The HOL reals Library

J. R. Harrison

University of Cambridge Computer Laboratory
New Museums Site
Pembroke Street
Cambridge, CB2 3QG
England.

21st July 1992

Contents

Chapter 1

The reals library

The reals library provides a theory of real numbers, including properties of the elementary transcendental functions. The theories are constructed purely definitionally, using Dedekind cuts of rationals.

1.1 Loading the library

Assuming HOL is installed properly, it is only necessary to execute the following ML command:

```
#load_library `reals`;;                                                    1
```

This makes the real theory hierarchy a parent, sets up the most useful theorems from the theories `REAL` and `TRANSC` to autoload, and provides an interface map in `real_interface_map` which makes real operations more presentable. Note however that this interface map is not set up automatically. If required, the following command will set it up:

```
#set_interface_map real_interface_map;;                                    2
```

However if there is an existing interface map, something rather more elaborate may be necessary. Various types are used in auxiliary ways within the theory hierarchy, but the only one which matters to most users is the type of real numbers, which is called simply `real`.

1.2 Theories in the reals library

The theory hierarchy underlying the library is quite extensive, but the dependencies are almost linear. In the list that follows, each theory depends on the ones above it, except that `LIM` and `SEQ` are independent; the theory `POWSER` brings the two together.

- `HRAT`: Positive rational numbers
- `HREAL`: Positive real numbers
- `REALAX`: Basic "axioms" characterizing the real numbers

- `REAL`: Main algebraic properties of real numbers
- `TOPOLOGY`: Basic topological definitions and theorems
- `NETS`: Moore-Smith convergence nets
- `LIM`: Limits, continuity and differentiation
- `SEQ`: Infinite sequences and series
- `POWSER`: Power series
- `TRANSC`: Transcendental functions

For most purposes, only the main definitions and theorems from `REAL` and `TRANSC` should be needed, and these are set up to autoload when the library is loaded. The intermediate theories `TOPOLOGY` to `POWSER` are a means to an end, and are not necessarily suitable as a foundation for a serious theory of mathematical analysis.

1.3 The user interface

HOL does not support subtypes, so the real numbers are formally a different type from the natual numbers. The "inclusion" mapping $\iota : num \rightarrow real$ is called `real_of_num`, but the ampersand (`&`) represents it in the standard interface map. interface map represents it by the ampersand `&`. Thus the real number "numerals" can be written as `&0`, `&1` etc. Occasionally it is necessary to bracket such combinations to avoid problems with associativity.

The arithmetic operations also have different names from their natural number counterparts:

- `real_neg` Negation (unary $-$)
- `real_add` Addition ($+$)
- `real_mul` Multiplication ($*$)
- `real_sub` Subtraction (binary $-$)
- `real_lt` Less-than ($<$)
- `real_le` Less-than-or-equal-to ($\leq$)
- `real_gt` Greater-than ($>$)
- `real_ge` Greater-than-or-equal-to ($\geq$)

- `real_inv` Multiplicative inverse (inv)

The interface map provided in `real_interface_map` when the library is loaded provides the normal names for these (except that unary minus is `--`), and replaces the natural number conterparts with `num_add` etc. This can make it clumsy to mix real and natural number arithmetic, but is usually satisfactory.

1.4 Other functions defined

Apart from the basic arithmetic operations listed above, the library defines various other constants, including the following:

- `abs`: Absolute value; $abs(x) = |x|$
- `acs`: Arc cosine; $0 \leq acs(x) \leq \pi$
- `asn`: Arc sine; $-\pi/2 \leq asn(x) \leq \pi/2$
- `atn`: Arc tangent; $-\pi/2 < atn(x) < \pi/2$
- `cos`: Cosine
- `exp`: Exponential function; $exp(x) = e^x$
- `ln`: Natural logarithm; base $e = exp(1)$
- `pi`: The constant $\pi = 3.14159\ldots$
- `pow`: Natural number power: $x\ pow\ n = x^n$
- `root`: n'th root; $root(n)x = \sqrt[n]{x}$
- `sin`: Sine
- `sqrt`: Square root; $sqrt(x) = \sqrt{x})$
- `tan`: Tangent

A list of theorems about each of these constants is provided — see the following sections for a full list.

Chapter 2

Pre-proved Theorems

The sections that follow list all theorems in the `reals` library, including definitions. The theorems are grouped into their theories, which corresponds quite closely to a natural logical partitioning. When the `reals` library is loaded, only theorems in `REAL` and `TRANSC` are set up to autoload (and not all of them). These should be all the theorems needed for day-to-day use of the library. Other definitions and theorems may, if desired, be recovered by using the `defintion` and `theorem` functions provided by ML. Note that all theorems are printed with the interface map set up as done by the library. In addition the symbol `-->` is used for the constants `tends_num_real` and `tends_real_real` in the theories SEQ and LIM respectively.

2.1 HRAT

`hrat_1` (HRAT)
```
  |- hrat_1 = mk_hrat($trat_eq trat_1)
```

`hrat_add` (HRAT)
```
  |- !g00014 g00015.
      g00014 hrat_add g00015 =
      mk_hrat
      ($trat_eq(($@(dest_hrat g00014)) trat_add ($@(dest_hrat g00015))))
```

`HRAT_ADD_ASSOC` (HRAT)
```
  |- !h i j. h hrat_add (i hrat_add j) = (h hrat_add i) hrat_add j
```

`HRAT_ADD_SYM` (HRAT)
```
  |- !h i. h hrat_add i = i hrat_add h
```

`HRAT_ADD_TOTAL` (HRAT)
```
  |- !h i. (h = i) \/ (?d. h = i hrat_add d) \/ (?d. i = h hrat_add d)
```

`HRAT_ARCH` (HRAT)
```
  |- !h. ?n d. hrat_sucint n = h hrat_add d
```

```
hrat_inv  (HRAT)
   |- !g00013.
        hrat_inv g00013 = mk_hrat($trat_eq(trat_inv($@(dest_hrat g00013))))

HRAT_LDISTRIB  (HRAT)
   |- !h i j.
        h hrat_mul (i hrat_add j) = (h hrat_mul i) hrat_add (h hrat_mul j)

hrat_mul  (HRAT)
   |- !g00016 g00017.
        g00016 hrat_mul g00017 =
        mk_hrat
        ($trat_eq(($@(dest_hrat g00016)) trat_mul ($@(dest_hrat g00017))))

HRAT_MUL_ASSOC  (HRAT)
   |- !h i j. h hrat_mul (i hrat_mul j) = (h hrat_mul i) hrat_mul j

HRAT_MUL_LID  (HRAT)
   |- !h. hrat_1 hrat_mul h = h

HRAT_MUL_LINV  (HRAT)
   |- !h. (hrat_inv h) hrat_mul h = hrat_1

HRAT_MUL_SYM  (HRAT)
   |- !h i. h hrat_mul i = i hrat_mul h

HRAT_NOZERO  (HRAT)
   |- !h i. ~(h hrat_add i = h)

HRAT_SUCINT  (HRAT)
   |- (hrat_sucint 0 = hrat_1) /\
      (!n. hrat_sucint(SUC n) = (hrat_sucint n) hrat_add hrat_1)

hrat_sucint  (HRAT)
   |- !g00018. hrat_sucint g00018 = mk_hrat($trat_eq(trat_sucint g00018))

hrat_tybij  (HRAT)
   |- (!a. mk_hrat(dest_hrat a) = a) /\
      (!r. (\c. ?x. c = $trat_eq x)r = (dest_hrat(mk_hrat r) = r))

hrat_TY_DEF  (HRAT)
   |- ?rep. TYPE_DEFINITION(\c. ?x. c = $trat_eq x)rep

trat_1  (HRAT)
   |- trat_1 = 0,0
```

```
trat_add  (HRAT)
  |- !x y x' y'.
       (x,y) trat_add (x',y') =
       PRE(((SUC x) * (SUC y')) + ((SUC x') * (SUC y))),
       PRE((SUC y) * (SUC y'))

TRAT_ADD_ASSOC  (HRAT)
  |- !h i j.
       (h trat_add (i trat_add j)) trat_eq ((h trat_add i) trat_add j)

TRAT_ADD_SYM  (HRAT)
  |- !h i. (h trat_add i) trat_eq (i trat_add h)

TRAT_ADD_SYM_EQ  (HRAT)
  |- !h i. h trat_add i = i trat_add h

TRAT_ADD_TOTAL  (HRAT)
  |- !h i.
       h trat_eq i \/
       (?d. h trat_eq (i trat_add d)) \/
       (?d. i trat_eq (h trat_add d))

TRAT_ADD_WELLDEFINED  (HRAT)
  |- !p q r. p trat_eq q ==> (p trat_add r) trat_eq (q trat_add r)

TRAT_ADD_WELLDEFINED2  (HRAT)
  |- !p1 p2 q1 q2.
       p1 trat_eq p2 /\ q1 trat_eq q2 ==>
       (p1 trat_add q1) trat_eq (p2 trat_add q2)

TRAT_ARCH  (HRAT)
  |- !h. ?n d. (trat_sucint n) trat_eq (h trat_add d)

trat_eq  (HRAT)
  |- !x y x' y'.
       (x,y) trat_eq (x',y') = ((SUC x) * (SUC y') = (SUC x') * (SUC y))

TRAT_EQ_AP  (HRAT)
  |- !p q. (p = q) ==> p trat_eq q

TRAT_EQ_EQUIV  (HRAT)
  |- !p q. p trat_eq q = ($trat_eq p = $trat_eq q)

TRAT_EQ_REFL  (HRAT)
  |- !p. p trat_eq p
```

TRAT_EQ_SYM (HRAT)
```
  |- !p q. p trat_eq q = q trat_eq p
```

TRAT_EQ_TRANS (HRAT)
```
  |- !p q r. p trat_eq q /\ q trat_eq r ==> p trat_eq r
```

trat_inv (HRAT)
```
  |- !x y. trat_inv(x,y) = y,x
```

TRAT_INV_WELLDEFINED (HRAT)
```
  |- !p q. p trat_eq q ==> (trat_inv p) trat_eq (trat_inv q)
```

TRAT_LDISTRIB (HRAT)
```
  |- !h i j.
       (h trat_mul (i trat_add j)) trat_eq
       ((h trat_mul i) trat_add (h trat_mul j))
```

trat_mul (HRAT)
```
  |- !x y x' y'.
       (x,y) trat_mul (x',y') =
       PRE((SUC x) * (SUC x')),PRE((SUC y) * (SUC y'))
```

TRAT_MUL_ASSOC (HRAT)
```
  |- !h i j.
       (h trat_mul (i trat_mul j)) trat_eq ((h trat_mul i) trat_mul j)
```

TRAT_MUL_LID (HRAT)
```
  |- !h. (trat_1 trat_mul h) trat_eq h
```

TRAT_MUL_LINV (HRAT)
```
  |- !h. ((trat_inv h) trat_mul h) trat_eq trat_1
```

TRAT_MUL_SYM (HRAT)
```
  |- !h i. (h trat_mul i) trat_eq (i trat_mul h)
```

TRAT_MUL_SYM_EQ (HRAT)
```
  |- !h i. h trat_mul i = i trat_mul h
```

TRAT_MUL_WELLDEFINED (HRAT)
```
  |- !p q r. p trat_eq q ==> (p trat_mul r) trat_eq (q trat_mul r)
```

TRAT_MUL_WELLDEFINED2 (HRAT)
```
  |- !p1 p2 q1 q2.
       p1 trat_eq p2 /\ q1 trat_eq q2 ==>
       (p1 trat_mul q1) trat_eq (p2 trat_mul q2)
```

```
TRAT_NOZERO  (HRAT)
  |- !h i. ~(h trat_add i) trat_eq h

TRAT_SUCINT  (HRAT)
  |- (trat_sucint 0) trat_eq trat_1 /\
     (!n. (trat_sucint(SUC n)) trat_eq ((trat_sucint n) trat_add trat_1))

trat_sucint  (HRAT)
  |- (trat_sucint 0 = trat_1) /\
     (!n. trat_sucint(SUC n) = (trat_sucint n) trat_add trat_1)

TRAT_SUCINT_0  (HRAT)
  |- !n. (trat_sucint n) trat_eq (n,0)
```

2.2 HREAL

```
CUT_BOUNDED  (HREAL)
  |- !X. ?x. ~cut X x

CUT_DOWN  (HREAL)
  |- !X x y. cut X x /\ y hrat_lt x ==> cut X y

CUT_ISACUT  (HREAL)
  |- !X. isacut(cut X)

CUT_NEARTOP_ADD  (HREAL)
  |- !X e. ?x. cut X x /\ ~cut X(x hrat_add e)

CUT_NEARTOP_MUL  (HREAL)
  |- !X u. hrat_1 hrat_lt u ==> (?x. cut X x /\ ~cut X(u hrat_mul x))

CUT_NONEMPTY  (HREAL)
  |- !X. ?x. cut X x

cut_of_hrat  (HREAL)
  |- !x. cut_of_hrat x = (\y. y hrat_lt x)

CUT_STRADDLE  (HREAL)
  |- !X x y. cut X x /\ ~cut X y ==> x hrat_lt y

CUT_UBOUND  (HREAL)
  |- !X x y. ~cut X x /\ x hrat_lt y ==> ~cut X y

CUT_UP  (HREAL)
  |- !X x. cut X x ==> (?y. cut X y /\ x hrat_lt y)
```

```
EQUAL_CUTS  (HREAL)
  |- !X Y. (cut X = cut Y) ==> (X = Y)

HRAT_DOWN  (HREAL)
  |- !x. ?y. y hrat_lt x

HRAT_DOWN2  (HREAL)
  |- !x y. ?z. z hrat_lt x /\ z hrat_lt y

HRAT_EQ_LADD  (HREAL)
  |- !x y z. (x hrat_add y = x hrat_add z) = (y = z)

HRAT_EQ_LMUL  (HREAL)
  |- !x y z. (x hrat_mul y = x hrat_mul z) = (y = z)

HRAT_GT_L1  (HREAL)
  |- !x y. hrat_1 hrat_lt ((hrat_inv x) hrat_mul y) = x hrat_lt y

HRAT_GT_LMUL1  (HREAL)
  |- !x y. y hrat_lt (x hrat_mul y) = hrat_1 hrat_lt x

HRAT_INV_MUL  (HREAL)
  |- !x y. hrat_inv(x hrat_mul y) = (hrat_inv x) hrat_mul (hrat_inv y)

hrat_lt  (HREAL)
  |- !x y. x hrat_lt y = (?d. y = x hrat_add d)

HRAT_LT_ADD2  (HREAL)
  |- !u v x y.
       u hrat_lt x /\ v hrat_lt y ==> (u hrat_add v) hrat_lt (x hrat_add y)

HRAT_LT_ADDL  (HREAL)
  |- !x y. x hrat_lt (x hrat_add y)

HRAT_LT_ADDR  (HREAL)
  |- !x y. y hrat_lt (x hrat_add y)

HRAT_LT_ANTISYM  (HREAL)
  |- !x y. ~(x hrat_lt y /\ y hrat_lt x)

HRAT_LT_GT  (HREAL)
  |- !x y. x hrat_lt y ==> ~y hrat_lt x

HRAT_LT_L1  (HREAL)
  |- !x y. ((hrat_inv x) hrat_mul y) hrat_lt hrat_1 = y hrat_lt x
```

```
HRAT_LT_LADD (HREAL)
  |- !x y z. (z hrat_add x) hrat_lt (z hrat_add y) = x hrat_lt y

HRAT_LT_LMUL (HREAL)
  |- !x y z. (z hrat_mul x) hrat_lt (z hrat_mul y) = x hrat_lt y

HRAT_LT_LMUL1 (HREAL)
  |- !x y. (x hrat_mul y) hrat_lt y = x hrat_lt hrat_1

HRAT_LT_MUL2 (HREAL)
  |- !u v x y.
       u hrat_lt x /\ v hrat_lt y ==> (u hrat_mul v) hrat_lt (x hrat_mul y)

HRAT_LT_NE (HREAL)
  |- !x y. x hrat_lt y ==> ~(x = y)

HRAT_LT_R1 (HREAL)
  |- !x y. (x hrat_mul (hrat_inv y)) hrat_lt hrat_1 = x hrat_lt y

HRAT_LT_RADD (HREAL)
  |- !x y z. (x hrat_add z) hrat_lt (y hrat_add z) = x hrat_lt y

HRAT_LT_REFL (HREAL)
  |- !x. ~x hrat_lt x

HRAT_LT_RMUL (HREAL)
  |- !x y z. (x hrat_mul z) hrat_lt (y hrat_mul z) = x hrat_lt y

HRAT_LT_RMUL1 (HREAL)
  |- !x y. (x hrat_mul y) hrat_lt x = y hrat_lt hrat_1

HRAT_LT_TOTAL (HREAL)
  |- !x y. (x = y) \/ x hrat_lt y \/ y hrat_lt x

HRAT_LT_TRANS (HREAL)
  |- !x y z. x hrat_lt y /\ y hrat_lt z ==> x hrat_lt z

HRAT_MEAN (HREAL)
  |- !x y. x hrat_lt y ==> (?z. x hrat_lt z /\ z hrat_lt y)

HRAT_MUL_RID (HREAL)
  |- !x. x hrat_mul hrat_1 = x

HRAT_MUL_RINV (HREAL)
  |- !x. x hrat_mul (hrat_inv x) = hrat_1
```

```
HRAT_RDISTRIB  (HREAL)
  |- !x y z.
       (x hrat_add y) hrat_mul z = (x hrat_mul z) hrat_add (y hrat_mul z)

HRAT_UP  (HREAL)
  |- !x. ?y. x hrat_lt y

hreal_1  (HREAL)
  |- hreal_1 = hreal(cut_of_hrat hrat_1)

hreal_add  (HREAL)
  |- !X Y.
       X hreal_add Y =
       hreal(\w. ?x y. (w = x hrat_add y) /\ cut X x /\ cut Y y)

HREAL_ADD_ASSOC  (HREAL)
  |- !X Y Z. X hreal_add (Y hreal_add Z) = (X hreal_add Y) hreal_add Z

HREAL_ADD_ISACUT  (HREAL)
  |- !X Y. isacut(\w. ?x y. (w = x hrat_add y) /\ cut X x /\ cut Y y)

HREAL_ADD_SYM  (HREAL)
  |- !X Y. X hreal_add Y = Y hreal_add X

HREAL_ADD_TOTAL  (HREAL)
  |- !X Y. (X = Y) \/ (?D. Y = X hreal_add D) \/ (?D. X = Y hreal_add D)

hreal_inv  (HREAL)
  |- !X.
       hreal_inv X =
       hreal
       (\w.
         ?d. d hrat_lt hrat_1 /\ (!x. cut X x ==> (w hrat_mul x) hrat_lt d))

HREAL_INV_ISACUT  (HREAL)
  |- !X.
       isacut
       (\w.
         ?d. d hrat_lt hrat_1 /\ (!x. cut X x ==> (w hrat_mul x) hrat_lt d))

HREAL_LDISTRIB  (HREAL)
  |- !X Y Z.
       X hreal_mul (Y hreal_add Z) =
       (X hreal_mul Y) hreal_add (X hreal_mul Z)

HREAL_LT  (HREAL)
  |- !X Y. X hreal_lt Y = (?D. Y = X hreal_add D)
```

```
hreal_lt (HREAL)
  |- !X Y. X hreal_lt Y = ~(X = Y) /\ (!x. cut X x ==> cut Y x)

HREAL_LT_LEMMA (HREAL)
  |- !X Y. X hreal_lt Y ==> (?x. ~cut X x /\ cut Y x)

HREAL_LT_TOTAL (HREAL)
  |- !X Y. (X = Y) \/ X hreal_lt Y \/ Y hreal_lt X

hreal_mul (HREAL)
  |- !X Y.
       X hreal_mul Y =
       hreal(\w. ?x y. (w = x hrat_mul y) /\ cut X x /\ cut Y y)

HREAL_MUL_ASSOC (HREAL)
  |- !X Y Z. X hreal_mul (Y hreal_mul Z) = (X hreal_mul Y) hreal_mul Z

HREAL_MUL_ISACUT (HREAL)
  |- !X Y. isacut(\w. ?x y. (w = x hrat_mul y) /\ cut X x /\ cut Y y)

HREAL_MUL_LID (HREAL)
  |- !X. hreal_1 hreal_mul X = X

HREAL_MUL_LINV (HREAL)
  |- !X. (hreal_inv X) hreal_mul X = hreal_1

HREAL_MUL_SYM (HREAL)
  |- !X Y. X hreal_mul Y = Y hreal_mul X

HREAL_NOZERO (HREAL)
  |- !X Y. ~(X hreal_add Y = X)

hreal_sub (HREAL)
  |- !Y X. Y hreal_sub X = hreal(\w. ?x. ~cut X x /\ cut Y(x hrat_add w))

HREAL_SUB_ADD (HREAL)
  |- !X Y. X hreal_lt Y ==> ((Y hreal_sub X) hreal_add X = Y)

HREAL_SUB_ISACUT (HREAL)
  |- !X Y.
       X hreal_lt Y ==> isacut(\w. ?x. ~cut X x /\ cut Y(x hrat_add w))

HREAL_SUP (HREAL)
  |- !P.
       (?X. P X) /\ (?Y. !X. P X ==> X hreal_lt Y) ==>
       (!Y. (?X. P X /\ Y hreal_lt X) = Y hreal_lt (hreal_sup P))
```

```
hreal_sup  (HREAL)
  |- !P. hreal_sup P = hreal(\w. ?X. P X /\ cut X w)

HREAL_SUP_ISACUT  (HREAL)
  |- !P.
       (?X. P X) /\ (?Y. !X. P X ==> X hreal_lt Y) ==>
       isacut(\w. ?X. P X /\ cut X w)

hreal_tybij  (HREAL)
  |- (!a. hreal(cut a) = a) /\ (!r. isacut r = (cut(hreal r) = r))

hreal_TY_DEF  (HREAL)
  |- ?rep. TYPE_DEFINITION isacut rep

isacut  (HREAL)
  |- !C.
       isacut C =
       (?x. C x) /\
       (?x. ~C x) /\
       (!x y. C x /\ y hrat_lt x ==> C y) /\
       (!x. C x ==> (?y. C y /\ x hrat_lt y))

ISACUT_HRAT  (HREAL)
  |- !h. isacut(cut_of_hrat h)
```

2.3 REALAX

```
HREAL_EQ_ADDL  (REALAX)
  |- !x y. ~(x = x hreal_add y)

HREAL_EQ_ADDR  (REALAX)
  |- !x y. ~(x hreal_add y = x)

HREAL_EQ_LADD  (REALAX)
  |- !x y z. (x hreal_add y = x hreal_add z) = (y = z)

HREAL_LT_ADD2  (REALAX)
  |- !x1 x2 y1 y2.
       x1 hreal_lt y1 /\ x2 hreal_lt y2 ==>
       (x1 hreal_add x2) hreal_lt (y1 hreal_add y2)

HREAL_LT_ADDL  (REALAX)
  |- !x y. x hreal_lt (x hreal_add y)
```

```
HREAL_LT_ADDR  (REALAX)
  |- !x y. ~(x hreal_add y) hreal_lt x

HREAL_LT_GT  (REALAX)
  |- !x y. x hreal_lt y ==> ~y hreal_lt x

HREAL_LT_LADD  (REALAX)
  |- !x y z. (x hreal_add y) hreal_lt (x hreal_add z) = y hreal_lt z

HREAL_LT_NE  (REALAX)
  |- !x y. x hreal_lt y ==> ~(x = y)

HREAL_LT_REFL  (REALAX)
  |- !x. ~x hreal_lt x

hreal_of_real  (REALAX)
  |- !g00032. hreal_of_real g00032 = hreal_of_treal($@(dest_real g00032))

hreal_of_treal  (REALAX)
  |- !x y. hreal_of_treal(x,y) = (@d. x = y hreal_add d)

HREAL_RDISTRIB  (REALAX)
  |- !x y z.
       (x hreal_add y) hreal_mul z =
       (x hreal_mul z) hreal_add (y hreal_mul z)

r0  (REALAX)
  |- r0 = mk_real($treal_eq treal_0)

r1  (REALAX)
  |- r1 = mk_real($treal_eq treal_1)

REAL_10  (REALAX)
  |- ~(r1 = r0)

real_add  (REALAX)
  |- !g00025 g00026.
       g00025 + g00026 =
       mk_real
       ($treal_eq(($@(dest_real g00025)) treal_add ($@(dest_real g00026))))

REAL_ADD_ASSOC  (REALAX)
  |- !x y z. x + (y + z) = (x + y) + z

REAL_ADD_LID  (REALAX)
  |- !x. r0 + x = x
```

```
REAL_ADD_LINV (REALAX)
  |- !x. (-- x) + x = r0

REAL_ADD_SYM (REALAX)
  |- !x y. x + y = y + x

real_inv (REALAX)
  |- !g00024.
       inv g00024 = mk_real($treal_eq(treal_inv($@(dest_real g00024))))

REAL_ISO_EQ (REALAX)
  |- !h i. h hreal_lt i = (real_of_hreal h) < (real_of_hreal i)

REAL_LDISTRIB (REALAX)
  |- !x y z. x * (y + z) = (x * y) + (x * z)

real_lt (REALAX)
  |- !g00029 g00030.
       g00029 < g00030 =
       ($@(dest_real g00029)) treal_lt ($@(dest_real g00030))

REAL_LT_IADD (REALAX)
  |- !x y z. y < z ==> (x + y) < (x + z)

REAL_LT_MUL (REALAX)
  |- !x y. r0 < x /\ r0 < y ==> r0 < (x * y)

REAL_LT_REFL (REALAX)
  |- !x. ~x < x

REAL_LT_TOTAL (REALAX)
  |- !x y. (x = y) \/ x < y \/ y < x

REAL_LT_TRANS (REALAX)
  |- !x y z. x < y /\ y < z ==> x < z

real_mul (REALAX)
  |- !g00027 g00028.
       g00027 * g00028 =
       mk_real
       ($treal_eq(($@(dest_real g00027)) treal_mul ($@(dest_real g00028))))

REAL_MUL_ASSOC (REALAX)
  |- !x y z. x * (y * z) = (x * y) * z

REAL_MUL_LID (REALAX)
  |- !x. r1 * x = x
```

```
REALP_MUL_LINV  (REALAX)
  |- !x. ~(x = r0) ==> ((inv x) * x = r1)

REAL_MUL_SYM  (REALAX)
  |- !x y. x * y = y * x

real_neg  (REALAX)
  |- !g00023.
       -- g00023 = mk_real($treal_eq(treal_neg($@(dest_real g00023))))

real_of_hreal  (REALAX)
  |- !g00031.
       real_of_hreal g00031 = mk_real($treal_eq(treal_of_hreal g00031))

REAL_POS  (REALAX)
  |- !X. r0 < (real_of_hreal X)

REAL_SUP_ALLPOS  (REALAX)
  |- !P.
       (!x. P x ==> r0 < x) /\ (?x. P x) /\ (?z. !x. P x ==> x < z) ==>
       (?s. !y. (?x. P x /\ y < x) = y < s)

real_tybij  (REALAX)
  |- (!a. mk_real(dest_real a) = a) /\
     (!r. (\c. ?x. c = $treal_eq x)r = (dest_real(mk_real r) = r))

real_TY_DEF  (REALAX)
  |- ?rep. TYPE_DEFINITION(\c. ?x. c = $treal_eq x)rep

SUP_ALLPOS_LEMMA1  (REALAX)
  |- (!x. P x ==> r0 < x) ==>
     ((?x. P x /\ y < x) =
      (?X. P(real_of_hreal X) /\ y < (real_of_hreal X)))

SUP_ALLPOS_LEMMA2  (REALAX)
  |- P(real_of_hreal X) = (\h. P(real_of_hreal h))X

SUP_ALLPOS_LEMMA3  (REALAX)
  |- (!x. P x ==> r0 < x) /\ (?x. P x) /\ (?z. !x. P x ==> x < z) ==>
     (?X. (\h. P(real_of_hreal h))X) /\
     (?Y. !X. (\h. P(real_of_hreal h))X ==> X hreal_lt Y)

SUP_ALLPOS_LEMMA4  (REALAX)
  |- !y. ~r0 < y ==> (!x. y < (real_of_hreal x))

treal_0  (REALAX)
  |- treal_0 = hreal_1,hreal_1
```

```
treal_1  (REALAX)
  |- treal_1 = hreal_1 hreal_add hreal_1,hreal_1

TREAL_10  (REALAX)
  |- ~treal_1 treal_eq treal_0

treal_add  (REALAX)
  |- !x1 y1 x2 y2.
      (x1,y1) treal_add (x2,y2) = x1 hreal_add x2,y1 hreal_add y2

TREAL_ADD_ASSOC  (REALAX)
  |- !x y z. x treal_add (y treal_add z) = (x treal_add y) treal_add z

TREAL_ADD_LID  (REALAX)
  |- !x. (treal_0 treal_add x) treal_eq x

TREAL_ADD_LINV  (REALAX)
  |- !x. ((treal_neg x) treal_add x) treal_eq treal_0

TREAL_ADD_SYM  (REALAX)
  |- !x y. x treal_add y = y treal_add x

TREAL_ADD_WELLDEF  (REALAX)
  |- !x1 x2 y1 y2.
      x1 treal_eq x2 /\ y1 treal_eq y2 ==>
      (x1 treal_add y1) treal_eq (x2 treal_add y2)

TREAL_ADD_WELLDEFR  (REALAX)
  |- !x1 x2 y.
      x1 treal_eq x2 ==> (x1 treal_add y) treal_eq (x2 treal_add y)

TREAL_BIJ  (REALAX)
  |- (!h. hreal_of_treal(treal_of_hreal h) = h) /\
     (!r.
       treal_0 treal_lt r = (treal_of_hreal(hreal_of_treal r)) treal_eq r)

TREAL_BIJ_WELLDEF  (REALAX)
  |- !h i. h treal_eq i ==> (hreal_of_treal h = hreal_of_treal i)

treal_eq  (REALAX)
  |- !x1 y1 x2 y2.
      (x1,y1) treal_eq (x2,y2) = (x1 hreal_add y2 = x2 hreal_add y1)

TREAL_EQ_AP  (REALAX)
  |- !p q. (p = q) ==> p treal_eq q
```

```
TREAL_EQ_EQUIV  (REALAX)
  |- !p q. p treal_eq q = ($treal_eq p = $treal_eq q)

TREAL_EQ_REFL  (REALAX)
  |- !x. x treal_eq x

TREAL_EQ_SYM  (REALAX)
  |- !x y. x treal_eq y = y treal_eq x

TREAL_EQ_TRANS  (REALAX)
  |- !x y z. x treal_eq y /\ y treal_eq z ==> x treal_eq z

treal_inv  (REALAX)
  |- !x y.
       treal_inv(x,y) =
       ((x = y) =>
        treal_0 |
        (y hreal_lt x =>
         ((hreal_inv(x hreal_sub y)) hreal_add hreal_1,hreal_1) |
         (hreal_1,(hreal_inv(y hreal_sub x)) hreal_add hreal_1)))

TREAL_INV_WELLDEF  (REALAX)
  |- !x1 x2. x1 treal_eq x2 ==> (treal_inv x1) treal_eq (treal_inv x2)

TREAL_ISO  (REALAX)
  |- !h i. h hreal_lt i ==> (treal_of_hreal h) treal_lt (treal_of_hreal i)

TREAL_LDISTRIB  (REALAX)
  |- !x y z.
       x treal_mul (y treal_add z) =
       (x treal_mul y) treal_add (x treal_mul z)

treal_lt  (REALAX)
  |- !x1 y1 x2 y2.
       (x1,y1) treal_lt (x2,y2) =
       (x1 hreal_add y2) hreal_lt (x2 hreal_add y1)

TREAL_LT_ADD  (REALAX)
  |- !x y z. y treal_lt z ==> (x treal_add y) treal_lt (x treal_add z)

TREAL_LT_MUL  (REALAX)
  |- !x y.
       treal_0 treal_lt x /\ treal_0 treal_lt y ==>
       treal_0 treal_lt (x treal_mul y)

TREAL_LT_REFL  (REALAX)
  |- !x. ~x treal_lt x
```

```
TREAL_LT_TOTAL  (REALAX)
  |- !x y. x treal_eq y \/ x treal_lt y \/ y treal_lt x

TREAL_LT_TRANS  (REALAX)
  |- !x y z. x treal_lt y /\ y treal_lt z ==> x treal_lt z

TREAL_LT_WELLDEF  (REALAX)
  |- !x1 x2 y1 y2.
       x1 treal_eq x2 /\ y1 treal_eq y2 ==>
       (x1 treal_lt y1 = x2 treal_lt y2)

TREAL_LT_WELLDEFL  (REALAX)
  |- !x y1 y2. y1 treal_eq y2 ==> (x treal_lt y1 = x treal_lt y2)

TREAL_LT_WELLDEFR  (REALAX)
  |- !x1 x2 y. x1 treal_eq x2 ==> (x1 treal_lt y = x2 treal_lt y)

treal_mul  (REALAX)
  |- !x1 y1 x2 y2.
       (x1,y1) treal_mul (x2,y2) =
       (x1 hreal_mul x2) hreal_add (y1 hreal_mul y2),
       (x1 hreal_mul y2) hreal_add (y1 hreal_mul x2)

TREAL_MUL_ASSOC  (REALAX)
  |- !x y z. x treal_mul (y treal_mul z) = (x treal_mul y) treal_mul z

TREAL_MUL_LID  (REALAX)
  |- !x. (treal_1 treal_mul x) treal_eq x

TREAL_MUL_LINV  (REALAX)
  |- !x.
       ~x treal_eq treal_0 ==> ((treal_inv x) treal_mul x) treal_eq treal_1

TREAL_MUL_SYM  (REALAX)
  |- !x y. x treal_mul y = y treal_mul x

TREAL_MUL_WELLDEF  (REALAX)
  |- !x1 x2 y1 y2.
       x1 treal_eq x2 /\ y1 treal_eq y2 ==>
       (x1 treal_mul y1) treal_eq (x2 treal_mul y2)

TREAL_MUL_WELLDEFR  (REALAX)
  |- !x1 x2 y.
       x1 treal_eq x2 ==> (x1 treal_mul y) treal_eq (x2 treal_mul y)

treal_neg  (REALAX)
  |- !x y. treal_neg(x,y) = y,x
```

TREAL_NEG_WELLDEF (REALAX)
```
|- !x1 x2. x1 treal_eq x2 ==> (treal_neg x1) treal_eq (treal_neg x2)
```

treal_of_hreal (REALAX)
```
|- !x. treal_of_hreal x = x hreal_add hreal_1,hreal_1
```

2.4 REAL

abs (REAL)
```
|- !x. abs x = ((& 0) <= x => x | -- x)
```

REAL_ABS_0 (REAL)
```
|- abs(& 0) = & 0
```

REAL_ABS_1 (REAL)
```
|- abs(& 1) = & 1
```

REAL_ABS_ABS (REAL)
```
|- !x. abs(abs x) = abs x
```

REAL_ABS_BETWEEN (REAL)
```
|- !x y d. (& 0) < d /\ (x - d) < y /\ y < (x + d) = (abs(y - x)) < d
```

REAL_ABS_BETWEEN1 (REAL)
```
|- !x y z. x < z /\ (abs(y - x)) < (z - x) ==> y < z
```

REAL_ABS_BETWEEN2 (REAL)
```
|- !x0 x y0 y.
     x0 < y0 /\
     (abs(x - x0)) < ((y0 - x0) / (& 2)) /\
     (abs(y - y0)) < ((y0 - x0) / (& 2)) ==>
     x < y
```

REAL_ABS_BOUND (REAL)
```
|- !x y d. (abs(x - y)) < d ==> y < (x + d)
```

REAL_ABS_BOUNDS (REAL)
```
|- !x k. (abs x) <= k = (-- k) <= x /\ x <= k
```

REAL_ABS_CASES (REAL)
```
|- !x. (x = & 0) \/ (& 0) < (abs x)
```

REAL_ABS_CIRCLE (REAL)
```
|- !x y h. (abs h) < ((abs y) - (abs x)) ==> (abs(x + h)) < (abs y)
```

```
REALABS_DIV  (REAL)
  |- !y. ~(y = & 0) ==> (!x. abs(x / y) = (abs x) / (abs y))

REAL_ABS_INV  (REAL)
  |- !x. ~(x = & 0) ==> (abs(inv x) = inv(abs x))

REAL_ABS_LE  (REAL)
  |- !x. x <= (abs x)

REAL_ABS_LT_MUL2  (REAL)
  |- !w x y z. (abs w) < y /\ (abs x) < z ==> (abs(w * x)) < (y * z)

REAL_ABS_MUL  (REAL)
  |- !x y. abs(x * y) = (abs x) * (abs y)

REAL_ABS_N  (REAL)
  |- !n. abs(& n) = & n

REAL_ABS_NEG  (REAL)
  |- !x. abs(-- x) = abs x

REAL_ABS_NZ  (REAL)
  |- !x. ~(x = & 0) = (& 0) < (abs x)

REAL_ABS_POS  (REAL)
  |- !x. (& 0) <= (abs x)

REAL_ABS_POW2  (REAL)
  |- !x. abs(x pow 2) = x pow 2

REAL_ABS_REFL  (REAL)
  |- !x. (abs x = x) = (& 0) <= x

REAL_ABS_SIGN  (REAL)
  |- !x y. (abs(x - y)) < y ==> (& 0) < x

REAL_ABS_SIGN2  (REAL)
  |- !x y. (abs(x - y)) < (-- y) ==> x < (& 0)

REAL_ABS_STILLNZ  (REAL)
  |- !x y. (abs(x - y)) < (abs y) ==> ~(x = & 0)

REAL_ABS_SUB  (REAL)
  |- !x y. abs(x - y) = abs(y - x)

REAL_ABS_SUB_ABS  (REAL)
  |- !x y. (abs((abs x) - (abs y))) <= (abs(x - y))
```

```
ABS_SUM (REAL)
  |- !f m n. (abs(Sum(m,n)f)) <= (Sum(m,n)(\n'. abs(f n')))

REAL_ABS_TRIANGLE (REAL)
  |- !x y. (abs(x + y)) <= ((abs x) + (abs y))

REAL_ABS_ZERO (REAL)
  |- !x. (abs x = & 0) = (x = & 0)

pow (REAL)
  |- (!x. x pow 0 = & 1) /\ (!x n. x pow (SUC n) = x * (x pow n))

POW_0 (REAL)
  |- !n. (& 0) pow (SUC n) = & 0

POW_1 (REAL)
  |- !x. x pow 1 = x

POW_2 (REAL)
  |- !x. x pow 2 = x * x

POW_2_LE1 (REAL)
  |- !n. (& 1) <= ((& 2) pow n)

POW_2_LT (REAL)
  |- !n. (& n) < ((& 2) pow n)

POW_ABS (REAL)
  |- !c n. (abs c) pow n = abs(c pow n)

POW_ADD (REAL)
  |- !c m n. c pow (m num_add n) = (c pow m) * (c pow n)

POW_INV (REAL)
  |- !c. ~(c = & 0) ==> (!n. inv(c pow n) = (inv c) pow n)

POW_LE (REAL)
  |- !n x y. (& 0) <= x /\ x <= y ==> (x pow n) <= (y pow n)

POW_M1 (REAL)
  |- !n. abs((--(& 1)) pow n) = & 1

POW_MINUS1 (REAL)
  |- !n. (--(& 1)) pow (2 num_mul n) = & 1

POW_MUL (REAL)
  |- !n x y. (x * y) pow n = (x pow n) * (y pow n)
```

```
POW_NZ  (REAL)
   |- !c n. ~(c = & 0) ==> ~(c pow n = & 0)

POW_PLUS1  (REAL)
   |- !e. (& 0) < e ==> (!n. ((& 1) + ((& n) * e)) <= (((& 1) + e) pow n))

POW_POS  (REAL)
   |- !x. (& 0) <= x ==> (!n. (& 0) <= (x pow n))

POW_POS_LT  (REAL)
   |- !x n. (& 0) < x ==> (& 0) < (x pow (SUC n))

REAL  (REAL)
   |- !n. &(SUC n) = (& n) + (& 1)

REAL_0  (REAL)
   |- r0 = & 0

REAL_1  (REAL)
   |- r1 = & 1

REAL_10  (REAL)
   |- ~(& 1 = & 0)

REAL_ADD  (REAL)
   |- !m n. (& m) + (& n) = &(m num_add n)

REAL_ADD2_SUB2  (REAL)
   |- !a b c d. (a + b) - (c + d) = (a - c) + (b - d)

REAL_ADD_ASSOC  (REAL)
   |- !x y z. x + (y + z) = (x + y) + z

REAL_ADD_LID  (REAL)
   |- !x. (& 0) + x = x

REAL_ADD_LID_UNIQ  (REAL)
   |- !x y. (x + y = y) = (x = & 0)

REAL_ADD_LINV  (REAL)
   |- !x. (-- x) + x = & 0

REAL_ADD_RID  (REAL)
   |- !x. x + (& 0) = x

REAL_ADD_RID_UNIQ  (REAL)
   |- !x y. (x + y = x) = (y = & 0)
```

```
REAL_ADD_RINV  (REAL)
  |- !x. x + (-- x) = & 0

REAL_ADD_SUB  (REAL)
  |- !x y. (x + y) - x = y

REAL_ADD_SUB2  (REAL)
  |- !x y. x - (x + y) = -- y

REAL_ADD_SYM  (REAL)
  |- !x y. x + y = y + x

REAL_ARCH  (REAL)
  |- !x. (& 0) < x ==> (!y. ?n. y < ((& n) * x))

REAL_ARCH_LEAST  (REAL)
  |- !y.
      (& 0) < y ==>
      (!x. (& 0) <= x ==> (?n. ((& n) * y) <= x /\ x < ((&(SUC n)) * y)))

REAL_DIFFSQ  (REAL)
  |- !x y. (x + y) * (x - y) = (x * x) - (y * y)

real_div  (REAL)
  |- !x y. x / y = x * (inv y)

REAL_DIV_LMUL  (REAL)
  |- !x y. ~(y = & 0) ==> (y * (x / y) = x)

REAL_DIV_LZERO  (REAL)
  |- !x. (& 0) / x = & 0

REAL_DIV_MUL2  (REAL)
  |- !x z. ~(x = & 0) /\ ~(z = & 0) ==> (!y. y / z = (x * y) / (x * z))

REAL_DIV_REFL  (REAL)
  |- !x. ~(x = & 0) ==> (x / x = & 1)

REAL_DIV_RMUL  (REAL)
  |- !x y. ~(y = & 0) ==> ((x / y) * y = x)

REAL_DOUBLE  (REAL)
  |- !x. x + x = (& 2) * x

REAL_DOWN  (REAL)
  |- !x. (& 0) < x ==> (?y. (& 0) < y /\ y < x)
```

```
REAL_DOWN2  (REAL)
  |- !x y. (& 0) < x /\ (& 0) < y ==> (?z. (& 0) < z /\ z < x /\ z < y)

REAL_ENTIRE  (REAL)
  |- !x y. (x * y = & 0) = (x = & 0) \/ (y = & 0)

REAL_EQ_IMP_LE  (REAL)
  |- !x y. (x = y) ==> x <= y

REAL_EQ_LADD  (REAL)
  |- !x y z. (x + y = x + z) = (y = z)

REAL_EQ_LMUL  (REAL)
  |- !x y z. (x * y = x * z) = (x = & 0) \/ (y = z)

REAL_EQ_LMUL2  (REAL)
  |- !x y z. ~(x = & 0) ==> ((y = z) = (x * y = x * z))

REAL_EQ_LMUL_IMP  (REAL)
  |- !x y z. ~(x = & 0) /\ (x * y = x * z) ==> (y = z)

REAL_EQ_NEG  (REAL)
  |- !x y. (-- x = -- y) = (x = y)

REAL_EQ_RADD  (REAL)
  |- !x y z. (x + z = y + z) = (x = y)

REAL_EQ_RMUL  (REAL)
  |- !x y z. (x * z = y * z) = (z = & 0) \/ (x = y)

REAL_EQ_RMUL_IMP  (REAL)
  |- !x y z. ~(z = & 0) /\ (x * z = y * z) ==> (x = y)

REAL_EQ_SUB_LADD  (REAL)
  |- !x y z. (x = y - z) = (x + z = y)

REAL_EQ_SUB_RADD  (REAL)
  |- !x y z. (x - y = z) = (x = z + y)

REAL_FACT_NZ  (REAL)
  |- !n. ~(&(FACT n) = & 0)

real_ge  (REAL)
  |- !x y. x >= y = y <= x

real_gt  (REAL)
  |- !x y. x > y = y < x
```

```
REAL_HALF_DOUBLE  (REAL)
  |- !x. (x / (& 2)) + (x / (& 2)) = x

REAL_INJ  (REAL)
  |- !m n. (& m = & n) = (m = n)

REAL_INV1  (REAL)
  |- inv(& 1) = & 1

REAL_INVINV  (REAL)
  |- !x. ~(x = & 0) ==> (inv(inv x) = x)

REAL_INV_1OVER  (REAL)
  |- !x. inv x = (& 1) / x

REAL_INV_LT1  (REAL)
  |- !x. (& 0) < x /\ x < (& 1) ==> (& 1) < (inv x)

REAL_INV_MUL  (REAL)
  |- !x y. ~(x = & 0) /\ ~(y = & 0) ==> (inv(x * y) = (inv x) * (inv y))

REAL_INV_NZ  (REAL)
  |- !x. ~(x = & 0) ==> ~(inv x = & 0)

REAL_INV_POS  (REAL)
  |- !x. (& 0) < x ==> (& 0) < (inv x)

REAL_LDISTRIB  (REAL)
  |- !x y z. x * (y + z) = (x * y) + (x * z)

REAL_LE  (REAL)
  |- !m n. (& m) <= (& n) = m num_le n

real_le  (REAL)
  |- !x y. x <= y = ~y < x

REAL_LE1_POW2  (REAL)
  |- !x. (& 1) <= x ==> (& 1) <= (x pow 2)

REAL_LET_ADD  (REAL)
  |- !x y. (& 0) <= x /\ (& 0) < y ==> (& 0) < (x + y)

REAL_LET_ADD2  (REAL)
  |- !w x y z. w <= x /\ y < z ==> (w + y) < (x + z)

REAL_LET_ANTISYM  (REAL)
  |- !x y. ~(x < y /\ y <= x)
```

```
REAL_LET_TOTAL  (REAL)
   |- !x y. x <= y \/ y < x

REAL_LET_TRANS  (REAL)
   |- !x y z. x <= y /\ y < z ==> x < z

REAL_LE_01  (REAL)
   |- (& 0) <= (& 1)

REAL_LE_ADD  (REAL)
   |- !x y. (& 0) <= x /\ (& 0) <= y ==> (& 0) <= (x + y)

REAL_LE_ADD2  (REAL)
   |- !w x y z. w <= x /\ y <= z ==> (w + y) <= (x + z)

REAL_LE_ADDL  (REAL)
   |- !x y. y <= (x + y) = (& 0) <= x

REAL_LE_ADDR  (REAL)
   |- !x y. x <= (x + y) = (& 0) <= y

REAL_LE_ANTISYM  (REAL)
   |- !x y. x <= y /\ y <= x = (x = y)

REAL_LE_DOUBLE  (REAL)
   |- !x. (& 0) <= (x + x) = (& 0) <= x

REAL_LE_LADD  (REAL)
   |- !x y z. (x + y) <= (x + z) = y <= z

REAL_LE_LDIV  (REAL)
   |- !x y z. (& 0) < x /\ y <= (z * x) ==> (y / x) <= z

REAL_LE_LMUL  (REAL)
   |- !x y z. (& 0) < x ==> ((x * y) <= (x * z) = y <= z)

REAL_LE_LMUL_IMP  (REAL)
   |- !x y z. (& 0) <= x /\ y <= z ==> (x * y) <= (x * z)

REAL_LE_LT  (REAL)
   |- !x y. x <= y = x < y \/ (x = y)

REAL_LE_MUL  (REAL)
   |- !x y. (& 0) <= x /\ (& 0) <= y ==> (& 0) <= (x * y)
```

```
REAL_LE_MUL2 (REAL)
  |- !x1 x2 y1 y2.
       (& 0) <= x1 /\ (& 0) <= y1 /\ x1 <= x2 /\ y1 <= y2 ==>
       (x1 * y1) <= (x2 * y2)

REAL_LE_NEG (REAL)
  |- !x y. (-- x) <= (-- y) = y <= x

REAL_LE_NEGL (REAL)
  |- !x. (-- x) <= x = (& 0) <= x

REAL_LE_NEGR (REAL)
  |- !x. x <= (-- x) = x <= (& 0)

REAL_LE_NEGTOTAL (REAL)
  |- !x. (& 0) <= x \/ (& 0) <= (-- x)

REAL_LE_POW2 (REAL)
  |- !x. (& 0) <= (x pow 2)

REAL_LE_RADD (REAL)
  |- !x y z. (x + z) <= (y + z) = x <= y

REAL_LE_RDIV (REAL)
  |- !x y z. (& 0) < x /\ (y * x) <= z ==> y <= (z / x)

REAL_LE_REFL (REAL)
  |- !x. x <= x

REAL_LE_RMUL (REAL)
  |- !x y z. (& 0) < z ==> ((x * z) <= (y * z) = x <= y)

REAL_LE_RMUL_IMP (REAL)
  |- !x y z. (& 0) <= x /\ y <= z ==> (y * x) <= (z * x)

REAL_LE_SQUARE (REAL)
  |- !x. (& 0) <= (x * x)

REAL_LE_SUB_LADD (REAL)
  |- !x y z. x <= (y - z) = (x + z) <= y

REAL_LE_SUB_RADD (REAL)
  |- !x y z. (x - y) <= z = x <= (z + y)

REAL_LE_TOTAL (REAL)
  |- !x y. x <= y \/ y <= x
```

```
REAL_LE_TRANS  (REAL)
   |- !x y z. x <= y /\ y <= z ==> x <= z

REAL_LINV_UNIQ  (REAL)
   |- !x y. (x * y = & 1) ==> (x = inv y)

REAL_LNEG_UNIQ  (REAL)
   |- !x y. (x + y = & 0) = (x = -- y)

REAL_LT  (REAL)
   |- !m n. (& m) < (& n) = m num_lt n

REAL_LT1_POW2  (REAL)
   |- !x. (& 1) < x ==> (& 1) < (x pow 2)

REAL_LTE_ADD  (REAL)
   |- !x y. (& 0) < x /\ (& 0) <= y ==> (& 0) < (x + y)

REAL_LTE_ADD2  (REAL)
   |- !w x y z. w < x /\ y <= z ==> (w + y) < (x + z)

REAL_LTE_ANTSYM  (REAL)
   |- !x y. ~(x <= y /\ y < x)

REAL_LTE_TOTAL  (REAL)
   |- !x y. x < y \/ y <= x

REAL_LTE_TRANS  (REAL)
   |- !x y z. x < y /\ y <= z ==> x < z

REAL_LT_01  (REAL)
   |- (& 0) < (& 1)

REAL_LT_1  (REAL)
   |- !x y. (& 0) <= x /\ x < y ==> (x / y) < (& 1)

REAL_LT_ADD  (REAL)
   |- !x y. (& 0) < x /\ (& 0) < y ==> (& 0) < (x + y)

REAL_LT_ADD1  (REAL)
   |- !x y. x <= y ==> x < (y + (& 1))

REAL_LT_ADD2  (REAL)
   |- !w x y z. w < x /\ y < z ==> (w + y) < (x + z)

REAL_LT_ADDL  (REAL)
   |- !x y. y < (x + y) = (& 0) < x
```

```
REAL_LT_ADDNEG (REAL)
  |- !x y z. y < (x + (-- z)) = (y + z) < x

REAL_LT_ADDNEG2 (REAL)
  |- !x y z. (x + (-- y)) < z = x < (z + y)

REAL_LT_ADDR (REAL)
  |- !x y. x < (x + y) = (& 0) < y

REAL_LT_ADD_SUB (REAL)
  |- !x y z. (x + y) < z = x < (z - y)

REAL_LT_ANTISYM (REAL)
  |- !x y. ~(x < y /\ y < x)

REAL_LT_FRACTION (REAL)
  |- !n d. 1 num_lt n ==> ((d / (& n)) < d = (& 0) < d)

REAL_LT_FRACTION_0 (REAL)
  |- !n d. ~(n = 0) ==> ((& 0) < (d / (& n)) = (& 0) < d)

REAL_LT_GT (REAL)
  |- !x y. x < y ==> ~y < x

REAL_LT_HALF1 (REAL)
  |- !d. (& 0) < (d / (& 2)) = (& 0) < d

REAL_LT_HALF2 (REAL)
  |- !d. (d / (& 2)) < d = (& 0) < d

REAL_LT_IADD (REAL)
  |- !x y z. y < z ==> (x + y) < (x + z)

REAL_LT_IMP_LE (REAL)
  |- !x y. x < y ==> x <= y

REAL_LT_IMP_NE (REAL)
  |- !x y. x < y ==> ~(x = y)

REAL_LT_INV (REAL)
  |- !x y. (& 0) < x /\ x < y ==> (inv y) < (inv x)

REAL_LT_LADD (REAL)
  |- !x y z. (x + y) < (x + z) = y < z

REAL_LT_LE (REAL)
  |- !x y. x < y = x <= y /\ ~(x = y)
```

REAL_LT_LMUL (REAL)
```
|- !x y z. (& 0) < x ==> ((x * y) < (x * z) = y < z)
```

REAL_LT_LMUL_0 (REAL)
```
|- !x y. (& 0) < x ==> ((& 0) < (x * y) = (& 0) < y)
```

REAL_LT_LMUL_IMP (REAL)
```
|- !x y z. y < z /\ (& 0) < x ==> (x * y) < (x * z)
```

REAL_LT_MUL (REAL)
```
|- !x y. (& 0) < x /\ (& 0) < y ==> (& 0) < (x * y)
```

REAL_LT_MUL2 (REAL)
```
|- !x1 x2 y1 y2.
     (& 0) <= x1 /\ (& 0) <= y1 /\ x1 < x2 /\ y1 < y2 ==>
     (x1 * y1) < (x2 * y2)
```

REAL_LT_MULTIPLE (REAL)
```
|- !n d. 1 num_lt n ==> (d < ((& n) * d) = (& 0) < d)
```

REAL_LT_NEG (REAL)
```
|- !x y. (-- x) < (-- y) = y < x
```

REAL_LT_NEGTOTAL (REAL)
```
|- !x. (x = & 0) \/ (& 0) < x \/ (& 0) < (-- x)
```

REAL_LT_NZ (REAL)
```
|- !n. ~(& n = & 0) = (& 0) < (& n)
```

REAL_LT_RADD (REAL)
```
|- !x y z. (x + z) < (y + z) = x < y
```

REAL_LT_RDIV (REAL)
```
|- !x y z. (& 0) < z ==> ((x / z) < (y / z) = x < y)
```

REAL_LT_RDIV_0 (REAL)
```
|- !y z. (& 0) < z ==> ((& 0) < (y / z) = (& 0) < y)
```

REAL_LT_REFL (REAL)
```
|- !x. ~x < x
```

REAL_LT_RMUL (REAL)
```
|- !x y z. (& 0) < z ==> ((x * z) < (y * z) = x < y)
```

REAL_LT_RMUL_0 (REAL)
```
|- !x y. (& 0) < y ==> ((& 0) < (x * y) = (& 0) < x)
```

```
REAL_LT_RMUL_IMP  (REAL)
  |- !x y z. x < y /\ (& 0) < z ==> (x * z) < (y * z)

REAL_LT_SUB_LADD  (REAL)
  |- !x y z. x < (y - z) = (x + z) < y

REAL_LT_SUB_RADD  (REAL)
  |- !x y z. (x - y) < z = x < (z + y)

REAL_LT_TOTAL  (REAL)
  |- !x y. (x = y) \/ x < y \/ y < x

REAL_LT_TRANS  (REAL)
  |- !x y z. x < y /\ y < z ==> x < z

REAL_MEAN  (REAL)
  |- !x y. x < y ==> (?z. x < z /\ z < y)

REAL_MUL  (REAL)
  |- !m n. (& m) * (& n) = &(m num_mul n)

REAL_MUL_ASSOC  (REAL)
  |- !x y z. x * (y * z) = (x * y) * z

REAL_MUL_LID  (REAL)
  |- !x. (& 1) * x = x

REAL_MUL_LINV  (REAL)
  |- !x. ~(x = & 0) ==> ((inv x) * x = & 1)

REAL_MUL_LZERO  (REAL)
  |- !x. (& 0) * x = & 0

REAL_MUL_RID  (REAL)
  |- !x. x * (& 1) = x

REAL_MUL_RINV  (REAL)
  |- !x. ~(x = & 0) ==> (x * (inv x) = & 1)

REAL_MUL_RZERO  (REAL)
  |- !x. x * (& 0) = & 0

REAL_MUL_SYM  (REAL)
  |- !x y. x * y = y * x

REAL_NEGNEG  (REAL)
  |- !x. --(-- x) = x
```

```
REAL_NEG_0  (REAL)
   |- --(& 0) = & 0

REAL_NEG_ADD  (REAL)
   |- !x y. --(x + y) = (-- x) + (-- y)

REAL_NEG_EQ  (REAL)
   |- !x y. (-- x = y) = (x = -- y)

REAL_NEG_EQ0  (REAL)
   |- !x. (-- x = & 0) = (x = & 0)

REAL_NEG_GE0  (REAL)
   |- !x. (& 0) <= (-- x) = x <= (& 0)

REAL_NEG_GT0  (REAL)
   |- !x. (& 0) < (-- x) = x < (& 0)

REAL_NEG_INV  (REAL)
   |- !x. ~(x = & 0) ==> (--(inv x) = inv(-- x))

REAL_NEG_LE0  (REAL)
   |- !x. (-- x) <= (& 0) = (& 0) <= x

REAL_NEG_LMUL  (REAL)
   |- !x y. --(x * y) = (-- x) * y

REAL_NEG_LT0  (REAL)
   |- !x. (-- x) < (& 0) = (& 0) < x

REAL_NEG_MINUS1  (REAL)
   |- !x. -- x = (--(& 1)) * x

REAL_NEG_MUL2  (REAL)
   |- !x y. (-- x) * (-- y) = x * y

REAL_NEG_RMUL  (REAL)
   |- !x y. --(x * y) = x * (-- y)

REAL_NEG_SUB  (REAL)
   |- !x y. --(x - y) = y - x

REAL_NOT_LE  (REAL)
   |- !x y. ~x <= y = y < x

REAL_NOT_LT  (REAL)
   |- !x y. ~x < y = y <= x
```

REAL_NZ_IMP_LT (REAL)
```
|- !n. ~(n = 0) ==> (& 0) < (& n)
```

real_of_num (REAL)
```
|- (& 0 = r0) /\ (!n. &(SUC n) = (& n) + r1)
```

REAL_OVER1 (REAL)
```
|- !x. x / (& 1) = x
```

REAL_POASQ (REAL)
```
|- !x. (& 0) < (x * x) = ~(x = & 0)
```

REAL_POS (REAL)
```
|- !n. (& 0) <= (& n)
```

REAL_POS_NZ (REAL)
```
|- !x. (& 0) < x ==> ~(x = & 0)
```

REAL_POW2_ABS (REAL)
```
|- !x. (abs x) pow 2 = x pow 2
```

REAL_RDISTRIB (REAL)
```
|- !x y z. (x + y) * z = (x * z) + (y * z)
```

REAL_RINV_UNIQ (REAL)
```
|- !x y. (x * y = & 1) ==> (y = inv x)
```

REAL_RNEG_UNIQ (REAL)
```
|- !x y. (x + y = & 0) = (y = -- x)
```

real_sub (REAL)
```
|- !x y. x - y = x + (-- y)
```

REAL_SUB_0 (REAL)
```
|- !x y. (x - y = & 0) = (x = y)
```

REAL_SUB_ABS (REAL)
```
|- !x y. ((abs x) - (abs y)) <= (abs(x - y))
```

REAL_SUB_ADD (REAL)
```
|- !x y. (x - y) + y = x
```

REAL_SUB_ADD2 (REAL)
```
|- !x y. y + (x - y) = x
```

```
REAL_SUB_INV2  (REAL)
   |- !x y.
        ~(x = & 0) /\ ~(y = & 0) ==> ((inv x) - (inv y) = (y - x) / (x * y))

REAL_SUB_LDISTRIB  (REAL)
   |- !x y z. x * (y - z) = (x * y) - (x * z)

REAL_SUB_LE  (REAL)
   |- !x y. (& 0) <= (x - y) = y <= x

REAL_SUB_LNEG  (REAL)
   |- !x y. (-- x) - y = --(x + y)

REAL_SUB_LT  (REAL)
   |- !x y. (& 0) < (x - y) = y < x

REAL_SUB_LZERO  (REAL)
   |- !x. (& 0) - x = -- x

REAL_SUB_NEG2  (REAL)
   |- !x y. (-- x) - (-- y) = y - x

REAL_SUB_RDISTRIB  (REAL)
   |- !x y z. (x - y) * z = (x * z) - (y * z)

REAL_SUB_REFL  (REAL)
   |- !x. x - x = & 0

REAL_SUB_RNEG  (REAL)
   |- !x y. x - (-- y) = x + y

REAL_SUB_RZERO  (REAL)
   |- !x. x - (& 0) = x

REAL_SUB_SUB  (REAL)
   |- !x y. (x - y) - x = -- y

REAL_SUB_SUB2  (REAL)
   |- !x y. x - (x - y) = y

REAL_SUB_TRIANGLE  (REAL)
   |- !a b c. (a - b) + (b - c) = a - c

REAL_SUMSQ  (REAL)
   |- !x y. ((x * x) + (y * y) = & 0) = (x = & 0) /\ (y = & 0)
```

```
REAL_SUP   (REAL)
  |- !P.
       (?x. P x) /\ (?z. !x. P x ==> x < z) ==>
       (!y. (?x. P x /\ y < x) = y < (sup P))

REAL_SUP_ALLPOS   (REAL)
  |- !P.
       (!x. P x ==> (& 0) < x) /\ (?x. P x) /\ (?z. !x. P x ==> x < z) ==>
       (?s. !y. (?x. P x /\ y < x) = y < s)

REAL_SUP_EXISTS   (REAL)
  |- !P.
       (?x. P x) /\ (?z. !x. P x ==> x < z) ==>
       (?s. !y. (?x. P x /\ y < x) = y < s)

REAL_SUP_LE   (REAL)
  |- !P.
       (?x. P x) /\ (?z. !x. P x ==> x <= z) ==>
       (!y. (?x. P x /\ y < x) = y < (sup P))

REAL_SUP_SOMEPOS   (REAL)
  |- !P.
       (?x. P x /\ (& 0) < x) /\ (?z. !x. P x ==> x < z) ==>
       (?s. !y. (?x. P x /\ y < x) = y < s)

REAL_SUP_UBOUND   (REAL)
  |- !P.
       (?x. P x) /\ (?z. !x. P x ==> x < z) ==> (!y. P y ==> y <= (sup P))

REAL_SUP_UBOUND_LE   (REAL)
  |- !P.
       (?x. P x) /\ (?z. !x. P x ==> x <= z) ==> (!y. P y ==> y <= (sup P))

SETOK_LE_LT   (REAL)
  |- !P.
       (?x. P x) /\ (?z. !x. P x ==> x <= z) =
       (?x. P x) /\ (?z. !x. P x ==> x < z)

Sum   (REAL)
  |- (Sum(n,0)f = & 0) /\ (Sum(n,SUC m)f = (Sum(n,m)f) + (f(n num_add m)))

sum   (REAL)
  |- (!n f. sum n 0 f = & 0) /\
     (!n m f. sum n(SUC m)f = (sum n m f) + (f(n num_add m)))

SUM_0   (REAL)
  |- !m n. Sum(m,n)(\r. & 0) = & 0
```

```
SUM_1  (REAL)
   |- !f n. Sum(n,1)f = f n

SUM_2  (REAL)
   |- !f n. Sum(n,2)f = (f n) + (f(n num_add 1))

SUM_ABS  (REAL)
   |- !f m n. abs(Sum(m,n)(\m. abs(f m))) = Sum(m,n)(\m. abs(f m))

SUM_ABS_LE  (REAL)
   |- !f m n. (abs(Sum(m,n)f)) <= (Sum(m,n)(\n'. abs(f n')))

SUM_ADD  (REAL)
   |- !f g m n. Sum(m,n)(\n'. (f n') + (g n')) = (Sum(m,n)f) + (Sum(m,n)g)

SUM_BOUND  (REAL)
   |- !f K m n.
        (!p. m num_le p /\ p num_lt (m num_add n) ==> (f p) <= K) ==>
        (Sum(m,n)f) <= ((& n) * K)

SUM_CMUL  (REAL)
   |- !f c m n. Sum(m,n)(\n'. c * (f n')) = c * (Sum(m,n)f)

Sum_DEF  (REAL)
   |- !m n f. Sum(m,n)f = sum m n f

SUM_DIFF  (REAL)
   |- !f m n. Sum(m,n)f = (Sum(0,m num_add n)f) - (Sum(0,m)f)

SUM_EQ  (REAL)
   |- !N. (!n. N num_le n ==> (f n = g n)) ==> (!n. Sum(N,n)f = Sum(N,n)g)

SUM_EQ_GEN  (REAL)
   |- !f g m n.
        (!r. m num_le r /\ r num_lt (n num_add m) ==> (f r = g r)) ==>
        (Sum(m,n)f = Sum(m,n)g)

SUM_GROUP  (REAL)
   |- !n k f. Sum(0,n)(\m. Sum(m num_mul k,k)f) = Sum(0,n num_mul k)f

SUM_LE  (REAL)
   |- !f g N.
        (!n. N num_le n ==> (f n) <= (g n)) ==>
        (!n. (Sum(N,n)f) <= (Sum(N,n)g))

SUM_NSUB  (REAL)
   |- !n f c. (Sum(0,n)f) - ((& n) * c) = Sum(0,n)(\p. (f p) - c)
```

```
SUM_OFFSET (REAL)
  |- !f n k.
       Sum(0,n)(\m. f(m num_add k)) = (Sum(0,n num_add k)f) - (Sum(0,k)f)

SUM_PERMUTE_0 (REAL)
  |- !n p.
       (!y. y num_lt n ==> (?! x. x num_lt n /\ (p x = y))) ==>
       (!f. Sum(0,n)(\n'. f(p n')) = Sum(0,n)f)

SUM_POS (REAL)
  |- !f. (!n. (& 0) <= (f n)) ==> (!m n. (& 0) <= (Sum(m,n)f))

SUM_POS_GEN (REAL)
  |- !f m.
       (!n. m num_le n ==> (& 0) <= (f n)) ==> (!n. (& 0) <= (Sum(m,n)f))

SUM_REINDEX (REAL)
  |- !f m k n. Sum(m num_add k,n)f = Sum(m,n)(\r. f(r num_add k))

SUM_SUBST (REAL)
  |- !f g m n.
       (!p. m num_le p /\ p num_lt (m num_add n) ==> (f p = g p)) ==>
       (Sum(m,n)f = Sum(m,n)g)

SUM_TWO (REAL)
  |- !f n p. (Sum(0,n)f) + (Sum(n,p)f) = Sum(0,n num_add p)f

SUM_ZERO (REAL)
  |- !f N.
       (!n. n num_ge N ==> (f n = & 0)) ==>
       (!m n. m num_ge N ==> (Sum(m,n)f = & 0))

sup (REAL)
  |- !P. sup P = (@s. !y. (?x. P x /\ y < x) = y < s)

SUP_LEMMA1 (REAL)
  |- !d.
       (!y. (?x. (\x. P(x + d))x /\ y < x) = y < s) ==>
       (!y. (?x. P x /\ y < x) = y < (s + d))

SUP_LEMMA2 (REAL)
  |- (?x. P x) ==> (?d x. (\x. P(x + d))x /\ (& 0) < x)

SUP_LEMMA3 (REAL)
  |- !d. (?z. !x. P x ==> x < z) ==> (?z. !x. (\x. P(x + d))x ==> x < z)
```

2.5 TOPOLOGY

```
ball (TOPOLOGY)
  |- !m x e. B m(x,e) = (\y. (dist m(x,y)) < e)

BALL_NEIGH (TOPOLOGY)
  |- !m x e. (& 0) < e ==> neigh(mtop m)(B m(x,e),x)

BALL_OPEN (TOPOLOGY)
  |- !m x e. (& 0) < e ==> open(mtop m)(B m(x,e))

closed (TOPOLOGY)
  |- !L S. closed L S = open L(compl S)

CLOSED_LIMPT (TOPOLOGY)
  |- !top S. closed top S = (!x. limpt top x S ==> S x)

compl (TOPOLOGY)
  |- !S. compl S = (\x. ~S x)

COMPL_MEM (TOPOLOGY)
  |- !S x. S x = ~compl S x

intersect_def (TOPOLOGY)
  |- !P Q. P intersect Q = (\x. P x /\ Q x)

ismet (TOPOLOGY)
  |- !m.
      ismet m =
      (!x y. (m(x,y) = & 0) = (x = y)) /\
      (!x y z. (m(y,z)) <= ((m(x,y)) + (m(x,z))))

ISMET_R1 (TOPOLOGY)
  |- ismet(\(x,y). abs(y - x))

istopology (TOPOLOGY)
  |- !L.
      istopology L =
      L null /\
      L universe /\
      (!a b. L a /\ L b ==> L(a intersect b)) /\
      (!P. P subset L ==> L(Union P))
```

```
limpt (TOPOLOGY)
  |- !top x S.
      limpt top x S =
      (!N. neigh top(N,x) ==> (?y. ~(x = y) /\ S y /\ N y))

METRIC_ISMET (TOPOLOGY)
  |- !m. ismet(dist m)

METRIC_NZ (TOPOLOGY)
  |- !m x y. ~(x = y) ==> (& 0) < (dist m(x,y))

METRIC_POS (TOPOLOGY)
  |- !m x y. (& 0) <= (dist m(x,y))

METRIC_SAME (TOPOLOGY)
  |- !m x. dist m(x,x) = & 0

METRIC_SYM (TOPOLOGY)
  |- !m x y. dist m(x,y) = dist m(y,x)

METRIC_TRIANGLE (TOPOLOGY)
  |- !m x y z. (dist m(x,z)) <= ((dist m(x,y)) + (dist m(y,z)))

metric_tybij (TOPOLOGY)
  |- (!a. metric(dist a) = a) /\ (!r. ismet r = (dist(metric r) = r))

metric_TY_DEF (TOPOLOGY)
  |- ?rep. TYPE_DEFINITION ismet rep

METRIC_ZERO (TOPOLOGY)
  |- !m x y. (dist m(x,y) = & 0) = (x = y)

mr1 (TOPOLOGY)
  |- mr1 = metric(\(x,y). abs(y - x))

MR1_ADD (TOPOLOGY)
  |- !x d. dist mr1(x,x + d) = abs d

MR1_ADD_LT (TOPOLOGY)
  |- !x d. (& 0) < d ==> (dist mr1(x,x + d) = d)

MR1_ADD_POS (TOPOLOGY)
  |- !x d. (& 0) <= d ==> (dist mr1(x,x + d) = d)

MR1_BETWEEN1 (TOPOLOGY)
  |- !x y z. x < z /\ (dist mr1(x,y)) < (z - x) ==> y < z
```

```
MR1_DEF  (TOPOLOGY)
   |- !x y. dist mr1(x,y) = abs(y - x)

MR1_LIMPT  (TOPOLOGY)
   |- !x. limpt(mtop mr1)x universe

MR1_SUB  (TOPOLOGY)
   |- !x d. dist mr1(x,x - d) = abs d

MR1_SUB_LE  (TOPOLOGY)
   |- !x d. (& 0) <= d ==> (dist mr1(x,x - d) = d)

MR1_SUB_LT  (TOPOLOGY)
   |- !x d. (& 0) < d ==> (dist mr1(x,x - d) = d)

mtop  (TOPOLOGY)
   |- !m.
        mtop m =
        topology
        (\S. !x. S x ==> (?e. (& 0) < e /\ (!y. (dist m(x,y)) < e ==> S y)))

mtop_istopology  (TOPOLOGY)
   |- !m.
        istopology
        (\S. !x. S x ==> (?e. (& 0) < e /\ (!y. (dist m(x,y)) < e ==> S y)))

MTOP_LIMPT  (TOPOLOGY)
   |- !m x S.
        limpt(mtop m)x S =
        (!e. (& 0) < e ==> (?y. ~(x = y) /\ S y /\ (dist m(x,y)) < e))

MTOP_OPEN  (TOPOLOGY)
   |- !m.
        open(mtop m)S =
        (!x. S x ==> (?e. (& 0) < e /\ (!y. (dist m(x,y)) < e ==> S y)))

neigh  (TOPOLOGY)
   |- !top N x. neigh top(N,x) = (?P. open top P /\ P subset N /\ P x)

null  (TOPOLOGY)
   |- null = (\x. F)

OPEN_NEIGH  (TOPOLOGY)
   |- !S top. open top S = (!x. S x ==> (?N. neigh top(N,x) /\ N subset S))

OPEN_OWN_NEIGH  (TOPOLOGY)
   |- !S top x. open top S /\ S x ==> neigh top(S,x)
```

```
OPEN_SUBOPEN  (TOPOLOGY)
  |- !S top.
       open top S = (!x. S x ==> (?P. P x /\ open top P /\ P subset S))

OPEN_UNOPEN  (TOPOLOGY)
  |- !S top. open top S = (Union(\P. open top P /\ P subset S) = S)

subset  (TOPOLOGY)
  |- !P Q. P subset Q = (!x. P x ==> Q x)

SUBSET_ANTISYM  (TOPOLOGY)
  |- !P Q. P subset Q /\ Q subset P = (P = Q)

SUBSET_REFL  (TOPOLOGY)
  |- !S. S subset S

SUBSET_TRANS  (TOPOLOGY)
  |- !P Q R. P subset Q /\ Q subset R ==> P subset R

TOPOLOGY  (TOPOLOGY)
  |- !L.
       open L null /\
       open L universe /\
       (!x y. open L x /\ open L y ==> open L(x intersect y)) /\
       (!P. P subset (open L) ==> open L(Union P))

topology_tybij  (TOPOLOGY)
  |- (!a. topology(open a) = a) /\
     (!r. istopology r = (open(topology r) = r))

topology_TY_DEF  (TOPOLOGY)
  |- ?rep. TYPE_DEFINITION istopology rep

TOPOLOGY_UNION  (TOPOLOGY)
  |- !L P. P subset (open L) ==> open L(Union P)

Union  (TOPOLOGY)
  |- !S. Union S = (\x. ?s. S s /\ s x)

union_def  (TOPOLOGY)
  |- !P Q. P union Q = (\x. P x \/ Q x)

universe  (TOPOLOGY)
  |- universe = (\x. T)
```

2.6 NETS

```
bounded  (NETS)
  |- !m g f.
       bounded(m,g)f =
       (?k x N. g N N /\ (!n. g n N ==> (dist m(f n,x)) real_lt k))

dorder  (NETS)
  |- !g.
       dorder g =
       (!x y.
         g x x /\ g y y ==> (?z. g z z /\ (!w. g w z ==> g w x /\ g w y)))

DORDER_LEMMA  (NETS)
  |- !g.
       dorder g ==>
       (!P Q.
         (?n. g n n /\ (!m. g m n ==> P m)) /\
         (?n. g n n /\ (!m. g m n ==> Q m)) ==>
         (?n. g n n /\ (!m. g m n ==> P m /\ Q m)))

DORDER_NGE  (NETS)
  |- dorder $>=

DORDER_TENDSTO  (NETS)
  |- !m x. dorder(tendsto(m,x))

LIM_TENDS  (NETS)
  |- !m1 m2 f x0 y0.
       limpt(mtop m1)x0 universe ==>
       ((f tends y0)(mtop m2,tendsto(m1,x0)) =
        (!e.
          (real_of_num 0) real_lt e ==>
          (?d.
            (real_of_num 0) real_lt d /\
            (!x.
              (real_of_num 0) real_lt (dist m1(x,x0)) /\
              (dist m1(x,x0)) real_le d ==>
              (dist m2(f x,y0)) real_lt e))))
```

```
LIM_TENDS2  (NETS)
  |- !m1 m2 f x0 y0.
       limpt(mtop m1)x0 universe ==>
       ((f tends y0)(mtop m2,tendsto(m1,x0)) =
        (!e.
          (real_of_num 0) real_lt e ==>
          (?d.
            (real_of_num 0) real_lt d /\
            (!x.
              (real_of_num 0) real_lt (dist m1(x,x0)) /\
              (dist m1(x,x0)) real_lt d ==>
              (dist m2(f x,y0)) real_lt e))))

MR1_BOUNDED  (NETS)
  |- !g f.
       bounded(mr1,g)f =
       (?k N. g N N /\ (!n. g n N ==> (abs(f n)) real_lt k))

MTOP_TENDS  (NETS)
  |- !d g x x0.
       (x tends x0)(mtop d,g) =
       (!e.
         (real_of_num 0) real_lt e ==>
         (?n. g n n /\ (!m. g m n ==> (dist d(x m,x0)) real_lt e)))

MTOP_TENDS_UNIQ  (NETS)
  |- !g d.
       dorder g ==>
       (x tends x0)(mtop d,g) /\ (x tends x1)(mtop d,g) ==>
       (x0 = x1)

NET_ABS  (NETS)
  |- !x x0.
       (x tends x0)(mtop mr1,g) ==>
       ((\n. abs(x n)) tends (abs x0))(mtop mr1,g)

NET_ADD  (NETS)
  |- !g.
       dorder g ==>
       (!x x0 y y0.
         (x tends x0)(mtop mr1,g) /\ (y tends y0)(mtop mr1,g) ==>
         ((\n. (x n) real_add (y n)) tends (x0 real_add y0))(mtop mr1,g))

NET_CONV_BOUNDED  (NETS)
  |- !g x x0. (x tends x0)(mtop mr1,g) ==> bounded(mr1,g)x
```

```
NET_CONV_IBOUNDED  (NETS)
  |- !g x x0.
       (x tends x0)(mtop mr1,g) /\ ~(x0 = real_of_num 0) ==>
       bounded(mr1,g)(\n. real_inv(x n))

NET_CONV_NZ  (NETS)
  |- !g x x0.
       (x tends x0)(mtop mr1,g) /\ ~(x0 = real_of_num 0) ==>
       (?N. g N N /\ (!n. g n N ==> ~(x n = real_of_num 0)))

NET_DIV  (NETS)
  |- !g.
       dorder g ==>
       (!x x0 y y0.
         (x tends x0)(mtop mr1,g) /\
         (y tends y0)(mtop mr1,g) /\
         ~(y0 = real_of_num 0) ==>
         ((\n. (x n) / (y n)) tends (x0 / y0))(mtop mr1,g))

NET_INV  (NETS)
  |- !g.
       dorder g ==>
       (!x x0.
         (x tends x0)(mtop mr1,g) /\ ~(x0 = real_of_num 0) ==>
         ((\n. real_inv(x n)) tends (real_inv x0))(mtop mr1,g))

NET_LE  (NETS)
  |- !g.
       dorder g ==>
       (!x x0 y y0.
         (x tends x0)(mtop mr1,g) /\
         (y tends y0)(mtop mr1,g) /\
         (?N. g N N /\ (!n. g n N ==> (x n) real_le (y n))) ==>
         x0 real_le y0)

NET_MUL  (NETS)
  |- !g.
       dorder g ==>
       (!x y x0 y0.
         (x tends x0)(mtop mr1,g) /\ (y tends y0)(mtop mr1,g) ==>
         ((\n. (x n) real_mul (y n)) tends (x0 real_mul y0))(mtop mr1,g))
```

NET_NEG (NETS)
```
  |- !g.
       dorder g ==>
       (!x x0.
         (x tends x0)(mtop mr1,g) =
         ((\n. real_neg(x n)) tends (real_neg x0))(mtop mr1,g))
```

NET_NULL (NETS)
```
  |- !g x x0.
       (x tends x0)(mtop mr1,g) =
       ((\n. (x n) real_sub x0) tends (real_of_num 0))(mtop mr1,g)
```

NET_NULL_ADD (NETS)
```
  |- !g.
       dorder g ==>
       (!x y.
         (x tends (real_of_num 0))(mtop mr1,g) /\
         (y tends (real_of_num 0))(mtop mr1,g) ==>
         ((\n. (x n) real_add (y n)) tends (real_of_num 0))(mtop mr1,g))
```

NET_NULL_CMUL (NETS)
```
  |- !g k x.
       (x tends (real_of_num 0))(mtop mr1,g) ==>
       ((\n. k real_mul (x n)) tends (real_of_num 0))(mtop mr1,g)
```

NET_NULL_MUL (NETS)
```
  |- !g.
       dorder g ==>
       (!x y.
         bounded(mr1,g)x /\ (y tends (real_of_num 0))(mtop mr1,g) ==>
         ((\n. (x n) real_mul (y n)) tends (real_of_num 0))(mtop mr1,g))
```

NET_SUB (NETS)
```
  |- !g.
       dorder g ==>
       (!x x0 y y0.
         (x tends x0)(mtop mr1,g) /\ (y tends y0)(mtop mr1,g) ==>
         ((\n. (x n) real_sub (y n)) tends (x0 real_sub y0))(mtop mr1,g))
```

SEQ_TENDS (NETS)
```
  |- !d x x0.
       (x tends x0)(mtop d,$>=) =
       (!e.
         (real_of_num 0) real_lt e ==>
         (?N. !n. n >= N ==> (dist d(x n,x0)) real_lt e))
```

```
tends (NETS)
  |- !s l top g.
      (s tends l)(top,g) =
      (!N. neigh top(N,l) ==> (?n. g n n /\ (!m. g m n ==> N(s m))))

tendsto (NETS)
  |- !m x y z.
      tendsto(m,x)y z =
      (real_of_num 0) real_lt (dist m(x,y)) /\
      (dist m(x,y)) real_le (dist m(x,z))
```

2.7 LIM

```
CHAIN_LEMMA1 (LIM)
  |- !f g x h.
      ((f(g(x + h))) - (f(g x))) / h =
      (((f(g(x + h))) - (f(g x))) / ((g(x + h)) - (g x))) *
      (((g(x + h)) - (g x)) / h)

CHAIN_LEMMA2 (LIM)
  |- !x y d. (abs(x - y)) < d ==> (abs x) < ((abs y) + d)

contl (LIM)
  |- !f x. f contl x = ((\h. f(x + h)) --> (f x))(& 0)

CONTL_LIM (LIM)
  |- !f x. f contl x = (f --> (f x))x

CONT_ADD (LIM)
  |- !x. f contl x /\ g contl x ==> (\x. (f x) + (g x)) contl x

CONT_ATTAINS (LIM)
  |- !f a b.
      a <= b /\ (!x. a <= x /\ x <= b ==> f contl x) ==>
      (?M.
        (!x. a <= x /\ x <= b ==> (f x) <= M) /\
        (?x. a <= x /\ x <= b /\ (f x = M)))

CONT_ATTAINS2 (LIM)
  |- !f a b.
      a <= b /\ (!x. a <= x /\ x <= b ==> f contl x) ==>
      (?M.
        (!x. a <= x /\ x <= b ==> M <= (f x)) /\
        (?x. a <= x /\ x <= b /\ (f x = M)))
```

```
CONT_BOUNDED (LIM)
  |- !f a b.
       a <= b /\ (!x. a <= x /\ x <= b ==> f contl x) ==>
       (?M. !x. a <= x /\ x <= b ==> (f x) <= M)

CONT_CONST (LIM)
  |- !x. (\x. k) contl x

CONT_DIV (LIM)
  |- !x.
       f contl x /\ g contl x /\ ~(g x = & 0) ==>
       (\x. (f x) / (g x)) contl x

CONT_HASSUP (LIM)
  |- !f a b.
       a <= b /\ (!x. a <= x /\ x <= b ==> f contl x) ==>
       (?M.
         (!x. a <= x /\ x <= b ==> (f x) <= M) /\
         (!N. N < M ==> (?x. a <= x /\ x <= b /\ N < (f x))))

CONT_INV (LIM)
  |- !x. f contl x /\ ~(f x = & 0) ==> (\x. inv(f x)) contl x

CONT_MUL (LIM)
  |- !x. f contl x /\ g contl x ==> (\x. (f x) * (g x)) contl x

CONT_NEG (LIM)
  |- !x. f contl x ==> (\x. --(f x)) contl x

CONT_SUB (LIM)
  |- !x. f contl x /\ g contl x ==> (\x. (f x) - (g x)) contl x

differentiable (LIM)
  |- !f x. f differentiable x = (?l. (f diffl l)x)

diffl (LIM)
  |- !f l x. (f diffl l)x = ((\h. ((f(x + h)) - (f x)) / h) --> l)(& 0)

DIFF_ADD (LIM)
  |- !f g l m x.
       (f diffl l)x /\ (g diffl m)x ==>
       ((\x. (f x) + (g x)) diffl (l + m))x

DIFF_CHAIN (LIM)
  |- !f g x.
       (f diffl l)(g x) /\ (g diffl m)x ==> ((\x. f(g x)) diffl (l * m))x
```

```
DIFF_CMUL  (LIM)
  |- !f c l x. (f diffl l)x ==> ((\x. c * (f x)) diffl (c * l))x

DIFF_CONST  (LIM)
  |- !k x. ((\x. k) diffl (& 0))x

DIFF_CONT  (LIM)
  |- !f l x. (f diffl l)x ==> f contl x

DIFF_DIV  (LIM)
  |- !f g l m.
      (f diffl l)x /\ (g diffl m)x /\ ~(g x = & 0) ==>
      ((\x. (f x) / (g x)) diffl
       (((l * (g x)) - (m * (f x))) / ((g x) pow 2)))
      x

DIFF_INV  (LIM)
  |- !f l x.
      (f diffl l)x /\ ~(f x = & 0) ==>
      ((\x. inv(f x)) diffl (--(l / ((f x) pow 2))))x

DIFF_ISCONST  (LIM)
  |- !f a b.
      a < b /\
      (!x. a <= x /\ x <= b ==> f contl x) /\
      (!x. a < x /\ x < b ==> (f diffl (& 0))x) ==>
      (!x. a <= x /\ x <= b ==> (f x = f a))

DIFF_ISCONST_ALL  (LIM)
  |- !f. (!x. (f diffl (& 0))x) ==> (!x y. f x = f y)

DIFF_ISCONST_END  (LIM)
  |- !f a b.
      a < b /\
      (!x. a <= x /\ x <= b ==> f contl x) /\
      (!x. a < x /\ x < b ==> (f diffl (& 0))x) ==>
      (f b = f a)

DIFF_LCONST  (LIM)
  |- !f x l.
      (f diffl l)x /\
      (?d. (& 0) < d /\ (!y. (abs(x - y)) < d ==> (f y = f x))) ==>
      (l = & 0)
```

```
DIFF_LDEC (LIM)
  |- !f x l.
       (f diffl l)x /\ l < (& 0) ==>
       (?d. (& 0) < d /\ (!h. (& 0) < h /\ h < d ==> (f x) < (f(x - h))))

DIFF_LINC (LIM)
  |- !f x l.
       (f diffl l)x /\ (& 0) < l ==>
       (?d. (& 0) < d /\ (!h. (& 0) < h /\ h < d ==> (f x) < (f(x + h))))

DIFF_LMAX (LIM)
  |- !f x l.
       (f diffl l)x /\
       (?d. (& 0) < d /\ (!y. (abs(x - y)) < d ==> (f y) <= (f x))) ==>
       (l = & 0)

DIFF_LMIN (LIM)
  |- !f x l.
       (f diffl l)x /\
       (?d. (& 0) < d /\ (!y. (abs(x - y)) < d ==> (f x) <= (f y))) ==>
       (l = & 0)

DIFF_MUL (LIM)
  |- !f g l m x.
       (f diffl l)x /\ (g diffl m)x ==>
       ((\x. (f x) * (g x)) diffl ((l * (g x)) + (m * (f x))))x

DIFF_NEG (LIM)
  |- !f l x. (f diffl l)x ==> ((\x. --(f x)) diffl (-- l))x

DIFF_POW (LIM)
  |- !n x. ((\x'. x' pow n) diffl ((& n) * (x pow (n num_sub 1))))x

DIFF_SUB (LIM)
  |- !f g l m x.
       (f diffl l)x /\ (g diffl m)x ==>
       ((\x. (f x) - (g x)) diffl (l - m))x

DIFF_SUM (LIM)
  |- !f f' m n x.
       (!r.
         m num_le r /\ r num_lt (m num_add n) ==>
         ((\x'. f r x') diffl (f' r x))x) ==>
       ((\x'. Sum(m,n)(\n'. f n' x')) diffl (Sum(m,n)(\r. f' r x)))x

DIFF_UNIQ (LIM)
  |- !f l m x. (f diffl l)x /\ (f diffl m)x ==> (l = m)
```

```
DIFF_X  (LIM)
  |- !x. ((\x. x) diffl (& 1))x

DIFF_XM1  (LIM)
  |- !x. ~(x = & 0) ==> ((\x. inv x) diffl (--((inv x) pow 2)))x

INTERVAL_LEMMA  (LIM)
  |- !a b x.
       a < x /\ x < b ==>
       (?d. (& 0) < d /\ (!y. (abs(x - y)) < d ==> a <= y /\ y <= b))

IVT  (LIM)
  |- !f a b y.
       a <= b /\
       ((f a) <= y /\ y <= (f b)) /\
       (!x. a <= x /\ x <= b ==> f contl x) ==>
       (?x. a <= x /\ x <= b /\ (f x = y))

IVT2  (LIM)
  |- !f a b y.
       a <= b /\
       ((f b) <= y /\ y <= (f a)) /\
       (!x. a <= x /\ x <= b ==> f contl x) ==>
       (?x. a <= x /\ x <= b /\ (f x = y))

IVT_SUPLEMMA  (LIM)
  |- a <= b /\
      ((f a) <= y /\ y <= (f b)) /\
      (!x. a <= x /\ x <= b ==> f contl x) ==>
      (?x. (\x. a <= x /\ x <= b /\ (f x) <= y)x) /\
      (?z. !x. (\x. a <= x /\ x <= b /\ (f x) <= y)x ==> x <= z)

IVT_SUPLEMMA2  (LIM)
  |- a <= b /\
      ((f a) <= y /\ y <= (f b)) /\
      (!x. a <= x /\ x <= b ==> f contl x) ==>
      a <= (sup(\x. a <= x /\ x <= b /\ (f x) <= y)) /\
      (sup(\x. a <= x /\ x <= b /\ (f x) <= y)) <= b
```

```
LIM (LIM)
  |- !f y0 x0.
       (f --> y0)x0 =
       (!e.
         (& 0) < e ==>
         (?d.
           (& 0) < d /\
           (!x.
             (& 0) < (abs(x - x0)) /\ (abs(x - x0)) < d ==>
             (abs((f x) - y0)) < e)))

LIM_ADD (LIM)
  |- !f g l m.
       (f --> l)x /\ (g --> m)x ==> ((\x. (f x) + (g x)) --> (l + m))x

LIM_BOUNDED (LIM)
  |- bounded(mr1,tendsto(mr1,x0))f =
     (?k d.
       (& 0) < d /\
       (!x. (& 0) < (abs(x - x0)) /\ (abs(x - x0)) < d ==> (abs(f x)) < k))

LIM_CONST (LIM)
  |- !k x. ((\x. k) --> k)x

LIM_DIV (LIM)
  |- !f g l m.
       (f --> l)x /\ (g --> m)x /\ ~(m = & 0) ==>
       ((\x. (f x) / (g x)) --> (l / m))x

LIM_EQUAL (LIM)
  |- !f g l x0.
       (!x. ~(x = x0) ==> (f x = g x)) ==> ((f --> l)x0 = (g --> l)x0)

LIM_INV (LIM)
  |- !f l. (f --> l)x /\ ~(l = & 0) ==> ((\x. inv(f x)) --> (inv l))x

LIM_MUL (LIM)
  |- !f g l m.
       (f --> l)x /\ (g --> m)x ==> ((\x. (f x) * (g x)) --> (l * m))x

LIM_NEG (LIM)
  |- !f l. (f --> l)x = ((\x. --(f x)) --> (-- l))x

LIM_NULL (LIM)
  |- !f l x. (f --> l)x = ((\x. (f x) - l) --> (& 0))x
```

```
LIM_NULL_MUL  (LIM)
  |- !x x0 y.
       bounded(mr1,tendsto(mr1,x0))x /\ (y --> (& 0))x0 ==>
       ((\u. (x u) * (y u)) --> (& 0))x0

LIM_SUB  (LIM)
  |- !f g l m.
       (f --> l)x /\ (g --> m)x ==> ((\x. (f x) - (g x)) --> (l - m))x

LIM_TRANSFORM  (LIM)
  |- !f g x0 l.
       ((\x. (f x) - (g x)) --> (& 0))x0 /\ (g --> l)x0 ==> (f --> l)x0

LIM_UNIQ  (LIM)
  |- !f l m x. (f --> l)x /\ (f --> m)x ==> (l = m)

LIM_X  (LIM)
  |- !x0. ((\x. x) --> x0)x0

MVT  (LIM)
  |- !f a b.
       a < b /\
       (!x. a <= x /\ x <= b ==> f contl x) /\
       (!x. a < x /\ x < b ==> f differentiable x) ==>
       (?l z.
         a < z /\ z < b /\ (f diffl l)z /\ ((f b) - (f a) = (b - a) * l))

MVT_LEMMA  (LIM)
  |- !f a b.
       (\x. (f x) - ((((f b) - (f a)) / (b - a)) * x))a =
       (\x. (f x) - ((((f b) - (f a)) / (b - a)) * x))b

ROLLE  (LIM)
  |- !f a b.
       a < b /\
       (f a = f b) /\
       (!x. a <= x /\ x <= b ==> f contl x) /\
       (!x. a < x /\ x < b ==> f differentiable x) ==>
       (?z. a < z /\ z < b /\ (f diffl (& 0))z)

SUP_BOUNDED  (LIM)
  |- !P a b.
       (?x. P x) /\ (!x. P x ==> a <= x /\ x <= b) ==>
       a <= (sup P) /\ (sup P) <= b

SUP_UBOUNDED  (LIM)
  |- !P b. (?x. P x) /\ (!x. P x ==> x <= b) ==> (sup P) <= b
```

```
tends_real_real  (LIM)
  |- !f l x0. (f --> l)x0 = (f tends l)(mtop mr1,tendsto(mr1,x0))
```

2.8 SEQ

```
ABS_NEG_LEMMA  (SEQ)
  |- !c. c <= (& 0) ==> (!x y. (abs x) <= (c * (abs y)) ==> (x = & 0))

cauchy  (SEQ)
  |- !f.
      cauchy f =
      (!e.
        (& 0) < e ==>
        (?N. !m n. m num_ge N /\ n num_ge N ==> (abs((f m) - (f n))) < e))

convergent  (SEQ)
  |- !f. convergent f = (?l. f --> l)

GP  (SEQ)
  |- !x. (abs x) < (& 1) ==> (\n. x pow n) sums (inv((& 1) - x))

GP_FINITE  (SEQ)
  |- !x.
      ~(x = & 1) ==>
      (!n. Sum(0,n)(\n'. x pow n') = ((x pow n) - (& 1)) / (x - (& 1)))

lim  (SEQ)
  |- !f. lim f = (@l. f --> l)

MAX_LEMMA  (SEQ)
  |- !s N. ?k. !n. n num_lt N ==> (abs(s n)) < k

mono  (SEQ)
  |- !f.
      mono f =
      (!m n. m num_le n ==> (f m) <= (f n)) \/
      (!m n. m num_le n ==> (f m) >= (f n))

MONO_SUC  (SEQ)
  |- !f. mono f = (!n. (f(SUC n)) >= (f n)) \/ (!n. (f(SUC n)) <= (f n))

SEQ  (SEQ)
  |- !x x0.
      x --> x0 =
      (!e. (& 0) < e ==> (?N. !n. n num_ge N ==> (abs((x n) - x0)) < e))
```

SEQ_ABS (SEQ)
```
|- !f. (\n. abs(f n)) --> (& 0) = f --> (& 0)
```

SEQ_ABS_IMP (SEQ)
```
|- !f l. f --> l ==> (\n. abs(f n)) --> (abs l)
```

SEQ_ADD (SEQ)
```
|- !x x0 y y0.
     x --> x0 /\ y --> y0 ==> (\n. (x n) + (y n)) --> (x0 + y0)
```

SEQ_BCONV (SEQ)
```
|- !f. bounded(mr1,$num_ge)f /\ mono f ==> convergent f
```

SEQ_BOUNDED (SEQ)
```
|- !s. bounded(mr1,$num_ge)s = (?k. !n. (abs(s n)) < k)
```

SEQ_CAUCHY (SEQ)
```
|- !f. cauchy f = convergent f
```

SEQ_CBOUNDED (SEQ)
```
|- !f. cauchy f ==> bounded(mr1,$num_ge)f
```

SEQ_CONST (SEQ)
```
|- !k. (\x. k) --> k
```

SEQ_DIRECT (SEQ)
```
|- !f. subseq f ==> (!N1 N2. ?n. n num_ge N1 /\ (f n) num_ge N2)
```

SEQ_DIV (SEQ)
```
|- !x x0 y y0.
     x --> x0 /\ y --> y0 /\ ~(y0 = & 0) ==>
     (\n. (x n) / (y n)) --> (x0 / y0)
```

SEQ_ICONV (SEQ)
```
|- !f.
     bounded(mr1,$num_ge)f /\ (!m n. m num_ge n ==> (f m) >= (f n)) ==>
     convergent f
```

SEQ_INV (SEQ)
```
|- !x x0. x --> x0 /\ ~(x0 = & 0) ==> (\n. inv(x n)) --> (inv x0)
```

SEQ_INV0 (SEQ)
```
|- !f.
     (!y. ?N. !n. n num_ge N ==> (f n) > y) ==> (\n. inv(f n)) --> (& 0)
```

SEQ_LE (SEQ)
```
|- !f g l m.
     f --> l /\ g --> m /\ (?N. !n. n num_ge N ==> (f n) <= (g n)) ==>
     l <= m
```

SEQ_LIM (SEQ)
```
|- !f. convergent f = f --> (lim f)
```

SEQ_MONOSUB (SEQ)
```
|- !s. ?f. subseq f /\ mono(\n. s(f n))
```

SEQ_MUL (SEQ)
```
|- !x x0 y y0.
     x --> x0 /\ y --> y0 ==> (\n. (x n) * (y n)) --> (x0 * y0)
```

SEQ_NEG (SEQ)
```
|- !x x0. x --> x0 = (\n. --(x n)) --> (-- x0)
```

SEQ_NEG_BOUNDED (SEQ)
```
|- !f. bounded(mr1,$num_ge)(\n. --(f n)) = bounded(mr1,$num_ge)f
```

SEQ_NEG_CONV (SEQ)
```
|- !f. convergent f = convergent(\n. --(f n))
```

SEQ_POWER (SEQ)
```
|- !c. (abs c) < (& 1) ==> (\n. c pow n) --> (& 0)
```

SEQ_POWER_ABS (SEQ)
```
|- !c. (abs c) < (& 1) ==> (\n. (abs c) pow n) --> (& 0)
```

SEQ_SBOUNDED (SEQ)
```
|- !s f. bounded(mr1,$num_ge)s ==> bounded(mr1,$num_ge)(\n. s(f n))
```

SEQ_SUB (SEQ)
```
|- !x x0 y y0.
     x --> x0 /\ y --> y0 ==> (\n. (x n) - (y n)) --> (x0 - y0)
```

SEQ_SUBLE (SEQ)
```
|- !f. subseq f ==> (!n. n num_le (f n))
```

SEQ_SUC (SEQ)
```
|- !f l. f --> l = (\n. f(SUC n)) --> l
```

SEQ_UNIQ (SEQ)
```
|- !x x1 x2. x --> x1 /\ x --> x2 ==> (x1 = x2)
```

```
SER_0  (SEQ)
  |- !f n. (!m. n num_le m ==> (f m = & 0)) ==> f sums (Sum(0,n)f)

SER_ABS  (SEQ)
  |- !f.
       summable(\n. abs(f n)) ==> (abs(suminf f)) <= (suminf(\n. abs(f n)))

SER_ACONV  (SEQ)
  |- !f. summable(\n. abs(f n)) ==> summable f

SER_ADD  (SEQ)
  |- !x x0 y y0.
       x sums x0 /\ y sums y0 ==> (\n. (x n) + (y n)) sums (x0 + y0)

SER_CAUCHY  (SEQ)
  |- !f.
       summable f =
       (!e. (& 0) < e ==> (?N. !m n. m num_ge N ==> (abs(Sum(m,n)f)) < e))

SER_CDIV  (SEQ)
  |- !x x0 c. x sums x0 ==> (\n. (x n) / c) sums (x0 / c)

SER_CMUL  (SEQ)
  |- !x x0 c. x sums x0 ==> (\n. c * (x n)) sums (c * x0)

SER_COMPAR  (SEQ)
  |- !f g.
       (?N. !n. n num_ge N ==> (abs(f n)) <= (g n)) /\ summable g ==>
       summable f

SER_COMPARA  (SEQ)
  |- !f g.
       (?N. !n. n num_ge N ==> (abs(f n)) <= (g n)) /\ summable g ==>
       summable(\k. abs(f k))

SER_GROUP  (SEQ)
  |- !f k.
       summable f /\ 0 num_lt k ==>
       (\n. Sum(n num_mul k,k)f) sums (suminf f)

SER_LE  (SEQ)
  |- !f g.
       (!n. (f n) <= (g n)) /\ summable f /\ summable g ==>
       (suminf f) <= (suminf g)
```

```
SER_LE2  (SEQ)
  |- !f g.
       (!n. (abs(f n)) <= (g n)) /\ summable g ==>
       summable f /\ (suminf f) <= (suminf g)
```

```
SER_NEG  (SEQ)
  |- !x x0. x sums x0 ==> (\n. --(x n)) sums (-- x0)
```

```
SER_OFFSET  (SEQ)
  |- !f.
       summable f ==>
       (!k. (\n. f(n num_add k)) sums ((suminf f) - (Sum(0,k)f)))
```

```
SER_PAIR  (SEQ)
  |- !f. summable f ==> (\n. Sum(2 num_mul n,2)f) sums (suminf f)
```

```
SER_POS_LE  (SEQ)
  |- !f n.
       summable f /\ (!m. n num_le m ==> (& 0) <= (f m)) ==>
       (Sum(0,n)f) <= (suminf f)
```

```
SER_POS_LT  (SEQ)
  |- !f n.
       summable f /\ (!m. n num_le m ==> (& 0) < (f m)) ==>
       (Sum(0,n)f) < (suminf f)
```

```
SER_POS_LT_PAIR  (SEQ)
  |- !f n.
       summable f /\
       (!d.
         (& 0) <
         ((f(n num_add (2 num_mul d))) +
          (f(n num_add ((2 num_mul d) num_add 1))))) ==>
       (Sum(0,n)f) < (suminf f)
```

```
SER_RATIO  (SEQ)
  |- !f c N.
       c < (& 1) /\
       (!n. n num_ge N ==> (abs(f(SUC n))) <= (c * (abs(f n)))) ==>
       summable f
```

```
SER_SUB  (SEQ)
  |- !x x0 y y0.
       x sums x0 /\ y sums y0 ==> (\n. (x n) - (y n)) sums (x0 - y0)
```

```
SER_ZERO  (SEQ)
  |- !f. summable f ==> f --> (& 0)
```

subseq (SEQ)
```
|- !f. subseq f = (!m n. m num_lt n ==> (f m) num_lt (f n))
```

SUBSEQ_SUC (SEQ)
```
|- !f. subseq f = (!n. (f n) num_lt (f(SUC n)))
```

suminf (SEQ)
```
|- !f. suminf f = (@s. f sums s)
```

summable (SEQ)
```
|- !f. summable f = (?s. f sums s)
```

SUMMABLE_SUM (SEQ)
```
|- !f. summable f ==> f sums (suminf f)
```

sums (SEQ)
```
|- !f s. f sums s = (\n. Sum(0,n)f) --> s
```

SUM_SUMMABLE (SEQ)
```
|- !f l. f sums l ==> summable f
```

SUM_UNIQ (SEQ)
```
|- !f x. f sums x ==> (x = suminf f)
```

tends_num_real (SEQ)
```
|- !x x0. x --> x0 = (x tends x0)(mtop mr1,$num_ge)
```

2.9 POWSER

diffs (POWSER)
```
|- !c. diffs c = (\n. (&(SUC n)) * (c(SUC n)))
```

DIFFS_EQUIV (POWSER)
```
|- !c x.
    summable(\n. (diffs c n) * (x pow n)) ==>
    (\n. (& n) * ((c n) * (x pow (n num_sub 1)))) sums
    (suminf(\n. (diffs c n) * (x pow n)))
```

DIFFS_LEMMA (POWSER)
```
|- !n c x.
    Sum(0,n)(\n'. (diffs c n') * (x pow n')) =
    (Sum(0,n)(\n'. (& n') * ((c n') * (x pow (n' num_sub 1))))) +
    ((& n) * ((c n) * (x pow (n num_sub 1))))
```

```
DIFFS_LEMMA2  (POWSER)
  |- !n c x.
      Sum(0,n)(\n. (& n) * ((c n) * (x pow (n num_sub 1)))) =
      (Sum(0,n)(\n. (diffs c n) * (x pow n))) -
      ((& n) * ((c n) * (x pow (n num_sub 1))))

DIFFS_NEG  (POWSER)
  |- !c. diffs(\n. --(c n)) = (\n. --(diffs c n))

POWDIFF  (POWSER)
  |- !n x y.
      (x pow (SUC n)) - (y pow (SUC n)) =
      (x - y) * (Sum(0,SUC n)(\p. (x pow p) * (y pow (n num_sub p))))

POWDIFF_LEMMA  (POWSER)
  |- !n x y.
      Sum(0,SUC n)(\p. (x pow p) * (y pow ((SUC n) num_sub p))) =
      y * (Sum(0,SUC n)(\p. (x pow p) * (y pow (n num_sub p))))

POWREV  (POWSER)
  |- !n x y.
      Sum(0,SUC n)(\p. (x pow p) * (y pow (n num_sub p))) =
      Sum(0,SUC n)(\p. (x pow (n num_sub p)) * (y pow p))

POWSER_INSIDE  (POWSER)
  |- !f x z.
      summable(\n. (f n) * (x pow n)) /\ (abs z) < (abs x) ==>
      summable(\n. (f n) * (z pow n))

POWSER_INSIDEA  (POWSER)
  |- !f x z.
      summable(\n. (f n) * (x pow n)) /\ (abs z) < (abs x) ==>
      summable(\n. (abs(f n)) * (z pow n))

TERMDIFF  (POWSER)
  |- !c K.
      summable(\n. (c n) * (K pow n)) /\
      summable(\n. (diffs c n) * (K pow n)) /\
      summable(\n. (diffs(diffs c)n) * (K pow n)) /\
      (abs x) < (abs K) ==>
      ((\x. suminf(\n. (c n) * (x pow n))) diffl
       (suminf(\n. (diffs c n) * (x pow n))))
      x
```

```
TERMDIFF_LEMMA1  (POWSER)
  |- !m z h.
       Sum(0,m)(\p. (((z + h) pow (m num_sub p)) * (z pow p)) - (z pow m)) =
       Sum
       (0,m)
       (\p.
         (z pow p) * (((z + h) pow (m num_sub p)) - (z pow (m num_sub p))))
```

```
TERMDIFF_LEMMA2  (POWSER)
  |- !z h.
       ~(h = & 0) ==>
       (((((z + h) pow n) - (z pow n)) / h) -
        ((& n) * (z pow (n num_sub 1))) =
        h *
        (Sum
         (0,n num_sub 1)
         (\p.
           (z pow p) *
           (Sum
            (0,(n num_sub 1) num_sub p)
            (\q.
              ((z + h) pow q) *
              (z pow (((n num_sub 2) num_sub p) num_sub q)))))))
```

```
TERMDIFF_LEMMA3  (POWSER)
  |- !z h n K.
       ~(h = & 0) /\ (abs z) <= K /\ (abs(z + h)) <= K ==>
       (abs
        (((((z + h) pow n) - (z pow n)) / h) -
         ((& n) * (z pow (n num_sub 1))))) <=
       ((& n) * ((&(n num_sub 1)) * ((K pow (n num_sub 2)) * (abs h))))
```

```
TERMDIFF_LEMMA4  (POWSER)
  |- !f K k.
       (& 0) < k /\
       (!h. (& 0) < (abs h) /\ (abs h) < k ==> (abs(f h)) <= (K * (abs h))) ==>
       (f tends_real_real (& 0))(& 0)
```

```
TERMDIFF_LEMMA5  (POWSER)
  |- !f g k.
       (& 0) < k /\
       summable f /\
       (!h.
         (& 0) < (abs h) /\ (abs h) < k ==>
         (!n. (abs(g h n)) <= ((f n) * (abs h)))) ==>
       ((\h. suminf(g h)) tends_real_real (& 0))(& 0)
```

2.10 TRANSC

```
ACS  (TRANSC)
  |- !y.
       (--(& 1)) <= y /\ y <= (& 1) ==>
       (& 0) <= (acs y) /\ (acs y) <= pi /\ (cos(acs y) = y)

acs  (TRANSC)
  |- !y. acs y = (@x. (& 0) <= x /\ x <= pi /\ (cos x = y))

ACS_BOUNDS  (TRANSC)
  |- !y.
       (--(& 1)) <= y /\ y <= (& 1) ==> (& 0) <= (acs y) /\ (acs y) <= pi

ACS_COS  (TRANSC)
  |- !y. (--(& 1)) <= y /\ y <= (& 1) ==> (cos(acs y) = y)

ASN  (TRANSC)
  |- !y.
       (--(& 1)) <= y /\ y <= (& 1) ==>
       (--(pi / (& 2))) <= (asn y) /\
       (asn y) <= (pi / (& 2)) /\
       (sin(asn y) = y)

asn  (TRANSC)
  |- !y.
       asn y =
       (@x. (--(pi / (& 2))) <= x /\ x <= (pi / (& 2)) /\ (sin x = y))

ASN_BOUNDS  (TRANSC)
  |- !y.
       (--(& 1)) <= y /\ y <= (& 1) ==>
       (--(pi / (& 2))) <= (asn y) /\ (asn y) <= (pi / (& 2))

ASN_SIN  (TRANSC)
  |- !y. (--(& 1)) <= y /\ y <= (& 1) ==> (sin(asn y) = y)

ATN  (TRANSC)
  |- !y.
       (--(pi / (& 2))) < (atn y) /\
       (atn y) < (pi / (& 2)) /\
       (tan(atn y) = y)
```

```
atn  (TRANSC)
  |- !y.
      atn y =
      (@x. (--(pi / (& 2))) < x /\ x < (pi / (& 2)) /\ (tan x = y))

ATN_BOUNDS  (TRANSC)
  |- !y. (--(pi / (& 2))) < (atn y) /\ (atn y) < (pi / (& 2))

ATN_TAN  (TRANSC)
  |- !y. tan(atn y) = y

cos  (TRANSC)
  |- !x.
      cos x =
      suminf
      (\n.
        ((\n'.
           (EVEN n' => ((--(& 1)) pow (n' DIV 2)) / (&(FACT n')) | & 0))
         n) *
        (x pow n))

COS_0  (TRANSC)
  |- cos(& 0) = & 1

COS_2  (TRANSC)
  |- (cos(& 2)) < (& 0)

COS_ACS  (TRANSC)
  |- !x. (& 0) <= x /\ x <= pi ==> (acs(cos x) = x)

COS_ADD  (TRANSC)
  |- !x y. cos(x + y) = ((cos x) * (cos y)) - ((sin x) * (sin y))

COS_BOUND  (TRANSC)
  |- !x. (abs(cos x)) <= (& 1)

COS_BOUNDS  (TRANSC)
  |- !x. (--(& 1)) <= (cos x) /\ (cos x) <= (& 1)

COS_CONVERGES  (TRANSC)
  |- !x.
      (\n.
        ((\n. (EVEN n => ((--(& 1)) pow (n DIV 2)) / (&(FACT n)) | & 0))n) *
        (x pow n)) sums
      (cos x)
```

```
COS_DOUBLE  (TRANSC)
  |- !x. cos((& 2) * x) = ((cos x) pow 2) - ((sin x) pow 2)

COS_FDIFF  (TRANSC)
  |- diffs(\n. (EVEN n => ((--(& 1)) pow (n DIV 2)) / (&(FACT n)) | & 0)) =
     (\n.
       --
       ((\n.
          (EVEN n =>
           & 0 |
           ((--(& 1)) pow ((n num_sub 1) DIV 2)) / (&(FACT n))))
        n))

COS_ISZERO  (TRANSC)
  |- ?! x. (& 0) <= x /\ x <= (& 2) /\ (cos x = & 0)

COS_NEG  (TRANSC)
  |- !x. cos(-- x) = cos x

COS_NPI  (TRANSC)
  |- !n. cos((& n) * pi) = (--(& 1)) pow n

COS_PAIRED  (TRANSC)
  |- !x.
      (\n.
        (((--(& 1)) pow n) / (&(FACT(2 num_mul n)))) *
        (x pow (2 num_mul n))) sums
      (cos x)

COS_PERIODIC  (TRANSC)
  |- !x. cos(x + ((& 2) * pi)) = cos x

COS_PERIODIC_PI  (TRANSC)
  |- !x. cos(x + pi) = --(cos x)

COS_PI  (TRANSC)
  |- cos pi = --(& 1)

COS_PI2  (TRANSC)
  |- cos(pi / (& 2)) = & 0

COS_POS_PI  (TRANSC)
  |- !x. (--(pi / (& 2))) < x /\ x < (pi / (& 2)) ==> (& 0) < (cos x)

COS_POS_PI2  (TRANSC)
  |- !x. (& 0) < x /\ x < (pi / (& 2)) ==> (& 0) < (cos x)
```

```
COS_SIN  (TRANSC)
  |- !x. cos x = sin((pi / (& 2)) - x)

COS_TOTAL  (TRANSC)
  |- !y.
      (--(& 1)) <= y /\ y <= (& 1) ==>
      (?! x. (& 0) <= x /\ x <= pi /\ (cos x = y))

COS_ZERO  (TRANSC)
  |- !x.
      (cos x = & 0) =
      (?n. ~EVEN n /\ (x = (& n) * (pi / (& 2)))) \/
      (?n. ~EVEN n /\ (x = --((& n) * (pi / (& 2)))))

COS_ZERO_LEMMA  (TRANSC)
  |- !x.
      (& 0) <= x /\ (cos x = & 0) ==>
      (?n. ~EVEN n /\ (x = (& n) * (pi / (& 2))))

DIFF_COS  (TRANSC)
  |- !x. (cos diffl (--(sin x)))x

DIFF_EXP  (TRANSC)
  |- !x. (exp diffl (exp x))x

DIFF_SIN  (TRANSC)
  |- !x. (sin diffl (cos x))x

DIFF_TAN  (TRANSC)
  |- !x. ~(cos x = & 0) ==> (tan diffl (inv((cos x) pow 2)))x

exp  (TRANSC)
  |- !x. exp x = suminf(\n. ((\n'. inv(&(FACT n')))n) * (x pow n))

EXP_0  (TRANSC)
  |- exp(& 0) = & 1

EXP_ADD  (TRANSC)
  |- !x y. exp(x + y) = (exp x) * (exp y)

EXP_ADD_MUL  (TRANSC)
  |- !x y. (exp(x + y)) * (exp(-- x)) = exp y

EXP_CONVERGES  (TRANSC)
  |- !x. (\n. ((\n. inv(&(FACT n)))n) * (x pow n)) sums (exp x)
```

```
EXP_FDIFF  (TRANSC)
  |- diffs(\n. inv(&(FACT n))) = (\n. inv(&(FACT n)))

EXP_INJ  (TRANSC)
  |- !x y. (exp x = exp y) = (x = y)

EXP_LE_X  (TRANSC)
  |- !x. (& 0) <= x ==> ((& 1) + x) <= (exp x)

EXP_LN  (TRANSC)
  |- !x. (exp(ln x) = x) = (& 0) < x

EXP_LT_1  (TRANSC)
  |- !x. (& 0) < x ==> (& 1) < (exp x)

EXP_MONO_IMP  (TRANSC)
  |- !x y. x < y ==> (exp x) < (exp y)

EXP_MONO_LE  (TRANSC)
  |- !x y. (exp x) <= (exp y) = x <= y

EXP_MONO_LT  (TRANSC)
  |- !x y. (exp x) < (exp y) = x < y

EXP_N  (TRANSC)
  |- !n x. exp((& n) * x) = (exp x) pow n

EXP_NEG  (TRANSC)
  |- !x. exp(-- x) = inv(exp x)

EXP_NEG_MUL  (TRANSC)
  |- !x. (exp x) * (exp(-- x)) = & 1

EXP_NEG_MUL2  (TRANSC)
  |- !x. (exp(-- x)) * (exp x) = & 1

EXP_NZ  (TRANSC)
  |- !x. ~(exp x = & 0)

EXP_POS_LE  (TRANSC)
  |- !x. (& 0) <= (exp x)

EXP_POS_LT  (TRANSC)
  |- !x. (& 0) < (exp x)

EXP_SUB  (TRANSC)
  |- !x y. exp(x - y) = (exp x) / (exp y)
```

EXP_TOTAL (TRANSC)
```
|- !y. (& 0) < y ==> (?x. exp x = y)
```

EXP_TOTAL_LEMMA (TRANSC)
```
|- !y.
    (& 1) <= y ==> (?x. (& 0) <= x /\ x <= (y - (& 1)) /\ (exp x = y))
```

ln (TRANSC)
```
|- !x. ln x = (@u. exp u = x)
```

LN_1 (TRANSC)
```
|- ln(& 1) = & 0
```

LN_DIV (TRANSC)
```
|- !x. (& 0) < x /\ (& 0) < y ==> (ln(x / y) = (ln x) - (ln y))
```

LN_EXP (TRANSC)
```
|- !x. ln(exp x) = x
```

LN_INJ (TRANSC)
```
|- !x y. (& 0) < x /\ (& 0) < y ==> ((ln x = ln y) = (x = y))
```

LN_INV (TRANSC)
```
|- !x. (& 0) < x ==> (ln(inv x) = --(ln x))
```

LN_MONO_LE (TRANSC)
```
|- !x y. (& 0) < x /\ (& 0) < y ==> ((ln x) <= (ln y) = x <= y)
```

LN_MONO_LT (TRANSC)
```
|- !x y. (& 0) < x /\ (& 0) < y ==> ((ln x) < (ln y) = x < y)
```

LN_MUL (TRANSC)
```
|- !x y. (& 0) < x /\ (& 0) < y ==> (ln(x * y) = (ln x) + (ln y))
```

LN_POW (TRANSC)
```
|- !n x. (& 0) < x ==> (ln(x pow n) = (& n) * (ln x))
```

pi (TRANSC)
```
|- pi = (& 2) * (@x. (& 0) <= x /\ x <= (& 2) /\ (cos x = & 0))
```

PI2 (TRANSC)
```
|- pi / (& 2) = (@x. (& 0) <= x /\ x <= (& 2) /\ (cos x = & 0))
```

PI2_BOUNDS (TRANSC)
```
|- (& 0) < (pi / (& 2)) /\ (pi / (& 2)) < (& 2)
```

```
PI_POS  (TRANSC)
  |- (& 0) < pi

root  (TRANSC)
  |- !n x. root n x = (@u. ((& 0) < x ==> (& 0) < u) /\ (u pow n = x))

ROOT_0  (TRANSC)
  |- !n. root(SUC n)(& 0) = & 0

ROOT_1  (TRANSC)
  |- !n. root(SUC n)(& 1) = & 1

ROOT_LN  (TRANSC)
  |- !n x. (& 0) < x ==> (!n. root(SUC n)x = exp((ln x) / (&(SUC n))))

ROOT_LT_LEMMA  (TRANSC)
  |- !n x. (& 0) < x ==> ((exp((ln x) / (&(SUC n)))) pow (SUC n) = x)

ROOT_POW_POS  (TRANSC)
  |- !n x. (& 0) <= x ==> ((root(SUC n)x) pow (SUC n) = x)

sin  (TRANSC)
  |- !x.
       sin x =
       suminf
       (\n.
         ((\n'.
            (EVEN n' =>
             & 0 |
             ((--(& 1)) pow ((n' num_sub 1) DIV 2)) / (&(FACT n'))))
          n) *
         (x pow n))

SIN_0  (TRANSC)
  |- sin(& 0) = & 0

SIN_ADD  (TRANSC)
  |- !x y. sin(x + y) = ((sin x) * (cos y)) + ((cos x) * (sin y))

SIN_ASN  (TRANSC)
  |- !x. (--(pi / (& 2))) <= x /\ x <= (pi / (& 2)) ==> (asn(sin x) = x)

SIN_BOUND  (TRANSC)
  |- !x. (abs(sin x)) <= (& 1)

SIN_BOUNDS  (TRANSC)
  |- !x. (--(& 1)) <= (sin x) /\ (sin x) <= (& 1)
```

```
SIN_CIRCLE (TRANSC)
  |- !x. ((sin x) pow 2) + ((cos x) pow 2) = & 1

SIN_CONVERGES (TRANSC)
  |- !x.
      (\n.
        ((\n.
           (EVEN n =>
            & 0 |
            ((--(& 1)) pow ((n num_sub 1) DIV 2)) / (&(FACT n))))
         n) *
        (x pow n)) sums
      (sin x)

SIN_COS (TRANSC)
  |- !x. sin x = cos((pi / (& 2)) - x)

SIN_COS_ADD (TRANSC)
  |- !x y.
      (((sin(x + y)) - (((sin x) * (cos y)) + ((cos x) * (sin y)))) pow 2) +
      (((cos(x + y)) - (((cos x) * (cos y)) - ((sin x) * (sin y)))) pow 2) =
      & 0

SIN_COS_NEG (TRANSC)
  |- !x.
      (((sin(-- x)) + (sin x)) pow 2) + (((cos(-- x)) - (cos x)) pow 2) =
      & 0

SIN_DOUBLE (TRANSC)
  |- !x. sin((& 2) * x) = (& 2) * ((sin x) * (cos x))

SIN_FDIFF (TRANSC)
  |- diffs
     (\n.
       (EVEN n =>
        & 0 |
        ((--(& 1)) pow ((n num_sub 1) DIV 2)) / (&(FACT n)))) =
     (\n. (EVEN n => ((--(& 1)) pow (n DIV 2)) / (&(FACT n)) | & 0))

SIN_NEG (TRANSC)
  |- !x. sin(-- x) = --(sin x)
```

```
SIN_NEGLEMMA  (TRANSC)
  |- !x.
       --(sin x) =
       suminf
       (\n.
         --
         (((\n.
             (EVEN n =>
              & 0 |
              ((--(& 1)) pow ((n num_sub 1) DIV 2)) / (&(FACT n))))
           n) *
          (x pow n)))

SIN_NPI  (TRANSC)
  |- !n. sin((& n) * pi) = & 0

SIN_PAIRED  (TRANSC)
  |- !x.
       (\n.
         (((--(& 1)) pow n) / (&(FACT((2 num_mul n) num_add 1)))) *
         (x pow ((2 num_mul n) num_add 1))) sums
       (sin x)

SIN_PERIODIC  (TRANSC)
  |- !x. sin(x + ((& 2) * pi)) = sin x

SIN_PERIODIC_PI  (TRANSC)
  |- !x. sin(x + pi) = --(sin x)

SIN_PI  (TRANSC)
  |- sin pi = & 0

SIN_PI2  (TRANSC)
  |- sin(pi / (& 2)) = & 1

SIN_POS  (TRANSC)
  |- !x. (& 0) < x /\ x < (& 2) ==> (& 0) < (sin x)

SIN_POS_PI  (TRANSC)
  |- !x. (& 0) < x /\ x < pi ==> (& 0) < (sin x)

SIN_POS_PI2  (TRANSC)
  |- !x. (& 0) < x /\ x < (pi / (& 2)) ==> (& 0) < (sin x)
```

SIN_TOTAL (TRANSC)
```
|- !y.
     (--(& 1)) <= y /\ y <= (& 1) ==>
     (?! x. (--(pi / (& 2))) <= x /\ x <= (pi / (& 2)) /\ (sin x = y))
```

SIN_ZERO (TRANSC)
```
|- !x.
     (sin x = & 0) =
     (?n. EVEN n /\ (x = (& n) * (pi / (& 2)))) \/
     (?n. EVEN n /\ (x = --((& n) * (pi / (& 2)))))
```

SIN_ZERO_LEMMA (TRANSC)
```
|- !x.
     (& 0) <= x /\ (sin x = & 0) ==>
     (?n. EVEN n /\ (x = (& n) * (pi / (& 2))))
```

sqrt (TRANSC)
```
|- !x. sqrt x = root 2 x
```

SQRT_0 (TRANSC)
```
|- sqrt(& 0) = & 0
```

SQRT_1 (TRANSC)
```
|- sqrt(& 1) = & 1
```

SQRT_POW2 (TRANSC)
```
|- !x. ((sqrt x) pow 2 = x) = (& 0) <= x
```

tan (TRANSC)
```
|- !x. tan x = (sin x) / (cos x)
```

TAN_0 (TRANSC)
```
|- tan(& 0) = & 0
```

TAN_ADD (TRANSC)
```
|- !x y.
     ~(cos x = & 0) /\ ~(cos y = & 0) /\ ~(cos(x + y) = & 0) ==>
     (tan(x + y) = ((tan x) + (tan y)) / ((& 1) - ((tan x) * (tan y))))
```

TAN_ATN (TRANSC)
```
|- !x. (--(pi / (& 2))) < x /\ x < (pi / (& 2)) ==> (atn(tan x) = x)
```

TAN_DOUBLE (TRANSC)
```
|- !x.
     ~(cos x = & 0) /\ ~(cos((& 2) * x) = & 0) ==>
     (tan((& 2) * x) = ((& 2) * (tan x)) / ((& 1) - ((tan x) pow 2)))
```

```
TAN_NEG (TRANSC)
  |- !x. tan(-- x) = --(tan x)

TAN_NPI (TRANSC)
  |- !n. tan((& n) * pi) = & 0

TAN_PERIODIC (TRANSC)
  |- !x. tan(x + ((& 2) * pi)) = tan x

TAN_PI (TRANSC)
  |- tan pi = & 0

TAN_POS_PI2 (TRANSC)
  |- !x. (& 0) < x /\ x < (pi / (& 2)) ==> (& 0) < (tan x)

TAN_TOTAL (TRANSC)
  |- !y. ?! x. (--(pi / (& 2))) < x /\ x < (pi / (& 2)) /\ (tan x = y)

TAN_TOTAL_LEMMA (TRANSC)
  |- !y. (& 0) < y ==> (?x. (& 0) < x /\ x < (pi / (& 2)) /\ y < (tan x))

TAN_TOTAL_POS (TRANSC)
  |- !y.
      (& 0) <= y ==> (?x. (& 0) <= x /\ x < (pi / (& 2)) /\ (tan x = y))
```

References

[1] N. G. de Bruijn. *Defining Reals Without the Use of Rationals.* Indagationes Mathematicae, vol. 38, no. 2, 1976.

[2] J. C. Burkill *A First Course in Mathematical Analysis.* Cambridge 1962.

[3] J. C. Burkill, H. Burkill. *A Second Course in Mathematical Analysis.* Cambridge 1970.

[4] C. W. Burrill. *Foundations of Real Numbers.* McGraw-Hill 1967.

[5] L. W. Cohen, G. Ehrlich. *Structure of the real number system.* Van Nostrand 1963.

[6] J. H. Conway. *On Numbers and Games.* Academic Press 1976.

[7] R. M. Dudley. *Real Analysis and Probability.* Wadsworth and Brooks / Cole 1989.

[8] F. Faltin, N. Metropolis, B. Ross, G.-C. Rota. *The Real Numbers as a Wreath Product.* Advances in Mathematics, vol. 16, pp. 278-304, 1975.

[9] C. Jones. *Completing the Rationals and Metric Spaces in LEGO.* Proc. 2nd Workshop on Logical Frameworks, Edinburgh 1991.

[10] L. S. Jutting. *Checking Landau's "Grundlagen" in the Automath System.* PhD thesis, Eindhoven University of Technology 1977.

[11] J. L. Kelley. *General Topology.* Van Nostrand 1955.

[12] E. Landau. *Foundations of analysis.* Chelsea 1951.

[13] H. A. Thurston. *The number system.* Blackie 1956.

Index