

Real Exponents as the Limits of Sequences of Rational Exponents

Jacques D. Fleuriot
University of Edinburgh

December 14, 2021

Abstract

In this formalisation, we construct real exponents as the limits of sequences of rational exponents. In particular, if $a \geq 1$ and $x \in \mathbb{R}$, we choose an increasing rational sequence r_n such that $\lim_{n \rightarrow \infty} r_n = x$. Then the sequence a^{r_n} is increasing and if r is any rational number such that $r > x$, a^{r_n} is bounded above by a^r . By the convergence criterion for monotone sequences, a^{r_n} converges. We define $a^x = \lim_{n \rightarrow \infty} a^{r_n}$ and show that it has the expected properties (for $a \geq 0$).

This particular construction of real exponents is needed instead of the usual one using the natural logarithm and exponential functions (which already exists in Isabelle) to support our mechanical derivation of Euler's exponential series as an "infinite polynomial". Aside from helping us avoid circular reasoning, this is, as far as we are aware, the first time real exponents are mechanised in this way within a proof assistant.

Contents

1	Rational Exponents	1
2	Real Exponents via Limits	7
3	Real Logarithms (Redefined)	22

```
theory RatPower
imports HOL.NthRoot
begin
```

1 Rational Exponents

A few lemmas about nth-root.

```

lemma real_root_mult_exp_cancel:
  "[[ 0 < x; 0 < m; 0 < n ]]"
  => root (m * n) (x ^ (k * n)) = root m (x ^ k)"
  <proof>

```

```

lemma real_root_mult_exp_cancel1:
  "[[ 0 < x; 0 < n ]]" => root n (x ^ (k * n)) = x ^ k"
  <proof>

```

```

lemma real_root_mult_exp_cancel2:
  "[[ 0 < x; 0 < m; 0 < n ]]"
  => root (n * m) (x ^ (n * k)) = root m (x ^ k)"
  <proof>

```

```

lemma real_root_mult_exp_cancel3:
  "[[ 0 < x; 0 < n ]]" => root n (x ^ (n * k)) = x ^ k"
  <proof>

```

Definition of rational exponents,

definition

```

powrat :: "[real, rat] => real"      (infixr "powQ" 80) where
"x powQ r = (if r > 0
              then root (nat (snd(quotient_of r)))
                    (x ^ (nat (fst(quotient_of r))))
              else root (nat (snd(quotient_of r)))
                    (1/x ^ (nat (- fst(quotient_of r))))))"

```

```

declare quotient_of_denom_pos' [simp]

```

```

lemma powrat_one_eq_one [simp]: "1 powQ a = 1"
  <proof>

```

```

lemma powrat_zero_eq_one [simp]: "x powQ 0 = 1"
  <proof>

```

```

lemma powrat_one [simp]: "x powQ 1 = x"
  <proof>

```

```

lemma powrat_mult_base:
  "(x * y) powQ r = (x powQ r) * (y powQ r)"
  <proof>

```

```

lemma powrat_divide:
  "(x / y) powQ r = (x powQ r)/(y powQ r)"
  <proof>

```

```

lemma powrat_zero_base [simp]:
  assumes "r ≠ 0" shows "0 powQ r = 0"

```

<proof>

lemma powrat_inverse:
"inverse y pow_Q r = inverse(y pow_Q r)"
<proof>

lemma powrat_minus:
"x pow_Q (-r) = inverse (x pow_Q r)"
<proof>

lemma powrat_gt_zero:
assumes "x > 0" shows "x pow_Q r > 0"
<proof>

lemma powrat_not_zero:
assumes "x ≠ 0" shows "x pow_Q r ≠ 0"
<proof>

lemma gcd_add_mult_commute: "gcd (m::'a::semiring_gcd) (n + k * m) =
gcd m n"
<proof>

lemma coprime_add_mult_iff1 [simp]:
"coprime (n + k * m) (m::'a::semiring_gcd) = coprime n m"
<proof>

lemma coprime_add_mult_iff2 [simp]:
"coprime (k * m + n) (m::'a::semiring_gcd) = coprime n m"
<proof>

lemma gcd_mult_div_cancel_left1 [simp]:
"gcd a b * (a div gcd a b) = (a::'a::semiring_gcd)"
<proof>

lemma gcd_mult_div_cancel_left2 [simp]:
"gcd b a * (a div gcd b a) = (a::'a::semiring_gcd)"
<proof>

lemma gcd_mult_div_cancel_right1 [simp]:
"(a div gcd a b) * gcd a b = (a::'a::semiring_gcd)"
<proof>

lemma gcd_mult_div_cancel_right2 [simp]:
"(a div gcd b a) * gcd b a = (a::'a::semiring_gcd)"
<proof>

```

lemma real_root_normalize_cancel:
  assumes "0 < x" and "a ≠ 0" and "b > 0"
  shows "root (nat(snd(Rat.normalize(a,b))))
        (x ^ nat(fst(Rat.normalize(a,b)))) =
        root (nat b) (x ^ (nat a))"
⟨proof⟩

lemma powrat_add_pos:
  assumes "0 < x" and "0 < r" and "0 < s"
  shows "x pow_Q (r + s) = (x pow_Q r) * (x pow_Q s)"
⟨proof⟩

lemma powrat_add_neg:
  assumes "0 < x" and "r < 0" and "s < 0"
  shows "x pow_Q (r + s) = (x pow_Q r) * (x pow_Q s)"
⟨proof⟩

lemma powrat_add_neg_pos:
  assumes pos_x: "0 < x" and
         neg_r: "r < 0" and
         pos_s: "0 < s"
  shows "x pow_Q (r + s) = (x pow_Q r) * (x pow_Q s)"
⟨proof⟩

lemma powrat_add_pos_neg:
  "[[ 0 < x; 0 < r; s < 0 ]]"
  ⇒ "x pow_Q (r + s) = (x pow_Q r) * (x pow_Q s)"
⟨proof⟩

lemma powrat_add:
  assumes "0 < x"
  shows "x pow_Q (r + s) = (x pow_Q r) * (x pow_Q s)"
⟨proof⟩

lemma powrat_diff:
  "0 < x ⇒ x pow_Q (a - b) = x pow_Q a / x pow_Q b"
⟨proof⟩

lemma powrat_mult_pos:
  assumes "0 < x" and "0 < r" and "0 < s"
  shows "x pow_Q (r * s) = (x pow_Q r) pow_Q s"
⟨proof⟩

lemma powrat_mult_neg:
  assumes "0 < x" "r < 0" and "s < 0"
  shows "x pow_Q (r * s) = (x pow_Q r) pow_Q s"
⟨proof⟩

```

```

lemma powrat_mult_neg_pos:
  assumes "0 < x" and "r < 0" and "0 < s"
  shows "x pow_Q (r * s) = (x pow_Q r) pow_Q s"
  <proof>

lemma powrat_mult_pos_neg:
  assumes "0 < x" and "0 < r" and "s < 0"
  shows "x pow_Q (r * s) = (x pow_Q r) pow_Q s"
  <proof>

lemma powrat_mult:
  assumes "0 < x" shows "x pow_Q (r * s) = (x pow_Q r) pow_Q s"
  <proof>

lemma powrat_less_mono:
  assumes "r < s" and "1 < x"
  shows "x pow_Q r < x pow_Q s"
  <proof>

lemma power_le_imp_le_base2:
  "[[ (a::'a::linordered_semiodom) ^ n ≤ b ^ n; 0 ≤ b; 0 < n ]
  ⇒ a ≤ b"
  <proof>

lemma powrat_le_mono:
  assumes "r ≤ s" and "1 ≤ x"
  shows "x pow_Q r ≤ x pow_Q s"
  <proof>

lemma powrat_less_cancel:
  "[[ x pow_Q r < x pow_Q s; 1 < x ] ⇒ r < s"
  <proof>

lemma powrat_less_cancel_iff [simp]:
  "1 < x ⇒ (x pow_Q r < x pow_Q s) = (r < s)"
  <proof>

lemma powrat_le_cancel_iff [simp]:
  "1 < x ⇒ (x pow_Q r ≤ x pow_Q s) = (r ≤ s)"
  <proof>

lemma power_inject_exp_less_one [simp]:
  "[[ 0 < a; (a::'a::{linordered_field}) < 1 ]
  ⇒ a ^ m = a ^ n ↔ m = n"
  <proof>

```

lemma *power_inject_exp_strong* [simp]:

$$\llbracket 0 < a; (a :: 'a :: \{\text{linordered_field}\}) \neq 1 \rrbracket$$

$$\implies a^m = a^n \iff m = n$$
 <proof>

lemma *nat_eq_cancel*: $0 < a \implies 0 < b \implies (\text{nat } a = \text{nat } b) = (a = b)$
 <proof>

lemma *powrat_inject_exp* [simp]:
 $1 < x \implies (x \text{ pow}_Q r = x \text{ pow}_Q s) = (s = r)$
 <proof>

lemma *powrat_inject_exp_less_one* [simp]:
 assumes $0 < x$ and $x < 1$
 shows $(x \text{ pow}_Q r = x \text{ pow}_Q s) = (s = r)$
 <proof>

lemma *powrat_inject_exp_strong* [simp]:
 $\llbracket 0 < x; x \neq 1 \rrbracket \implies (x \text{ pow}_Q r = x \text{ pow}_Q s) = (s = r)$
 <proof>

lemma *powrat_less_1_cancel_iff* [simp]:
 assumes $x_0: 0 < x$ and $x_1: x < 1$
 shows $(x \text{ pow}_Q r < x \text{ pow}_Q s) = (s < r)$
 <proof>

lemma *powrat_le_1_cancel_iff* [simp]:
 $\llbracket 0 < x; x < 1 \rrbracket \implies (x \text{ pow}_Q r \leq x \text{ pow}_Q s) = (s \leq r)$
 <proof>

lemma *powrat_ge_one*: $x \geq 1 \implies r \geq 0 \implies x \text{ pow}_Q r \geq 1$
 <proof>

lemma *isCont_powrat*:
 assumes $0 < x$ shows $\text{isCont } (\lambda x. x \text{ pow}_Q r) x$
 <proof>

lemma *LIMSEQ_powrat_base*:
 $\llbracket X \longrightarrow a; a > 0 \rrbracket \implies (\lambda n. (X n) \text{ pow}_Q q) \longrightarrow a \text{ pow}_Q q$
 <proof>

lemma *powrat_inverse_of_nat_ge_one* [simp]:
 $a \geq 1 \implies a \text{ pow}_Q (\text{inverse } (\text{of_nat } n)) \geq 1$
 <proof>

lemma *powrat_inverse_of_nat_le_self* [simp]:

assumes "1 ≤ a" shows "a pow_Q inverse (rat_of_nat n) ≤ a"
<proof>

lemma BseqI2': "∀n≥N. norm (X n) ≤ K ⇒ Bseq X"
<proof>

lemma Bseq_powrat_inverse_of_nat_ge_one:
"a ≥ 1 ⇒ Bseq (λn. a pow_Q (inverse (of_nat n)))"
<proof>

lemma decseq_powrat_inverse_of_nat_ge_one:
"a ≥ 1 ⇒ decseq (λn. a pow_Q (inverse (of_nat (Suc n))))"
<proof>

lemma convergent_powrat_inverse_Suc_of_nat_ge_one:
assumes "a ≥ 1"
shows "convergent (λn. a pow_Q (inverse (of_nat (Suc n))))"
<proof>

lemma convergent_powrat_inverse_of_nat_ge_one:
assumes "a ≥ 1" shows "convergent (λn. a pow_Q (inverse (of_nat n)))"
<proof>

lemma LIMSEQ_powrat_inverse_of_nat_ge_one:
assumes "a ≥ 1" shows "(λn. a pow_Q (inverse (of_nat n))) → 1"
<proof>

lemma LIMSEQ_powrat_inverse_of_nat_pos_less_one:
assumes a0: "0 < a" and a1: "a < 1"
shows "(λn. a pow_Q (inverse (of_nat n))) → 1"
<proof>

lemma LIMSEQ_powrat_inverse_of_nat:
"a > 0 ⇒ (λn. a pow_Q (inverse (of_nat n))) → 1"
<proof>

lemma real_root_eq_powrat_inverse:
assumes "n > 0" shows "root n x = x pow_Q (inverse (of_nat n))"
<proof>

lemma powrat_power_eq:
"0 < a ⇒ a pow_Q rat_of_nat n = a ^ n"
<proof>

end

2 Real Exponents via Limits

```
theory RealPower
imports RatPower
begin

instance rat :: ab_group_add
⟨proof⟩

instance rat :: field
⟨proof⟩

instance rat :: comm_ring_1
⟨proof⟩

instantiation rat :: dist
begin
definition rat_dist_def:
  "dist x y = of_rat (abs(y - x))"

instance ⟨proof⟩
end

instantiation rat :: dist_norm
begin
definition rat_norm_def:
  "norm (q::rat) = of_rat |q|"

instance
⟨proof⟩
end

instantiation rat :: metric_space
begin

definition uniformity_rat_def [code del]:
  "(uniformity :: (rat × rat) filter) =
    (INF e∈{0 <..}. principal {(x::rat), y}. dist x y < e)"

definition open_rat_def [code del]:
  "open (U :: rat set) ↔
    (∀x∈U. eventually (λ(x', y). x' = x → y ∈ U) uniformity)"

instance
⟨proof⟩

end

instance rat :: topological_space
```


<proof>

lemma LIMSEQ_squeeze:

assumes abc: " $\forall n. a\ n \leq b\ n \wedge b\ n \leq c\ n$ "

and alim: " $a \longrightarrow (L::\text{real})$ "

and clim: " $c \longrightarrow L$ " shows " $b \longrightarrow L$ "

<proof>

primrec incratsseq :: "real \Rightarrow nat \Rightarrow rat" where

"incratsseq x 0 = ($\epsilon q. x - 1 < \text{of_rat } q \wedge \text{of_rat } q < x$)"

| "incratsseq x (Suc n) =

($\epsilon q. \max (\text{of_rat}(\text{incratsseq } x\ n)) (x - 1/(n + 2)) < \text{of_rat } q \wedge$
of_rat q < x)"

lemma incratsseq_0_gt [simp]: " $x - 1 < \text{of_rat}(\text{incratsseq } x\ 0)$ "

<proof>

lemma incratsseq_0_less [simp]: " $\text{of_rat}(\text{incratsseq } x\ 0) < x$ "

<proof>

declare incratsseq.simps [simp del]

lemma incratsseq_ub [simp]:

"real_of_rat (incratsseq x n) < x"

<proof>

lemma incratsseq_incseq [simp]:

"incratsseq x n < incratsseq x (Suc n)"

<proof>

lemma incratsseq_lb [simp]: " $x - 1/(\text{Suc } n) < \text{real_of_rat } (\text{incratsseq } x\ n)$ "

<proof>

lemma incseq_incratsseq [simp]:

"incseq (incratsseq x)"

<proof>

lemma LIMSEQ_rat_real_ex:

" $\exists r. \text{incseq } r \wedge (\lambda n. \text{of_rat } (r\ n)) \longrightarrow (x::\text{real})$ "

<proof>

lemma incseq_incseq_powrat:

" $[1 \leq a; \text{incseq } r] \implies \text{incseq } (\lambda n. a \text{ pow}_Q (r\ n))$ "

<proof>

lemma ex_less_of_rat: " $\exists r. (x :: 'a :: \text{archimedean_field}) < \text{of_rat } r$ "

<proof>

lemma *powrat_incseq_bounded*:

" $[1 \leq a; \forall n. r\ n < \text{of_rat } q; \text{incseq } r] \implies a \text{ pow}_Q (r\ n) \leq a \text{ pow}_Q q$ "
<proof>

lemma *Bseq_powrat_incseq*:

assumes " $1 \leq a$ "
and "*incseq* *r*"
and " $\forall n. \text{of_rat}(r\ n) \leq (x :: 'a :: \text{archimedean_field})$ "
shows "*Bseq* $(\lambda n. a \text{ pow}_Q (r\ n))$ "
<proof>

lemma *convergent_powrat_incseq*:

" $[1 \leq a; \text{incseq } r; \forall n. r\ n \leq x] \implies \text{convergent } (\lambda n. a \text{ pow}_Q (r\ n))$ "
<proof>

definition

incseq_of :: "*real* \implies (*nat* \implies *rat*)" **where**
"*incseq_of* *x* = (*SOME* *r. incseq* *r* \wedge $(\lambda n. \text{of_rat } (r\ n)) \longrightarrow x$)"

lemma *LIMSEQ_incseq_of [simp]*:

" $(\lambda n. \text{of_rat } (\text{incseq_of } x\ n)) \longrightarrow x$ "
<proof>

lemma *incseq_incseq_of [simp]*:

"*incseq* (*incseq_of* *x*)"
<proof>

lemma *incseq_of_Suc [simp]*:

"*incseq_of* *x* *n* \leq *incseq_of* *x* (*Suc* *n*)"
<proof>

lemma *incseq_of_rat_incseq_of*:

"*incseq* $(\lambda n. \text{of_rat}(\text{incseq_of } x\ n)) :: 'a :: \text{linordered_field}$ "
<proof>

lemma *incseq_of_rat*:

"*incseq* *s* \implies *incseq* $(\lambda n. \text{of_rat}(s\ n)) :: 'a :: \text{linordered_field}$ "
<proof>

lemma *incseq_rat_le_real*:

" $[\text{incseq } s; (\lambda n. \text{of_rat } (s\ n)) \longrightarrow x] \implies \text{of_rat } (s\ n) \leq (x :: \text{real})$ "
<proof>

lemma *incseq_of_le_self*: "*of_rat*(*incseq_of* *x* *n*) \leq *x*"

<proof>

lemma *powrat_incseq_of_bounded*:
 "[$1 \leq a; x < \text{of_rat } r$] $\implies a \text{ pow}_Q (\text{incseq_of } x \ n) \leq a \text{ pow}_Q r$ "
 <proof>

lemma *incseq_powrat_insec_of*:
 " $1 \leq a \implies \text{incseq } (\lambda n. a \text{ pow}_Q (\text{incseq_of } x \ n))$ "
 <proof>

lemma *Bseq_powrat_incseq_of*: " $1 \leq a \implies Bseq (\lambda n. a \text{ pow}_Q (\text{incseq_of } x \ n))$ "
 <proof>

lemma *convergent_powrat_incseq_of*: " $1 \leq a \implies \text{convergent } (\lambda n. a \text{ pow}_Q (\text{incseq_of } x \ n))$ "
 <proof>

We're now ready to define real exponentation.

definition
powa :: "[real,real] \Rightarrow real" (infixr "powa" 80) where
 "a powa x = (THE y. ($\lambda n. a \text{ pow}_Q (\text{incseq_of } x \ n)$) \longrightarrow y)"

real exponents.

definition
powreal :: "[real,real] \Rightarrow real" (infixr "pow_R" 80) where
 "a pow_R x = (if $0 < a \wedge a < 1$ then (inverse a) powa (-x)
 else if $a \geq 1$ then a powa x else 0)"

lemma *powreal_eq_powa*:
 " $a \geq 1 \implies a \text{ pow}_R x = a \text{ powa } x$ "
 <proof>

lemma *LIMSEQ_powrat_incseq_of_ex1*:
 " $1 \leq a \implies \exists !y. (\lambda n. a \text{ pow}_Q (\text{incseq_of } x \ n)) \longrightarrow y$ "
 <proof>

lemma *LIMSEQ_powa*:
 " $1 \leq a \implies (\lambda n. a \text{ pow}_Q (\text{incseq_of } x \ n)) \longrightarrow a \text{ powa } x$ "
 <proof>

lemma *lemma_incseq_incseq_diff_inverse*:
 " $\text{incseq } s \implies \text{incseq } (\lambda n. (s \ n :: \text{rat}) - 1/\text{of_nat}(\text{Suc } n))$ "
 <proof>

lemma *lemma_incseq_diff_inverse_ub*:
 assumes "incseq s"
 and " $(\lambda n. \text{of_rat } (s \ n)) \longrightarrow x$ "
 shows " $\text{of_rat}(s \ n - 1/\text{of_nat}(\text{Suc } n)) < (x :: \text{real})$ "
 <proof>

```

lemma lemma_LIMSEQ_incseq_diff_inverse:
  assumes "(λn. of_rat (s n)) ⟶ x"
  shows "(λn. of_rat(s n - 1/of_nat(Suc n))) ⟶ (x::real)"
⟨proof⟩

lemma lemma_LIMSEQ_powrat_diff_inverse:
  assumes "1 ≤ a"
  and "(λn. a pow_Q (s n)) ⟶ y"
  shows "(λn. a pow_Q (s n - 1/of_nat(Suc n))) ⟶ y"
⟨proof⟩

lemma lemma_LIMSEQ_powrat_diff_inverse2:
  assumes "1 < a"
  and "(λn. a pow_Q (s n - 1/of_nat(Suc n))) ⟶ y"
  shows "(λn. a pow_Q (s n)) ⟶ y"
⟨proof⟩

lemma lemma_seq_point_gt_ex:
  "[[ (λn. of_rat (r n)) ⟶ (x::real); y < x ]]"
  ⇒ ∃ (m::nat). y < of_rat(r m)"
⟨proof⟩

lemma lemma_seq_point_gt_ex2:
  "[[ (λn. of_rat (r n)) ⟶ (x::real); of_rat y < x ]]"
  ⇒ (∃ m. y < r m)"
⟨proof⟩

primrec interlaced_index :: "(nat ⇒ rat) ⇒ (nat ⇒ rat) ⇒ nat ⇒ nat"
where
  "interlaced_index r s 0 = 0"
| "interlaced_index r s (Suc n) =
    (LEAST m. if odd n then r (interlaced_index r s n) < s m
    else s (interlaced_index r s n) < r m)"

definition interlaced_seq :: "(nat ⇒ rat) ⇒ (nat ⇒ rat) ⇒ nat ⇒ rat"
where
  "interlaced_seq r s n = (if odd n then r (interlaced_index r s n)
    else s (interlaced_index r s n))"

lemma incseq_interlaced_seq:
  assumes "(λn. of_rat (r n)) ⟶ (x::real)"
  and "(λn. of_rat (s n)) ⟶ (x::real)"
  and "∀ n. of_rat (r n) < x"
  and "∀ n. of_rat (s n) < x"
  shows "incseq (interlaced_seq r s)"
⟨proof⟩

lemma incseq_of_rat_interlaced_seq:

```

```

"[[ (λn. of_rat (r n)) → (x::real);
  (λn. of_rat (s n)) → (x::real);
  ∀n. of_rat (r n) < x; ∀n. of_rat (s n) < x ]]
⇒ incseq (λn. real_of_rat (interlaced_seq r s n))"
⟨proof⟩

lemma interlaced_seq_bounded:
"[[ ∀n. of_rat (r n) < x; ∀n. of_rat (s n) < x ]]
⇒ of_rat (interlaced_seq r s n) < x"
⟨proof⟩

lemma convergent_interlaced_seq:
assumes "(λn. of_rat (r n)) → (x::real)"
and "(λn. of_rat (s n)) → (x::real)"
and "∀n. of_rat (r n) < x"
and "∀n. of_rat (s n) < x"
shows "convergent (λn. real_of_rat (interlaced_seq r s n))"
⟨proof⟩

lemma convergent_powrat_interlaced_seq:
"[[ 1 ≤ a; (λn. of_rat (r n)) → (x::real);
  (λn. of_rat (s n)) → (x::real);
  ∀n. of_rat (r n) < x; ∀n. of_rat (s n) < x ]]
⇒ convergent (λn. a pow_Q (interlaced_seq r s n))"
⟨proof⟩

lemma LIMSEQ_even_odd_subseq_LIMSEQ:
assumes "(λn. (X (2 * n))) → a" "(λn. (X (Suc(2 * n)))) → a"
shows "X → (a :: 'a::real_normed_vector)"
⟨proof⟩

lemma incseqD2: "incseq r ⇒ r m < r n ⇒ m < n"
⟨proof⟩

lemma subseq_interlaced_index_even:
assumes "incseq r"
and "incseq s"
and "(λn. of_rat (r n)) → (x::real)"
and "(λn. of_rat (s n)) → (x::real)"
and "∀n. of_rat (r n) < x"
and "∀n. of_rat (s n) < x"
shows "strict_mono (λn. interlaced_index r s (2 * n))"
⟨proof⟩

lemma subseq_interlaced_index_odd:
assumes "incseq r"
and "incseq s"

```

```

and "(λn. of_rat (r n)) ⟶ (x::real)"
and "(λn. of_rat (s n)) ⟶ (x::real)"
and "∀n. of_rat (r n) < x"
and "∀n. of_rat (s n) < x"
shows "strict_mono (λn. interlaced_index r s (Suc (2 * n)))"
⟨proof⟩

lemma interlaced_seq_even:
  "interlaced_seq r s (2*n) = s (interlaced_index r s (2*n))"
  ⟨proof⟩

lemma interlaced_seq_odd:
  "interlaced_seq r s (Suc (2*n)) = r (interlaced_index r s (Suc (2*n)))"
  ⟨proof⟩

lemma powa_indep_incseq_of:
  assumes "1 ≤ a"
  and "incseq r"
  and "incseq s"
  and "(λn. real_of_rat (r n)) ⟶ x"
  and "(λn. real_of_rat (s n)) ⟶ x"
  and "(λn. a pow_Q (r n)) ⟶ y"
  and "(λn. a pow_Q (s n)) ⟶ z"
shows "y = z"
⟨proof⟩

lemma powa_indep_incseq_of':
  "[[ 1 ≤ a; incseq r;
    (λn. real_of_rat (r n)) ⟶ x;
    (λn. a pow_Q (r n)) ⟶ y ]]
  ⇒ (λn. a pow_Q (incseq_of x n)) ⟶ y"
  ⟨proof⟩

lemma lemma_incseq_incseq_of_diff_inverse:
  "incseq (λn. incseq_of x n - 1/of_nat(Suc n))"
  ⟨proof⟩

lemma lemma_incseq_of_diff_inverse_ub:
  "of_rat(incseq_of x n - 1/of_nat(Suc n)) < x"
  ⟨proof⟩

lemma lemma_LIMSEQ_incseq_of_diff_inverse:
  "(λn. of_rat(incseq_of x n - 1/of_nat(Suc n))) ⟶ x"
  ⟨proof⟩

lemma powa_add:
  assumes "1 ≤ a"

```

shows "a powa (x + y) = a powa x * a powa y"
 <proof>

lemma real_inverse_ge_one_lemma:
 "[0 < (a::real); a < 1] \implies inverse a \geq 1"
 <proof>

lemma real_inverse_gt_one_lemma:
 "[0 < (a::real); a < 1] \implies inverse a > 1"
 <proof>

lemma real_inverse_bet_one_one_lemma:
 "1 < (a::real) \implies 0 < inverse a \wedge inverse a < 1"
 <proof>

lemma powreal_add:
 "a pow_R (x + y) = a pow_R x * a pow_R y"
 <proof>

lemma powa_one_eq_one [simp]: "1 powa a = 1"
 <proof>

lemma powreal_one_eq_one [simp]: "1 pow_R a = 1"
 <proof>

lemma powa_zero_eq_one [simp]: "a \geq 1 \implies a powa 0 = 1"
 <proof>

lemma powreal_zero_eq_one [simp]: "a > 0 \implies a pow_R 0 = 1"
 <proof>

lemma powr_zero_eq_one_iff [simp]: "x pow_R 0 = (if x \leq 0 then 0 else 1)"
 <proof>

lemma powa_one_gt_zero [simp]: "1 \leq a \implies a powa 1 = a"
 <proof>

lemma powa_minus_one:
 assumes "1 \leq a" shows "a powa -1 = inverse a"
 <proof>

lemma powreal_minus_one: "0 \leq a \implies a pow_R -1 = inverse a"
 <proof>

lemma powreal_one [simp]: "a \geq 0 \implies a pow_R 1 = a"
 <proof>

lemma powa_gt_zero:

```

    assumes "a ≥ 1"
    shows "a powa x > 0"
  ⟨proof⟩

lemma powreal_gt_zero: "a > 0 ⇒ a powR x > 0"
  ⟨proof⟩

lemma powreal_not_zero: "a > 0 ⇒ a powR x ≠ 0"
  ⟨proof⟩

lemma powreal_minus:
  "a powR -x = inverse (a powR x)"
  ⟨proof⟩

lemma powreal_minus_base_ge_one:
  "a powR (-x) = (inverse a) powR x"
  ⟨proof⟩

lemma powreal_inverse:
  "inverse (a powR x) = (inverse a) powR x"
  ⟨proof⟩

lemma powa_minus: "a ≥ 1 ⇒ a powa (-x) = inverse (a powa x)"
  ⟨proof⟩

lemma powa_mult_base:
  assumes "1 ≤ a" and "1 ≤ b"
  shows "(a * b) powa x = (a powa x) * (b powa x)"
  ⟨proof⟩

lemma powreal_mult_base_lemma1:
  "[[ 1 ≤ a; 1 ≤ b ]
   ⇒ (a * b) powR x = (a powR x) * (b powR x)"
  ⟨proof⟩

lemma powreal_mult_base_lemma2:
  assumes "1 ≤ a"
  and "0 < b"
  and "b < 1"
  shows "(a * b) powR x = (a powR x) * (b powR x)"
  ⟨proof⟩

lemma powreal_mult_base_lemma3:
  assumes "0 < a"
  and "a < 1"
  and "0 < b"
  and "b < 1"
  shows "(a * b) powR x = (a powR x) * (b powR x)"

```


<proof>

lemma powreal_mult_base:

assumes " $0 \leq a$ " and " $0 \leq b$ "

shows " $(a * b) \text{ pow}_R x = (a \text{ pow}_R x) * (b \text{ pow}_R x)$ "

<proof>

lemma incseq_le_all: " $\text{incseq } X \implies X \longrightarrow L \implies \forall n. X n \leq (L::\text{real})$ "

<proof>

lemma powa_powrat_eq:

assumes " $a \geq 1$ " shows " $a \text{ pow}_a (\text{of_rat } q) = a \text{ pow}_Q q$ "

<proof>

lemma realpow_powrat_eq: " $a > 0 \implies a \text{ pow}_R (\text{of_rat } q) = a \text{ pow}_Q q$ "

<proof>

lemma LIMSEQ_real_root:

" $\llbracket X \longrightarrow a; m > 0 \rrbracket \implies (\lambda n. \text{root } m (X n)) \longrightarrow (\text{root } m a)$ "

<proof>

lemma powa_powrat_lemma1:

assumes " $a \geq 1$ " and " $p \geq 0$ "

shows " $(a \text{ pow}_a x) \text{ pow}_Q p = (a \text{ pow}_a (x * \text{of_rat } p))$ "

<proof>

lemma powa_powrat_lemma2:

assumes " $a \geq 1$ " and " $p < 0$ "

shows " $(a \text{ pow}_a x) \text{ pow}_Q p = (a \text{ pow}_a (x * \text{of_rat } p))$ "

<proof>

lemma powa_powrat_lemma:

" $a \geq 1 \implies (a \text{ pow}_a x) \text{ pow}_Q p = (a \text{ pow}_a (x * \text{of_rat } p))$ "

<proof>

lemma LIMSEQ_iff2:

fixes $L :: "'a::\text{metric_space}$ "

shows " $(X \longrightarrow L) = (\forall m::\text{nat}>0. \exists no. \forall n \geq no. \text{dist } (X n) L < \text{inverse } m)$ "

<proof>

lemma LIM_def2:

" $f \rightarrow a \rightarrow L = (\forall m::\text{nat}>0. \exists s>0. \forall x. x \neq a \wedge \text{dist } x a < s \longrightarrow \text{dist } (f x) L < \text{inverse } m)$ "

for $a :: "'a::\text{metric_space}$ " and $L :: "'b::\text{metric_space}$ "

<proof>

```

lemma powa_ge_one:
  assumes "a ≥ 1"
  and "x ≥ 0"
  shows "a powa x ≥ 1"
  <proof>

lemma powreal_ge_one: "a ≥ 1 ⇒ x ≥ 0 ⇒ a powR x ≥ 1"
  <proof>

lemma powreal_ge_one2:
  "[[ 0 < a; a < 1; x ≤ 0 ] ⇒ a powR x ≥ 1"
  <proof>

lemma inverse_of_real_nat_of_rat_of_nat:
  "inverse (real_of_nat n) = of_rat (inverse (of_nat n))"
  <proof>

lemma LIMSEQ_powa_inverse_of_nat:
  "a ≥ 1 ⇒ (λn. a powa inverse (real_of_nat n)) → 1"
  <proof>

lemma incseq_of_le_mono:
  assumes "r ≤ s"
  shows "∃N. ∀n≥N. incseq_of r n ≤ incseq_of s n"
  <proof>

lemma powa_le_mono:
  assumes "r ≤ s"
  and "a ≥ 1"
  shows "a powa r ≤ a powa s"
  <proof>

lemma powreal_le_mono:
  "[[ r ≤ s; a ≥ 1 ] ⇒ a powR r ≤ a powR s"
  <proof>

lemma powreal_le_anti_mono:
  "[[ r ≤ s; 0 < a; a < 1 ] ⇒ a powR r ≥ a powR s"
  <proof>

lemma powreal_less_cancel:
  "[[ a powR r < a powR s; a ≥ 1 ] ⇒ r < s"
  <proof>

lemma powa_less_mono:
  assumes "r < s" and "a > 1"
  shows "a powa r < a powa s"
  <proof>

```

```

lemma powreal_less_anti_mono:
  assumes "r < s"
  and "0 < a"
  and "a < 1"
shows "a powR r > a powR s"
⟨proof⟩

lemma powreal_less_mono:
  "[[ r < s; a > 1 ] ] ⇒ a powR r < a powR s"
⟨proof⟩

lemma powa_le_cancel:
  "[[ a powa r ≤ a powa s; a > 1 ] ] ⇒ r ≤ s"
⟨proof⟩

lemma powreal_le_cancel:
  "[[ a powR r ≤ a powR s; a > 1 ] ] ⇒ r ≤ s"
⟨proof⟩

lemma powreal_less_cancel_iff [simp]:
  "1 < a ⇒ (a powR r < a powR s) = (r < s)"
⟨proof⟩

lemma powreal_le_cancel_iff [simp]:
  "1 < a ⇒ (a powR r ≤ a powR s) = (r ≤ s)"
⟨proof⟩

lemma powreal_inject_exp1 [simp]:
  "1 < a ⇒ (a powR r = a powR s) = (s = r)"
⟨proof⟩

lemma powreal_eq_one_iff [simp]:
  "a powR x = 1 ⇔ x = 0" if "a > 1"
⟨proof⟩

lemma powreal_inject_base_less_one [simp]:
  "0 < a ⇒ a < 1 ⇒ (a powR r = a powR s) = (s = r)"
⟨proof⟩

lemma powreal_inject [simp]:
  "0 < a ⇒ a ≠ 1 ⇒ (a powR r = a powR s) = (s = r)"
⟨proof⟩

lemma powreal_gt_one: "a > 1 ⇒ x > 0 ⇒ a powR x > 1"
⟨proof⟩

lemma isCont_powa_exponent_at_zero:
  assumes "a > 1" shows "isCont (λx. a powa x) 0"

```

<proof>

lemma LIM_powa_exponent_at_zero: "1 < a \implies ($\lambda h.$ a powa h) $\xrightarrow{0}$ 1"
<proof>

lemma isCont_powa_exponent_gt_one:
 assumes "a > 1"
 shows "isCont ($\lambda x.$ a powa x) x"
<proof>

lemma isCont_powreal_exponent_gt_one:
 "a > 1 \implies isCont ($\lambda x.$ a pow_R x) x"
<proof>

lemma isCont_powreal_exponent_less_one:
 assumes "0 < a"
 and "a < 1"
 shows "isCont ($\lambda x.$ a pow_R x) x"
<proof>

lemma isCont_powreal_exponent:
 assumes a_gt_0: "0 < a" shows "isCont ($\lambda x.$ a pow_R x) x"
<proof>

lemma real_of_rat_abs:
 "real_of_rat(abs a) = abs(of_rat a)"
<proof>

lemma isCont_powrat_exponent:
 assumes "0 < a" shows "isCont ($\lambda x.$ a pow_Q x) x"
<proof>

lemma LIMSEQ_powrat_exponent:
 "[[X \longrightarrow x; a > 0]] \implies ($\lambda n.$ a pow_Q (X n)) \longrightarrow a pow_Q x"
<proof>

lemma powa_mult:
 assumes "1 \leq a" and "0 \leq x"
 shows "(a powa x) powa y = a powa (x * y)"
<proof>

lemma powreal_mult1:
 "[[1 \leq a; 0 \leq x]] \implies (a pow_R x) pow_R y = a pow_R (x * y)"
<proof>

lemma powreal_mult2:

assumes "0 < a" and "a < 1" and "0 ≤ x"
 shows "(a pow_R x) pow_R y = a pow_R (x * y)"
 ⟨proof⟩

lemma powreal_mult3:
 "[0 < a; 0 ≤ x] ⇒ (a pow_R x) pow_R y = a pow_R (x * y)"
 ⟨proof⟩

lemma powreal_mult4:
 assumes a0: "0 < a" and x0: "x ≤ 0"
 shows "(a pow_R x) pow_R y = a pow_R (x * y)"
 ⟨proof⟩

lemma powreal_mult:
 "(a pow_R x) pow_R y = a pow_R (x * y)"
 ⟨proof⟩

lemma powreal_divide:
 "[0 ≤ a; 0 ≤ b] ⇒ (a/b) pow_R x = (a pow_R x) / (b pow_R x)"
 ⟨proof⟩

lemma powreal_divide2:
 "0 ≤ a ⇒ a pow_R (x - y) = (a pow_R x) / (a pow_R y)"
 ⟨proof⟩

lemma powreal_less_mono_base:
 assumes r0: "r > 0" and a0: "0 < a" and ab: "a < b"
 shows "a pow_R r < b pow_R r"
 ⟨proof⟩

lemma powreal_less_antimono_base:
 assumes "r < 0" and "0 < a" and "a < b"
 shows "a pow_R r > b pow_R r"
 ⟨proof⟩

lemma powa_power_eq:
 assumes "a ≥ 1" shows "a powa (of_nat n) = a ^ n"
 ⟨proof⟩

lemma powreal_power_eq:
 "a > 0 ⇒ a pow_R (of_nat n) = a ^ n"
 ⟨proof⟩

lemma powreal_power_eq2:
 "0 ≤ a ⇒ 0 < n ⇒ a ^ n = (if a = 0 then 0 else a pow_R (real n))"
 ⟨proof⟩

lemma powreal_mult_power: "a > 0 ⇒ a pow_R (n * x) = (a pow_R x) ^ n"

```

    <proof>

lemma powreal_int:
  assumes "x > 0"
  shows "x powR i = (if i ≥ 0 then x ^ nat i else 1 / x ^ nat (-i))"
  <proof>

lemma powreal_numeral: "0 ≤ x ⇒ x powR numeral n = x ^ numeral n"
  <proof>

lemma root_powreal_inverse:
  assumes "0 < n" and "0 ≤ x"
  shows "root n x = x powR (1/n)"
  <proof>

lemma powreal_inverse_of_nat_gt_one:
  "[[ 1 < a; n ≠ 0 ]] ⇒ a powR (inverse (of_nat n)) > 1"
  <proof>

end

```

3 Real Logarithms (Redefined)

```

theory Log
imports RealPower
begin

```

We can now directly define real logarithm of x to base a .

definition

```

  Log :: "[real,real] ⇒ real" where
  "Log a x = (THE y. a powR y = x)"

```

lemma *IVT_simple*:

```

  "[[ f (a::real) ≤ (y::real); y ≤ f b; a ≤ b;
    ∀ x. a ≤ x ∧ x ≤ b ⟶ isCont f x ]]
  ⟹ ∃ x. f x = y"
  <proof>

```

lemma *inj_on_powreal*:

```

  "0 < a ⟹ a ≠ 1 ⟹ inj_on (λx. a powR x) UNIV"
  <proof>

```

lemma *LIMSEQ_powreal_minus_nat*:

```

  "a > 1 ⟹ (λn. a powR (-real n)) ⟶ 0"
  <proof>

```

lemma *LIMSEQ_less_Ex*:

```

  "[[ X ⟶ (x::real); x < y ]] ⟹ ∃ n. X n < y"
  <proof>

```

```

lemma powreal_IVT_upper_lemma:
  assumes "a > (1::real)" and "x > 0"
  shows "∃n::nat. a powR (-real n) < x"
⟨proof⟩

lemma powreal_IVT_lower_lemma:
  assumes "a > (1::real)"
  and "x > 0"
  shows "∃n::nat. x < a powR (real n)"
⟨proof⟩

lemma powreal_surj:
  assumes "a > 1"
  and "x > 0"
  shows "∃y. a powR y = x"
⟨proof⟩

lemma powreal_surj2:
  "[[ 0 < a; a < 1; x > 0 ]] ⇒ ∃y. a powR y = x"
⟨proof⟩

lemma powreal_ex1_eq:
  assumes "a > 0"
  and "a ≠ 1"
  and "x > 0"
  shows "∃! y. a powR y = x"
⟨proof⟩

lemma powreal_Log_cancel [simp]:
  "[[ a > 0; a ≠ 1; x > 0 ]] ⇒ a powR (Log a x) = x"
⟨proof⟩

lemma Log_powreal_cancel [simp]:
  "[[ 0 < a; a ≠ 1 ]] ⇒ Log a (a powR y) = y"
⟨proof⟩

lemma Log_mult:
  "[[ 0 < a; a ≠ 1; 0 < x; 0 < y ]]
  ⇒ Log a (x * y) = Log a x + Log a y"
⟨proof⟩

lemma Log_one [simp]: "[[ 0 < a; a ≠ 1 ]] ⇒ Log a 1 = 0"
⟨proof⟩

lemma Log_eq_one [simp]: "[[ 0 < a; a ≠ 1 ]] ⇒ Log a a = 1"
⟨proof⟩

lemma Log_inverse:

```

" $[a > 0; a \neq 1; x > 0] \implies \text{Log } a (\text{inverse } x) = - \text{Log } a x$ "
<proof>

lemma *Log_divide*:
" $[0 < a; a \neq 1; 0 < x; 0 < y]$
 $\implies \text{Log } a (x/y) = \text{Log } a x - \text{Log } a y$ "
<proof>

lemma *Log_less_cancel_iff [simp]*:
assumes " $1 < a$ "
and " $0 < x$ "
and " $0 < y$ "
shows " $(\text{Log } a x < \text{Log } a y) = (x < y)$ "
<proof>

lemma *Log_inj*: assumes " $1 < b$ " **shows** "*inj_on* ($\text{Log } b$) $\{0 <..\}$ "
<proof>

lemma *Log_le_cancel_iff [simp]*:
" $[1 < a; 0 < x; 0 < y] \implies (\text{Log } a x \leq \text{Log } a y) = (x \leq y)$ "
<proof>

lemma *zero_less_Log_cancel_iff [simp]*:
" $1 < a \implies 0 < x \implies 0 < \text{Log } a x \longleftrightarrow 1 < x$ "
<proof>

lemma *zero_le_Log_cancel_iff [simp]*:
" $1 < a \implies 0 < x \implies 0 \leq \text{Log } a x \longleftrightarrow 1 \leq x$ "
<proof>

lemma *Log_less_zero_cancel_iff [simp]*:
" $1 < a \implies 0 < x \implies \text{Log } a x < 0 \longleftrightarrow x < 1$ "
<proof>

lemma *Log_le_zero_cancel_iff [simp]*:
" $1 < a \implies 0 < x \implies \text{Log } a x \leq 0 \longleftrightarrow x \leq 1$ "
<proof>

lemma *one_less_Log_cancel_iff [simp]*:
" $1 < a \implies 0 < x \implies 1 < \text{Log } a x \longleftrightarrow a < x$ "
<proof>

lemma *one_le_Log_cancel_iff [simp]*:
" $1 < a \implies 0 < x \implies 1 \leq \text{Log } a x \longleftrightarrow a \leq x$ "
<proof>

lemma *Log_less_one_cancel_iff [simp]*:
" $1 < a \implies 0 < x \implies \text{Log } a x < 1 \longleftrightarrow x < a$ "
<proof>


```
lemma Log_le_one_cancel_iff[simp]:  
  "1 < a  $\implies$  0 < x  $\implies$  Log a x  $\leq$  1  $\longleftrightarrow$  x  $\leq$  a"  
  <proof>
```

```
lemma Log_powreal:  
  assumes "0 < x"  
  and "1 < b"  
  and "b  $\neq$  1"  
  shows "Log b (x powR y) = y * Log b x"  
  <proof>
```

```
lemma Log_nat_power:  
  assumes "0 < x"  
  and "1 < b" and "b  $\neq$  1"  
  shows "Log b (x ^ n) = real n * Log b x"  
  <proof>
```

```
end
```