt:
  "fallfactpow (r + s) k / fact k =
    ( $\sum_{i=0..k}. ((r::'a::field\_char\_0)\ gchoose\ i) * (s\ gchoose\ (k - i))$ )"
  <proof>

end

```

## 2 Various supporting lemmas

```

theory AuxiliaryNSA
imports "HOL-Nonstandard_Analysis.HSeries" "Real_Power.Log"
begin

```

These results can be moved to the indicated theories.

StarDef.thy lemmas

```

lemma Iset_star_of_empty [simp]: "Iset(star_of {}) = {}"
  <proof>

```

```

lemma Iset_star_n_empty [simp]: "Iset (star_n ( $\lambda n. \{ \}$ )) = {}"
  <proof>

```

```

lemma Iset_eq_cancel:
  "(Iset (star_n X) = Iset (star_n Y)) = (eventually ( $\lambda n. X\ n = Y\ n$ )  $\mathcal{U}$ )"
  <proof>

```

```

lemma Collect_starP_starset_eq: "{x. (*p* P) x} = *s* {x. P x}"
  <proof>

```

Hypernat.thy lemmas

```

lemma Nats_hypnat_of_nat_iff: "(n  $\in$  Nats) = ( $\exists m. n = hypnat\_of\_nat\ m$ )"
  <proof>

```

```

lemma HNatInfinite_add_one_cancel:

```

```

    "N + 1 ∈ HNatInfinite ⇒ N ∈ HNatInfinite"
  ⟨proof⟩

lemma of_hypnat_of_nat [simp]: "of_hypnat(hypnat_of_nat n) = of_nat n"
  ⟨proof⟩

lemma of_nat_hypreal_of_hypnat:
  "of_nat n = of_hypnat (hypnat_of_nat n)"
  ⟨proof⟩

lemma hSuc_eq_add_one:
  "∧n. hSuc n = n + 1"
  ⟨proof⟩

lemma HNatInfinite_hSuc_diff_one [simp]:
  "N ∈ HNatInfinite ⇒ hSuc (N - 1) = N"
  ⟨proof⟩

lemma HNatInfinite_atLeastAtMost_hSuc:
  "N ∈ HNatInfinite ⇒ {M..<N} = {M..<hSuc(N - 1)}"
  ⟨proof⟩

HyperDef.thy lemmas

lemma hypreal_of_nat_less_hypreal_of_hypnat_iff:
  "∧a b. (hypreal_of_nat a < of_hypnat b) = (hypnat_of_nat a < b)"
  ⟨proof⟩

lemma hypreal_of_nat_le_hypreal_of_hypnat_iff:
  "∧a b. (hypreal_of_nat a ≤ hypreal_of_hypnat b) = (hypnat_of_nat
a ≤ b)"
  ⟨proof⟩

lemma hyperpow_zero [simp]: "x pow 0 = 1"
  ⟨proof⟩

lemma hyperpow_of_nat: "∧x. x pow of_nat n = x ^ n"
  ⟨proof⟩

lemma hyperpow_hSuc_zero [simp]:
  "∧N. (0::'a::{power, semiring_0} star) pow hSuc N = 0"
  ⟨proof⟩

lemma starfun_power: "∧n. (*f* (^) x) n = star_of x pow n"
  ⟨proof⟩

lemma hyperpow_diff:
  "∧x n k. n ≥ k ⇒ x ≠ 0 ⇒ ((x::'a::{field star} pow n)/(x pow k)
= x pow (n - k))"
  ⟨proof⟩

```

```

lemma hyperpow_divide:
  " $\bigwedge x y n. ((x::'a::field\ star)/y)\ pow\ n = (x\ pow\ n)/(y\ pow\ n)$ "
  <proof>

lemma hyperpow_diff_cancel:
  " $k \leq n \implies x\ pow\ k * (x::'a::monoid\_mult\ star)\ pow\ (n - k) = x\ pow\ n$ "
  <proof>

lemma hyperpow_0_left:
  " $\bigwedge n. 0\ pow\ n = (if\ n = 0\ then\ 1\ else\ (0::'a::\{power,\ semiring\_1\}\ star))$ "
  <proof>

lemma hyperpow_eq_0_iff [simp]:
  " $\bigwedge x n. (x\ pow\ n = 0) \iff$ 
     $(x = (0::'a::\{mult\_zero,\ zero\_neq\_one,\ semiring\_1\_no\_zero\_divisors,\ power\}\ star) \wedge n \neq 0)$ "
  <proof>

lemma hyperpow_less_zero_eq:
  " $\bigwedge x n. (x\ pow\ n < (0::'a::\{linordered\_idom\}\ star)) = (\neg\ 2\ dvd\ n \wedge x < 0)$ "
  <proof>

lemma star_n_divide: "star_n X / star_n Y = star_n ( $\lambda n. X\ n / Y\ n$ )"
  <proof>

NSA.thy lemmas

lemma approx_zero_abs_zero_iff [iff]:
  " $(abs\ x \approx 0) = ((x::hypreal) \approx 0)$ "
  <proof>

lemma Infinitesimal_divide_HFinite:
  " $\llbracket (x::'a::real\_normed\_div\_algebra\ star) \in Infinitesimal; y \in HFinite - Infinitesimal \rrbracket$ 
     $\implies x/y \in Infinitesimal$ "
  <proof>

lemma approx_1_not_Infinitesimal:
  " $x \approx (1::'a::real\_normed\_div\_algebra\ star) \implies x \notin Infinitesimal$ "
  <proof>

lemma HFinite_add_imp_HFinite:
  " $\llbracket x \geq (0::hypreal); y \geq 0; x + y \in HFinite \rrbracket \implies y \in HFinite$ "
  <proof>

lemma Infinitesimal_hrealpow_cancel [simp]:
  assumes "n > 0"

```

shows "(x ^ n ∈ Infinitesimal) = ((x::'a::real\_normed\_div\_algebra star) ∈ Infinitesimal)"  
 ⟨proof⟩

lemma hrealpow\_approx:  
 "[(x::'a::{real\_normed\_algebra, one, monoid\_mult} star) ≈ b; b ∈ HFinite] ⇒ x ^ n ≈ b ^ n"  
 ⟨proof⟩

lemma hrealpow\_approx\_one:  
 "(x::'a::{real\_normed\_algebra, one, monoid\_mult} star) ≈ 1 ⇒ x ^ n ≈ 1"  
 ⟨proof⟩

lemma zero\_not\_HInfinite [simp]: "0 ∉ HInfinite"  
 ⟨proof⟩

lemma HInfinite\_hyperpow\_not\_zero:  
 "(x::'a::{real\_normed\_field} star) ∈ HInfinite ⇒ x pow n ≠ 0"  
 ⟨proof⟩

lemma HFinite\_divide:  
 "[(x::'a::real\_normed\_div\_algebra star) ∈ HFinite; y ∉ Infinitesimal] ⇒ x/y ∈ HFinite"  
 ⟨proof⟩

lemma HFinite\_hyperpow\_HFinite:  
 "[(x::'a::{monoid\_mult, real\_normed\_algebra} star) ∈ HFinite; n ∈ ℕ] ⇒ x pow n ∈ HFinite"  
 ⟨proof⟩

lemma hyperpow\_approx:  
 assumes HFinite\_a: "a ∈ HFinite"  
 and approx\_ab: "(a::hypreal) ≈ b"  
 shows "a pow of\_nat n ≈ b pow of\_nat n"  
 ⟨proof⟩

lemma Infinitesimal\_approx\_hyperpow:  
 assumes Infinitesimal\_a: "a ∈ Infinitesimal"  
 and approx\_ab: "(a::hypreal) ≈ b"  
 shows "a pow n ≈ b pow n"  
 ⟨proof⟩

lemma hnorm\_hypreal\_of\_hypnat [simp]:  
 "∧n. hnorm (of\_hypnat n::'a::real\_normed\_algebra\_1 star) = of\_hypnat n"  
 ⟨proof⟩

lemma HFinite\_of\_nat [simp]: "of\_nat n ∈ HFinite"

*<proof>*

**lemma** *HInfinite\_def2*: "HInfinite = {x.  $\forall n \in \text{Nats. hypreal\_of\_hypnat } n < \text{hnorm } x$ }"

*<proof>*

**lemma** *HFinite\_def2*: "HFinite = {x.  $\exists n \in \text{Nats. hnorm } x < \text{hypreal\_of\_hypnat } n$ }"

*<proof>*

**lemma** *HFinite\_less\_Nat\_Ex*:

"(x::hypreal)  $\in$  HFinite  $\implies \exists n \in \text{Nats. } x < n$ "

*<proof>*

**lemma** *HNatInfinite\_of\_hypnat\_HInfinite*:

"x  $\in$  HNatInfinite  $\implies$  (of\_hypnat x :: 'a::real\_normed\_algebra\_1 star)  $\in$  HInfinite"

*<proof>*

**lemma** *HInfinite\_ge\_HNatInfinite*:

" $\llbracket n \in \text{HNatInfinite; hypreal\_of\_hypnat } n \leq x \rrbracket \implies x \in \text{HInfinite}$ "

*<proof>*

**lemma** *HNatInfinite\_of\_hypnat\_HInfinite\_iff*:

"(of\_hypnat n :: 'a::real\_normed\_algebra\_1 star)  $\in$  HInfinite = (n  $\in$  HNatInfinite)"

*<proof>*

**lemma** *HNatInfinite\_inverse\_of\_hypnat\_Infinitesimal\_iff*:

"N  $\neq 0 \implies (1/\text{hypreal\_of\_hypnat } N \in \text{Infinitesimal}) = (N \in \text{HNatInfinite})"$

*<proof>*

**lemma** *Nats\_hypreal\_of\_hypnat\_HFinite*:

"n  $\in$  Nats  $\implies$  of\_hypnat n  $\in$  HFinite"

*<proof>*

**lemma** *approx\_divide\_HFinite\_diff\_Infinitesimal*:

" $\llbracket (x :: 'a::\{\text{real\_normed\_div\_algebra, field}\} \text{star}) \approx y;$   
z  $\in$  HFinite - Infinitesimal  
 $\rrbracket \implies x/z \approx y/z$ "

*<proof>*

**lemma** *HFinite\_diff*:

" $\llbracket x \in \text{HFinite; } y \in \text{HFinite} \rrbracket \implies x - y \in \text{HFinite}$ "

*<proof>*

**lemma** *Infinitesimal\_interval2a*:

" $\llbracket e \in \text{Infinitesimal; } e' \in \text{Infinitesimal};$   
hnorm e'  $\leq$  hnorm x ; hnorm x  $\leq$  hnorm e  $\rrbracket \implies x \in \text{Infinitesimal}$ "

*<proof>*

**lemma** *Infinitesimal\_HInfinite\_divide:*

"[[ (x :: 'a :: {real\_normed\_div\_algebra, field} star) ∈ HFinite; y ∈ HInfinite ]]  
⇒ x/y ∈ Infinitesimal"

*<proof>*

**lemma** *approx\_divide\_HInfinite:*

"[(x :: 'a :: {real\_normed\_div\_algebra, field} star) ≈ y; z ∈ HInfinite]  
⇒ x/z ≈ y/z"

*<proof>*

**lemma** *approx\_1\_mult\_HFinite\_HFinite:*

assumes *approx\_1*: "x ≈ (1 :: 'a :: real\_normed\_div\_algebra star)"

and *HFinite\_prod*: "x\*y ∈ HFinite"

shows "y ∈ HFinite"

*<proof>*

Star.thy lemmas

**instance** *star* :: (semiring\_modulo) semiring\_modulo

*<proof>*

**declare** *of\_bool\_def* [*transfer\_unfold*]

**instance** *star* :: (semiring\_parity) semiring\_parity

*<proof>*

**lemma** *starfun\_starfun\_n\_eq:*

"\*f\* f = \*fn\* (λn. f)"

*<proof>*

**lemma** *starfun\_n\_zero:*

"(\*fn\* (λm n :: 'b. 0 :: 'a :: zero)) = (λn. star\_of 0)"

*<proof>*

**lemma** *InternalFuns\_starfun\_n* [*simp*]: "\*fn\* f ∈ InternalFuns"

*<proof>*

**lemma** *InternalFuns\_starfun* [*simp*]: "\*f\* f ∈ InternalFuns"

*<proof>*

**lemma** *InternalFuns\_starfun2* [*simp*]: "(\*f2\* f) a ∈ InternalFuns"

*<proof>*

**lemma** *starfun\_n\_hyperpow*: "(λn. (\*f2\* (^)) ((\*fn\* f) n) ((\*fn\* g) n))

z = (\*fn\* (λ i x. (f i x) ^ (g i x))) z"

*<proof>*

**lemma** *InternalFuns\_hSuc* [simp]: "hSuc  $\in$  InternalFuns"  
 <proof>

**lemma** *InternalSets\_starset\_n* [simp]: " $*sn*$  X  $\in$  InternalSets"  
 <proof>

**lemma** *InternalSets\_singleton* [simp]: "{x}  $\in$  InternalSets"  
 <proof>

**lemma** *starset\_n\_singleton* [simp]: " $*sn*$  ( $\lambda n.$  {X n}) = {star\_n X}"  
 <proof>

**lemma** *starset\_n\_singleton2*: " $*sn*$  ( $\lambda n.$  {X n}) = {x. x = star\_n X}"  
 <proof>

**lemma** *starset\_n\_empty* [simp]: " $*sn*$  ( $\lambda n.$  {}) = {}"  
 <proof>

**lemma** *InternalSets\_empty* [simp]: "{}  $\in$  InternalSets"  
 <proof>

**lemma** *starset\_n\_subset*:  
 " $*sn*$  X  $\subseteq$   $*sn*$  Y = (eventually ( $\lambda n.$  X n  $\subseteq$  Y n)  $\mathcal{U}$ )"  
 <proof>

**lemma** *starfun\_n\_mult\_star\_n\_right*:  
 " $*fn*$  F n \* star\_n X = ( $*fn*$  ( $\lambda n m.$  F n m \* X n)) n"  
 <proof>

**lemma** *starfun\_n\_mult\_star\_n\_left*:  
 "star\_n X \* ( $*fn*$  F) n = ( $*fn*$  ( $\lambda n m.$  X n \* F n m)) n"  
 <proof>

**lemma** *starfun\_n\_diff*:  
 " $*fn*$  f z -  $*fn*$  g z = ( $*fn*$  ( $\lambda i x.$  f i x - g i x)) z"  
 <proof>

**lemma** *starfun\_n\_inverse*: "inverse (( $*fn*$  f) x) = ( $*fn*$  ( $\lambda i x.$  inverse ((f i) x))) x"  
 <proof>

**lemma** *InternalFuns\_const\_fun* [simp]:  
 " $(\lambda n.$  c)  $\in$  InternalFuns"  
 <proof>

**lemma** *starfun\_n\_divide*:  
 " $((*fn* f) z :: 'a::division\_ring \text{star}) / (*fn* g) z = (*fn* (\lambda i x.$   
 f i x / g i x)) z"  
 <proof>

**lemma** *InternalFuns\_divide*:  
 "[  $f \in \text{InternalFuns}; g \in \text{InternalFuns}$  ]  
 $\implies (\lambda n. (f\ n :: 'a::\text{division\_ring\_star})/g\ n) \in \text{InternalFuns}$ "  
 <proof>

**lemma** *InternalFuns\_id\_fun [simp]*: " $(\lambda n. n) \in \text{InternalFuns}$ "  
 <proof>

**lemma** *FreeUltrafilterNat\_star\_n\_Infinitesimal*:  
 assumes "eventually  $(\lambda n. \text{norm } (X\ n) < \text{inverse}(\text{Suc } n))\ \mathcal{U}$ "  
 shows " $\text{star}_n\ X \in \text{Infinitesimal}$ "  
 <proof>

**lemma** *starset\_n\_atLeastLessThan\_eq*:  
 " $\text{*sn* } (\lambda n. \{(X\ n)..<(Y\ n)\}) = \{\text{star}_n\ X..<\text{star}_n\ Y\}$ "  
 <proof>

**lemma** *starset\_n\_atLeast\_zero\_LessThan\_eq*:  
 " $\text{*sn* } (\lambda n. \{0..<(X\ n)\}) = \{0..<\text{star}_n\ X\}$ "  
 <proof>

**lemma** *starset\_atLeastAtMost*:  
 " $\{\text{of\_nat } m \ .. \text{of\_nat } n\} = \text{*s* } \{m \ .. n\}$ "  
 <proof>

**lemma** *starset\_n\_atLeastAtMost*:  
 " $\text{*sn* } (\lambda n. \{(X\ n)..(Y\ n)\}) = \{\text{star}_n\ X.. \text{star}_n\ Y\}$ "  
 <proof>

**lemma** *starset\_n\_atLeast\_zero\_AtMost*:  
 " $\text{*sn* } (\lambda n. \{0..(X\ n)\}) = \{0.. \text{star}_n\ X\}$ "  
 <proof>

**lemma** *InternalSets\_atLeastLessThan [simp]*: " $\{M..<N\} \in \text{InternalSets}$ "  
 <proof>

**lemma** *InternalSets\_atLeastAtMost [simp]*: " $\{M..N\} \in \text{InternalSets}$ "  
 <proof>

**lemma** *le\_inverse\_HNatInfinite\_Infinitesimal*:  
 "[ $n \in \text{HNatInfinite}; \text{hnorm } x \leq \text{inverse } (\text{hypreal\_of\_hypnat } n)$ ]  
 $\implies x \in \text{Infinitesimal}$ "  
 <proof>

**lemma** *InternalFuns\_hyperpow*:

$f \in \text{InternalFuns} \implies (g : 'a \text{ star} \Rightarrow \text{nat star}) \in \text{InternalFuns}$   
 $\implies (\lambda n. f n \text{ pow } g n) \in \text{InternalFuns}$   
 <proof>

**lemma** *InternalFuns\_hyperpowf [simp]:*  
 $f \in \text{InternalFuns} \implies (\lambda n. x \text{ pow } (f n)) \in \text{InternalFuns}$   
 <proof>

**lemma** *InternalFuns\_hyperpow\_simple [simp]:*  $(\lambda n. x \text{ pow } n) \in \text{InternalFuns}$   
 <proof>

NatStar.thy lemmas

**lemma** *InternalFuns\_add:*  
 $\llbracket f \in \text{InternalFuns}; g \in \text{InternalFuns} \rrbracket \implies (\lambda n. f n + g n) \in \text{InternalFuns}$   
 <proof>

**lemma** *InternalFuns\_mult:*  
 $\llbracket f \in \text{InternalFuns}; g \in \text{InternalFuns} \rrbracket \implies (\lambda n. f n * g n) \in \text{InternalFuns}$   
 <proof>

**lemma** *InternalFuns\_diff:*  
 $\llbracket f \in \text{InternalFuns}; g \in \text{InternalFuns} \rrbracket \implies (\lambda n. f n - g n) \in \text{InternalFuns}$   
 <proof>

**lemma** *InternalFuns\_minus:*  
 $f \in \text{InternalFuns} \implies (\lambda n. - f n) \in \text{InternalFuns}$   
 <proof>

**lemma** *InternalFuns\_mult\_const\_left:*  
 $f \in \text{InternalFuns} \implies (\lambda n. c * f n) \in \text{InternalFuns}$   
 <proof>

**lemma** *InternalFuns\_mult\_const\_right:*  
 $f \in \text{InternalFuns} \implies (\lambda n. f n * c) \in \text{InternalFuns}$   
 <proof>

**lemma** *InternalFuns\_mult\_of\_hypnat [simp]:*  
 $X \in \text{InternalFuns} \implies (\lambda n. \text{of\_hypnat } n * X n) \in \text{InternalFuns}$   
 <proof>

**lemma** *finite\_InternalSets:*  
 assumes "finite X"  
 shows "X ∈ InternalSets"  
 <proof>

**lemma** *InternalFuns\_of\_hypnat:*  
 assumes "f ∈ InternalFuns"  
 shows " $(\lambda n. \text{of\_hypnat } (f n)) \in \text{InternalFuns}$ "  
 <proof>

```

lemma insert_star_n_starset_n:
  "insert (star_n X) (*sn* A) = *sn* (λn. insert (X n) (A n))"
  ⟨proof⟩

lemma inj_starfun_n:
  "inj (*fn* f) = (∀F n in U. inj (f n))"
  ⟨proof⟩

lemma surj_starfun_n:
  "surj (*fn* f) = (∀F n in U. surj (f n))"
  ⟨proof⟩

lemma bij_starfun_n:
  "bij (*fn* f) = (∀F n in U. bij (f n))"
  ⟨proof⟩

lemma
  assumes "bij (*fn* f)"
  shows "(inv (*fn* f)) = *fn* (λn. inv (f n))"
  ⟨proof⟩

lemma starfun_n_o:
  "(λi. (*fn* F) ((*fn* G) i)) = (*fn* (λi m. F i (G i m)))"
  ⟨proof⟩

lemma starfun_starfun_n_o:
  "(λi. (*f* f) ((*fn* F) i)) = (*fn* (λi m. f (F i m)))"
  ⟨proof⟩

lemma InternalFuns_o2:
  "f ∈ InternalFuns ⇒ g ∈ InternalFuns ⇒ (λn. f (g n)) ∈ InternalFuns"
  ⟨proof⟩

lemma InternalFuns_abs_starfun_n [simp]:
  "(λi. abs ((*fn* F) i)) ∈ InternalFuns"
  ⟨proof⟩

lemma InternalFuns_abs:
  "f ∈ InternalFuns ⇒ (λi. abs (f i)) ∈ InternalFuns"
  ⟨proof⟩

HSEQ.thy lemmas

lemma LIMSEQ_0_Infinitesimal:
  assumes "X ⟶ 0"
  shows "star_n X ∈ Infinitesimal"
  ⟨proof⟩

```

Alternative definitions for exponential and real powers to replace the one in

the AFP session eventually.

```
lemma powrat_def2: "x powQ r = root (nat (snd(quotient_of r))) (x powi (fst(quotient_of r)))"
⟨proof⟩
```

```
lemma powrat_powi: "x powQ (of_int n) = x powi n"
⟨proof⟩
```

```
lemma real_root_eq_powrat_inverse': "n > 0 ⇒ x powQ (1 / of_nat n) = root n x"
⟨proof⟩
```

```
lemma "a powa x = lim (λn. a powQ (incseq_of x n))"
⟨proof⟩
```

Next two proofs by Zuzana Frankovska

```
lemma HNatInfinite_pow_real_Infinitesimal:
  assumes "0 ≤ (x::real)" and "x < 1" and "N ∈ HNatInfinite"
  shows "(star_of x) pow N ≈ 0"
⟨proof⟩
```

```
lemma HNatInfinite_pow_Infinitesimal:
  assumes "0 < (x::hypreal)" and "x < 1" and "¬(x ≈ 1)" and "N ∈ HNatInfinite"
  shows "x pow N ≈ 0"
⟨proof⟩
```

```
lemma Infinitesimal_hyperpow_HNatInfinite:
  assumes "hnorm (x::'a::real_normed_div_algebra star) < 1"
  and "N ∈ HNatInfinite"
  and "¬(hnorm x ≈ 1)"
  shows "x pow N ∈ Infinitesimal"
⟨proof⟩
```

HSeries.thy lemmas

```
lemma NSsums_sumhr_HNatInfinite_approx_iff:
  "(f NSsums s) = (∀N ∈ HNatInfinite. sumhr(0, N, f) ≈ of_real s)"
⟨proof⟩
```

```
lemma sumhr_approx:
  "sumhr (0, M, f) ≈ sumhr (0, N, f) ⇒ sumhr (M, N, f) ≈ 0"
⟨proof⟩
```

end

### 3 Hyperinteger numbers

```
theory HyperInt
  imports AuxiliaryNSA "HOL-Nonstandard_Analysis.NSA"
```

begin

type\_synonym hypint = "int star"

abbreviation

hypint\_of\_int :: "int  $\Rightarrow$  int star" where  
"hypint\_of\_int  $\equiv$  star\_of"

abbreviation

hypint\_of\_hypnat :: "hypnat  $\Rightarrow$  hypint" where  
"hypint\_of\_hypnat  $\equiv$  of\_hypnat"

### 3.1 Properties of Hyperintegers

Many properties hold by transfer from the integers

lemma hypint\_nat\_mult\_less\_mono2\_lemma:

"(i::hypint)<j  $\Longrightarrow$  0<k  $\Longrightarrow$  of\_nat k \* i < of\_nat k \* j"  
<proof>

lemma hypint\_mult\_less\_mono2\_lemma:

" $\bigwedge$  i j k. (i::hypint)<j  $\Longrightarrow$  0<k  $\Longrightarrow$  of\_hypnat k \* i < of\_hypnat k \* j"  
<proof>

lemma zero\_le\_imp\_eq\_hypint: " $\bigwedge$ k. (0::hypint)  $\leq$  k  $\Longrightarrow$   $\exists$ n. k = of\_hypnat n"  
<proof>

lemma zero\_less\_imp\_eq\_hypint: " $\bigwedge$ k. (0::hypint) < k  $\Longrightarrow$   $\exists$ n>0. k = of\_hypnat n"  
<proof>

lemma hypint\_less\_mono2: " $\bigwedge$ i j k. [ $i$ < $j$ ; (0::hypint) < k]  $\Longrightarrow$  k\*i < k\*j"  
<proof>

lemma hypint\_less\_imp\_add1\_le: " $\bigwedge$ w z. w < z  $\Longrightarrow$  w + (1::hypint)  $\leq$  z"  
<proof>

lemma hypint\_less\_iff\_hSuc\_add:

" $\bigwedge$ w z. (w :: hypint) < z  $\longleftrightarrow$  ( $\exists$ n. z = w + of\_hypnat (hSuc n))"  
<proof>

lemma hypint\_le\_iff\_add:

" $\bigwedge$ w z. (w :: hypint)  $\leq$  z  $\longleftrightarrow$  ( $\exists$ n. z = w + of\_hypnat n)"  
<proof>

lemma hypint\_le\_iff\_diff:

" $\bigwedge$ w z. (w :: hypint)  $\leq$  z  $\longleftrightarrow$  ( $\exists$ n. w = z - of\_hypnat n)"

*<proof>*

**lemma** *hypint\_less\_zero\_eq\_minus\_of\_hypnat*: "x < (0::hypint)  $\implies \exists n.$   
x = - hypint\_of\_hypnat (n)"  
*<proof>*

**lemma** *hypint\_less\_add1\_eq*: " $\bigwedge w z. (w < z + (1::hypint)) = (w < z \mid w = z)$ "  
*<proof>*

**lemma** *add1\_hypint\_le\_eq*: " $\bigwedge w z. (w + (1::hypint) \leq z) = (w < z)$ "  
*<proof>*

**lemma** *hypint\_le\_diff1\_eq [simp]*: " $\bigwedge w z. (w \leq z - (1::hypint)) = (w < z)$ "  
*<proof>*

**lemma** *hypint\_le\_add1\_eq\_le [simp]*: " $\bigwedge w z. (w < z + (1::hypint)) = (w \leq z)$ "  
*<proof>*

**lemma** *hypint\_one\_le\_iff\_zero\_less*: " $\bigwedge z. ((1::hypint) \leq z) = (0 < z)$ "  
*<proof>*

### 3.2 Embedding of the hyperintegers into other types

**definition**

*of\_hypint* :: "hypint  $\implies$  'a::linordered\_idom star" where  
*of\_hypint\_def [transfer\_unfold]*: "*of\_hypint* = \*f\* *of\_int*"

**lemma** *of\_hypint\_of\_int [simp]*: "*of\_hypint* (*of\_int* z) = *of\_int* z"  
*<proof>*

**lemma** *of\_hypint\_hypint\_of\_int [simp]*: "*of\_hypint* (*hypint\_of\_int* z) =  
*of\_int* z"  
*<proof>*

**lemma** *of\_hypint\_of\_hypnat [simp]*: " $\bigwedge z. \text{of\_hypint } (\text{of\_hypnat } z) = \text{of\_hypnat } z$ "  
*<proof>*

**lemma** *of\_hypint\_of\_nat [simp]*: "*of\_hypint* (*of\_nat* z) = *of\_nat* z"  
*<proof>*

**lemma** *of\_hypint\_0 [simp]*: "*of\_hypint* 0 = 0"  
*<proof>*

**lemma** *of\_hypint\_1 [simp]*: "*of\_hypint* 1 = 1"  
*<proof>*

**lemma** *of\_hypint\_add [simp]*: " $\bigwedge w z. \text{of\_hypint } (w+z) = \text{of\_hypint } w + \text{of\_hypint } z$ "

$z$   
*<proof>*

**lemma** *of\_hypint\_minus [simp]*: " $\wedge z. \text{of\_hypint } (-z) = - (\text{of\_hypint } z)$ "  
*<proof>*

**lemma** *of\_hypint\_diff [simp]*: " $\wedge w z. \text{of\_hypint } (w - z) = \text{of\_hypint } w - \text{of\_hypint } z$ "  
*<proof>*

**lemma** *of\_hypint\_mult [simp]*: " $\wedge w z. \text{of\_hypint } (w * z) = \text{of\_hypint } w * \text{of\_hypint } z$ "  
*<proof>*

**lemma** *of\_hypint\_of\_nat\_eq [simp]*: " $\text{of\_hypint } (\text{of\_nat } n) = \text{of\_nat } n$ "  
*<proof>*

**lemma** *of\_hypint\_nat\_power*: " $\text{of\_hypint } (z ^ n) = \text{of\_hypint } z ^ n$ "  
*<proof>*

**lemma** *of\_hypint\_eq\_iff [simp]*:  
" $\wedge w z. \text{of\_hypint } w = \text{of\_hypint } z \longleftrightarrow w = z$ "  
*<proof>*

**lemma** *of\_hypint\_eq\_0\_iff [simp]*:  
" $\wedge z. \text{of\_hypint } z = 0 \longleftrightarrow z = 0$ "  
*<proof>*

**lemma** *of\_hypint\_0\_eq\_iff [simp]*:  
" $\wedge z. 0 = \text{of\_hypint } z \longleftrightarrow z = 0$ "  
*<proof>*

**lemma** *of\_hypint\_le\_iff [simp]*:  
" $\wedge w z. \text{of\_hypint } w \leq \text{of\_hypint } z \longleftrightarrow w \leq z$ "  
*<proof>*

**lemma** *of\_hypint\_less\_iff [simp]*:  
" $\wedge w z. \text{of\_hypint } w < \text{of\_hypint } z \longleftrightarrow w < z$ "  
*<proof>*

**lemma** *of\_hypint\_0\_le\_iff [simp]*:  
" $\wedge z. 0 \leq \text{of\_hypint } z \longleftrightarrow 0 \leq z$ "  
*<proof>*

**lemma** *of\_hypint\_le\_0\_iff [simp]*:  
" $\wedge z. \text{of\_hypint } z \leq 0 \longleftrightarrow z \leq 0$ "  
*<proof>*

**lemma** *of\_hypint\_0\_less\_iff [simp]*:

" $\wedge z. 0 < \text{of\_hypint } z \iff 0 < z$ "  
(proof)

**lemma** *of\_hypint\_less\_0\_iff [simp]*:  
" $\wedge z. \text{of\_hypint } z < 0 \iff z < 0$ "  
(proof)

**lemma** *of\_hypint\_abs*: " $\wedge z. \text{of\_hypint } (\text{abs } z) = \text{abs}(\text{of\_hypint } z)$ "  
(proof)

**lemma** *of\_hypint\_add\_one*: " $\text{of\_hypint } z + (1 :: \text{real star}) = \text{of\_hypint } (z + 1)$ "  
(proof)

### 3.3 Embedding the naturals into the hyperintegers

**abbreviation**

*hypint\_of\_nat* :: "nat => hypint" where  
"*hypint\_of\_nat* == *of\_nat*"

**lemma** *HypintNat\_eq*: " $\text{Nats} = \{z. \exists N. z = \text{hypint\_of\_nat } N\}$ "  
(proof)

**lemma** *hypint\_of\_nat*:  
"*hypint\_of\_nat* *m* = *star\_n* (%*n*. *int m*)"  
(proof)

**lemma** *HypintNat\_ge\_zero*: " $n \in \mathbb{N} \implies 0 \leq (n :: \text{hypint})$ "  
(proof)

**lemma** *hypint\_of\_hypnat\_hypint\_of\_nat\_eq [simp]*:  
" $\wedge n. \text{hypint\_of\_hypnat } (\text{hypnat\_of\_nat } n) = \text{hypint\_of\_nat } n$ "  
(proof)

**lemma** *abs\_hypint\_of\_hypnat [simp]*: " $|\text{hypint\_of\_hypnat } n| = \text{hypint\_of\_hypnat } n$ "  
(proof)

**lemma** *hypint\_abs\_less\_one\_iff [simp]*: " $\wedge z. (|z| < 1) = (z = (0 :: \text{hypint}))$ "  
(proof)

**lemma** *abs\_hypint\_mult\_eq\_1*:  
" $\wedge m n. |m * n| = 1 \implies |m| = (1 :: \text{hypint})$ "  
(proof)

**lemma** *pos\_hypint\_mult\_eq\_1\_iff\_lemma*:  
" $\wedge m n. (m * n = 1) \implies m = (1 :: \text{hypint}) \mid m = -1$ "  
(proof)

**lemma** *pos\_hypint\_mult\_eq\_1\_iff*:  

$$\bigwedge m n. 0 < (m::\text{hypint}) \implies (m * n = 1) = (m = 1 \wedge n = 1)$$
*<proof>*

**lemma** *hypint\_mult\_eq\_1\_iff*:  

$$\bigwedge m n. (m*n = (1::\text{hypint})) = ((m = 1 \wedge n = 1) \vee (m = -1 \wedge n = -1))$$
*<proof>*

### 3.4 Magnitude of a hyperinteger

**definition**

*hypnat* :: "hypint => hypnat"

**where**

*[transfer\_unfold]*: "hypnat  $\equiv$  \*f\* nat"

**lemma** *hypnat\_hypint [simp]*: " $\bigwedge n. \text{hypnat} (\text{of\_hypnat } n) = n$ "  
*<proof>*

**lemma** *hypnat\_zero [simp]*: "hypnat 0 = 0"  
*<proof>*

**lemma** *hypnat\_one [simp]*: "hypnat 1 = 1"  
*<proof>*

**lemma** *hypint\_hypnat\_eq [simp]*: " $\bigwedge z. \text{of\_hypnat} (\text{hypnat } z) = (\text{if } 0 \leq z \text{ then } z \text{ else } 0)$ "  
*<proof>*

**corollary** *hypnat\_0\_le*: " $0 \leq z \implies \text{of\_hypnat} (\text{hypnat } z) = z$ "  
*<proof>*

**lemma** *hypnat\_le\_0 [simp]*: " $\bigwedge z. z \leq 0 \implies \text{hypnat } z = 0$ "  
*<proof>*

**lemma** *hypnat\_le\_eq\_zle*: " $\bigwedge w z. 0 < w \vee 0 \leq z \implies (\text{hypnat } w \leq \text{hypnat } z) = (w \leq z)$ "  
*<proof>*

**corollary** *hypnat\_mono\_iff*: " $0 < z \implies (\text{hypnat } w < \text{hypnat } z) = (w < z)$ "  
*<proof>*

**corollary** *hypnat\_less\_eq\_less*: " $0 \leq w \implies (\text{hypnat } w < \text{hypnat } z) = (w < z)$ "  
*<proof>*

**lemma** *less\_hypnat\_conj [simp]*: " $\bigwedge w z. (\text{hypnat } w < \text{hypnat } z) = (0 < z \wedge w < z)$ "  
*<proof>*

**lemma** *nonneg\_eq\_hypint*:

```

fixes z :: hypint
assumes "0 ≤ z" and "∧m. z = of_hypnat m ⇒ P"
shows P
⟨proof⟩

lemma hypnat_eq_iff:
  "∧ m w. (hypnat w = m) = (if 0 ≤ w then w = of_hypnat m else m=0)"
⟨proof⟩

corollary hypnat_eq_iff2: "(m = hypnat w) = (if 0 ≤ w then w = of_hypnat
m else m=0)"
⟨proof⟩

lemma hypnat_less_iff: "∧m w. 0 ≤ w ⇒ (hypnat w < m) = (w < of_hypnat
m)"
⟨proof⟩

lemma hypnat_0_iff[simp]: "∧i. hypnat i = 0 ⇔ i≤0"
⟨proof⟩

lemma hypint_eq_iff: "(of_hypnat m = z) = (m = hypnat z ∧ 0 ≤ z)"
⟨proof⟩

lemma zero_less_hypnat_eq [simp]: "(0 < hypnat z) = (0 < z)"
⟨proof⟩

lemma hypnat_add_distrib:
  "∧z z'. [ (0::hypint) ≤ z; 0 ≤ z' ] ⇒ hypnat (z+z') = hypnat
z + hypnat z'"
⟨proof⟩

lemma hypnat_diff_distrib:
  "∧z z'. [ (0::hypint) ≤ z'; z' ≤ z ] ⇒ hypnat (z-z') = hypnat
z - hypnat z'"
⟨proof⟩

lemma hypnat_minus_hypint [simp]: "∧n. hypnat (- (of_hypnat n)) = 0"
⟨proof⟩

lemma less_hypnat_eq_hypint_less: "∧m z. (m < hypnat z) = (of_hypnat
m < z)"
⟨proof⟩

lemma hypnat_add_one: "0 ≤ z ⇒ hypnat z + (1 :: nat star) = hypnat
(z + 1)"
⟨proof⟩

```

### 3.5 The divides relation

**lemma** *hypint\_dvd\_antisym\_nonneg*:

" $\wedge m n. 0 \leq m \implies 0 \leq n \implies m \text{ dvd } n \implies n \text{ dvd } m \implies m = (n::\text{hypint})$ "  
 $\langle \text{proof} \rangle$

**lemma** *hypint\_dvd\_antisym\_abs*:

" $\wedge a b. [(a::\text{hypint}) \text{ dvd } b; b \text{ dvd } a] \implies |a| = |b|$ "  
 $\langle \text{proof} \rangle$

**lemma** *hyperint\_dvd\_diffD*: " $\wedge k m n. k \text{ dvd } m - n \implies k \text{ dvd } n \implies k \text{ dvd } (m::\text{hypint})$ "

$\langle \text{proof} \rangle$

**lemma** *hypint\_dvd\_reduce*: " $\wedge k m n. (k \text{ dvd } n + k * m) = (k \text{ dvd } (n::\text{hypint}))$ "

$\langle \text{proof} \rangle$

**lemma** *hypint\_dvd\_imp\_le\_hypint*:

" $\wedge d i. [i \neq 0; d \text{ dvd } (i::\text{hypint})] \implies |d| \leq |i|$ "  
 $\langle \text{proof} \rangle$

**lemma** *hypint\_dvd\_not\_less*:

" $\wedge m n. [0 < m; m < (n::\text{hypint})] \implies \neg n \text{ dvd } m$ "  
 $\langle \text{proof} \rangle$

**lemma** *hypint\_dvd\_mult\_cancel*:

" $\wedge k m n. [k * m \text{ dvd } k * n; k \neq (0::\text{hypint})] \implies m \text{ dvd } n$ "  
 $\langle \text{proof} \rangle$

**theorem** *dvd\_hypint\_of\_hypnat*:

" $\wedge x y. ((x::\text{hypnat}) \text{ dvd } y) = ((\text{hypint\_of\_hypnat } x) \text{ dvd } (\text{hypint\_of\_hypnat } y))$ "  
 $\langle \text{proof} \rangle$

**lemma** *hypint\_dvd1\_eq[simp]*: " $\wedge x. (x::\text{hypint}) \text{ dvd } 1 = (|x| = 1)$ "

$\langle \text{proof} \rangle$

**lemma** *hypint\_dvd\_mult\_cancel1*:

" $\wedge m n. m \neq (0::\text{hypint}) \implies (m * n \text{ dvd } m) = (|n| = 1)$ "  
 $\langle \text{proof} \rangle$

**lemma** *hypint\_of\_hypnat\_dvd\_iff*:

" $\wedge m z. ((\text{of\_hypnat } m) \text{ dvd } z) = (m \text{ dvd } \text{hypnat } (\text{abs } z))$ "  
 $\langle \text{proof} \rangle$

**lemma** *dvd\_hypint\_iff*: " $\wedge m z. (z \text{ dvd } \text{of\_hypnat } m) = (\text{hypnat } (\text{abs } z) \text{ dvd } m)$ "

$\langle \text{proof} \rangle$

**lemma** *hypnat\_dvd\_iff*: " $(\text{hypnat } z \text{ dvd } m) = (\text{if } 0 \leq z \text{ then } (z \text{ dvd } \text{of\_hypnat } m))$ "

m) else m = 0)"  
 ⟨proof⟩

**lemma eq\_hypnat\_hypnat\_iff:**  
 "0 ≤ z ⇒ 0 ≤ z' ⇒ hypnat z = hypnat z' ↔ z = z'"  
 ⟨proof⟩

**lemma hypnat\_power\_eq:**  
 "∧z. 0 ≤ z ⇒ hypnat (z ^ n) = hypnat z ^ n"  
 ⟨proof⟩

**lemma hypint\_dvd\_imp\_le:**  
 "∧n z. [ z dvd n; 0 < n ] ⇒ z ≤ (n::hypint)"  
 ⟨proof⟩

**lemma hypint\_period:**  
 "∧a c d t x. (a::hypint) dvd d ⇒ a dvd (x + t) ↔ a dvd ((x + c \* d) + t)"  
 ⟨proof⟩

### 3.6 Division on hypint

**lemma hypint\_of\_nat\_div:** "hypint\_of\_nat(m div n) = hypint\_of\_nat m div (hypint\_of\_nat n)"  
 ⟨proof⟩

**lemma hypint\_of\_hypnat\_div:** "∧m n. hypint\_of\_hypnat(m div n) = hypint\_of\_hypnat m div (hypint\_of\_hypnat n)"  
 ⟨proof⟩

**lemma HypintNats\_div:** "[ (n::hypint) ∈ ℕ; m ∈ ℕ ] ⇒ n div m ∈ ℕ"  
 ⟨proof⟩

**lemma hypint\_div\_monol:** "∧a a' b. [ a ≤ a'; 0 < (b::hypint) ] ⇒ a div b ≤ a' div b"  
 ⟨proof⟩

**lemma hypint\_div\_neg\_pos\_less0:** "∧a b. [ a < (0::hypint); 0 < b ] ⇒ a div b < 0"  
 ⟨proof⟩

**lemma hypint\_div\_nonneg\_neg\_le0:** "∧a b. [ (0::hypint) ≤ a; b < 0 ] ⇒ a div b ≤ 0"  
 ⟨proof⟩

**lemma hypint\_div\_nonpos\_pos\_le0:** "∧a b. [ (a::hypint) ≤ 0; b > 0 ] ⇒ a div b ≤ 0"  
 ⟨proof⟩

**lemma** *hypint\_pos\_imp\_div\_nonneg\_iff*: " $\bigwedge a b. (0::\text{hypint}) < b \implies (0 \leq a \text{ div } b) = (0 \leq a)$ "

*<proof>*

**lemma** *hypint\_neg\_imp\_div\_nonneg\_iff*:

" $\bigwedge a b. b < (0::\text{hypint}) \implies (0 \leq a \text{ div } b) = (a \leq (0::\text{hypint}))$ "

*<proof>*

**lemma** *hypint\_pos\_imp\_div\_neg\_iff*: " $\bigwedge a b. (0::\text{hypint}) < b \implies (a \text{ div } b < 0) = (a < 0)$ "

*<proof>*

**lemma** *hypint\_neg\_imp\_div\_neg\_iff*: " $\bigwedge a b. b < (0::\text{hypint}) \implies (a \text{ div } b < 0) = (0 < a)$ "

*<proof>*

**lemma** *hypint\_nonneg1\_imp\_div\_pos\_iff*:

" $\bigwedge a b. (0::\text{hypint}) \leq a \implies (a \text{ div } b > 0) = (a \geq b \wedge b > 0)$ "

*<proof>*

### 3.7 Properties of the set of embedded integers

**lemma** *of\_int\_eq\_star\_of [simp]*: "*of\_int = star\_of*"

*<proof>*

**lemma** *Ints\_eq\_Standard*: " $(\mathbb{Z} :: \text{int star set}) = \text{Standard}$ "

*<proof>*

**lemma** *hypint\_of\_int\_mem\_Ints [simp]*: "*hypint\_of\_int*  $z \in \mathbb{Z}$ "

*<proof>*

**lemma** *hypint\_of\_nat\_Suc [simp]*:

"*hypint\_of\_nat* (*Suc*  $n$ ) = *hypint\_of\_nat*  $n$  +  $(1::\text{hypint})$ "

*<proof>*

**lemma** *of\_int\_eq\_add [rule\_format]*:

" $\forall d::\text{hypint}. \text{of\_int } m = \text{of\_int } n + d \longrightarrow d \in \text{range of\_int}$ "

*<proof>*

### 3.8 Infinite hyperintegers

**definition**

*HIntInfinite* :: "*hypint set*" where

"*HIntInfinite* =  $\{z. z \notin \mathbb{Z}\}$ "

**lemma** *Ints\_not\_HIntInfinite\_iff*: " $(x \in \text{Ints}) = (x \notin \text{HIntInfinite})$ "

*<proof>*

**lemma** *HIntInfinite\_not\_Ints\_iff*: " $(x \in \text{HIntInfinite}) = (x \notin \text{Ints})$ "

*<proof>*

**lemma** *star\_of\_neq\_HIntInfinite*: " $Z \in \text{HIntInfinite} \implies \text{star\_of } n \neq Z$ "  
*<proof>*

**lemma** *HIntInfinite\_minus\_iff [simp]*:  
" $(- Z \in \text{HIntInfinite}) = (Z \in \text{HIntInfinite})$ "  
*<proof>*

**lemma** *star\_of\_less\_pos\_HIntInfinite*:  
assumes " $0 < Z$ " and " $Z \in \text{HIntInfinite}$ "  
shows " $\text{star\_of } z < Z$ "  
*<proof>*

**lemma** *star\_of\_gt\_neg\_HIntInfinite*:  
assumes " $Z < 0$ "  
and " $Z \in \text{HIntInfinite}$ "  
shows " $Z < \text{star\_of } z$ "  
*<proof>*

**lemma** *zero\_not\_HIntInfinite [simp]*: " $0 \notin \text{HIntInfinite}$ "  
*<proof>*

**lemma** *star\_of\_less\_abs\_HIntInfinite*:  
" $Z \in \text{HIntInfinite} \implies \text{star\_of } z < \text{abs } Z$ "  
*<proof>*

**lemma** *hypint\_of\_nat\_less\_abs\_HIntInfinite*:  
" $y \in \text{HIntInfinite} \implies \text{hypint\_of\_nat } n < |y|$ "  
*<proof>*

**lemma** *Ints\_less\_pos\_HIntInfinite*: " $\llbracket 0 < y; x \in \mathbb{Z}; y \in \text{HIntInfinite} \rrbracket \implies x < y$ "  
*<proof>*

**lemma** *Ints\_gt\_neg\_HIntInfinite*: " $\llbracket y < 0; x \in \mathbb{Z}; y \in \text{HIntInfinite} \rrbracket \implies y < x$ "  
*<proof>*

**lemma** *Ints\_less\_abs\_HIntInfinite*: " $\llbracket x \in \mathbb{Z}; y \in \text{HIntInfinite} \rrbracket \implies x < \text{abs } y$ "  
*<proof>*

**lemma** *Ints\_le\_abs\_HIntInfinite*: " $\llbracket x \in \mathbb{Z}; y \in \text{HIntInfinite} \rrbracket \implies x \leq \text{abs } y$ "  
*<proof>*

**lemma** *Nats\_less\_abs\_HIntInfinite*: " $\llbracket x \in \mathbb{N}; y \in \text{HIntInfinite} \rrbracket \implies x$ "

< abs y"  
<proof>

**lemma** Nats\_le\_abs\_HIntInfinite: "[ x ∈ ℕ; y ∈ HIntInfinite ] ⇒ x < abs y"  
<proof>

**lemma** Ints\_pos\_downward\_closed:  
"[ 0 ≤ y; x ∈ Ints; (y::hypint) ≤ x ] ⇒ y ∈ ℤ"  
<proof>

**lemma** Ints\_neg\_upward\_closed:  
"[ y ≤ 0; x ∈ Ints; x ≤ (y::hypint) ] ⇒ y ∈ ℤ"  
<proof>

**lemma** HIntInfinite\_pos\_upward\_closed:  
"[ 0 ≤ x; x ∈ HIntInfinite; x ≤ y ] ⇒ y ∈ HIntInfinite"  
<proof>

**lemma** HIntInfinite\_neg\_downward\_closed:  
"[ x ≤ 0; x ∈ HIntInfinite; y ≤ x ] ⇒ y ∈ HIntInfinite"  
<proof>

**lemma** HIntInfinite\_pos\_add:  
"[ 0 ≤ x; 0 ≤ y; x ∈ HIntInfinite ] ⇒ x + y ∈ HIntInfinite"  
<proof>

**lemma** HIntInfinite\_neg\_add:  
"[ x ≤ 0; y ≤ 0; x ∈ HIntInfinite ] ⇒ x + y ∈ HIntInfinite"  
<proof>

**lemma** HIntInfinite\_add\_Ints:  
"[ x ∈ HIntInfinite; y ∈ Ints ] ⇒ x + y ∈ HIntInfinite"  
<proof>

**lemma** HIntInfinite\_diff:  
"[ x ∈ HIntInfinite; y ∈ Ints ] ⇒ x - y ∈ HIntInfinite"  
<proof>

**lemma** Ints\_def2: "ℤ = {z. ∃Z. z = hypint\_of\_int Z}"  
<proof>

**lemma** HIntInfinite\_hypint\_of\_hypnat\_iff:  
"(hypint\_of\_hypnat n ∈ HIntInfinite) = (n ∈ HNatInfinite)"  
<proof>

**lemma** two\_hyperpow\_ge\_one [simp]: "(1::hypint) ≤ 2 pow n"  
<proof>

### 3.9 Embedding of hypertintegers in hyperreals

abbreviation

`hypreal_of_hypint` :: "hypint  $\Rightarrow$  hypreal" where  
"hypreal\_of\_hypint  $\equiv$  of\_hypint"

**lemma** `of_hypint_numeral [simp]`: "of\_hypint (numeral v) = numeral v"  
 $\langle$ proof $\rangle$

**lemma** `real_of_int_le_real_of_nat_iff`: "(real\_of\_int a  $\leq$  real\_of\_nat b) = (a  $\leq$  int b)"  
 $\langle$ proof $\rangle$

**lemma** `minus_real_of_int_le_real_of_nat_iff`: "(- real\_of\_int a  $\leq$  real\_of\_nat b) = (- a  $\leq$  int b)"  
 $\langle$ proof $\rangle$

**lemma** `hypreal_of_hypint_le_hypreal_of_hypnat_iff`:  
" $\wedge$  a b. (hypreal\_of\_hypint a  $\leq$  of\_hypnat b) = (a  $\leq$  of\_hypnat b)"  
 $\langle$ proof $\rangle$

**lemma** `minus_hypreal_of_hypint_le_hypreal_of_hypnat_iff`:  
" $\wedge$  a b. (- hypreal\_of\_hypint a  $\leq$  hypreal\_of\_hypnat b) = (- a  $\leq$  of\_hypnat b)"  
 $\langle$ proof $\rangle$

**lemma** `hnorm_of_hypint [simp]`:  
" $\wedge$  z. hnorm (of\_hypint z :: 'a :: {real\_normed\_algebra\_1, linordered\_idom} star) = abs(of\_hypint z)"  
 $\langle$ proof $\rangle$

**lemma** `of_hypnat_hypnat [simp]`: " $\wedge$  z. 0  $\leq$  z  $\implies$  of\_hypnat(hypnat z) = of\_hypint z"  
 $\langle$ proof $\rangle$

**lemma** `HInfinite_def3`: "HInfinite = {x.  $\forall$  z  $\in$  Ints. hypreal\_of\_hypint z < hnorm x}"  
 $\langle$ proof $\rangle$

**lemma** `HInfinite_of_hypint_HNatInfinite_hypnat`:  
" $\llbracket$  (of\_hypint x :: 'a :: {real\_normed\_algebra\_1, linordered\_idom} star)  $\in$  HInfinite; x > 0  $\rrbracket$   
 $\implies$  hypnat x  $\in$  HNatInfinite"  
 $\langle$ proof $\rangle$

**lemma** `HNatInfinite_hypnat_HInfinite_of_hypint`:  
"hypnat x  $\in$  HNatInfinite  
 $\implies$  (of\_hypint x :: 'a :: {real\_normed\_algebra\_1, linordered\_idom} star)  $\in$  HInfinite"

*<proof>*

end

## 4 Internal sets and functions

theory Internal

imports "HOL-Nonstandard\_Analysis.NatStar" HyperInt AuxiliaryNSA

begin

definition

n\_star :: "'a star  $\Rightarrow$  (nat  $\Rightarrow$  'a)" where  
"n\_star x = (SOME y. x = star\_n y)"

definition

n\_starset :: "'a star set  $\Rightarrow$  (nat  $\Rightarrow$  'a set)" ("\*ns\* \_" [80] 80) where  
"\*ns\* A = (SOME As. A = Iset (star\_n As))"

definition

n\_starfun :: "('a star  $\Rightarrow$  'b star)  $\Rightarrow$  (nat  $\Rightarrow$  ('a  $\Rightarrow$  'b))" ("\*nf\* \_"  
[80] 80) where  
"\*nf\* F = (SOME Fs. F = Ifun (star\_n Fs))"

definition

starfun2\_n :: "(nat  $\Rightarrow$  ('a  $\Rightarrow$  'b  $\Rightarrow$  'c))  $\Rightarrow$  ('a star  $\Rightarrow$  'b star  $\Rightarrow$  'c  
star)" ("\*fn2\* \_" [80] 80) where  
"\*fn2\* F = ( $\lambda$ x. Ifun (star\_n F  $\star$  x))"

definition

n\_starfun2 :: "('a star  $\Rightarrow$  'b star  $\Rightarrow$  'c star)  $\Rightarrow$  (nat  $\Rightarrow$  ('a  $\Rightarrow$  'b  
 $\Rightarrow$  'c))" ("\*nf2\* \_" [80] 80) where  
"\*nf2\* F = (SOME Gs. F = ( $\lambda$ x. Ifun (star\_n Gs  $\star$  x)))"

definition

InternalFuns2 :: "('a star  $\Rightarrow$  'b star  $\Rightarrow$  'c star) set" where  
"InternalFuns2 = {X.  $\exists$ F. X = \*fn2\* F}"

lemma n\_star\_def2: "n\_star x = (SOME y. y  $\in$  Rep\_star x)"

*<proof>*

lemma star\_n\_star [simp]:

"star\_n (n\_star x) = x"

*<proof>*

lemma n\_star\_star\_n\_eq\_ultra:

"eventually ( $\lambda$ n. n\_star (star\_n X) n = X n)  $\mathcal{U}$ "

*<proof>*

lemma n\_star\_star\_of\_eq\_ultra:

```

"eventually (λn. n_star (star_of X) n = X) U"
⟨proof⟩

lemma n_starset_def2: "*ns* S = (SOME Sn. S = *sn* Sn)"
⟨proof⟩

lemma Iset_n_starset_eq [simp]:
  assumes "X ∈ InternalSets"
  shows "Iset (star_n (*ns* X)) = X"
⟨proof⟩

lemma Iset_n_starset_empty [simp]:
  "Iset(star_n (*ns* {})) = {}"
⟨proof⟩

lemma n_starset_eq_ultra:
  "eventually (λn. (*ns* (Iset (star_n X))) n = X n) U"
⟨proof⟩

lemma n_starset_starset_n_eq_ultra:
  "eventually (λn. (*ns* *sn* X) n = X n) U"
⟨proof⟩

lemma n_starset_starset_n_eq_ultra2:
  "eventually (λn. X n = (*ns* *sn* X) n) U"
⟨proof⟩

lemma n_starset_empty_ultra:
  "eventually (λn. (*ns* {}) n = {}) U"
⟨proof⟩

lemma P_n_starset_starset_n_ultra_iff:
  "eventually (λn. P ((*ns* *sn* X) n)) U = (eventually (λn. P (X n))
U)"
⟨proof⟩

lemma InternalSets_starset_n_starset [simp]:
  assumes "A ∈ InternalSets"
  shows "*sn* (*ns* A) = A"
⟨proof⟩

lemma starset_n_star_n:
  "(star_n X ∈ *sn* As) = (eventually (λn. (X n) ∈ (As n)) U)"
⟨proof⟩

lemma n_starfun_def2: "*nf* F = (SOME Fn. F = *fn* Fn)"
⟨proof⟩

lemma InternalFuns_starfun_n_starfun [simp]:

```

```

    assumes "F ∈ InternalFuns"
    shows "*fn* (*nf* F) = F"
  ⟨proof⟩

lemma Ifun_eq_cancel:
  "(Ifun (star_n F) = Ifun (star_n G)) = (eventually (λn. F n = G n) U)"
  ⟨proof⟩

lemma starfun_n_eq_cancel:
  "(*fn* F = *fn* G) = (eventually (λn. F n = G n) U)"
  ⟨proof⟩

lemma n_starfun_starfun_n_eq_ultra:
  "eventually (λn. (*nf* (*fn* F)) n = F n) U"
  ⟨proof⟩

lemma n_starfun_starfun_eq_ultra:
  "eventually (λn. (*nf* (*f* f)) n = f) U"
  ⟨proof⟩

lemma P_n_starfun_starfun_n_ultra_iff:
  "eventually (λn. P ((*nf* *fn* X) n)) U = (eventually (λn. P (X n))
  U)"
  ⟨proof⟩

lemma starfun2_eq_starfun2n: "*f2* f = *fn2* (λn. f)"
  ⟨proof⟩

lemma starfun2_n_Ifun_starfun_n: "*fn2* f = (λx. Ifun ((*fn* f) x))"
  ⟨proof⟩

lemma starfun2_n_Ifun_starfun_n2: "( *fn2* f) x y = ((*fn* f) x) * y"
  ⟨proof⟩

The definition of binary internal functions is as expected

lemma starfun2_n: "(*fn2* f) (star_n X) (star_n Y) = star_n (λn. f n
  (X n) (Y n))"
  ⟨proof⟩

lemma n_starfun2_starfun2_n_eq_ultra:
  "eventually (λn. (*nf2* (*fn2* F)) n = F n) U"
  ⟨proof⟩

lemma n_starfun2_starfun2_eq_ultra: "eventually (λn. (*nf2* (*f2* f))
  n = f) U"
  ⟨proof⟩

lemma InternalFuns2_starfun2_n_starfun2 [simp]:
  assumes "f ∈ InternalFuns2"

```

shows " $*fn2* (*nf2* f) = f$ "  
<proof>

lemma transfer\_fun2\_eq [transfer\_intro]:  
"[[  $\bigwedge X Y. f (star_n X) (star_n Y) = g (star_n X) (star_n Y)$   
 $\equiv$  eventually  $(\lambda n. F n (X n) (Y n) = G n (X n) (Y n)) \mathcal{U}$  ]]  
 $\implies f = g \equiv$  eventually  $(\lambda n. F n = G n) \mathcal{U}$ "  
<proof>

lemma starfun2\_n\_eq\_cancel:  
" $(*fn2* F = *fn2* G) =$  (eventually  $(\lambda n. F n = G n) \mathcal{U})$ "  
<proof>

lemma n\_starfun\_starfun2\_n\_eq\_ultra:  
" $\forall_F n$  in  $\mathcal{U}. (*nf* (*fn2* f) (star_n X)) n = f n (X n)$ "  
<proof>

lemma starfun2\_n\_minus: " $- (*fn2* f) x y = (*fn2* (\lambda i j x. - (f i j) x)) x y$ "  
<proof>

lemma starfun2\_n\_diff:  
" $(*fn2* f) y z - (*fn2* g) y z = (*fn2* (\lambda i j x. f i j x - g i j x)) y z$ "  
<proof>

lemma starfun2\_n\_inverse: "inverse  $((*fn2* f) x y) = (*fn2* (\lambda i j x. inverse ((f i j) x))) x y$ "  
<proof>

lemma starfun2\_n\_mult:  
" $(*fn2* f) y z * (*fn2* g) y z = (*fn2* (\lambda i j x. f i j x * g i j x)) y z$ "  
<proof>

lemma starfun2\_n\_divide:  
" $((*fn2* f) y z :: 'a :: division\_ring star) / (*fn2* g) y z =$   
 $(*fn2* (\lambda i j x. f i j x / g i j x)) y z$ "  
<proof>

lemma starfun\_n\_starfun2\_omega: " $(*fn* F) z = (*f2* F) whn z$ "  
<proof>

lemma InternalFuns2\_divide:  
"[[  $f \in$  InternalFuns2;  $g \in$  InternalFuns2 ]]  
 $\implies (\lambda m n. (f m n :: 'a :: division\_ring star) / g m n) \in$  InternalFuns2"  
<proof>

lemma InternalFuns2\_mult:

" $\llbracket f \in \text{InternalFuns2}; g \in \text{InternalFuns2} \rrbracket \implies (\lambda m n. f m n * g m n) \in \text{InternalFuns2}$ "  
 <proof>

**lemma** *InternalFuns2\_diff*:  
 " $\llbracket f \in \text{InternalFuns2}; g \in \text{InternalFuns2} \rrbracket \implies (\lambda m n. f m n - g m n) \in \text{InternalFuns2}$ "  
 <proof>

**lemma** *InternalFuns2\_minus*:  
 " $f \in \text{InternalFuns2} \implies (\lambda m n. - f m n) \in \text{InternalFuns2}$ "  
 <proof>

**lemma** *InternalFuns\_starfun2\_n [simp]*: " $*fn2* f \in \text{InternalFuns2}$ "  
 <proof>

**lemma** *ext\_obj*: " $(\forall x y. f x y = g x y) \implies f = g$ "  
 <proof>

**lemma** *InternalFuns2\_const\_fun*: " $(\lambda n m. c) \in \text{InternalFuns2}$ "  
 <proof>

**lemma** *InternalFuns2\_InternalFuns\_iff*:  
 " $((\lambda m n. f m) \in \text{InternalFuns2}) = (f \in \text{InternalFuns})$ "  
 <proof>

**lemma** *InternalSets\_hnorm\_InternalFuns\_gt [simp]*:  
 assumes " $X \in \text{InternalFuns}$ "  
 shows " $\{n. \text{hnorm}(X n) > m\} \in \text{InternalSets}$ "  
 <proof>

**lemma** *InternalSets\_hnorm\_starfun\_gt [simp]*:  
 " $\bigwedge m. \{n. \text{hnorm}((*f* X) n) > m\} \in \text{InternalSets}$ "  
 <proof>

## 4.1 Overspill theorem

Theorem 3.3.18, Nonstandard Analysis, A. Robinson

**lemma** *InternalFuns\_ub*:  
 assumes " $X \in \text{InternalFuns}$ "  
 and " $\forall n \in \mathbb{N}. \text{hnorm}(X n) \leq m$ "  
 shows " $\exists v \in \text{HNatInfinite}. \forall n < v. \text{hnorm}(X n) \leq m$ "  
 <proof>

**lemma** *starfun\_least*:  
 " $\forall n \in \mathbb{N}. \text{hnorm}((*f* X) n) \leq m \implies \exists v \in \text{HNatInfinite}. \forall n < v. \text{hnorm}(*f* X) n \leq m$ "  
 <proof>

## 4.2 The Sequential Theorem

Theorem 3.3.20, Nonstandard Analysis, A. Robinson

```

lemma InternalFuns_Sequential_Theorem:
  assumes Int_f: "f ∈ InternalFuns"
  and f_approx_zero: "∀n∈Nats. f n ≈ (0::'a::{real_normed_div_algebra,
semiring_1_cancel} star)"
  shows "∃N∈HNatInfinite.∀n<N. f n ≈ 0"
⟨proof⟩

lemma InternalFuns_Sequential_Theorem2:
  assumes "f ∈ InternalFuns" "g ∈ InternalFuns"
  "∀n∈Nats. f n ≈ (g n :: 'a::{real_normed_div_algebra, semiring_1_cancel}
star)"
  shows "∃N∈HNatInfinite.∀n<N. f n ≈ g n"
⟨proof⟩

lemma Sequential_Theorem:
  "∀n∈Nats. (*f* f) n ≈ (0::'a::{real_normed_div_algebra, semiring_1_cancel}
star)
  ⇒ ∃N∈HNatInfinite.∀n<N. (*f* f) n ≈ 0"
⟨proof⟩

lemma Sequential_Theorem2:
  "∀n∈Nats. (*f* f) n ≈ ((*f* g) n :: 'a::{real_normed_div_algebra, semiring_1_cancel}
star)
  ⇒ ∃N∈HNatInfinite.∀n<N. (*f* f) n ≈ (*f* g) n"
⟨proof⟩

lemma hyperpower_starfun_n:
  "(x::'a::{power star} pow n = ( *fn* (λn m. (n_star x) n ^ m)) n"
  ⟨proof⟩

lemma InternalFuns2_hyperpower [simp]: "(λm. (pow) x) ∈ InternalFuns2"
⟨proof⟩

lemma hyperpow_approx_one_ex:
  assumes "x ≈ 1"
  shows "∃N∈HNatInfinite.∀n<N. x pow n ≈ (1::'a::{real_normed_div_algebra
star})"
  ⟨proof⟩

end

theory Hyperfinite_Set
imports "HOL-Nonstandard_Analysis.NatStar" Internal
begin

```

## 5 Hyperfinite sets

**definition**

```
hyperfinite :: "'a star set  $\Rightarrow$  bool" where
  "hyperfinite S  $\equiv$  unstar (star_n ( $\lambda$ n. finite ((*ns* S) n)))"
```

Usual definition of a hyperfinite set

**lemma hyperfinite\_iff:**

```
"hyperfinite S = (eventually ( $\lambda$ n. finite ((*ns* S) n))  $\mathcal{U}$ )"
<proof>
```

**lemma hyperfinite\_starset\_n\_iff:**

```
"hyperfinite (*sn* X) = (eventually ( $\lambda$ n. finite (X n))  $\mathcal{U}$ )"
<proof>
```

**lemma hyperfinite\_empty [simp]:** "hyperfinite {}"

<proof>

**lemma hyperfinite\_Iset\_iff\_starP\_finite:**

```
"hyperfinite (Iset X) = (*p* finite) X"
<proof>
```

**lemma finite\_hyperfinite\_starset:**

```
"finite X  $\implies$  hyperfinite (*s* X)"
<proof>
```

**lemma hyperfinite\_starset\_finite:**

```
"hyperfinite (*s* X)  $\implies$  finite X"
<proof>
```

**lemma hyperfinite\_starset\_iff\_finite:**

```
"hyperfinite (*s* X) = finite X"
<proof>
```

### 5.1 Properties of hyperfinite sets

**lemma hyperfinite\_UnI:**

```
"[[ X  $\in$  InternalSets; Y  $\in$  InternalSets;
  hyperfinite X; hyperfinite Y ]]  $\implies$  hyperfinite (X  $\cup$  Y)"
<proof>
```

**lemma hyperfinite\_starset\_n\_UnI:**

```
"[[ hyperfinite (*sn* X); hyperfinite (*sn* Y) ]]  $\implies$  hyperfinite (*sn*
X  $\cup$  *sn* Y)"
<proof>
```

**lemma hyperfinite\_starset\_n\_Un [iff]:**

```
"hyperfinite ((*sn* X)  $\cup$  (*sn* Y))  $\iff$  hyperfinite (*sn* X)  $\wedge$  hyperfinite
(*sn* Y)"
<proof>
```

```

lemma hyperfinite_singleton [simp]: "hyperfinite {x}"
⟨proof⟩

lemma finite_hyperfinite:
  assumes "finite X"
  shows "hyperfinite X"
⟨proof⟩

lemma hyperfinite_insert [simp]:
  "A ∈ InternalSets ⇒ hyperfinite (insert a A) ⇔ hyperfinite A"
⟨proof⟩

lemma hyperfinite_starset_n_insert [simp]:
  "hyperfinite (insert a (*sn* A)) ⇔ hyperfinite (*sn* A)"
⟨proof⟩

lemma hyperfinite_subset:
  assumes "A ∈ InternalSets" and "B ∈ InternalSets" and
    "hyperfinite B" and "A ⊆ B"
  shows "hyperfinite A"
⟨proof⟩

lemma hyperfinite_starset_n_subset:
  "[[ hyperfinite (*sn* Y); (*sn* X) ⊆ (*sn* Y) ]] ⇒ hyperfinite (*sn* X)"
⟨proof⟩

lemma hyperfinite_Int [simp, intro]:
  "[[ X ∈ InternalSets; Y ∈ InternalSets;
    hyperfinite X ∨ hyperfinite Y ]] ⇒ hyperfinite (X ∩ Y)"
⟨proof⟩

lemma hyperfinite_IntI1:
  "[[ X ∈ InternalSets; Y ∈ InternalSets; hyperfinite X ]] ⇒ hyperfinite (X ∩ Y)"
⟨proof⟩

lemma hyperfinite_IntI2:
  "[[ X ∈ InternalSets; Y ∈ InternalSets; hyperfinite Y ]] ⇒ hyperfinite (X ∩ Y)"
⟨proof⟩

lemma hyperfinite_starset_n_Int [simp, intro]:
  "hyperfinite (*sn* X) ∨ hyperfinite (*sn* Y) ⇒ hyperfinite (*sn* X
  ∩ *sn* Y)"
⟨proof⟩

```

```

lemma hyperfinite_starset_n_IntI1:
  "hyperfinite (*sn* X)  $\implies$  hyperfinite (*sn* X  $\cap$  *sn* Y)"
  <proof>

lemma hyperfinite_starset_n_IntI2:
  "hyperfinite (*sn* Y)  $\implies$  hyperfinite (*sn* X  $\cap$  *sn* Y)"
  <proof>

lemma hyperfinite_Diff [simp, intro]:
  "[[ X  $\in$  InternalSets; Y  $\in$  InternalSets; hyperfinite X ]]  $\implies$  hyperfinite
(X - Y)"
  <proof>

lemma hyperfinite_starset_n_Diff [simp, intro]:
  "hyperfinite (*sn* X)  $\implies$  hyperfinite (*sn* X - *sn* Y)"
  <proof>

lemma hyperfinite_Diff2 [simp]:
  assumes "X  $\in$  InternalSets"
  and "Y  $\in$  InternalSets"
  and "hyperfinite Y"
  shows "hyperfinite (X - Y)  $\longleftrightarrow$  hyperfinite X"
  <proof>

lemma hyperfinite_starset_n_Diff2 [simp]:
  "hyperfinite (*sn* X)
 $\implies$  hyperfinite (*sn* X - *sn* Y)  $\longleftrightarrow$  hyperfinite (*sn* X)"
  <proof>

lemma hyperfinite_Diff_insert [iff]:
  "[[ X  $\in$  InternalSets; Y  $\in$  InternalSets ]]
 $\implies$  hyperfinite (X - insert a Y)  $\longleftrightarrow$  hyperfinite (X - Y)"
  <proof>

lemma hyperfinite_starset_n_Diff_insert [iff]:
  "hyperfinite (*sn* X - insert a (*sn* Y))  $\longleftrightarrow$  hyperfinite (*sn* X -
*sn* Y)"
  <proof>

lemma hyperfinite_Compl [simp]:
  "[[ X  $\in$  InternalSets; hyperfinite (X :: 'a star set) ]]
 $\implies$  hyperfinite (- X)  $\longleftrightarrow$  hyperfinite (UNIV :: 'a star set)"
  <proof>

lemma hyperfinite_starset_n_Compl [simp]:
  "hyperfinite (*sn* X :: 'a star set)
 $\implies$  hyperfinite (- (*sn* X))  $\longleftrightarrow$  hyperfinite (UNIV :: 'a star set)"
  <proof>

```

```

lemma hyperfinite_Collect_not[simp]:
  "[[ {x :: 'a star. P x} ∈ InternalSets; hyperfinite {x :: 'a star. P
x} ]]"
  ⇒ hyperfinite {x. ¬ P x} ↔ hyperfinite (UNIV :: 'a star set)"
⟨proof⟩

```

```

lemma hyperfinite_starset_Collect_not[simp]:
  "hyperfinite {x :: 'a star. (*P* P) x}
  ⇒ hyperfinite {x. ¬ (*P* P) x} ↔ hyperfinite (UNIV :: 'a star
set)"
⟨proof⟩

```

```

lemma hypnat_hyperfinite_atLeastLessThan [simp]:
  "hyperfinite {M::hypnat..

```

```

lemma hyperfinite_hypnat_set_atLeastAtMost:
  "hyperfinite {of_nat m .. of_nat n} :: hypnat set)"
⟨proof⟩

```

## 5.2 Hyperfold function

```

definition hyperfold :: "('a star ⇒ 'b star ⇒ 'b star) ⇒ 'b star ⇒
'a star set => 'b star" where
  "hyperfold f z A = star_n (λn. Finite_Set.fold ((*nf2* f) n) (n_star
z n) ((*ns* A) n))"

```

hyperfold is indeed an internal function

```

lemma hyperfold:
  "hyperfold (*fn2* Fs) (star_n Zs) (*sn* As) =
  star_n (λn. Finite_Set.fold (Fs n) (Zs n) (As n))"
⟨proof⟩

```

```

lemma hyperfold_empty [simp]: "f ∈ InternalFuns2 ⇒ hyperfold f z {}
= z"
⟨proof⟩

```

```

definition hyperfold_image ::
  "('b star ⇒ 'b star ⇒ 'b star) ⇒ ('a star ⇒ 'b star) ⇒ 'b star
⇒ 'a star set ⇒ 'b star"
  where "hyperfold_image f g = hyperfold (λx y. f (g x) y)"

```

end

## 6 Hyperfinite summation

```

theory HyperSum
  imports "HOL-Nonstandard_Analysis.HSeries" Hyperfinite_Set HyperInt

```

begin

definition (in comm\_monoid\_add) hypersum :: "('b star  $\Rightarrow$  'a star)  $\Rightarrow$  'b star set  $\Rightarrow$  'a star" where  
"hypersum f A = star\_n ( $\lambda n$ . sum ((\*nf\* f) n) ((\*ns\* A) n))"

notation hypersum (" $\sum_h$ ")  
syntax "\_hypersum" :: "pttrn  $\Rightarrow$  'b star set  $\Rightarrow$  'c star  $\Rightarrow$  'c star" (" $(2\sum_h$   
(\_/∈\_)./ \_)" [0, 51, 10] 10)  
translations " $\sum_h i \in A. b$ " == "CONST hypersum ( $\lambda i. b$ ) A"

lemma hypersum:

"hypersum (\*fn\* f) (\*sn\* A) = star\_n ( $\lambda n$ . sum (f n) (A n))"  
<proof>

lemma hypersum\_starfun2\_n:

"hypersum ((\*fn2\* f) (star\_n X)) (\*sn\* A) = star\_n ( $\lambda n$ . sum (f n (X n)) (A n))"  
<proof>

## 6.1 Hypersum properties

lemma hypersum\_empty [simp]:

assumes "f  $\in$  InternalFuns" shows "hypersum f {} = 0"  
<proof>

lemma hypersum\_starfun\_n\_empty [simp]: "hypersum (\*fn\* f) {} = 0"

<proof>

lemma setsum\_singleton [simp]: "sum f {x} = f x"

<proof>

lemma hypersum\_singleton [simp]:

assumes "f  $\in$  InternalFuns"  
shows "hypersum f {a} = f a"

<proof>

lemma hypersum\_F\_neutral\_ivl [simp]:

"hypersum ( $\lambda n. 0$ ) {m.. $n$ } = 0"  
<proof>

lemma hypersum\_head\_hSuc:

assumes "f  $\in$  InternalFuns" and "m  $\leq$  n"  
shows "hypersum f {m..n} = f m + hypersum f {hSuc m..n}"

<proof>

lemma hypersum\_head\_upt\_hSuc:

assumes "f  $\in$  InternalFuns" and "m < n"

shows "hypersum f {m.. $n$ } = f m + hypersum f {hSuc m.. $n$ }"  
 <proof>

lemma hypersum\_ub\_add\_hypnat:  
 assumes "f ∈ InternalFuns"  
 and "(m::hypnat) ≤ n + 1"  
 shows "hypersum f {m.. $n + p$ } = hypersum f {m.. $n$ } + hypersum f {n + 1.. $n + p$ }"  
 <proof>

lemma hypersum\_op\_ivl\_Suc[simp]:  
 assumes "f ∈ InternalFuns"  
 shows "hypersum f {m.. $\text{hSuc } n$ } = (if n < m then 0 else hypersum f {m.. $n$ } + f(n))"  
 <proof>

lemma hypersum\_insert:  
 assumes "f ∈ InternalFuns"  
 and "A ∈ InternalSets"  
 and "hyperfinite A"  
 and "a ∉ A"  
 shows "hypersum f (insert a A) = f a + hypersum f A"  
 <proof>

lemma hypersum\_subset\_diff:  
 assumes "f ∈ InternalFuns"  
 and "A ∈ InternalSets"  
 and "B ∈ InternalSets"  
 and "B ⊆ A"  
 and "hyperfinite A"  
 shows "hypersum f A = hypersum f (A - B) + hypersum f B"  
 <proof>

lemma hypersum\_diff:  
 assumes "f ∈ InternalFuns"  
 and "A ∈ InternalSets"  
 and "B ∈ InternalSets"  
 and "B ⊆ A"  
 and "hyperfinite A"  
 shows "hypersum f (A - B) = hypersum f A - ((hypersum f B)::('a::ab\_group\_add\_star))"  
 <proof>

lemma hypersum\_add\_hypnat\_ivl:  
 assumes "f ∈ InternalFuns"  
 and "m ≤ n" and "n ≤ p"  
 shows "hypersum f {m.. $n$ } + hypersum f {n.. $p$ } = hypersum f {m.. $p$ ::hypnat}"  
 <proof>

```

lemma hypersum_diff_hypnat_iv1:
  "[[ (f::hypnat=>'a::ab_group_add_star) ∈ InternalFuns; m ≤ n; n ≤ p
  ]]
  ==> hypersum f {m..<p} - hypersum f {m..<n} = hypersum f {n..<p::hypnat}"
  <proof>

lemma InternalFuns2_hypersum:
  assumes "(λj i. f j i) ∈ InternalFuns2"
  shows "(λi. hypersum (f i) {m..n}) ∈ InternalFuns"
  <proof>

lemma hypersum_starfun_atLeastAtMost:
  "hypersum (*f* f) {M..N} = (*f2* (λm n. sum f {m..n})) M N"
  <proof>

lemma hypersum_starfun_atLeastLessThan:
  "hypersum (*f* f) {M..<N} = (*f2* (λm n. sum f {m..<n})) M N"
  <proof>

lemma hypersum_shift_bounds_cl_hSuc_iv1:
  assumes "f ∈ InternalFuns"
  shows "hypersum f {hSuc m..hSuc n} = hypersum (λi. f(hSuc i)){m..n}"
  <proof>

lemma hypersum_shift_bounds_hSuc_iv1:
  assumes "f ∈ InternalFuns"
  shows "hypersum f {hSuc m..<hSuc n} = hypersum (λi. f(hSuc i)){m..<n}"
  <proof>

lemma lemma_n_starfun2_add_eq:
  "eventually (λn. ((*nf2* (λx. (+) ((*fn* f) x))) n) = (λx. (+) (f n x)))
  U"
  <proof>

lemma (in comm_monoid_add) setsum_def2:
  "sum f A = (if finite A then (Finite_Set.fold (λx y. (+) (f x) y) 0
  A) else 0)"
  <proof>

lemma hypersum_def2:
  assumes "f ∈ InternalFuns"
  shows "hypersum f A = (if hyperfinite A then hyperfold_image (+) f 0
  A else 0)"
  <proof>

```

## 6.2 The sumhr function as a hypersum.

```
lemma sumhr_hypersum_eq:
  "sumhr(M,N,f) = hypersum (*f* f) {M..<N}"
  ⟨proof⟩
```

```
lemma NSsums_hypersum_HNatInfinite_approx_iff:
  "(f NSsums s) = (∀N∈HNatInfinite. hypersum (*f* f) {0..<N} ≈ of_real
s)"
  ⟨proof⟩
```

## 6.3 Hyper-convergence

**definition**

```
HyperCauchy :: "(hypnat ⇒ 'a::real_normed_vector star) ⇒ bool" where
  "HyperCauchy X = (∀M∈HNatInfinite. ∀N∈HNatInfinite. X M ≈ X N)"
```

**definition**

```
HyperSummable :: "(hypnat ⇒ 'a::real_normed_vector star) ⇒ bool" where
  "HyperSummable F = HyperCauchy (λN. hypersum F {0..<N})"
```

**lemma NSCauchy\_HyperCauchy\_starfun\_iff:**

```
"NSCauchy X = HyperCauchy (*f* X)"
  ⟨proof⟩
```

**lemma Cauchy\_HyperCauchy\_starfun\_iff:**

```
"Cauchy X = HyperCauchy (*f* X)"
  ⟨proof⟩
```

**lemma NSsummable\_HyperSummable\_starfun\_iff:**

```
"NSsummable f = HyperSummable (*f* f)"
  ⟨proof⟩
```

**lemma starfun\_setsum\_atLeastLessThan\_eq\_hypesetsum\_fun:**

```
"(*f* (λn. sum f {k n..<g n})) = (λN. hypersum (*f* f) {( (*f* k) N)..<
(( *f* g) N)})"
  ⟨proof⟩
```

**lemma starfun\_setsum\_atLeast\_zero\_hypesetsum\_fun:**

```
"(*f* (λn. sum f {0..<g n})) = (λN. hypersum (*f* f) {0..< (( *f* g)
N)})"
  ⟨proof⟩
```

**lemma starfun\_setsum\_atLeast\_zero\_hypesetsum:**

```
"(*f* (λn. sum f {0..<n})) = (λN. hypersum (*f* f) {0..<N})"
  ⟨proof⟩
```

**lemma hypersum\_atLeast\_zero\_starfun:**

```
"hypersum (*f* f) {0..<N} = (*f* (λn. sum f {0..<n})) N"
```

*<proof>*

**lemma** *hypersum\_starfun\_atLeast0LessThan:*

"hypersum (\*f\* f) {0..<N::nat star} = (\*f\* (λn. ∑ i<n. f i)) N"

*<proof>*

**lemma** *hypersum\_atLeast\_zero\_star\_n\_starfun\_n:*

"hypersum (\*fn\* f) {0..<star\_n X} = star\_n((λn. sum (f n) {0..<X n}))"

*<proof>*

**lemma** *hypersum\_atLeast\_zero\_starfun\_n:*

"hypersum (\*fn\* f) {0..<N} = (\*fn\* (λn m. sum (f n) {0..<m})) N"

*<proof>*

**lemma** *hypersum\_geometric:*

assumes "x ≠ 1"

shows "hypersum (λn. x pow n) {0..<n} = (x pow n - 1) / (x - 1::'a::field star)"

*<proof>*

**lemma** *HyperSummable\_geometric:*

assumes "hnorm (x::'a::{real\_normed\_field} star) < 1"

and "¬hnorm x ≈ 1"

shows "HyperSummable (λN. x pow N)"

*<proof>*

**lemma** *summable\_convergent\_sumr\_iff:*

"summable f = convergent (λn. sum f {0..<n})"

*<proof>*

**lemma** *HyperSummable\_starfun\_summable\_iff:*

"(HyperSummable (\*f\* f)::hypnat => 'a::banach star) = (summable f)"

*<proof>*

**lemma** *HyperSummable\_def2:*

"HyperSummable f = (∃s. ∀N∈HNatInfinite. hypersum f {0..<N} ≈ s)"

*<proof>*

**lemma** *HyperSummable\_def3:*

assumes "f ∈ InternalFuns"

shows "HyperSummable f = (∀M∈HNatInfinite.∀N∈HNatInfinite. hypersum f {M..<N} ≈ 0)"

*<proof>*

**lemma** *HyperSummable\_starfun\_n:*

"HyperSummable (\*fn\* f) = (∀M∈HNatInfinite.∀N∈HNatInfinite. hypersum

$(\text{*fn* } f) \{M..<N\} \approx 0)$ "  
 ⟨proof⟩

**lemma** *atLeastLessThanhSuc\_atLeastAtMost*:  
 " $\bigwedge m n. \{m..<hSuc\ n\} = \{m..n\}$ "  
 ⟨proof⟩

**lemma** *hypersum\_op\_ivl\_Suc2[simp]*:  
 " $\text{hypersum } (\text{*fn* } f) \{m..<hSuc\ n\} = (\text{if } n < m \text{ then } 0 \text{ else hypersum } (\text{*fn* } f) \{m..<n\} + (\text{*fn* } f) n)$ "  
 ⟨proof⟩

**lemma** *hypersum\_op\_ivl\_Suc2'*:  
 " $\text{hypersum } (\text{*fn* } f) \{m..(n::\text{hypnat})\} = (\text{if } n < m \text{ then } 0 \text{ else hypersum } (\text{*fn* } f) \{m..<n\} + (\text{*fn* } f) n)$ "  
 ⟨proof⟩

**lemma** *HyperSummable\_starfun\_n\_HNatInfinite*:  
 assumes "*HyperSummable* ( $\text{*fn* } f)$ "  
 and " $K \in \text{HNatInfinite}$ "  
 shows " $(\text{*fn* } f) K \approx 0$ "  
 ⟨proof⟩

**lemma** *HyperSummable\_def'*: "*HyperSummable*  $F = \text{HyperCauchy } (\lambda N. \text{hypersum } F \{0..N\})$ "  
 ⟨proof⟩

**lemma** *HyperSummable\_def3'*:  
 assumes " $f \in \text{InternalFuns}$ "  
 shows "*HyperSummable*  $f = (\forall M \in \text{HNatInfinite}. \forall N \in \text{HNatInfinite}. \text{hypersum } f \{M..N\} \approx 0)$ "  
 ⟨proof⟩

**lemma** *HyperSummable\_def4*:  
 "*HyperSummable* ( $\text{*fn* } f) = (\exists s. \forall N \in \text{HNatInfinite}. \text{hypersum } (\text{*fn* } f) \{0..N\} \approx s)$ "  
 ⟨proof⟩

**lemma** *HyperSummable\_shift\_hSuc*:  
 assumes " $f \in \text{InternalFuns}$ "  
 and "*HyperSummable*  $f$ "  
 shows "*HyperSummable*  $(\lambda n. f (hSuc\ n))$ "  
 ⟨proof⟩

**lemma** *InternalFuns\_hypersum\_starfun\_n\_Interval*:  
 " $(\lambda N. \text{hypersum } (\text{*fn* } F) \{M..<N\}) \in \text{InternalFuns}$ "  
 ⟨proof⟩

```

lemma InternalFuns_hypersum:
  assumes "f ∈ InternalFuns"
  shows "(λn. hypersum f {m..<n}) ∈ InternalFuns"
  ⟨proof⟩

lemma InternalFuns_hypersum_starfun_n_Interval2:
  "(λN. hypersum (*fn* F) {M..N}) ∈ InternalFuns"
  ⟨proof⟩

lemma InternalFuns_hypersum2:
  "f ∈ InternalFuns ⇒ (λn. hypersum f {m..n}) ∈ InternalFuns"
  ⟨proof⟩

lemma NSsummable_NSCauchy_setsum:
  "NSsummable f = NSCauchy (λn. sum f {0..<n})"
  ⟨proof⟩

lemma Hypersum_Comparison_Theorem_Nats:
  assumes "f ∈ InternalFuns" "g ∈ InternalFuns"
    "HyperSummable f" "HyperSummable g"
    "∀n∈Nats. f n ≈ g n"
    "N ∈ Nats"
  shows "hypersum f {0..<N} ≈ hypersum g {0..<N}"
  ⟨proof⟩

lemma Hypersum_Comparison_Theorem_HNatInfinite:
  assumes int_f: "f ∈ InternalFuns"
    and int_g: "(g::nat star ⇒ 'a::{real_normed_div_algebra, semiring_1_cancel}
star) ∈ InternalFuns"
    and hyperSums: "HyperSummable f" "HyperSummable g"
    and inf_close_fg: "∀n∈Nats. f n ≈ g n"
    and inf_N: "N ∈ HNatInfinite"
  shows "hypersum f {0..<N} ≈ hypersum g {0..<N}"
  ⟨proof⟩

Full version of the Summation Comparison Theorem

lemma Hypersum_Comparison_Theorem:
  "[[ f ∈ InternalFuns;
  (g::nat star ⇒ 'a::{real_normed_div_algebra, semiring_1_cancel}
star) ∈ InternalFuns;
  HyperSummable f; HyperSummable g; ∀n∈Nats. f n ≈ g n ]
  ⇒ hypersum f {0..<N} ≈ hypersum g {0..<N}"
  ⟨proof⟩

lemma Hypersum_Comparison_Theorem':
  "[[ f ∈ InternalFuns;
  (g::nat star ⇒ 'a::{real_normed_div_algebra, semiring_1_cancel}

```

```

star) ∈ InternalFuns;
  HyperSummable f; HyperSummable g; ∀ n ∈ Nats. f n ≈ g n ]
  ⇒ hypersum f {0..N} ≈ hypersum g {0..N}"
⟨proof⟩

lemma HyperSummable_hypersum_approx:
  "[[ HyperSummable f; M ∈ HNatInfinite; N ∈ HNatInfinite ]
  ⇒ hypersum f {0..M} ≈ hypersum f {0..N}"
⟨proof⟩

lemma hnorm_hypersum:
  "[[ f ∈ InternalFuns; A ∈ InternalSets ] ⇒ hnorm (hypersum f A) ≤
hypersum (λ i. hnorm (f i)) A"
⟨proof⟩

lemma hypersum_mono':
  assumes internal_f: "f ∈ InternalFuns"
    and internal_g: "g ∈ InternalFuns"
    and internal_setK: "K ∈ InternalSets"
    and fg_mono: "(∀ i. i ∈ K → f i ≤ ((g i)::('b::ordered_comm_monoid_add
star)))"
  shows "hypersum f K ≤ hypersum g K"
⟨proof⟩

lemma hypersum_mono:
  "[[ f ∈ InternalFuns; g ∈ InternalFuns; K ∈ InternalSets;
  (∀ i. i ∈ K ⇒ f i ≤ ((g i)::('b::ordered_comm_monoid_add star)))
  ]
  ]
  ⇒ hypersum f K ≤ hypersum g K"
⟨proof⟩

lemma hypersum_nonneg:
  assumes "f ∈ InternalFuns" "A ∈ InternalSets"
    "∀ x ∈ A. (0::'a::ordered_comm_monoid_add star) ≤ f x"
  shows "0 ≤ hypersum f A"
⟨proof⟩

lemma abs_hypersum_abs [simp]:
  "[[ (f :: 'a star ⇒ ('b::ordered_ab_group_add_abs) star) ∈ InternalFuns;
A ∈ InternalSets ]
  ]
  ⇒ abs(hypersum (λ i. abs(f i)) A) = hypersum (λ i. abs(f i)) A"
⟨proof⟩

lemma InternalFuns_hnorm_starfun_n [simp]:
  "(λ i. hnorm ((*fn* F) i)) ∈ InternalFuns"
⟨proof⟩

lemma InternalFuns_hnorm:

```

```

    "f ∈ InternalFuns ⇒ (λn. hnorm (f n)) ∈ InternalFuns"
  ⟨proof⟩

lemma HyperSummable_Cauchy:
  "HyperSummable (*fn* F) ⇒ (∀ e ∈ Reals. e > 0 → (∃ N. ∀ m ≥ N. ∀ n. hnorm
  (hypersum (*fn* F) {m..<n}) < e))"
  ⟨proof⟩

lemma Cauchy_HyperSummable:
  "(∀ e ∈ Reals. e > 0 → (∃ N ∈ Nats. ∀ m ≥ N. ∀ n. hnorm (hypersum (*fn* F)
  {m..<n}) < e)) ⇒ HyperSummable (*fn* F)"
  ⟨proof⟩

lemma Cauchy_HyperSummable':
  "(∀ e ∈ Reals. e > 0 → (∃ N ∈ Nats. ∀ m ≥ N. ∀ n ≥ N. hnorm (hypersum (*fn*
  F) {m..<n}) < e)) ⇒
  HyperSummable (*fn* F)"
  ⟨proof⟩

lemma HyperSummable_Cauchy':
  assumes H: "HyperSummable (*fn* F)"
  shows "(∀ e ∈ Reals. e > 0 → (∃ N. ∀ m ≥ N. ∀ n ≥ N. hnorm (hypersum (*fn*
  F) {m..<n}) < e))"
  ⟨proof⟩

lemma HyperSummable_comparison_test:
  assumes "f ∈ InternalFuns" "g ∈ InternalFuns"
  "∃ k ∈ Nats. ∀ n. k ≤ n → hnorm ((f::hypnat => 'a::banach star)
  n) ≤ g n"
  "HyperSummable g"
  shows "HyperSummable f"
  ⟨proof⟩

lemma HyperSummable_hypreal_comparison_test:
  "[[ (f::hypnat ⇒ hypreal) ∈ InternalFuns;
  g ∈ InternalFuns;
  ∀ n. f n ≥ 0;
  ∃ k ∈ Nats. ∀ n. k ≤ n → f n ≤ g n;
  HyperSummable g ] ] ⇒ HyperSummable f"
  ⟨proof⟩

lemma HyperSummable_comparison_test_norm:
  "f ∈ InternalFuns
  ⇒ HyperSummable (λn. hnorm ((f::hypnat => 'a::banach star) n))
  ⇒ HyperSummable f"
  ⟨proof⟩

lemma hypersum_atLeastLessThan_starfun_n:
  "hypersum (*fn* f) {M..<N} = (*fn2* (λi m n. sum (f i) {m..<n})) M"

```

*N*"  
<proof>

**lemma** *hypersum\_minus*:  
"[[  $f \in \text{InternalFuns}; A \in \text{InternalSets}$  ]]  
 $\implies \text{hypersum } (\lambda x. - (f x)::'a::\text{ab\_group\_add\_star}) A = - \text{hypersum } f$   
*A*"  
<proof>

**lemma** *HyperSummable\_minus*:  
"[[  $f \in \text{InternalFuns}; \text{HyperSummable } f$  ]]  $\implies \text{HyperSummable } (\lambda n. - f n)$ "  
<proof>

**lemma** *HyperSummable\_negf\_iff*:  
" $f \in \text{InternalFuns} \implies (\text{HyperSummable } (\lambda n. - f n)) = (\text{HyperSummable } f)$ "  
<proof>

**lemma** *hypersum\_left\_distrib*:  
assumes " $f \in \text{InternalFuns}$ " " $A \in \text{InternalSets}$ "  
shows " $\text{hypersum } f A * (r::'a::\text{semiring\_0\_star}) = \text{hypersum } (\lambda n. f n * r) A$ "  
<proof>

**lemma** *hypersum\_right\_distrib*:  
assumes " $f \in \text{InternalFuns}$ " " $A \in \text{InternalSets}$ "  
shows " $(r::'a::\text{semiring\_0\_star}) * \text{hypersum } f A = \text{hypersum } (\lambda n. r * f n) A$ "  
<proof>

**lemma** *hypersum\_product*:  
"[[  $f \in \text{InternalFuns}; g \in \text{InternalFuns}; A \in \text{InternalSets}; B \in \text{InternalSets}$  ]]  
]  
 $\implies \text{hypersum } f A * \text{hypersum } g B =$   
 $\text{hypersum } (\lambda i. \text{hypersum } (\lambda j. f i * (g j::'a::\text{semiring\_0\_star})) B)$   
*A*"  
<proof>

**lemma** *starfun2\_to\_starfun\_n\_lemma*:  
" $(\lambda n. (*f2* f) (\text{star}_n X) ((*fn* F) n)) = *fn* (\lambda n m. f (X n) (F n m))$ "  
<proof>

**lemma** *starfun2\_to\_starfun\_n\_lemma2*:  
" $(\lambda n. (*f2* f) ((*fn* F) n) (\text{star}_n X)) = *fn* (\lambda n m. f (F n m) (X n))$ "  
<proof>

**lemma** *hypersum\_divide\_distrib*:  
assumes " $f \in \text{InternalFuns}$ "  
" $A \in \text{InternalSets}$ "

```

  shows "hypersum f A / (r::'a::field star) = hypersum (λn. f n / r)
A"
⟨proof⟩

```

```

lemma HyperSummable_HFinite_mult_left:
  "[[ f ∈ InternalFuns; (c::'a::{real_normed_algebra} star) ∈ HFinite;
HyperSummable f ] ]
  ⇒ HyperSummable (λn. c * f n)"
⟨proof⟩

```

```

lemma HyperSummable_divide:
  assumes "f ∈ InternalFuns"
          "HyperSummable f"
          "r ∉ Infinitesimal"
  shows "HyperSummable (λn. f n / (r::'a::{real_normed_field} star))"
⟨proof⟩

```

end

## 7 The floor and ceiling functions extended to hypernumbers

```

theory HyperArchimedean
imports HyperInt
begin

```

### 7.1 The nonstandard floor function

```

definition
  hfloor :: "'a::floor_ceiling star ⇒ hypint"  (<<open_block notation=<mixfix
floor>>[_]>>) where
  [transfer_unfold]: "hfloor x = (*f* floor) x"

```

```

lemma hfloor:
  "hfloor (star_n X) = star_n (λn. floor (X n))"
⟨proof⟩

```

```

lemma hfloor_correct: "∧x. of_hypint (hfloor x) ≤ x ∧ x < of_hypint
(hfloor x + 1)"
⟨proof⟩

```

```

lemma hfloor_unique: "∧x z. [ of_hypint z ≤ x; x < of_hypint z + 1 ]
⇒ hfloor x = z"
⟨proof⟩

```

```

lemma hfloor_eq_zero [simp]: "[[ 0 ≤ x; x < 1 ] ] ⇒ hfloor x = 0"
⟨proof⟩

```

**lemma** *of\_hypint\_floor\_le*: "of\_hypint (hfloor x) ≤ x"  
 ⟨proof⟩

**lemma** *le\_hfloor\_iff*: " $\bigwedge x z. z \leq \text{hfloor } x \iff \text{of\_hypint } z \leq x$ "  
 ⟨proof⟩

**lemma** *hfloor\_less\_iff*: " $\bigwedge x z. \text{hfloor } x < z \iff x < \text{of\_hypint } z$ "  
 ⟨proof⟩

**lemma** *less\_hfloor\_iff*: " $\bigwedge x z. z < \text{hfloor } x \iff \text{of\_hypint } z + 1 \leq x$ "  
 ⟨proof⟩

**lemma** *hfloor\_le\_iff*: " $\bigwedge x z. \text{hfloor } x \leq z \iff x < \text{of\_hypint } z + 1$ "  
 ⟨proof⟩

**lemma** *hfloor\_mono*: " $\bigwedge x y. x \leq y \implies \text{hfloor } x \leq \text{hfloor } y$ "  
 ⟨proof⟩

**lemma** *hfloor\_less\_cancel*: " $\bigwedge x y. \text{hfloor } x < \text{hfloor } y \implies x < y$ "  
 ⟨proof⟩

**lemma** *hfloor\_of\_hypint [simp]*: " $\bigwedge z. \text{hfloor } (\text{of\_hypint } z) = z$ "  
 ⟨proof⟩

**lemma** *hfloor\_of\_int [simp]*: " $\text{hfloor } (\text{of\_int } z) = \text{hypint\_of\_int } z$ "  
 ⟨proof⟩

**lemma** *hfloor\_of\_hypnat [simp]*: " $\bigwedge n. \text{hfloor } (\text{of\_hypnat } n) = \text{of\_hypnat } n$ "  
 ⟨proof⟩

**lemma** *hfloor\_of\_nat [simp]*: " $\text{hfloor } (\text{of\_nat } n) = \text{of\_nat } n$ "  
 ⟨proof⟩

Floor with numerals

**lemma** *hfloor\_zero [simp]*: " $\text{hfloor } 0 = 0$ "  
 ⟨proof⟩

**lemma** *hfloor\_one [simp]*: " $\text{hfloor } 1 = 1$ "  
 ⟨proof⟩

**lemma** *hfloor\_star\_of [simp]*: " $\text{hfloor } (\text{star\_of } x) = \text{star\_of } x$ "  
 ⟨proof⟩

**lemma** *star\_of\_hfloor [simp]*: " $\text{star\_of}(\text{floor } x) = \text{hfloor } (\text{star\_of } x)$ "  
 ⟨proof⟩

**lemma** *hfloor\_number\_of [simp]*: " $\text{hfloor } (\text{numeral } v :: \text{hypreal}) = (\text{numeral } v)$ "  
 ⟨proof⟩

```

v :: hypint)"
⟨proof⟩

lemma zero_le_hfloor [simp]: "0 ≤ hfloor x ↔ 0 ≤ x"
  ⟨proof⟩

lemma one_le_hfloor [simp]: "1 ≤ hfloor x ↔ 1 ≤ x"
  ⟨proof⟩

lemma numeral_le_hfloor [simp]: "∧x. numeral v ≤ hfloor x ↔ numeral
v ≤ x"
  ⟨proof⟩

lemma zero_less_hfloor [simp]: "0 < hfloor x ↔ 1 ≤ x"
  ⟨proof⟩

lemma one_less_hfloor [simp]: "1 < hfloor x ↔ 2 ≤ x"
  ⟨proof⟩

lemma numeral_less_hfloor [simp]:
  "∧x. numeral v < hfloor x ↔ numeral v + 1 ≤ x"
  ⟨proof⟩

lemma hfloor_le_zero [simp]: "hfloor x ≤ 0 ↔ x < 1"
  ⟨proof⟩

lemma hfloor_le_one [simp]: "hfloor x ≤ 1 ↔ x < 2"
  ⟨proof⟩

lemma hfloor_le_numeral [simp]:
  "∧x. hfloor x ≤ numeral v ↔ x < numeral v + 1"
  ⟨proof⟩

lemma hfloor_less_zero [simp]: "hfloor x < 0 ↔ x < 0"
  ⟨proof⟩

lemma hfloor_less_one [simp]: "hfloor x < 1 ↔ x < 1"
  ⟨proof⟩

lemma hfloor_less_numeral [simp]:
  "∧x. hfloor x < numeral v ↔ x < numeral v"
  ⟨proof⟩

lemma hfloor_add_of_hypint [simp]: "hfloor (x + of_hypint z) = hfloor
x + z"
  ⟨proof⟩

lemma hfloor_add_of_int [simp]: "∧x. hfloor (x + of_int z) = hfloor
x + of_int z"

```

*<proof>*

**lemma** *hfloor\_add\_numeral [simp]:*  
"hfloor (x + numeral v) = hfloor x + numeral v"  
*<proof>*

**lemma** *hfloor\_add\_one [simp]:* "hfloor (x + 1) = hfloor x + 1"  
*<proof>*

**lemma** *hfloor\_diff\_of\_hypint [simp]:* "hfloor (x - of\_hypint z) = hfloor  
x - z"  
*<proof>*

**lemma** *hfloor\_diff\_of\_int [simp]:* "hfloor (x - of\_int z) = hfloor x -  
of\_int z"  
*<proof>*

**lemma** *hfloor\_diff\_numeral [simp]:*  
"hfloor (x - numeral v) = hfloor x - numeral v"  
*<proof>*

**lemma** *hfloor\_diff\_one [simp]:* "hfloor (x - 1) = hfloor x - 1"  
*<proof>*

**lemma** *hfloor\_add\_one\_gt\_zero:*  
" $\bigwedge y. 0 < y \implies 0 < \text{hfloor } y + 1$ "  
*<proof>*

## 7.2 The nonstandard ceiling function

**definition**

*hceiling :: "'a::floor\_ceiling star => hypint" (<<open\_block notation=<mixfix  
ceiling>>[\_]>>) where*  
*[transfer\_unfold]: "hceiling x = - hfloor (- x)"*

**lemma** *hceiling\_correct:* "of\_hypint (hceiling x) - 1 < x  $\wedge$  x  $\leq$  of\_hypint  
(hceiling x)"  
*<proof>*

**lemma** *hceiling\_unique:* "[of\_hypint z - 1 < x; x  $\leq$  of\_hypint z]  $\implies$  hceiling  
x = z"  
*<proof>*

**lemma** *le\_of\_int\_hceiling:* "x  $\leq$  of\_hypint (hceiling x)"  
*<proof>*

**lemma** *hceiling\_le\_iff:* "hceiling x  $\leq$  z  $\iff$  x  $\leq$  of\_hypint z"  
*<proof>*

**lemma less\_hceiling\_iff:**  $"z < \text{hceiling } x \longleftrightarrow \text{of\_hypint } z < x"$   
 ⟨proof⟩

**lemma hceiling\_less\_iff:**  $"\text{hceiling } x < z \longleftrightarrow x \leq \text{of\_hypint } z - 1"$   
 ⟨proof⟩

**lemma le\_hceiling\_iff:**  $"z \leq \text{hceiling } x \longleftrightarrow \text{of\_hypint } z - 1 < x"$   
 ⟨proof⟩

**lemma hceiling\_mono:**  $"x \geq y \implies \text{hceiling } x \geq \text{hceiling } y"$   
 ⟨proof⟩

**lemma hceiling\_less\_cancel:**  $"\text{hceiling } x < \text{hceiling } y \implies x < y"$   
 ⟨proof⟩

**lemma hceiling\_of\_hypint [simp]:**  $"\text{hceiling } (\text{of\_hypint } z) = z"$   
 ⟨proof⟩

**lemma hceiling\_of\_int [simp]:**  $"\text{hceiling } (\text{of\_int } z) = \text{of\_int } z"$   
 ⟨proof⟩

**lemma hceiling\_of\_hypnat [simp]:**  $"\text{hceiling } (\text{of\_hypnat } n) = \text{of\_hypnat } n"$   
 ⟨proof⟩

**lemma hceiling\_of\_nat [simp]:**  $"\text{hceiling } (\text{of\_nat } n) = \text{of\_nat } n"$   
 ⟨proof⟩

Ceiling with numerals

**lemma hceiling\_zero [simp]:**  $"\text{hceiling } 0 = 0"$   
 ⟨proof⟩

**lemma hceiling\_one [simp]:**  $"\text{hceiling } 1 = 1"$   
 ⟨proof⟩

**lemma hceiling\_numeral [simp]:**  $"\text{hceiling } (\text{numeral } v) = \text{numeral } v"$   
 ⟨proof⟩

**lemma hceiling\_le\_zero [simp]:**  $"\text{hceiling } x \leq 0 \longleftrightarrow x \leq 0"$   
 ⟨proof⟩

**lemma hceiling\_le\_one [simp]:**  $"\text{hceiling } x \leq 1 \longleftrightarrow x \leq 1"$   
 ⟨proof⟩

**lemma hceiling\_le\_numeral [simp]:**  
 $"\bigwedge x. \text{hceiling } x \leq \text{numeral } v \longleftrightarrow x \leq \text{numeral } v"$   
 ⟨proof⟩

**lemma hceiling\_less\_zero [simp]:**  $"\text{hceiling } x < 0 \longleftrightarrow x \leq -1"$

$\langle proof \rangle$

**lemma** `hceiling_less_one [simp]: "hceiling x < 1  $\longleftrightarrow$  x  $\leq$  0"`  
 $\langle proof \rangle$

**lemma** `hceiling_less_numeral [simp]:`  
 $\text{"}\wedge x. \text{ hceiling } x < \text{ numeral } v \longleftrightarrow x \leq \text{ numeral } v - 1\text{"}$   
 $\langle proof \rangle$

**lemma** `zero_le_hceiling [simp]: "0  $\leq$  hceiling x  $\longleftrightarrow$  -1 < x"`  
 $\langle proof \rangle$

**lemma** `one_le_hceiling [simp]: "1  $\leq$  hceiling x  $\longleftrightarrow$  0 < x"`  
 $\langle proof \rangle$

**lemma** `numeral_le_hceiling [simp]:`  
 $\text{"}\wedge x. \text{ numeral } v \leq \text{ hceiling } x \longleftrightarrow \text{ numeral } v - 1 < x\text{"}$   
 $\langle proof \rangle$

**lemma** `zero_less_hceiling [simp]: "0 < hceiling x  $\longleftrightarrow$  0 < x"`  
 $\langle proof \rangle$

**lemma** `one_less_hceiling [simp]: "1 < hceiling x  $\longleftrightarrow$  1 < x"`  
 $\langle proof \rangle$

**lemma** `numeral_less_hceiling [simp]:`  
 $\text{"}\wedge x. \text{ numeral } v < \text{ hceiling } x \longleftrightarrow \text{ numeral } v < x\text{"}$   
 $\langle proof \rangle$

**lemma** `hceiling_add_of_hypint [simp]: "hceiling (x + of_hypint z) = hceiling x + z"`  
 $\langle proof \rangle$

**lemma** `hceiling_add_of_int [simp]: "hceiling (x + of_int z) = hceiling x + of_int z"`  
 $\langle proof \rangle$

**lemma** `hceiling_add_numeral [simp]:`  
 $\text{"hceiling (x + numeral v) = hceiling x + numeral v"}$   
 $\langle proof \rangle$

**lemma** `hceiling_add_one [simp]: "hceiling (x + 1) = hceiling x + 1"`  
 $\langle proof \rangle$

**lemma** `hceiling_diff_of_hypint [simp]: "hceiling (x - of_hypint z) = hceiling x - z"`  
 $\langle proof \rangle$

**lemma** `hceiling_diff_of_int [simp]: "hceiling (x - of_int z) = hceiling`

$x - \text{of\_int } z$   
*<proof>*

**lemma** *hceiling\_diff\_numeral* [simp]:  
"hceiling (x - numeral v) = hceiling x - numeral v"  
*<proof>*

**lemma** *hceiling\_diff\_one* [simp]: "hceiling (x - 1) = hceiling x - 1"  
*<proof>*

**lemma** *hfloor\_minus*: "hfloor (- x) = - hceiling x"  
*<proof>*

**lemma** *hceiling\_minus*: "hceiling (- x) = - hfloor x"  
*<proof>*

### 7.3 The nonstandard fractional part

#### definition

*hpart* :: "'a::floor\_ceiling star => 'a star" where  
[transfer\_unfold]: "hpart x = x - of\_hypint (hfloor x)"

#### notation

*hpart* ( $\{\}_-$ )

#### notation (HTML output)

*hpart* ( $\{\}_-$ )

**lemma** *hpart\_zero* [simp]: "hpart(0) = 0"  
*<proof>*

**lemma** *hpart\_ge\_zero* [simp]: " $0 \leq \text{hpart } x$ "  
*<proof>*

**lemma** *hpart\_less\_one* [simp]: "hpart x < 1"  
*<proof>*

**lemma** *hpart\_le\_one* [simp]: "hpart x  $\leq$  1"  
*<proof>*

**lemma** *hypreal\_eq\_hfloor\_hpart\_add*:  
"x = of\_hypint (hfloor x) + hpart x"  
*<proof>*

**lemma** *hfloor\_eq\_self\_diff\_hpart*: "of\_hypint (hfloor x) = x - hpart x"  
*<proof>*

**lemma** *HFinite\_hpart* [simp]: " $\bigwedge x. \text{hpart } (x::\text{hypreal}) \in \text{HFinite}$ "  
 <proof>

**lemma** *hpart\_not\_less\_one* [simp]: " $\neg 1 < \{x\}$ "  
 <proof>

**lemma** *hpart\_mult\_less\_cancel* [simp]: " $(x * \{y\} < x) = (0 < x)$ "  
 <proof>

**lemma** *hfloor\_of\_hypint\_divide* [simp]:  
 " $\bigwedge x y. \text{hfloor } ((\text{of\_hypint } x :: \text{hypreal}) / \text{of\_hypint } y) = (x \text{ div } y)$ "  
 <proof>

**lemma** *hfloor\_of\_hypint\_of\_hypnat\_divide* [simp]:  
 " $\bigwedge x y. \text{hfloor } ((\text{of\_hypint } x :: \text{hypreal}) / \text{of\_hypnat } y) = (x \text{ div } (\text{of\_hypnat } y))$ "  
 <proof>

**lemma** *floor\_real\_of\_nat\_real\_of\_int\_divide* [simp]:  
 " $\text{floor } (\text{real } x / \text{real } y) = \text{int } x \text{ div } y$ "  
 <proof>

**lemma** *hfloor\_of\_hypnat\_of\_hypint\_divide* [simp]:  
 " $\bigwedge x y. \text{hfloor } ((\text{of\_hypnat } x :: \text{hypreal}) / \text{of\_hypint } y) = (\text{of\_hypnat } x) \text{ div } y$ "  
 <proof>

**lemma** *of\_hypnat\_hypnat\_of\_hypint* [simp]:  
 " $0 \leq x \implies \text{of\_hypnat}(\text{hypnat}(\text{hfloor } x)) = \text{of\_hypint } (\text{hfloor } x)$ "  
 <proof>

## 7.4 Miscellaneous properties

**lemma** *HFinite\_hypnat\_hfloor\_Nat*:  
 assumes " $(y::\text{hypreal}) \in \text{HFinite}$ "  
 shows " $\text{hypnat } (\text{hfloor } y) \in \mathbb{N}$ "  
 <proof>

**lemma** *HFinite\_between\_Nats*:  
 assumes " $(x::\text{hypreal}) \in \text{HFinite}$ "  
 and " $x > 0$ "  
 shows " $\exists n \in \mathbb{N}. n \leq x \wedge x < n + 1$ "  
 <proof>

**lemma** *hfloor\_eq\_self\_diff\_hpart2*:  
 " $\bigwedge x. 0 \leq x \implies \text{of\_hypnat } (\text{hypnat } (\text{hfloor } x)) = x - \text{hpart } x$ "  
 <proof>

**lemma** *hypreal\_HInfinite\_hfloor*:

```

"(x::hypreal) ∈ HInfinite ⇒ hypreal_of_hypint (hfloor x) ∈ HInfinite"
⟨proof⟩

lemma hypreal_HInfinite_hfloor_cancel:
  "hypreal_of_hypint (hfloor x) ∈ HInfinite ⇒ (x::hypreal) ∈ HInfinite"
⟨proof⟩

lemma hypreal_HNatInfinite_hfloor:
  "⌊ (x::hypreal) ∈ HInfinite; x > 0 ⌋ ⇒ hypnat (hfloor x) ∈ HNatInfinite"
⟨proof⟩

lemma hypreal_HNatInfinite_hfloor_cancel:
  "hypnat (hfloor x) ∈ HNatInfinite ⇒ (x::hypreal) ∈ HInfinite"
⟨proof⟩

lemma hypreal_HNatInfinite_hfloor_inverse_Infinitesimal:
  "⌊ (e::hypreal) ∈ Infinitesimal; e > 0 ⌋ ⇒ hypnat (hfloor (1 / e))
∈ HNatInfinite"
⟨proof⟩

lemma hypreal_Infinitesimal_HNatInfinite_hfloor_inverse_iff:
  assumes pos: "e > 0"
  shows "(e::hypreal) ∈ Infinitesimal = (hypnat (hfloor (1 / e))
∈ HNatInfinite)" (is "?L = ?R")
⟨proof⟩

lemma hypreal_Infinitesimal_hfloor_inverse_HNatInfinite:
  "hypnat (hfloor (1 / e)) ∈ HNatInfinite ⇒ (e::hypreal) ∈ Infinitesimal"
⟨proof⟩

lemma inverse_hfloor_inverse_approx:
  "e ∈ Infinitesimal ⇒ inverse (hypreal_of_hypint(hfloor(1 / e))) ≈
e"
⟨proof⟩

lemma Infinitesimal_hfloor_divide_mult:
  assumes infmal_x: "x ∈ Infinitesimal"
  and not_zero_x: "x ≠ (0::hypreal)"
  shows "x * of_hypint(hfloor (z/x)) ≈ z"
⟨proof⟩

lemma Infinitesimal_hfloor_inverse_mult_self:
  assumes infmal_x: "x ∈ Infinitesimal"
  and not_zero_x: "x ≠ (0::hypreal)"
  shows "x * of_hypint(hfloor (inverse x)) ≈ 1"
⟨proof⟩

lemma Infinitesimal_hfloor_inverse_mult_self_pos:

```

```

    "[[ e ∈ Infinitesimal; e > (0::hypreal) ]] ⇒ e * of_hypint(hfloor(1/e))
    ≈ 1"
    ⟨proof⟩

```

end

## 8 The hyperfactorial and hyperbinomial coefficient functions

```

theory HyperBinomial
imports FallFactorial HyperSum HyperArchimedean Internal
begin

```

### 8.1 The hyperbinomial coefficients

definition

```

    hchoose :: "nat star ⇒ nat star ⇒ nat star" (infixl "hchoose" 65)

```

where

```

    hyperbinomial_def [transfer_unfold]:    "(hchoose) = *f2* (choose)"

```

```

lemma star_choose: "(star_n (N::nat⇒nat)) hchoose (star_n K) = star_n
(λn. N n choose K n)"
⟨proof⟩

```

```

lemma star_of_choose [simp]: "star_of (n choose k) = (star_of n) hchoose
(star_of k)"
⟨proof⟩

```

```

lemma star_choose_zero_hypnat [simp]: "∧n. (n::hypnat) hchoose 0 = 1"
⟨proof⟩

```

```

lemma zero_star_choose_hypnat [simp]: "∧n k. n < (k::hypnat) ⇒ n hchoose
k = 0"
⟨proof⟩

```

```

lemma star_choose_reduce_hypnat:
    "∧n k. [[0 < (n::hypnat); 0 < k]]
    ⇒ n hchoose k = n - 1 hchoose k + (n - 1 hchoose (k - 1))"
⟨proof⟩

```

```

lemma star_choose_plus_one_hypnat: "∧n k. ((n::hypnat) + 1) hchoose
(k + 1) =
    (n hchoose (k + 1)) + (n hchoose k)"
⟨proof⟩

```

```

lemma star_choose_hSuc_hypnat: "∧n k. (hSuc n) hchoose (hSuc k) =
    (n hchoose (hSuc k)) + (n hchoose k)"
⟨proof⟩

```

**lemma** *star\_choose\_self\_hypnat* [simp]: " $\bigwedge n. ((n::\text{hypnat}) \text{hchoose } n) = 1$ "  
 <proof>

**lemma** *star\_choose\_one\_hypnat* [simp]: " $\bigwedge n. (n::\text{hypnat}) \text{hchoose } 1 = n$ "  
 <proof>

**lemma** *star\_plus\_one\_choose\_self\_hypnat* [simp]: " $\bigwedge n. (n::\text{hypnat}) + 1 \text{hchoose } n = n + 1$ "  
 <proof>

**lemma** *hSuc\_choose\_self\_hypnat* [simp]: " $\bigwedge n. (\text{hSuc } n) \text{hchoose } n = \text{hSuc } n$ "  
 <proof>

**lemma** *choose\_pos\_nat* [rule\_format]: " $\bigwedge k n. k \leq (n::\text{hypnat}) \implies (n \text{hchoose } k) > 0$ "  
 <proof>

## 8.2 The hyperfactorial function

**definition**

*hfact\_def* [transfer\_unfold]: " $\text{hfact} \equiv *f* \text{fact}$ "

**lemma** *star\_choose\_altdef\_hypnat*:  
 " $\bigwedge n k. (k::\text{hypnat}) \leq n \implies n \text{hchoose } k = \text{hfact } n \text{div } (\text{hfact } k * \text{hfact } (n - k))$ "  
 <proof>

**lemma** *star\_choose\_dvd\_hypnat*: " $\bigwedge n k. (k::\text{hypnat}) \leq n \implies \text{hfact } k * \text{hfact } (n - k) \text{dvd } \text{hfact } n$ "  
 <proof>

**lemma** *hfact\_of\_nat*: " $\text{hfact } (\text{hypnat\_of\_nat } n) = \text{of\_nat } (\text{fact } n)$ "  
 <proof>

**lemma** *HFinite\_hfact\_of\_nat* [simp]:  
 " $\text{of\_hypnat } (\text{hfact } (\text{hypnat\_of\_nat } n)) \in \text{HFinite}$ "  
 <proof>

**lemma** *hfact\_nat\_in\_Nats*: " $n \in \mathbb{N} \implies \text{hfact } (n::\text{hypnat}) \in \mathbb{N}$ "  
 <proof>

**lemma** *star\_choose\_le\_hyperpow*:  
 " $\bigwedge r n. r \leq n \implies n \text{hchoose } r \leq n \text{pow } r$ "  
 <proof>

The binomial theorem extended to hyperreal and hypernaturals, characterized via hyperfinite sums

**lemma** *Iset\_interval*:  
 "Iset (star\_n (λn. {0..X n})) = {0..star\_n X}"  
 ⟨proof⟩

**lemma** *n\_star\_interval\_ultra*: "eventually (λn. (\*ns\* {0..star\_n X}) n = {0..X n}) U"  
 ⟨proof⟩

**lemma** *lemma\_starfun\_n\_binomial*:  
 "\*fn\* (λn k. of\_nat (Xb n choose k) \* X n ^ k \* Xa n ^ (Xb n - k)) =  
 (λK. of\_hypnat ((\*f2\* (choose)) (star\_n Xb) K) \* (\*f2\* (^)) (star\_n X) K \*  
 (\*f2\* (^)) (star\_n Xa) (star\_n Xb - K))"  
 ⟨proof⟩

### 8.3 The hyperbinomial theorem

**lemma** *hyperbinomial\_ring*:  
 "(a + b::'a::{comm\_semiring\_1,semiring\_1\_cancel} star) pow N =  
 hypersum (λK. of\_hypnat (N hchoose K) \* a pow K \* b pow  
 (N - K)) {0..N}"  
 ⟨proof⟩

**lemma** *hyperbinomial\_simple*:  
 "(1 + a::'a::{comm\_semiring\_1,semiring\_1\_cancel} star) pow N =  
 hypersum (λi. of\_hypnat (N hchoose i) \* a pow i) {0..N}"  
 ⟨proof⟩

**lemma** *lemma\_InternalFuns\_hyperbinomial*:  
 "hypreal\_of\_hypnat (star\_n X hchoose star\_n Xc) \*  
 star\_n Xa pow star\_n Xc \* star\_n Xb pow (star\_n X - star\_n Xc) =  
 star\_n (λn. real (X n choose Xc n) \* Xa n ^ Xc n \* Xb n ^ (X n - Xc  
 n))"  
 ⟨proof⟩

**lemma** *InternalFuns\_hyperbinomial [simp]*:  
 "(λK. hypreal\_of\_hypnat (N hchoose K) \* a pow K \* b pow (N - K)) ∈  
 InternalFuns"  
 ⟨proof⟩

### 8.4 Nonstandard falling factorial

**definition**  
 hfallfactpow :: "'a::comm\_ring\_1 star ⇒ hypnat ⇒ 'a star" where  
 [transfer\_unfold]: "hfallfactpow a n = (\*f2\* fallfactpow) a n"

**lemma** *hfallfactpow\_0[simp]*: "∧a. hfallfactpow a 0 = 1"  
 ⟨proof⟩

**lemma** `hfallfactpow_1[simp]`: " $\bigwedge a. \text{hfallfactpow } a \ 1 = a$ " *<proof>*  
**lemma** `hfallfactpow_hSuc0[simp]`: " $\bigwedge a. \text{hfallfactpow } a \ (\text{hSuc } 0) = a$ " *<proof>*

**lemma** `hfallfactpow_hSuc`: " $\bigwedge a \ n. \text{hfallfactpow } a \ (\text{hSuc } n) = \text{hfallfactpow } a \ n * (a - \text{of\_hypnat } n)$ "  
*<proof>*

**lemma** `hfallfactpow_hSuc'`: " $\text{hfallfactpow } a \ (n + 1) = \text{hfallfactpow } a \ n * (a - \text{of\_hypnat } n)$ "  
*<proof>*

**lemma** `hfallfactpow_rec`: " $\bigwedge a \ n. \text{hfallfactpow } a \ (\text{hSuc } n) = a * \text{hfallfactpow } (a - 1) \ n$ "  
*<proof>*

**lemma** `hfallfactpow_rec'`: " $\bigwedge a \ n. \text{hfallfactpow } a \ (n + 1) = a * \text{hfallfactpow } (a - 1) \ n$ "  
*<proof>*

**lemma** `hfallfactpow_base_zero[simp]`: " $\bigwedge n. \text{hfallfactpow } 0 \ (\text{hSuc } n) = 0$ "  
*<proof>*

**lemma** `hfallfactpow_base_zero'[simp]`: " $\bigwedge n. \text{hfallfactpow } 0 \ (n + 1) = 0$ "  
*<proof>*

**lemma** `hfallfactpow_fact`: " $\bigwedge n. \text{of\_hypnat}(\text{hfact } n) = \text{hfallfactpow } (\text{of\_hypnat } n) \ n$ "  
*<proof>*

**lemma** `hfallfactpow_of_nat_eq_0_lemma`:  
" $\bigwedge k \ n. k > n \implies \text{hfallfactpow } (\text{of\_hypnat } n :: 'a :: \text{idom } \text{star}) \ k = 0$ "  
*<proof>*

**lemma** `hyperbinomial_hfallfactpow`:  
" $\bigwedge k \ n. \text{of\_hypnat } (n \ \text{hchoose } k) = (\text{hfallfactpow } (\text{of\_hypnat } n) \ k :: 'a :: \{\text{field\_char\_0}\} \ \text{star}) / \text{hfact } k$ "  
*<proof>*

**lemma** `hyperbinomial3`:  
" $(a + b :: 'a :: \{\text{field\_char\_0}\} \ \text{star}) \ \text{pow } n = \text{hypersum } (\lambda k. \text{hfallfactpow } (\text{of\_hypnat } n) \ k / \text{hfact } k * a \ \text{pow } k * b \ \text{pow } (n - k)) \ \{0..n\}$ "  
*<proof>*

**lemma** `hfallfactpow_le_power_self [simp]`:  
" $\bigwedge k. \text{hfallfactpow } (\text{of\_hypnat } k :: 'a :: \{\text{linordered\_idom}\} \ \text{star}) \ k \leq \text{of\_hypnat } k \ \text{pow } k$ "  
*<proof>*

**lemma** *hfallfactpow\_le\_hyperpow*:  
 $\bigwedge k j. \text{of\_hypnat } k \leq j \implies \text{hfallfactpow } (j :: 'a :: \{\text{linordered\_idom}\} \text{ star})$   
 $k \leq j \text{ pow } k$   
*<proof>*

**lemma** *hfallfactpow\_ge\_zero [simp]*:  
 $\bigwedge k n. \text{hfallfactpow } (\text{of\_hypnat } n) k \geq (0 :: 'a :: \{\text{linordered\_idom}\} \text{ star})$   
*<proof>*

**lemma** *fact\_Suc\_ge\_two\_pow*:  $\text{fact } (n + 1) \geq (2 :: \text{nat})^n$   
*<proof>*

**lemma** *hfact\_hSuc\_ge\_two\_pow*:  
 $\bigwedge n. (\text{hfact } (n + 1) :: \text{hypreal}) \geq 2 \text{ pow } n$   
*<proof>*

**lemma** *HInfinite\_diff\_of\_nat\_divide\_approx\_one*:  
*assumes*  $x \in \text{HInfinite}$   
*shows*  $(x - \text{of\_nat } k) / x \approx (1 :: 'a :: \{\text{real\_normed\_field}\} \text{ star})$   
*<proof>*

**lemma** *HInfinite\_hfallfactpow\_divide\_hrealpow*:  
*assumes*  $j \in \text{HInfinite}$   
*shows*  $\text{hfallfactpow } j (\text{of\_nat } k) / (j \wedge k) \approx (1 :: 'a :: \{\text{real\_normed\_field}\} \text{ star})$   
*<proof>*

**lemma** *HInfinite\_hfallfactpow\_divide\_hyperpow\_of\_nat*:  
 $j \in \text{HInfinite}$   
 $\implies \text{hfallfactpow } j (\text{of\_nat } k) / (j \text{ pow } \text{of\_nat } k) \approx (1 :: 'a :: \{\text{real\_normed\_field}\} \text{ star})$   
*<proof>*

**lemma** *HInfinite\_hfallfactpow\_divide\_hyperpow\_Nats*:  
 $\llbracket k \in \mathbb{N}; j \in \text{HInfinite} \rrbracket \implies \text{hfallfactpow } j k / (j \text{ pow } k) \approx (1 :: 'a :: \{\text{real\_normed\_field}\} \text{ star})$   
*<proof>*

This is the one that we want to deal with Euler's claim that  $(j - 1)(j - 2) \dots (j - k) / j^k = 1!$

**lemma** *HNatInfinite\_hfallfactpow\_divide\_hyperpow\_of\_nat*:  
 $j \in \text{HNatInfinite}$   
 $\implies \text{hfallfactpow } (\text{of\_hypnat } j) (\text{of\_nat } k) / (\text{of\_hypnat } j \text{ pow } \text{of\_nat } k)$   
 $\approx (1 :: 'a :: \{\text{real\_normed\_field}\} \text{ star})$   
*<proof>*

**lemma** *hfallfactpow\_base\_starfun2*:  
 $\text{hfallfactpow } a = (*f2* \text{fallfactpow}) a$

*<proof>*

**lemma** *InternalFuns\_hfallfactpow [simp]: "hfallfactpow a ∈ InternalFuns"*  
*<proof>*

**lemma** *hfallfactpow\_starfun2:*  
*"hfallfactpow = \*f2\* fallfactpow"*  
*<proof>*

**lemma** *InternalFuns2\_hfallfactpow [simp]: "hfallfactpow ∈ InternalFuns2"*  
*<proof>*

**lemma** *InternalFuns2\_hfallfactpow\_base [simp]: "(λn. hfallfactpow a) ∈ InternalFuns2"*  
*<proof>*

**lemma** *hfallfactpow: "hfallfactpow (star\_n X) (star\_n Y) = star\_n (λn. fallfactpow (X n) (Y n))"*  
*<proof>*

**lemma** *InternalFuns\_hfallfactpow\_fun:*  
*assumes "f ∈ InternalFuns"*  
*shows "(λn. hfallfactpow a (f n)) ∈ InternalFuns"*  
*<proof>*

**lemma** *InternalFuns\_hfallfactpow\_divide:*  
*"(λk. (hfallfactpow (of\_hypnat j) k / (of\_hypnat j pow k))) :: 'a :: {comm\_ring\_1, division\_ring} star ) ∈ InternalFuns"*  
*<proof>*

**lemma** *hfact:*  
*"hfact (star\_n X) = star\_n (λn. fact (X n))"*  
*<proof>*

**lemma** *InternalFuns\_hfact\_fun [simp]:*  
*"f ∈ InternalFuns ⇒ (λn. hfact (f n)) ∈ InternalFuns"*  
*<proof>*

**lemma** *InternalFuns\_hfact [simp]: "(λn. hfact n) ∈ InternalFuns"*  
*<proof>*

**lemma** *InternalFuns2\_hfact [simp]: "(λm. hfact) ∈ InternalFuns2"*  
*<proof>*

**lemma** *Infinitesimal\_hyperpow\_star\_of\_less\_one:*  
*assumes "hnorm ((star\_of x)::'a::real\_normed\_algebra\_1 star) < 1"*  
*"n ∈ HNatInfinite"*  
*shows "((star\_of x)::'a::real\_normed\_algebra\_1 star) pow n ∈ Infinitesimal"*  
*<proof>*

**lemma** *InternalFuns\_abs\_pow\_fun*:  
 assumes *Int\_f*: "*f* ∈ *InternalFuns*" shows " $(\lambda n. |x \text{ pow } f \ n|) \in \text{InternalFuns}$ "  
 ⟨*proof*⟩

**lemma** *hfact\_gt\_zero [simp]*: " $\bigwedge n. (0 :: 'a :: \text{linordered\_semidom } \text{star}) < \text{hfact } (n :: \text{hypnat})$ "  
 ⟨*proof*⟩

**lemma** *hypnat\_fact\_zero [simp]* : " $\text{hfact } (0 :: \text{nat } \text{star}) = 1$ "  
 ⟨*proof*⟩

## 8.5 Nonstandard version of Pochhammer's symbol i.e. the rising factorial

**definition**

*hpochhammer* :: "'a::comm\_semiring\_1 star ⇒ hypnat ⇒ 'a star" where  
 [*transfer\_unfold*]: "*hpochhammer* *x n* = (\*f2\* *pochhammer*) *x n*"

**lemma** *fact\_fact\_pochhammer\_mult*:  
 " $n \geq k \implies \text{fact } n = (\text{fact } k) * \text{pochhammer } (k + 1) (n - k)$ "  
 ⟨*proof*⟩

**lemma** *fact\_fact\_hpochhammer\_mult*:  
 " $\bigwedge n \ k. n \geq k \implies \text{hfact } n = (\text{hfact } k) * \text{hpochhammer } (k + 1) (n - k)$ "  
 ⟨*proof*⟩

**lemma** *power\_le\_pochhammer*:  
 assumes " $0 \leq x$ "  
 shows " $(x \wedge^n :: 'a :: \text{linordered\_semidom}) \leq \text{pochhammer } x \ n$ "  
 ⟨*proof*⟩

**lemma** *hyperpow\_le\_hpochhammer*:  
 " $\bigwedge x \ n. 0 \leq x \implies (x \text{ pow } n :: 'a :: \text{linordered\_semidom } \text{star}) \leq \text{hpochhammer } x \ n$ "  
 ⟨*proof*⟩

**lemma** *of\_nat\_pochhammer\_of\_nat*:  
 assumes " $0 \leq z$ " shows " $\text{of\_nat } (\text{pochhammer } (\text{nat } z) \ n) = \text{pochhammer } (\text{of\_int } z) \ n$ "  
 ⟨*proof*⟩

**lemma** *of\_hypnat\_hpochhammer\_of\_hypnat*:  
 " $\bigwedge z \ n. 0 \leq z \implies \text{of\_hypnat } (\text{hpochhammer } (\text{hypnat } z) \ n) = \text{hpochhammer } (\text{of\_hypint } z) \ n$ "  
 ⟨*proof*⟩

```

lemma hyperpow_divide_fact_le_lemma':
  assumes "y ∈ HFinite"
  and "(0 :: real star) < y"
  and "hypnat (hfloor y + 1) ≤ n"
  shows "|y pow n| / hypreal_of_hypnat (hfact n)
        ≤ y pow n /
           ((hypreal_of_hypint (hfloor y) + 2) pow
            (n - hypnat (hfloor y + 1))) *
           hypreal_of_hypnat (hfact (hypnat (hfloor y + 1))))"
⟨proof⟩

lemma hfloor_one_minus_epsilon: "hfloor(1 - ε) = 0"
⟨proof⟩

lemma hypreal_hfloor_approx_zero [simp]: "[ (x::hypreal) ≈ 0; x > 0 ]
⇒ hfloor x = 0"
⟨proof⟩

lemma hfloor_approx_zero [simp]: "x ≈ 0 ⇒ hfloor (hnorm x) = 0"
⟨proof⟩

lemma InternalFuns_expf_term' [simp]:
  "(λN. y pow N / hypreal_of_hypnat (hfact N)) ∈ InternalFuns"
⟨proof⟩

lemma InternalFuns_expf_term [simp]:
  "(λN. y pow N / (hfact N :: real star)) ∈ InternalFuns"
⟨proof⟩

lemma HyperSummable_hyperpow_divide_fact_Infinitesimal:
  assumes HFinite_y: "(y::hypreal) ∈ HFinite"
  and y_gt_0: "y > 0"
  and y_approx_0: "y ≈ 0"
  shows "HyperSummable (λN. y pow N/of_hypnat(hfact N))"
⟨proof⟩

lemma HyperSummable_hyperpow_divide_fact_not_Infinitesimal:
  assumes HFinite_y: "(y::hypreal) ∈ HFinite"
  and y_gt_0: "y > 0"
  and y_not_approx_0: "¬ y ≈ 0"
  shows "HyperSummable (λN. y pow N/of_hypnat(hfact N))"
⟨proof⟩

lemma HyperSummable_hyperpow_divide_fact_pos:
  "[ (y::hypreal) ∈ HFinite; y > 0 ] ⇒ HyperSummable (λN. y pow N/of_hypnat(hfact
N))"
⟨proof⟩

```

```

lemma HyperSummable_hyperpow_divide_fact_neg:
  assumes "(y::hypreal) ∈ HFinite"
  and "y < 0"
  shows "HyperSummable (λN. y pow N/of_hypnat(hfact N))"
⟨proof⟩

lemma InternalFuns_star_choose: "(hchoose) k ∈ InternalFuns"
⟨proof⟩

lemma InternalFuns_star_choose_from [simp]: "(λn. n hchoose k) ∈ InternalFuns"
⟨proof⟩

lemma hypnat_fact_not_zero [simp]: "hfact n ≠ (0::hypnat)"
⟨proof⟩

lemma hfact_nonzero [simp]:
  "∧k. hfact k ≠ (0::'a::{semiring_char_0,semiring_no_zero_divisors}
star)"
⟨proof⟩

lemma HyperSummable_geometric':
  assumes "hnorm (x::'a::{real_normed_field} star) < 1"
  "¬hnorm x ≈ 1"
  shows "HyperSummable (λN. x pow (hSuc N))"
⟨proof⟩

lemma HyperSummable_hyperpow_divide_fact_zero:
  "HyperSummable (λn. 0 pow n / hypreal_of_hypnat (hfact n))"
⟨proof⟩

lemma HyperSummable_exp':
  "(y::hypreal) ∈ HFinite ⇒ HyperSummable (λN. y pow N/of_hypnat(hfact
N))"
⟨proof⟩

lemma of_hypnat_hfact:
  "∧n. of_hypnat(hfact n) = hfact n"
⟨proof⟩

lemma HyperSummable_exp:
  "(y::hypreal) ∈ HFinite ⇒ HyperSummable (λN. y pow N/hfact N)"
⟨proof⟩

lemma lemma_n_starfun_extract_binomial_hfallfactpow:
  "eventually (λn. ((*nf* (λk. of_hypnat ((*f2* (choose)) (star_n Xb)
k) *

```

```

- k) *
(*f2* fallfactpow) (star_n X) (star_n Xb
(*f2* fallfactpow) (star_n Xa) k)) n) =
(λk. of_nat(Xb n choose k) *
fallfactpow (X n) (Xb n - k) *
fallfactpow (Xa n) k)) U"

```

*<proof>*

**lemma hyperbinomial\_fallfactpow\_ring:**

```

"hfallfactpow (a + b :: 'a::{comm_ring_1} star) n =
(hypersum (λk. of_hypnat(n hchoose k) * hfallfactpow a (n - k) * hfallfactpow
b k)) {0..n}"

```

*<proof>*

**lemma hyperbinomial\_fallfactpow\_ring2:**

```

"∧a b n. hfallfactpow (a + b :: 'a::{comm_ring_1} star) n =
(hypersum (λk. of_hypnat(n hchoose k) * hfallfactpow a k * hfallfactpow
b (n - k))) {0..n}"

```

*<proof>*

**lemma hyperbinomial\_hfact:**

```

"∧n k. k ≤ n ⇒ of_hypnat (n hchoose k) = (hfact n :: 'a:: field_char_0
star) / (hfact k * hfact (n - k))"

```

*<proof>*

**lemma hypersum\_cong:**

```

assumes ifun_g: "g ∈ InternalFuns"
and ifun_h: "h ∈ InternalFuns"
and iset_A: "A ∈ InternalSets"
and "A = B"
and g_h: "∧x. x ∈ B ⇒ g x = h x"
shows "hypersum g A = hypersum h B"

```

*<proof>*

**lemma hyperbinomial\_vandermonde:**

```

"(r + s) pow k/hfact k =
hypersum (λi. (r::'a :: field_char_0 star) pow i / hfact i * s pow
(k - i) / hfact (k - i)) {0..k}"

```

*<proof>*

**lemma hyperbinomial\_hfallfactpow\_vandermonde:**

```

"hfallfactpow (r + s) k/hfact k =
hypersum (λi. hfallfactpow (r::'a :: field_char_0 star) i / hfact
i * hfallfactpow s (k - i) / hfact (k - i)) {0..k}"

```

*<proof>*

```

lemma hyperbinomial_fallfactpow_ring3:
  "hfallfactpow (a + b :: 'a::{field_char_0} star) n =
    (hypersum ( $\lambda k. \text{hfallfactpow (of_hypnat } n) k / \text{hfact } k * \text{hfallfactpow } a (n - k) * \text{hfallfactpow } b k$ )) {0..n}"
  <proof>

```

Setting up a few lemmas to prove properties about infinitesimal exponents

```

lemma binomial_expand_first_two_terms:
  "0 ≤ u ⇒ ∃ x ≥ 0. (1 + (u :: 'a::{linordered_idom,power}))n = 1 + of_nat
  n * u + x"
  <proof>

```

```

lemma hyperbinomial_expand_first_two_terms:
  " $\bigwedge u n. 0 \leq u \Rightarrow \exists x \geq 0. (1 + (u :: 'a::{linordered_idom,power} \text{star}))^n = 1 + \text{of\_hypnat } n * u + x$ "
  <proof>

```

```

lemma hyperbinomial_expand_first_two_terms':
  " $\bigwedge u n. 0 \leq u \Rightarrow \exists x \geq 0. ((u :: 'a::{linordered_idom,power} \text{star}) + 1)^n = 1 + \text{of\_hypnat } n * u + x$ "
  <proof>

```

end

## 9 Hyperreal powers

**theory** HyperrealPower

**imports** "Real\_Power.Log" "HOL-Nonstandard\_Analysis.NatStar" HyperBinomial  
begin

**definition**

hpow :: "[hypreal,hypreal] ⇒ hypreal" (infixr "hpow" 80) where  
[transfer\_unfold]: "x hpow a = starfun2 (pow<sub>R</sub>) x a"

```

lemma "(hpow) ∈ InternalFuns2"
  <proof>

```

```

lemma hpow: "(star_n X) hpow (star_n Y) = star_n ( $\lambda n. (X n) \text{pow}_R (Y n)$ )"
  <proof>

```

### 9.1 Tranferred properties

```

lemma hpow_one_eq_one [simp]: " $\bigwedge a. 1 \text{ hpow } a = 1$ "
  <proof>

```

```

lemma hpow_zero_eq_one [simp]: " $\bigwedge a. a > 0 \Rightarrow a \text{ hpow } 0 = 1$ "
  <proof>

```

**lemma** *hpow\_minus\_one*: " $\bigwedge a. 0 < a \implies a \text{ hpow } -1 = \text{inverse } a$ "  
 <proof>

**lemma** *hpow\_one [simp]*: " $\bigwedge a. a > 0 \implies a \text{ hpow } 1 = a$ "  
 <proof>

**lemma** *hpow\_gt\_zero*: " $\bigwedge a x. a > 0 \implies a \text{ hpow } x > 0$ "  
 <proof>

**lemma** *hpow\_not\_zero*: " $\bigwedge a x. a > 0 \implies a \text{ hpow } x \neq 0$ "  
 <proof>

**lemma** *hpow\_minus*: " $\bigwedge a x. a > 0 \implies a \text{ hpow } (-x) = \text{inverse } (a \text{ hpow } x)$ "  
 <proof>

**lemma** *hpow\_inverse*:  
 " $\bigwedge a x. a > 0 \implies \text{inverse } (a \text{ hpow } x) = (\text{inverse } a) \text{ hpow } x$ "  
 <proof>

**lemma** *hpow\_mult\_base*:  
 " $\bigwedge a b x. [ 0 < a; 0 < b ] \implies (a * b) \text{ hpow } x = (a \text{ hpow } x) * (b \text{ hpow } x)$ "  
 <proof>

**lemma** *hpow\_mult*:  
 " $\bigwedge a x y. 0 < a \implies (a \text{ hpow } x) \text{ hpow } y = a \text{ hpow } (x * y)$ "  
 <proof>

**lemma** *hpow\_divide*:  
 " $\bigwedge a b x. [ 0 < a; 0 < b ] \implies (a/b) \text{ hpow } x = (a \text{ hpow } x) / (b \text{ hpow } x)$ "  
 <proof>

**lemma** *hpow\_divide2*: " $\bigwedge a x y. a > 0 \implies a \text{ hpow } (x - y) = (a \text{ hpow } x) / (a \text{ hpow } y)$ "  
 <proof>

**lemma** *hpow\_ge\_one*: " $\bigwedge a x. a \geq 1 \implies x \geq 0 \implies a \text{ hpow } x \geq 1$ "  
 <proof>

**lemma** *hpow\_ge\_one2*:  
 " $\bigwedge a x. [ 0 < a; a < 1; x \leq 0 ] \implies a \text{ hpow } x \geq 1$ "  
 <proof>

**lemma** *hpow\_le\_mono*:  
 " $\bigwedge r s a. [ r \leq s; a \geq 1 ] \implies a \text{ hpow } r \leq a \text{ hpow } s$ "  
 <proof>

**lemma** *hpow\_le\_anti\_mono*:  
 " $\bigwedge r s a. [ r \leq s; 0 < a; a < 1 ] \implies a \text{ hpow } r \geq a \text{ hpow } s$ "

*<proof>*

**lemma** *hpow\_less\_cancel*:

" $\bigwedge r s a. \llbracket a \text{ hpow } r < a \text{ hpow } s; a \geq 1 \rrbracket \implies r < s$ "  
*<proof>*

**lemma** *hpow\_less\_anti\_mono*:

" $\bigwedge r s a. \llbracket r < s; 0 < a; a < 1 \rrbracket \implies a \text{ hpow } r > a \text{ hpow } s$ "  
*<proof>*

**lemma** *hpow\_less\_mono*:

" $\bigwedge r s a. \llbracket r < s; a > 1 \rrbracket \implies a \text{ hpow } r < a \text{ hpow } s$ "  
*<proof>*

**lemma** *hpow\_le\_cancel*:

" $\bigwedge r s a. \llbracket a \text{ hpow } r \leq a \text{ hpow } s; a > 1 \rrbracket \implies r \leq s$ "  
*<proof>*

**lemma** *hpow\_less\_cancel\_iff [simp]*:

" $\bigwedge r s a. 1 < a \implies (a \text{ hpow } r < a \text{ hpow } s) = (r < s)$ "  
*<proof>*

**lemma** *hpow\_le\_cancel\_iff [simp]*:

" $\bigwedge r s a. 1 < a \implies (a \text{ hpow } r \leq a \text{ hpow } s) = (r \leq s)$ "  
*<proof>*

**lemma** *hpow\_inject\_exp1 [simp]*:

" $\bigwedge r s a. 1 < a \implies (a \text{ hpow } r = a \text{ hpow } s) = (s = r)$ "  
*<proof>*

**lemma** *hpow\_inject\_exp2 [simp]*:

" $\bigwedge r s a. 0 < a \implies a < 1 \implies (a \text{ hpow } r = a \text{ hpow } s) = (s = r)$ "  
*<proof>*

**lemma** *hpow\_inject [simp]*:

" $\bigwedge r s a. 0 < a \implies a \neq 1 \implies (a \text{ hpow } r = a \text{ hpow } s) = (s = r)$ "  
*<proof>*

**lemma** *hpow\_gt\_one*: " $\bigwedge a x. a > 1 \implies x > 0 \implies a \text{ hpow } x > 1$ "

*<proof>*

**lemma** *hpow\_less\_mono\_base*:

" $\bigwedge a b r. \llbracket r > 0; 0 < a; a < b \rrbracket \implies a \text{ hpow } r < b \text{ hpow } r$ "  
*<proof>*

**lemma** *hpow\_less\_antimono\_base*:

" $\bigwedge a b r. \llbracket r < 0; 0 < a; a < b \rrbracket \implies a \text{ hpow } r > b \text{ hpow } r$ "  
*<proof>*

**lemma hpow\_hyperpow\_eq:**  

$$\bigwedge a n. a > 0 \implies a \text{ hpow } (\text{of\_hypnat } n) = a \text{ pow } n"$$
 <proof>

**lemma hpow\_hyperpow\_eq2:**  

$$\bigwedge a n. 0 \leq a \implies 0 < n \implies a \text{ pow } n = (\text{if } (a = 0) \text{ then } 0 \text{ else } a \text{ hpow } (\text{of\_hypnat } n))"$$
 <proof>

**lemma hpow\_hypint:**  

$$\bigwedge x i. x > 0 \implies x \text{ hpow } (\text{of\_hypint } i) = (\text{if } i \geq 0 \text{ then } x \text{ pow hypnat } i \text{ else } 1 / x \text{ pow hypnat } (-i))"$$
 <proof>

**lemma hpow\_power\_eq:**  

$$\bigwedge a. a > 0 \implies a \text{ hpow hypreal\_of\_nat } n = a ^ n"$$
 <proof>

**lemma hpow\_inverse\_of\_hypnat\_gt\_one:**  

$$\bigwedge a N. \llbracket a > 1; N \neq 0 \rrbracket \implies a \text{ hpow } (\text{inverse}(\text{of\_hypnat } N)) > 1"$$
 <proof>

**lemma hpow\_inverse\_of\_nat\_gt\_one:**  

$$\bigwedge a. \llbracket a > 1; N \neq 0 \rrbracket \implies a \text{ hpow } (\text{inverse}(\text{of\_nat } N)) > 1"$$
 <proof>

**lemma Infinitesimal\_hyperpow\_less\_one\_lemma:**  

$$\bigwedge x r. \llbracket \text{hnorm } (x :: 'a :: \text{real\_normed\_algebra}_1 \text{ star}) < 1; 0 < r \rrbracket \implies \exists n_0. \forall n \geq n_0. \text{hnorm } (x \text{ pow } n) < r"$$
 <proof>

**lemma hpow\_add:** 
$$\bigwedge a x y. 0 < a \implies a \text{ hpow } (x + y) = a \text{ hpow } x * a \text{ hpow } y"$$
 <proof>

**lemma hpow\_eq\_square:** 
$$\bigwedge x. 0 < x \implies x \text{ hpow } 2 = x * x"$$
 <proof>

**lemma hpow\_minus\_eq\_hpow:**  

$$\llbracket 0 < a; 0 < b; a \text{ hpow } -x = b \text{ hpow } y \rrbracket \implies a \text{ hpow } x = b \text{ hpow } -y"$$
 <proof>

**lemma hpow\_half\_divide\_eq:**  

$$0 < y \implies y \text{ hpow } (1/2) / y = \text{inverse}(y \text{ hpow } (1/2))"$$
 <proof>

**lemma less\_hpow\_half\_lemma:**  

$$\llbracket 0 < x; 0 < y; x < y \text{ hpow } (1/2) \rrbracket \implies x/y < \text{inverse}(y \text{ hpow } (1/2))"$$

*<proof>*

**lemma** *le\_hpow\_half\_lemma*:

" $\llbracket 0 < x; 0 < y; x \leq y \text{ hpow } (1/2) \rrbracket \implies x/y \leq \text{inverse}(y \text{ hpow } (1/2))$ "  
*<proof>*

## 9.2 Relating pow and hpow

**lemma** *hpow\_hyperpow\_eq\_hpow*: " $0 < a \implies (a \text{ hpow } x) \text{ pow } n = a \text{ hpow } (x * \text{of\_hypnat } n)$ "

*<proof>*

**lemma** *hpow\_divide\_pow\_cancel*:

assumes " $a > 1$ " " $j \neq 0$ "

shows " $(a \text{ hpow } (z/\text{of\_hypnat } j)) \text{ pow } j = a \text{ hpow } z$ "

*<proof>*

**lemma** *hpow\_inverse\_of\_hypnat\_cancel*:

" $\llbracket 0 < a; 0 < N \rrbracket \implies a = (a \text{ hpow } (\text{inverse}(\text{of\_hypnat } N))) \text{ hpow } (\text{of\_hypnat } N)$ "

*<proof>*

**lemma** *hpow\_inverse\_of\_hypnat\_pow\_cancel*:

" $\llbracket 0 < a; 0 < N \rrbracket \implies (a \text{ hpow } (\text{inverse}(\text{of\_hypnat } N))) \text{ pow } N = a$ "

*<proof>*

**lemma** *hpow\_inverse\_of\_hypnat\_pow\_cancel2*:

" $\llbracket a > 1; N \in \text{HNatInfinite} \rrbracket \implies a = (a \text{ hpow } (\text{inverse}(\text{of\_hypnat } N))) \text{ pow } N$ "

*<proof>*

**lemma** *HNatInfinite\_hpow\_inverse\_gt\_one*:

" $\llbracket a > 1; N \in \text{HNatInfinite} \rrbracket \implies a \text{ hpow } (\text{inverse}(\text{of\_hypnat } N)) > 1$ "

*<proof>*

**lemma** *HNatInfinite\_hpow\_eq\_Ex*:

assumes " $a > 1$ " and " $N \in \text{HNatInfinite}$ "

shows " $\exists u > 0. a \text{ hpow } (\text{inverse}(\text{of\_hypnat } N)) = 1 + u$ "

*<proof>*

## 9.3 Some nonstandard theorems

**lemma** *hpow\_inverse\_hypreal\_of\_nat\_approx\_one\_cancel*:

assumes " $a > 0$ "

and " $m > 0$ "

and " $a \text{ hpow } \text{inverse}(\text{hypreal\_of\_nat } m) \approx 1$ "

shows " $a \approx 1$ "

*<proof>*

**lemma** *HNatInfinite\_hpow\_inverse\_approx\_one\_cancel*:

"[ a > 0; ¬(a ≈ 1); (a hpow (inverse(of\_hypnat N))) ≈ 1; N ≠ 0 ] ⇒  
 N ∈ HNatInfinite"  
 ⟨proof⟩

lemma HFinite\_hpow1:  
 assumes "a ∈ HFinite"  
 and "x ∈ HFinite"  
 and "1 ≤ a"  
 shows "a hpow x ∈ HFinite"  
 ⟨proof⟩

lemma HFinite\_hpow2:  
 assumes "a ∈ HFinite - Infinitesimal"  
 and "x ∈ HFinite"  
 and "0 < a"  
 and "a < 1"  
 shows "a hpow x ∈ HFinite"  
 ⟨proof⟩

lemma HFinite\_hpow3:  
 "[ a ∈ HFinite - Infinitesimal; x ∈ HFinite; 0 < a ] ⇒ a hpow x ∈  
 HFinite"  
 ⟨proof⟩

lemma HFinite\_hpow\_not\_Infinitesimal:  
 assumes "a ∈ HFinite"  
 and "x ∈ HFinite"  
 and "1 ≤ a"  
 shows "a hpow x ∉ Infinitesimal"  
 ⟨proof⟩

lemma HFinite\_hpow\_HFinite\_Diff\_Infinitesimal:  
 "[ a ∈ HFinite; x ∈ HFinite; a ≥ 1 ] ⇒ a hpow x ∈ HFinite - Infinitesimal"  
 ⟨proof⟩

lemma approx\_hpow\_minus\_hpow:  
 "[ 0 < a; 0 < b; a hpow -x ≈ b hpow y; a hpow x ∈ HFinite - Infinitesimal  
 ] ⇒ a hpow x ≈ b hpow - y"  
 ⟨proof⟩

lemma HNatInfinite\_hpow\_inverse\_approx\_one:  
 assumes "a ∈ HFinite"  
 and "a > 1"  
 and "N ∈ HNatInfinite"  
 shows "(a hpow (inverse(of\_hypnat N))) ≈ 1"  
 ⟨proof⟩

lemma HNatInfinite\_hpow\_inverse\_diff\_one\_Infinitesimal:

"[[ a ∈ HFinite; a > 1; N ∈ HNatInfinite ]] ⇒ (a hpow (inverse(of\_hypnat N))) - 1 ∈ Infinitesimal"  
 ⟨proof⟩

**lemma** HNatInfinite\_hpov\_inverse\_approx\_one2:  
 "[[ a ∈ HFinite; a > 1; N ∈ HNatInfinite ]] ⇒ (a hpow (1/of\_hypnat N))  
 ≈ 1"  
 ⟨proof⟩

One of the results that we want on our way to derive the exponential series  
 a la Euler

**lemma** Infinitesimal\_pos\_hpov\_approx\_one:  
 assumes HFinite\_a: "a ∈ HFinite"  
 and a\_gt1: "a > 1"  
 and e\_gt0: "e > 0"  
 and Infinitesimal\_e: "e ∈ Infinitesimal"  
 shows "a hpow e ≈ 1"  
 ⟨proof⟩

**lemma** Infinitesimal\_neg\_hpov\_approx\_one:  
 "[[ a ∈ HFinite; a > 1; e > 0; e ∈ Infinitesimal ]] ⇒ a hpow -e ≈ 1"  
 ⟨proof⟩

And the next result, without e>0 restriction!

**lemma** Infinitesimal\_hpov\_approx\_one:  
 assumes "a ∈ HFinite"  
 and "a > 1"  
 and " e ∈ Infinitesimal"  
 shows "a hpow e ≈ 1"  
 ⟨proof⟩

**lemma** hpow\_HFinite\_approx\_1\_lemma1:  
 assumes x\_finite: "x ∈ HFinite"  
 and a\_infclose\_1: "a ≈ 1"  
 and a\_ge\_1: "a ≥ 1" and x\_gt\_0: "x > 0"  
 shows "a hpow x ≈ 1"  
 ⟨proof⟩

**lemma** approx\_hpov:  
 assumes "y ∈ HFinite" and "a ∈ HFinite"  
 and "a > 1" and "x ≈ y"  
 shows "a hpow x ≈ a hpow y"  
 ⟨proof⟩

end

## 10 The hyper logarithm

theory HyperLog

```

imports "Real_Power.Log" "HOL-Nonstandard_Analysis.NatStar" HyperrealPower
begin

```

**definition**

```

  hLog :: "[hypreal,hypreal] => hypreal" where
    [transfer_unfold]: "hLog a x = starfun2 Log a x"

```

**lemma** hLog:

```

  "hLog (star_n X) (star_n Y) = star_n ( $\lambda$ n. Log (X n) (Y n))"
<proof>

```

**lemma** hpow\_hLog\_cancel [simp]:

```

  " $\wedge$ a x. [ $0 < a$ ;  $a \neq 1$ ;  $x > 0$ ]  $\implies$  a hpow (hLog a x) = x"
<proof>

```

**lemma** hLog\_hpow\_cancel [simp]:

```

  " $\wedge$ a y. [ $0 < a$ ;  $a \neq 1$ ]  $\implies$  hLog a (a hpow y) = y"
<proof>

```

**lemma** hLog\_mult:

```

  " $\wedge$ a x y. [ $0 < a$ ;  $a \neq 1$ ;  $0 < x$ ;  $0 < y$ ]  $\implies$  hLog a (x * y) = hLog a x + hLog a y"
<proof>

```

**lemma** hLog\_one [simp]:

```

  " $\wedge$ a. [ $0 < a$ ;  $a \neq 1$ ]  $\implies$  hLog a 1 = 0"
<proof>

```

**lemma** hLog\_eq\_one [simp]:

```

  " $\wedge$ a. [ $0 < a$ ;  $a \neq 1$ ]  $\implies$  hLog a a = 1"
<proof>

```

**lemma** hLog\_inverse:

```

  " $\wedge$ a x. [ $0 < a$ ;  $a \neq 1$ ;  $x > 0$ ]  $\implies$  hLog a (inverse x) = - hLog a x"
<proof>

```

**lemma** hLog\_divide:

```

  " $\wedge$ a x y. [ $0 < a$ ;  $a \neq 1$ ;  $0 < x$ ;  $0 < y$ ]  $\implies$  hLog a (x/y) = hLog a x - hLog a y"
<proof>

```

**lemma** hLog\_less\_cancel\_iff [simp]:

```

  " $\wedge$ a x y. [ $1 < a$ ;  $0 < x$ ;  $0 < y$ ]  $\implies$  (hLog a x < hLog a y) = (x < y)"
<proof>

```

**lemma** hLog\_inj: assumes "1 < b" shows "inj\_on (hLog b) {0 <..}"

```

<proof>

```

```

lemma hLog_le_cancel_iff [simp]:
  " $\wedge a x y. [1 < a; 0 < x; 0 < y] \implies (hLog a x \leq hLog a y) = (x \leq y)$ "
  <proof>

lemma zero_less_hLog_cancel_iff [simp]:
  " $\wedge a x. 1 < a \implies 0 < x \implies 0 < hLog a x \longleftrightarrow 1 < x$ "
  <proof>

lemma zero_le_hLog_cancel_iff [simp]: "1 < a  $\implies 0 < x \implies 0 \leq hLog a x \longleftrightarrow 1 \leq x$ "
  <proof>

lemma hLog_less_zero_cancel_iff [simp]: "1 < a  $\implies 0 < x \implies hLog a x < 0 \longleftrightarrow x < 1$ "
  <proof>

lemma hLog_le_zero_cancel_iff [simp]: "1 < a  $\implies 0 < x \implies hLog a x \leq 0 \longleftrightarrow x \leq 1$ "
  <proof>

lemma one_less_hLog_cancel_iff [simp]: "1 < a  $\implies 0 < x \implies 1 < hLog a x \longleftrightarrow a < x$ "
  <proof>

lemma one_le_hLog_cancel_iff [simp]: "1 < a  $\implies 0 < x \implies 1 \leq hLog a x \longleftrightarrow a \leq x$ "
  <proof>

lemma hLog_less_one_cancel_iff [simp]: "1 < a  $\implies 0 < x \implies hLog a x < 1 \longleftrightarrow x < a$ "
  <proof>

lemma hLog_le_one_cancel_iff [simp]: "1 < a  $\implies 0 < x \implies hLog a x \leq 1 \longleftrightarrow x \leq a$ "
  <proof>

lemma hLog_hpow: " $\wedge b x y. [0 < x; 1 < b; b \neq 1] \implies hLog b (x hpow y) = y * hLog b x$ "
  <proof>

lemma hLog_nat_power:
  " $\wedge b x n. [0 < x; 1 < b; b \neq 1] \implies hLog b (x pow n) = of_hypnat n * hLog b x$ "
  <proof>

end

```

## 11 Deriving the exponential series Euler-style

```
theory EulerExponential
imports HyperBinomial HyperLog
begin
```

### 11.1 Euler's witness: Introductio, paragraph 114

We show explicitly that Euler's witness (that he denotes by  $k$ ) is a finite quantity. The existence is merely stated by Euler and finiteness is only argued using an example, with Euler merely stating that:

"We see that  $k$  is a finite number which depends on the value of the base  $a$ ".

```
lemma Introductio_114_HFinite_witness:
  assumes infinitesimal_exponent: "e ∈ Infinitesimal"
  and exponent_gt_zero: "e > 0"
  and base_finite: "a ∈ HFinite"
  and base_greater_1: "a > 1"
  shows "(a hpow e - 1)/e ∈ HFinite"
⟨proof⟩
```

Proposition 114, with some finite  $k$ .

```
lemma Introductio_114_pos:
  assumes infinitesimal_exponent: "e ∈ Infinitesimal"
  and exponent_gt_zero: "e > 0"
  and base_finite: "a ∈ HFinite"
  and base_greater_1: "a > 1"
  shows "∃k∈HFinite. k > 0 ∧ a hpow e = 1 + k * e"
⟨proof⟩
```

This case, not discussed explicitly by Euler, is also needed to eventually deal with finite, negative exponents.

```
lemma Introductio_114_neg:
  assumes infinitesimal_exponent: "e ∈ Infinitesimal"
  and exponent_less_zero: "e < 0"
  and base_finite: "a ∈ HFinite"
  and base_greater_1: "a > 1"
  shows "∃k∈HFinite. k > 0 ∧ a hpow e = 1 + k * e"
⟨proof⟩
```

```
lemma Introductio_114_HFinite_witness2:
  assumes infinitesimal_exponent: "e ∈ Infinitesimal"
  and exponent_less_zero: "e < 0"
  and base_finite: "a ∈ HFinite"
  and base_greater_1: "a > 1"
  shows "(a hpow e - 1)/e ∈ HFinite"
⟨proof⟩
```

```
lemma Introductio_114_HFinite_witness_full:
```

```

    assumes "x ∈ Infinitesimal"
    and "a ∈ HFinite"
    and "a > 1"
    shows "(a hpow x - 1)/x ∈ HFinite"
  ⟨proof⟩

```

## 11.2 Defining Euler's finite witness

In fact, let us introduce a definition that will enable us to characterize Euler's  $k$ , (called the scaling factor by McKinzie and Tuckey but also left undefined by them).

```

definition eulerK :: "hypreal ⇒ hypreal ⇒ hypreal" where
  "eulerK a x ≡ (a hpow x - 1)/ x"

```

```

lemma eulerK_neg_exponent:
  assumes "a > 0" and "e ≠ 0"
  shows "eulerK a (-e) = eulerK a e / (a hpow e)"
  ⟨proof⟩

```

Equivalent multiplicative form.

```

corollary eulerK_neg_exponent_mult:
  assumes "a > 0" and "e ≠ 0"
  shows "eulerK a (-e) * (a hpow e) = eulerK a e"
  ⟨proof⟩

```

```

lemma eulerK_neg_approx:
  assumes "a ∈ HFinite" and "a > 1" and "x ∈ Infinitesimal"
  shows "eulerK a (-x) ≈ eulerK a x"
  ⟨proof⟩

```

```

corollary eulerK_epsilon_neg_approx:
  assumes "a ∈ HFinite" and "a > 1"
  shows "eulerK a (-ε) ≈ eulerK a ε"
  ⟨proof⟩

```

```

lemma Introductio_114_eulerK_witness:
  assumes "x ∈ Infinitesimal"
  and "a ∈ HFinite"
  and "a > 1"
  shows "eulerK a x ∈ HFinite"
  ⟨proof⟩

```

```

lemma HFinite_eulerK_inverse_wnh:
  assumes "a ∈ HFinite"
  and "a > 1"
  shows "eulerK a (1/of_hypnat whn) ∈ HFinite"
  ⟨proof⟩

```

```

lemma Introductio_114_eulerK_epsilon_witness:
  assumes "a ∈ HFinite"
  and     "a > 1"
  shows "eulerK a ε ∈ HFinite"
  ⟨proof⟩

lemma eulerK_neg_e_HFinite:
  assumes "e ∈ Infinitesimal" "e > 0" "a ∈ HFinite" "a > 1"
  shows "eulerK a (-e) ∈ HFinite"
  ⟨proof⟩

lemma Introductio_114_neg_eulerK_epsilon_witness:
  assumes "a ∈ HFinite" "a > 1"
  shows "eulerK a (-ε) ∈ HFinite"
  ⟨proof⟩

lemma eulerK_neg_e_gt_zero:
  assumes "e ∈ Infinitesimal" "e > 0" "a ∈ HFinite" "a > 1"
  shows "eulerK a (-e) > 0"
  ⟨proof⟩

lemma eulerK_neg_epsilon_gt_zero:
  assumes "a ∈ HFinite" "a > 1"
  shows "eulerK a (-ε) > 0"
  ⟨proof⟩

lemma Introductio_114_eulerK:
  shows "a > 0 ⇒ a hpow e = 1 + eulerK a e * e"
  ⟨proof⟩

lemma Introductio_114_eulerK':
  shows "e ≠ 0 ⇒ a hpow e = 1 + eulerK a e * e"
  ⟨proof⟩

lemma Introductio_114_eulerK_epsilon:
  shows "a hpow ε = 1 + eulerK a ε * ε"
  ⟨proof⟩

lemma Introductio_114_epsilon:
  "a > 1 ⇒ ε = hLog a (1 + eulerK a ε * ε)"
  ⟨proof⟩

lemma eulerK_product_neg_e:
  assumes "a > 0" and "e ≠ 0"
  shows "eulerK a (-e) * (-e) = eulerK a e * (-e) / (a hpow e)"
  ⟨proof⟩

lemma eulerK_pow_neg_e:
  assumes "a > 0" and "e ≠ 0"

```

shows "(eulerK a (-e) \* (-e)) pow of\_nat n = (eulerK a e \* (-e)) pow of\_nat n / (a hpow e) ^ n"  
 <proof>

### 11.3 Euler's Introductio, Paragraph 115: first expansion

This expansion is given by Euler who implicitly uses the Binomial theorem.

lemma *Introductio\_115\_expansion\_hchoose*:

"a > 0  $\implies$  (a hpow x) pow j = hypersum ( $\lambda i.$  of\_hypnat (j hchoose i) \* (eulerK a x \* x) pow i) {0..j}"  
 <proof>

lemma *Introductio\_115\_expansion\_hffp*:

"a > 0  $\implies$   
 (a hpow x) pow j =  
 hypersum ( $\lambda i.$  (hfallfactpow (of\_hypnat j) i) / hfact i \* (eulerK a x \* x) pow i) {0..j}"  
 <proof>

lemma *Introductio\_115\_expansion\_hchoose'*:

"e  $\neq$  0  $\implies$  (a hpow e) pow j = hypersum ( $\lambda i.$  of\_hypnat (j hchoose i) \* (eulerK a e \* e) pow i) {0..j}"  
 <proof>

lemma *Introductio\_115\_expansion\_hchoose\_epsilon*:

"(a hpow  $\varepsilon$ ) pow j = hypersum ( $\lambda i.$  of\_hypnat (j hchoose i) \* (eulerK a  $\varepsilon$  \*  $\varepsilon$ ) pow i) {0..j}"  
 <proof>

lemma *HFinite\_Infinitesimal\_hpow\_cancel\_approx*:

assumes *infinitesimal\_exponent*: "e  $\in$  Infinitesimal"  
 and *exponent\_not\_zero*: "e  $\neq$  0"  
 and *base\_finite*: "a  $\in$  HFinite"  
 and *base\_greater\_1*: "a > 1"  
 and *exponentz\_HFinite*: "z  $\in$  HFinite"  
 shows "(a hpow e) hpow of\_hypint(hfloor(z/e))  $\approx$  a hpow z"  
 <proof>

lemma *hpow\_hypersum\_pos\_hchoose*:

assumes "e  $\in$  Infinitesimal" and "e > 0"  
 and "a  $\in$  HFinite" and "a > 1"  
 and "z  $\in$  HFinite" and "z > 0"  
 defines "N  $\equiv$  hypnat (hfloor (z / e))"  
 shows "a hpow z  $\approx$  ( $\sum_h i \in \{0..N\}.$  hypreal\_of\_hypnat (N hchoose i) \* (eulerK a e \* e) pow i)"  
 <proof>

lemma *hpow\_hypersum\_pos\_hffp*:

```

assumes "e ∈ Infinitesimal" and "e > 0"
      and "a ∈ HFinite" and "a > 1"
      and "z ∈ HFinite" and "z > 0"
defines "N ≡ hypnat (hfloor (z / e))"
shows "a hpow z ≈
      (∑h i∈{0..N}. hfallfactpow (of_hypnat N) i / hfact i * (eulerK
a e * e) pow i)"
⟨proof⟩

```

```

lemma HFinite_Infinitesimal_hpow_hypersum_pos_eulerK_epsilon:
  assumes "a ∈ HFinite" and "a > 1"
  and "z ∈ HFinite" and "z > 0"
  defines "N ≡ hypnat (hfloor (z/ε))"
  shows "a hpow z ≈
        (∑h i∈{0..N}. hypreal_of_hypnat (N hchoose i) * (eulerK
a ε * ε) pow i)"
  ⟨proof⟩

```

```

lemma Infinitesimal_pow_hfallfactpow_le_pow:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and e_gt_zero: "e > (0::hypreal)"
  and exponentz_HFinite: "z ∈ HFinite - Infinitesimal"
  and exponentz_gt_zero: "z > 0"
  defines "N ≡ hypnat (hfloor (z / e))"
  shows "e pow n * hfallfactpow (of_hypnat N) n ≤ z pow n"
  ⟨proof⟩

```

```

lemma HyperSummable_exp_terms:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and e_gt_zero: "e > (0::hypreal)"
  and exponentz_HFinite: "z ∈ HFinite - Infinitesimal"
  and exponentz_gt_zero: "z > 0"
  and k_HFinite: "k ∈ HFinite"
  and k_gt_zero: "k > 0"
  defines "N ≡ hypnat (hfloor (z / e))"
  shows "HyperSummable
        (λi. hfallfactpow (of_hypnat N) i / hfact i * (k * e) pow
i)"
  ⟨proof⟩

```

We compare against the previous series this time to show hypersum for negative exponent is also summable

```

lemma HyperSummable_exp_terms2:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and e_gt_zero: "e > (0::hypreal)"
  and exponentz_HFinite: "z ∈ HFinite - Infinitesimal"
  and exponentz_less_zero: "z < 0"
  and k_HFinite: "k ∈ HFinite"
  and k_gt_zero: "k > 0"

```

```

    defines "N ≡ hypnat (hfloor (-z / e))"
    shows "HyperSummable (λi. hfallfactpow (of_hypnat N) i / hfact
i * (k * -e) pow i)"
⟨proof⟩

lemma HyperSummable_binomial_neg_hchoose:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
    and infitesimal_e_gt_zero: "e > 0"
    and base_finite: "a ∈ HFinite"
    and base_greater_1: "a > 1"
    and exponentz_HFinite: "z ∈ HFinite - Infinitesimal"
    and exponentz_less_zero: "z < 0"
  defines "N ≡ hypnat (hfloor(-z/e))"
  shows "HyperSummable (λi. of_hypnat (N hchoose i) * (eulerK a e * -e)
pow i)"
⟨proof⟩

lemma HyperSummable_binomial_neg_hchoose':
  assumes infinitesimal_e: "e ∈ Infinitesimal"
    and infitesima_e_gt_zero: "e > 0"
    and base_finite: "a ∈ HFinite"
    and base_greater_1: "a > 1"
    and exponentz_HFinite: "z ∈ HFinite - Infinitesimal"
    and exponentz_less_zero: "z < 0"
  defines "N ≡ hypnat (hfloor(-z/e))"
  shows "HyperSummable (λi. of_hypnat (N hchoose i) * (eulerK a (-e)
* -e) pow i)"
⟨proof⟩

lemma hpow_hypersum_neg_hchoose:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
    and infinitesimal_e_gt_zero: "e > 0"
    and base_finite: "a ∈ HFinite"
    and base_greater_1: "a > 1"
    and exponentz_HFinite: "z ∈ HFinite"
    and exponentz_less_zero: "z < 0"
  defines "N ≡ hypnat (hfloor (- z/e))"
  shows "a hpow z ≈
(∑h i ∈ {0..N}. of_hypnat (N hchoose i) * (eulerK a (- e)
* - e) pow i)"
⟨proof⟩

lemma hyperpow_nat_approx:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
    and e_gt_0: "e > (0::hypreal)"
    and z_HFinite_not_Infinitesimal: "z ∈ HFinite - Infinitesimal"
    and z_gt_0: "z > 0"
  shows "z pow of_nat i ≈ (hfallfactpow (of_hypnat(hypnat(hfloor(z/e))))
(of_nat i)) * (e pow of_nat i)"

```

*<proof>*

```
lemma hyperpow_nat_approx':
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and e_gt_0: "e > (0::hypreal)"
  and z_HFinite_not_Infinitesimal: "z ∈ HFinite - Infinitesimal"
  and z_gt_0: "z > 0"
  and Nats_i: "i ∈ ℕ"
  shows "z pow i ≈ (hfallfactpow (of_hypnat(hypnat(hfloor(z/e)))) i)
* (e pow i)"
<proof>
```

```
lemma HFinite_not_Infinitesimal_pow_of_nat_eq_hfallfactpow_neg:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and e_gt_0: "e > (0::hypreal)"
  and z_HFinite_not_Infinitesimal: "z ∈ HFinite - Infinitesimal"
  and z_less_0: "z < 0"
  shows "z pow of_nat i ≈ (hfallfactpow (of_hypnat(hypnat(hfloor(-z/e))))
(of_nat i)) * ((-e) pow of_nat i)"
<proof>
```

```
lemma HFinite_not_Infinitesimal_pow_of_nat_eq_hfallfactpow2:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and e_gt_0: "e > (0::hypreal)"
  and z_HFinite_not_Infinitesimal: "z ∈ HFinite - Infinitesimal"
  and z_gt_0: "z > 0"
  and HFinite_k: "k ∈ HFinite"
  shows "(k * z) pow of_nat i ≈
(hfallfactpow (of_hypnat(hypnat(hfloor(z/e)))) (of_nat i))
* ((k * e) pow of_nat i)"
<proof>
```

```
lemma HFinite_not_Infinitesimal_pow_of_nat_eq_hfallfactpow2_neg:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and e_gt_0: "e > (0::hypreal)"
  and z_HFinite_not_Infinitesimal: "z ∈ HFinite - Infinitesimal"
  and z_less_0: "z < 0"
  and HFinite_k: "k ∈ HFinite"
  shows "(k * z) pow of_nat i ≈
(hfallfactpow (of_hypnat(hypnat(hfloor(-z/e)))) (of_nat i))
* ((k * -e) pow of_nat i)"
<proof>
```

```
lemma HFinite_not_Infinitesimal_pow_of_nat_eq_hfallfactpow_divide:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and e_gt_0: "e > (0::hypreal)"
  and z_HFinite_not_Infinitesimal: "z ∈ HFinite - Infinitesimal"
  and z_gt_0: "z > 0"
  and HFinite_k: "k ∈ HFinite"
```

```

shows "(k * z) pow of_nat i / fact i ≈
      (hfallfactpow (of_hypnat(hypnat(hfloor(z/e)))) (of_nat i)) / fact
i * ((k * e) pow of_nat i)"
⟨proof⟩

```

```

lemma HFinite_not_Infinitesimal_pow_of_nat_eq_hfallfactpow_divide_neg:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and e_gt_0: "e > (0::hypreal)"
  and z_HFinite_not_Infinitesimal: "z ∈ HFinite - Infinitesimal"
  and z_less_0: "z < 0"
  and HFinite_k: "k ∈ HFinite"
  shows "(k * z) pow of_nat i / fact i ≈
        (hfallfactpow (of_hypnat(hypnat(hfloor(-z/e)))) (of_nat i)) / fact
i * ((k * -e) pow of_nat i)"
⟨proof⟩

```

```

lemma binomial_summand_HFinite_neg:
  assumes "e ∈ Infinitesimal" "e > 0"
  "z ∈ HFinite - Infinitesimal" "z < 0"
  "k ∈ HFinite" "k > 0"
  "n = of_nat m"
  shows "hypreal_of_hypnat (hypnat (hfloor(-z/e)) hchoose n) * (k * -e)
pow n ∈ HFinite"
⟨proof⟩

```

These two binomial summands are infinitely close for standard  $n$ .

```

lemma hchoose_approx_for_Nats:
  assumes Infinitesimal_e: "e ∈ Infinitesimal"
  and Infinitesimal_e_gt_0: "e > 0"
  and base_finite: "a ∈ HFinite"
  and base_greater_1: "a > 1"
  and exponentz_HFinite: "z ∈ HFinite - Infinitesimal"
  and exponentz_less_zero: "z < 0"
  shows "∀n∈ℕ. hypreal_of_hypnat (hypnat (hfloor(-z/e)) hchoose n) *
(eulerK a e * -e) pow n ≈
      hypreal_of_hypnat (hypnat (hfloor(-z/e)) hchoose n) *
(eulerK a (-e) * -e) pow n"
⟨proof⟩

```

```

lemma hypersetsum_neg_approx:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and infinitesimal_e_gt_zero: "e > 0"
  and base_finite: "a ∈ HFinite"
  and base_greater_1: "a > 1"
  and exponentz_HFinite: "z ∈ HFinite - Infinitesimal"
  and exponentz_less_zero: "z < 0"
  shows "hypersum
      (λi. of_hypnat (hypnat (hfloor (- z/e)) hchoose i) *

```

```

      (eulerK a e * -e) pow i)
{0..hypnat (hfloor (- z/e))} ≈
hypersum
(λi. of_hypnat (hypnat (hfloor (- z/e)) hchoose i) *
(eulerK a (-e) * -e) pow i)
{0..hypnat (hfloor (- z/e))}"
⟨proof⟩

lemma hpow_hypersum_neg_hchoose':
  assumes "e ∈ Infinitesimal"
    and infinitesimal_e_gt_zero: "e > 0"
    and base_finite: "a ∈ HFinite"
    and base_greater_1: "a > 1"
    and exponentz_HFinite: "z ∈ HFinite - Infinitesimal"
    and exponentz_less_zero: "z < 0"
  shows "a hpow z ≈
    hypersum (λi. of_hypnat (hypnat (hfloor (- z/e)) hchoose
i) * (eulerK a e * -e) pow i)
    {0..hypnat (hfloor (- z/e))}"
⟨proof⟩

```

Euler's series in Paragraph 115, with finite, non-infinitesimal  $z > 0$ :

```

lemma hpow_approx_series_pos:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
    and e_gt_zero: "e > 0"
    and base_finite: "a ∈ HFinite"
    and base_greater_1: "a > 1"
    and exponentz_HFinite: "z ∈ HFinite - Infinitesimal"
    and exponentz_gt_zero: "z > 0"
  defines "N ≡ hypnat(hfloor(z/e))"
  shows "a hpow z ≈ hypersum (λi. (eulerK a e * z) pow i / hfact i)
{0..N}"
⟨proof⟩

```

```

lemma hpow_approx_series_pos_epsilon:
  assumes base_finite: "a ∈ HFinite"
    and base_greater_1: "a > 1"
    and exponentz_HFinite: "z ∈ HFinite - Infinitesimal"
    and exponentz_gt_zero: "z > 0"
  shows "a hpow z ≈
    hypersum (λi. (eulerK a ε * z) pow i / hfact i) {0..hypnat(hfloor(z/ε))}"
⟨proof⟩

```

Euler's series in Paragraph 115, with finite, non-infinitesimal  $z < 0$ :

```

lemma hpow_approx_series_neg:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
    and e_gt_zero: "e > 0"
    and base_finite: "a ∈ HFinite"
    and base_greater_1: "a > 1"

```

```

and exponentz_HFinite: "z ∈ HFinite - Infinitesimal"
and exponentz_less_zero: "z < 0"
shows "a hpow z ≈
      hypersum (λi. (eulerK a e * z) pow i / hfact i) {0..hypnat(hfloor(-z/e))}"

```

⟨proof⟩

Euler's series in Paragraph 115, with infinitesimal z:

```

lemma hpow_approx_series_infinitesimal':
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and base_finite: "a ∈ HFinite"
  and base_greater_1: "a > 1"
  and exponentz_Infinitesimal: "z ≈ 0"
  shows "a hpow z ≈
        hypersum (λi. (eulerK a e * z) pow i / hfact i) {0..< hSuc

```

n}"

⟨proof⟩

```

lemma hpow_approx_series_infinitesimal:
  assumes "e ∈ Infinitesimal"
  and base_finite: "a ∈ HFinite"
  and base_greater_1: "a > 1"
  and exponentz_Infinitesimal: "z ≈ 0"
  shows "a hpow z ≈
        hypersum (λi. (eulerK a e * z) pow i / hfact i) {0..n}"

```

⟨proof⟩

```

lemma hpow_approx_series_infinitesimal_epsilon:
  assumes base_finite: "a ∈ HFinite"
  and base_greater_1: "a > 1"
  and exponentz_Infinitesimal: "z ≈ 0"
  shows "a hpow z ≈
        hypersum (λi. (eulerK a ε * z) pow i / hfact i) {0..n}"

```

⟨proof⟩

Euler's power series in Paragraph 115:

```

lemma hpow_approx_powseries:
  assumes infinitesimal_e: "e ∈ Infinitesimal"
  and e_gt_zero: "e > 0"
  and base_finite: "a ∈ HFinite"
  and base_greater_1: "a > 1"
  and exponentz_HFinite: "z ∈ HFinite"
  shows "a hpow z ≈
        hypersum (λi. (eulerK a e * z) pow i / hfact i) {0..hypnat(hfloor(|z|/e))}"

```

⟨proof⟩

## 11.4 Euler's Introductio, Paragraph 116: exponential function.

First version of main theorem, with Euler's  $k$  explicit:

```
lemma Euler_hpow_approx_powseries':
  assumes base_finite: "a ∈ HFinite"
    and base_greater_1: "a > 1"
    and exponentz_HFinite: "z ∈ HFinite"
    and index_infinite: "n ∈ HNatInfinite"
  shows "a hpow z ≈
        hypersum (λi. (eulerK a (1 / of_hypnat n) * z) pow i / hfact
i) {0..n}"
⟨proof⟩
```

We can remove the dependency between the infinite sum and the scaling factor as the hypersequence is hypersummable (hyper Cauchy) and so infinitely close for any infinite  $m$  and  $n$ , giving us Euler's result for power series.

```
lemma Euler_hpow_approx_powseries:
  assumes base_finite: "a ∈ HFinite"
    and base_greater_1: "a > 1"
    and exponentz_HFinite: "z ∈ HFinite"
    and pow_infinite: "n ∈ HNatInfinite"
    and index_infinite: "m ∈ HNatInfinite"
  shows "a hpow z ≈
        hypersum (λi. (eulerK a (1 / of_hypnat n) * z) pow i / hfact
i) {0..m}"
⟨proof⟩
```

```
lemma epsilon_inverse_hSuc_wnh:
  "ε = 1/hypreal_of_hypnat (hSuc whn)"
⟨proof⟩
```

We derive another simpler version of Euler's result by fixing our positive infinitesimal.

```
lemma Euler_hpow_approx_powseries_epsilon:
  assumes base_finite: "a ∈ HFinite"
    and base_greater_1: "a > 1"
    and exponentz_HFinite: "z ∈ HFinite"
  shows "a hpow z ≈
        hypersum (λi. (eulerK a ε * z) pow i / hfact i) {0..whn}"
⟨proof⟩
```

Theorem as stated by McKinzie and Tucker, 2001:

```
lemma Euler_hpow_approx_powseries_MT:
  assumes base_finite: "a ∈ HFinite"
    and base_greater_1: "a > 1"
  shows "∃k∈HFinite.
        ∀z∈HFinite.
```

$\forall n \in \text{HNatInfinite}.$   
 $a \text{ hpow } z \approx \text{hypersum } (\lambda i. (k * z) \text{ pow } i / \text{hfact } i)$

{0..n}"  
 <proof>

**lemma binomial\_term\_le\_inv\_fact:**  
 assumes "k ≤ N" and "(0::hypreal) < of\_hypnat N"  
 shows "hypreal\_of\_hypnat (N hchoose k) \* (1 / of\_hypnat N) pow k  
 ≤ 1 / hfact k"  
 <proof>

**lemma geometric\_half\_sum\_lt\_two:**  
 " $\sum_h ((\text{pow } (1 / 2)) \{0..M\} < (2::hypreal))$ "  
 <proof>

**lemma inv\_fact\_sum\_le\_three:**  
 "hypersum ( $\lambda k. 1 / (\text{hfact } k)$ ) {0..N} ≤ (3 ::hypreal)"  
 <proof>

**lemma HNatInfinite\_one\_plus\_inv\_pow\_HFinite:**  
 assumes "n ∈ HNatInfinite"  
 shows "(1 + 1 / hypreal\_of\_hypnat n) pow n ∈ HFinite"  
 <proof>

**lemma hpow\_approx\_powseries\_exp\_lemma:**  
 assumes exponentz\_HFinite: "z ∈ HFinite"  
 and index\_infinite: "m ∈ HNatInfinite"  
 and pow\_infinite: "n ∈ HNatInfinite"  
 shows "((1 + 1/of\_hypnat n) pow n) hpow z ≈  
 hypersum ( $\lambda i. z \text{ pow } i / \text{hfact } i$ ) {0..m}"  
 <proof>

**lemma hpow\_approx\_powseries\_exp\_lemma2:**  
 assumes "z ∈ HFinite"  
 shows "((1 + 1/of\_hypnat whn) pow whn) hpow z ≈ hypersum ( $\lambda i. z \text{ pow } i / \text{hfact } i$ ) {0..whn}"  
 <proof>

Setting the exponent z=1, gives us the series to evaluate the base, which in Introductio 122, Euler denotes by the symbol e. We do the same below.

**lemma hpow\_approx\_powseries\_exp\_lemma3:**  
 "(1 + (1::hypreal)/of\_hypnat whn) pow whn ≈ hypersum ( $\lambda i. 1 / \text{hfact } i$ ) {0..whn}"  
 <proof>

## 11.5 Euler's Introductio, Paragraph 122: exponential series expansion for the base $\epsilon$

We can define a Euler's  $\epsilon$  as follows although to pin it down to a unique real number, we should use the standard part function.

**definition** *euler\_e* ( $\epsilon$ ) **where**  
 $\epsilon = (1 + 1/\text{of\_hypnat } whn) \text{ pow } whn$

Basic properties of  $\epsilon$ .

**lemma** *euler\_e\_pos* [*simp*]: " $(\epsilon::\text{hypreal}) > 0$ "  
 $\langle \text{proof} \rangle$

The (infinitely close) series for  $\epsilon$

**lemma** *Euler\_e*:  
 $\epsilon \approx \text{hypersum } (\lambda i. (1::\text{hypreal}) / \text{hfact } i) \{0..whn\}$   
 $\langle \text{proof} \rangle$

Euler's Exponential series as a hyperfinite hypersum i.e. an infinite polynomial:

**lemma** *Euler\_exp\_powseries*:  
**assumes** " $z \in \text{HFinite}$ "  
**shows** " $\epsilon \text{ hpow } z \approx \text{hypersum } (\lambda i. z \text{ pow } i / \text{hfact } i) \{0..whn\}$ "  
 $\langle \text{proof} \rangle$

**lemma** *euler\_e\_hpow\_zero* [*simp*]: " $\epsilon \text{ hpow } 0 = 1$ "  
 $\langle \text{proof} \rangle$

**lemma** *euler\_e\_hpow\_one* [*simp*]: " $\epsilon \text{ hpow } 1 = \epsilon$ "  
 $\langle \text{proof} \rangle$

**lemma** *hexp\_gt\_one*: " $0 < x \implies 1 < \epsilon \text{ hpow } x$ "  
 $\langle \text{proof} \rangle$

**lemma** *HFinite\_euler\_e* [*simp*]:  
 $(\epsilon::\text{real star}) \in \text{HFinite}$   
 $\langle \text{proof} \rangle$

**lemma** *euler\_e\_gt\_one* [*simp*]:  
 $(\epsilon::\text{real star}) > 1$   
 $\langle \text{proof} \rangle$

**lemma** *euler\_e\_ne\_one*: " $(\epsilon::\text{hypreal}) \neq 1$ "  
 $\langle \text{proof} \rangle$

**lemma** "*eulerK*  $\epsilon (1 / \text{hypreal\_of\_hypnat } whn) = 1$ "  
 $\langle \text{proof} \rangle$

```

lemma euler_e_ge_two: "( $\epsilon$ ::hypreal)  $\geq 2$ "
<proof>

lemma euler_e_power_ge_two_power: "( $\epsilon$ ::hypreal)  $^ n \geq 2 ^ n$ "
<proof>

lemma euler_e_power_bound:
  assumes "a  $\in$  HFinite" "a  $\geq 1$ "
  shows " $\exists n. (\epsilon$ ::hypreal)  $^ n \geq a$ "
<proof>

lemma hpow_euler_e_eulerK_approx:
  assumes "a  $\in$  HFinite"
  and "a > 1"
  shows " $\epsilon$  hpow (eulerK a (1/of_hypnat whn))  $\approx a$ "
<proof>

lemma euler_e_hpow_not_approx_one_pos:
  assumes "d  $\in$  HFinite" "d > 0" "d  $\notin$  Infinitesimal"
  shows " $\neg (\epsilon$  hpow d  $\approx 1$ )"
<proof>

lemma euler_e_hpow_not_approx_one_neg:
  assumes "d  $\in$  HFinite" "d < 0" "d  $\notin$  Infinitesimal"
  shows " $\neg (\epsilon$  hpow d  $\approx 1$ )"
<proof>

lemma euler_e_hpow_not_approx_one:
  assumes "d  $\in$  HFinite" "d  $\neq 0$ " "d  $\notin$  Infinitesimal"
  shows " $\neg (\epsilon$  hpow d  $\approx 1$ )"
<proof>

corollary euler_e_hpow_approx_one_iff_Infinitesimal:
  assumes "d  $\in$  HFinite"
  shows " $(\epsilon$  hpow d  $\approx 1) \iff (d \in$  Infinitesimal)"
<proof>

lemma euler_e_hpow_approx_inject:
  assumes " $\epsilon$  hpow x  $\approx \epsilon$  hpow y"
  and "x  $\in$  HFinite"
  and "y  $\in$  HFinite"
  shows "x  $\approx$  y"
<proof>

```

## 11.6 Hypernatural logarithm

Let us define the hypernatural logarithm of a hyperreal  $x$

**definition**  $L_n$  where

$$"L_n x \equiv hLog \ \epsilon \ x"$$

**lemma** *Ln\_euler\_e [simp]*: "Ln  $\epsilon = 1$ "  
 ⟨*proof*⟩

**lemma** *hpow\_e\_Ln [simp]*:  
 assumes "a > 0"  
 shows " $\epsilon$  hpow (Ln a) = a"  
 ⟨*proof*⟩

**lemma** *Ln\_ge\_zero*: "a  $\geq 1 \implies$  Ln a  $\geq 0$ "  
 ⟨*proof*⟩

**lemma** *Ln\_inverse*: "x > 0  $\implies$  Ln (inverse x) = - Ln x"  
 ⟨*proof*⟩

**lemma** *Ln\_less\_zero*: "a > 0  $\implies$  a < 1  $\implies$  Ln a < 0"  
 ⟨*proof*⟩

**lemma** *Ln\_HFinite*:  
 assumes "a  $\in$  HFinite - Infinitesimal" "a > 0"  
 shows "Ln a  $\in$  HFinite"  
 ⟨*proof*⟩

**lemma** *Ln\_eulerK\_approx*:  
 assumes "a  $\in$  HFinite" "a > 1"  
 shows "Ln a  $\approx$  eulerK a (1 / hypreal\_of\_hypnat whn)"  
 ⟨*proof*⟩

This is just to show that natural logarithm would have the expected definition (if we take the standard part of the RHS)

**lemma** *Ln\_approx\_def*:  
 assumes "a  $\in$  HFinite" "a > 1"  
 shows "Ln a  $\approx$  (a hpow (1 / hypreal\_of\_hypnat whn) - 1) \* of\_hypnat whn"  
 ⟨*proof*⟩

Euler's power series in terms of hypernatural logarithm.

**lemma** *hpow\_powseries\_Ln*:  
 assumes "a  $\in$  HFinite"  
 and "a > 1"  
 and "z  $\in$  HFinite"  
 shows "a hpow z  $\approx$  ( $\sum_h i \in \{0..whn\}. (Ln a * z)^i / hfact i$ )"  
 ⟨*proof*⟩

## 11.7 Linking the Euler and Isabelle library exponential functions.

Just for the sake of it: we link our current derivation of the exponential series to the Isabelle library one.

```
lemma exp_NSsums: "(λn. z ^ n / fact n) NSsums (exp z)"
⟨proof⟩
```

```
lemma hypersum_exp_approx:
  assumes "N ∈ HNatInfinite"
  shows "hypersum (*f* (λn. z ^ n / fact n)) {0..<N} ≈ of_real (exp z)"
⟨proof⟩
```

```
lemma starfun_exp_of_real: "(*f* exp) (of_real z) = of_real (exp z)"
⟨proof⟩
```

```
lemma starfun_exp_term_eq:
  "(*f* (λn. z ^ n / fact n)) n = (of_real z) pow n / hfact n"
⟨proof⟩
```

```
lemma HyperSummable_starfun_real_exp:
  "HyperSummable (*f* (λn. (z::real) ^ n / fact n))"
⟨proof⟩
```

```
lemma hypersum_starfun_exp_approx_euler_sum:
  "hypersum (*f* (λn. z ^ n / fact n)) {0..N} ≈
  hypersum (λi. (of_real z) pow i / hfact i) {0..N}"
⟨proof⟩
```

Theorem linking the two representations of the exponential series.

```
theorem euler_e_hpow_approx_starfun_exp:
  "ε hpow (of_real z) ≈ (*f* exp) (of_real z)"
⟨proof⟩
```

```
corollary euler_e_hpow_approx_exp:
  "ε hpow (hypreal_of_real z) ≈ hypreal_of_real (exp z)"
⟨proof⟩
```

The standard part of our Euler exponential is (the hyperreal embedding of) Isabelle's exponential function

```
corollary st_euler_e_hpow_eq_exp:
  "st(ε hpow (hypreal_of_real z)) = hypreal_of_real (exp z)"
⟨proof⟩
```

We can show that the standard part of our Euler  $\epsilon$  is (literally!) the real  $e$ .

```
corollary euler_e_approx_exp1: "(ε::hypreal) ≈ of_real (exp 1)"
⟨proof⟩
```

**corollary** *st\_euler\_e\_approx\_exp1*: "st ( $\epsilon$ ::hypreal) = of\_real (exp 1)"  
 ⟨proof⟩

## 11.8 Derivative of Euler's exponential function.

We can also show that our Euler exponential function is its own derivative.

**lemma** *euler\_e\_hpow\_eulerK\_approx\_base*:

assumes *inf\_e*: "e ∈ Infinitesimal"  
 and *e\_pos*: "e > 0"  
 and *a\_HF*: "a ∈ HFinite"  
 and *a\_gt1*: "a > 1"

shows " $\epsilon$  hpow (eulerK a e)  $\approx$  a"

⟨proof⟩

**lemma** *eulerK\_approx\_Ln\_pos*:

assumes "e ∈ Infinitesimal" "e > 0" "a ∈ HFinite" "a > 1"  
 shows "eulerK a e  $\approx$  Ln a"

⟨proof⟩

Extend the previous lemma to deal with negative infinitesimals.

**lemma** *eulerK\_approx\_Ln\_neg*:

assumes "e ∈ Infinitesimal" "e < 0" "a ∈ HFinite" "a > 1"  
 shows "eulerK a e  $\approx$  Ln a"

⟨proof⟩

The combined result for any nonzero infinitesimal.

**theorem** *eulerK\_approx\_Ln*:

assumes "e ∈ Infinitesimal" "e  $\neq$  0" "a ∈ HFinite" "a > 1"  
 shows "eulerK a e  $\approx$  Ln a"

⟨proof⟩

**lemma** *hpow\_difference\_quotient*:

assumes "a > 0"

shows "(a hpow (x + e) - a hpow x) / e = a hpow x \* eulerK a e"

⟨proof⟩

Nonstandard derivative of the general exponential

**theorem** *NSderivative\_hpow*:

assumes *a\_HF*: "a ∈ HFinite"

and *a\_gt1*: "a > 1"

and *x\_HF*: "x ∈ HFinite"

and *e\_inf*: "e ∈ Infinitesimal"

and *e\_ne0*: "e  $\neq$  0"

shows "(a hpow (x + e) - a hpow x) / e  $\approx$  Ln a \* (a hpow x)"

⟨proof⟩

Euler's exponential function is its own (nonstandardly-expressed) derivative (as expected).

**theorem** *NSderivative\_euler\_e*:  
 assumes " $x \in HFinite$ " " $e \in Infinitesimal$ " " $e \neq 0$ "  
 shows " $(\epsilon \text{ hpow } (x + e) - \epsilon \text{ hpow } x) / e \approx \epsilon \text{ hpow } x$ "  
*<proof>*

**lemma** *NSderivative\_hpow\_at\_zero*:  
 assumes " $a \in HFinite$ " " $a > 1$ " " $e \in Infinitesimal$ " " $e \neq 0$ "  
 shows " $(a \text{ hpow } e - 1) / e \approx \text{Ln } a$ "  
*<proof>*

The derivative of  $\epsilon \text{ hpow } x$  at  $x=0$  is 1.

**lemma** *NSderivative\_euler\_e\_at\_zero*:  
 assumes " $e \in Infinitesimal$ " " $e \neq 0$ "  
 shows " $(\epsilon \text{ hpow } e - 1) / e \approx 1$ "  
*<proof>*

**lemma** " $(\epsilon \text{ hpow } \epsilon - 1) / \epsilon \approx 1$ "  
*<proof>*

**end**