

Combinatorial q -Analogues

Manuel Eberl

January 24, 2025

Abstract

This entry defines the q -analogues of various combinatorial symbols, namely:

- The q -bracket $[n]_q = \frac{1-q^n}{1-q}$ for $n \in \mathbb{Z}$
- The q -factorial $[n]_q! = [1]_q [2]_q \cdots [n]_q$ for $n \in \mathbb{Z}$
- The q -binomial coefficients $\binom{n}{k}_q = \frac{[n]_q!}{[k]_q! [n-k]_q!}$ for $n, k \in \mathbb{N}$ (also known as Gaussian binomial coefficients or Gaussian polynomials)
- The infinite q -Pochhammer symbol $(a; q)_\infty = \prod_{n=0}^{\infty} (1 - aq^n)$
- Euler's ϕ function $\phi(q) = (q; q)_\infty$
- The finite q -Pochhammer symbol $(a; q)_n = (a; q)_\infty / (aq^n; q)_\infty$ for $n \in \mathbb{Z}$

Proofs for many basic properties are provided, notably for the q -binomial theorem:

$$(-a; q)_n = \prod_{k=0}^{n-1} (1 + aq^k) = \sum_{k=0}^n \binom{n}{k}_q a^k q^{k(k-1)/2}$$

Additionally, two identities of Euler are formalised that give power series expansions for $(a; q)_\infty$ and $1/(a; q)_\infty$ in powers of a :

$$(a; q)_\infty = \prod_{k=0}^{\infty} (1 - aq^k) = \sum_{n=0}^{\infty} \frac{(-a)^n q^{n(n-1)/2}}{(1-q) \cdots (1-q^n)}$$
$$\frac{1}{(a; q)_\infty} = \prod_{k=0}^{\infty} \frac{1}{1 - aq^k} = \sum_{n=0}^{\infty} \frac{a^n}{(1-q) \cdots (1-q^n)}$$

Contents

1	Auxiliary material	3
1.1	Additional facts about infinite products	3
1.2	Miscellanea	6
2	q-analogues of basic combinatorial symbols	8
2.1	The q -bracket $[n]_q$	8
2.2	The q -factorial $[n]_q!$	11
2.3	q -binomial coefficients $\binom{n}{k}_q$	14
2.4	The Gaussian polynomials	16
2.5	The finite Pochhammer symbol $(a; q)_n$	18
3	The infinite q-Pochhammer symbol $(a; q)_\infty$	23
3.1	Definition and basic properties	23
3.2	Uniform convergence and its consequences	25
3.3	Bounds for $(a; q)_n$ and $\binom{n}{k}_q$ in terms of $(a; q)_\infty$	26
3.4	Limits of the q -binomial coefficients	27
3.5	Useful identities	27
3.6	Two series expansions by Euler	28
3.7	Euler's function	30
4	q-binomial identities	31
4.1	The q -binomial theorem	31
4.2	The infinite q -binomial theorem	32
4.3	The q -Vandermonde identity	32

1 Auxiliary material

1.1 Additional facts about infinite products

```
theory More_Infinite_Products
  imports "HOL-Analysis.Analysis"
begin
```

```
lemma uniform_limit_singleton: "uniform_limit {x} f g F  $\longleftrightarrow$  (( $\lambda n. f$ 
n x)  $\longrightarrow$  g x) F"
  <proof>
```

```
lemma uniformly_convergent_on_singleton:
  "uniformly_convergent_on {x} f  $\longleftrightarrow$  convergent ( $\lambda n. f$  n x)"
  <proof>
```

```
lemma uniformly_convergent_on_subset:
  assumes "uniformly_convergent_on A f" "B  $\subseteq$  A"
  shows "uniformly_convergent_on B f"
  <proof>
```

```
lemma raw_has_prod_imp_nonzero:
  assumes "raw_has_prod f N P" "n  $\geq$  N"
  shows "f n  $\neq$  0"
  <proof>
```

```
lemma has_prod_imp_tendsto:
  fixes f :: "nat  $\Rightarrow$  'a :: {semidom, t2_space}"
  assumes "f has_prod P"
  shows "( $\lambda n. \prod_{k \leq n}. f k$ )  $\longrightarrow$  P"
  <proof>
```

```
lemma has_prod_imp_tendsto':
  fixes f :: "nat  $\Rightarrow$  'a :: {semidom, t2_space}"
  assumes "f has_prod P"
  shows "( $\lambda n. \prod_{k < n}. f k$ )  $\longrightarrow$  P"
  <proof>
```

```
lemma convergent_prod_tendsto_imp_has_prod:
  fixes f :: "nat  $\Rightarrow$  'a :: real_normed_field"
  assumes "convergent_prod f" "( $\lambda n. \prod_{i \leq n}. f i$ )  $\longrightarrow$  P"
  shows "f has_prod P"
  <proof>
```

```
lemma has_prod_group_nonzero:
```

```

fixes f :: "nat  $\Rightarrow$  'a :: {semidom, t2_space}"
assumes "f has_prod P" "k > 0" "P  $\neq$  0"
shows "(\ $\lambda$ n. ( $\prod$  i $\in$ {n*k.. $n$ *k+k}. f i)) has_prod P"
<proof>

```

```

lemma has_prod_group:
  fixes f :: "nat  $\Rightarrow$  'a :: real_normed_field"
  assumes "f has_prod P" "k > 0"
  shows "(\ $\lambda$ n. ( $\prod$  i $\in$ {n*k.. $n$ *k+k}. f i)) has_prod P"
<proof>

```

```

lemma has_prod_nonneg:
  assumes "f has_prod P" " $\bigwedge$ n. f n  $\geq$  (0::real)"
  shows "P  $\geq$  0"
<proof>

```

```

lemma has_prod_pos:
  assumes "f has_prod P" " $\bigwedge$ n. f n > (0::real)"
  shows "P > 0"
<proof>

```

```

lemma prod_ge_produinf:
  fixes f :: "nat  $\Rightarrow$  'a::{linordered_idom, linorder_topology}"
  assumes "f has_prod a" " $\bigwedge$ i. 0  $\leq$  f i" " $\bigwedge$ i. i  $\geq$  n  $\implies$  f i  $\leq$  1"
  shows "prod f {.. $n$ }  $\geq$  produinf f"
<proof>

```

```

lemma has_prod_less:
  fixes F G :: real
  assumes less: "f m < g m"
  assumes f: "f has_prod F" and g: "g has_prod G"
  assumes pos: " $\bigwedge$ n. 0 < f n" and le: " $\bigwedge$ n. f n  $\leq$  g n"
  shows "F < G"
<proof>

```

Cauchy's criterion for the convergence of infinite products, adapted to proving uniform convergence: let $f_k(x)$ be a sequence of functions such that

1. $f_k(x)$ has uniformly bounded partial products, i.e. there exists a constant C such that $\prod_{k=0}^m f_k(x) \leq C$ for all m and $x \in A$.
2. For any $\varepsilon > 0$ there exists a number $M \in \mathbb{N}$ such that, for any $m, n \geq M$ and all $x \in A$ we have $|\left(\prod_{k=m}^n f_k(x)\right) - 1| < \varepsilon$

Then $\prod_{k=0}^n f_k(x)$ converges to $\prod_{k=0}^{\infty} f_k(x)$ uniformly for all $x \in A$.

```

lemma uniformly_convergent_prod_Cauchy:

```

```

fixes f :: "nat  $\Rightarrow$  'a :: topological_space  $\Rightarrow$  'b :: {real_normed_div_algebra,
comm_ring_1, banach}"
assumes C: " $\bigwedge x m. x \in A \implies \text{norm } (\prod_{k < m}. f\ k\ x) \leq C$ "
assumes "  $\bigwedge e. e > 0 \implies \exists M. \forall x \in A. \forall m \geq M. \forall n \geq m. \text{dist } (\prod_{k=m..n}. f$ 
k x) 1 < e"
shows "uniformly_convergent_on A ( $\lambda N x. \prod_{n < N}. f\ n\ x$ )"
<proof>

```

By instantiating the set A in this result with a singleton set, we obtain the “normal” Cauchy criterion for infinite products:

```

lemma convergent_prod_Cauchy_sufficient:
fixes f :: "nat  $\Rightarrow$  'b :: {real_normed_div_algebra, comm_ring_1, banach}"
assumes "  $\bigwedge e. e > 0 \implies \exists M. \forall m\ n. M \leq m \longrightarrow m \leq n \longrightarrow \text{dist } (\prod_{k=m..n}. f$ 
k) 1 < e"
shows "convergent_prod f"
<proof>

```

We now prove that the Cauchy criterion for pointwise convergence is both necessary and sufficient.

```

lemma convergent_prod_Cauchy_necessary:
fixes f :: "nat  $\Rightarrow$  'b :: {real_normed_field, banach}"
assumes "convergent_prod f" "e > 0"
shows "  $\exists M. \forall m\ n. M \leq m \longrightarrow m \leq n \longrightarrow \text{dist } (\prod_{k=m..n}. f\ k) 1 <$ 
e"
<proof>

```

```

lemma convergent_prod_Cauchy_iff:
fixes f :: "nat  $\Rightarrow$  'b :: {real_normed_field, banach}"
shows "convergent_prod f  $\longleftrightarrow (\forall e > 0. \exists M. \forall m\ n. M \leq m \longrightarrow m \leq n \longrightarrow$ 
dist  $(\prod_{k=m..n}. f\ k) 1 < e)$ "
<proof>

```

```

lemma uniform_limit_suminf:
fixes f :: "nat  $\Rightarrow$  'a :: topological_space  $\Rightarrow$  'b :: {metric_space, comm_monoid_add}"
assumes "uniformly_convergent_on X ( $\lambda n x. \sum_{k < n}. f\ k\ x$ )"
shows "uniform_limit X ( $\lambda n x. \sum_{k < n}. f\ k\ x$ ) ( $\lambda x. \sum_k. f\ k\ x$ ) sequentially"
<proof>

```

```

lemma uniformly_convergent_on_prod:
fixes f :: "nat  $\Rightarrow$  'a :: topological_space  $\Rightarrow$  'b :: {real_normed_div_algebra,
comm_ring_1, banach}"
assumes cont: " $\bigwedge n. \text{continuous\_on } A (f\ n)$ "
assumes A: "compact A"
assumes conv_sum: "uniformly_convergent_on A ( $\lambda N x. \sum_{n < N}. \text{norm } (f$ 
n x))"
shows "uniformly_convergent_on A ( $\lambda N x. \prod_{n < N}. 1 + f\ n\ x$ )"
<proof>

```

```

lemma uniformly_convergent_on_prod':
  fixes f :: "nat  $\Rightarrow$  'a :: topological_space  $\Rightarrow$  'b :: {real_normed_div_algebra,
comm_ring_1, banach}"
  assumes cont: " $\bigwedge n$ . continuous_on A (f n)"
  assumes A: "compact A"
  assumes conv_sum: "uniformly_convergent_on A ( $\lambda N x$ .  $\sum_{n < N}$ . norm (f
n x - 1))"
  shows "uniformly_convergent_on A ( $\lambda N x$ .  $\prod_{n < N}$ . f n x)"
  <proof>

end
theory Q_Library
  imports "HOL-Analysis.Analysis" "HOL-Computational_Algebra.Computational_Algebra"
begin

```

1.2 Miscellanea

```

lemma prod_uminus: "( $\prod_{x \in A}$ . -f x :: 'a :: comm_ring_1) = (-1) ^ card
A * ( $\prod_{x \in A}$ . f x)"
  <proof>

```

```

lemma prod_diff_swap:
  fixes f :: "'a  $\Rightarrow$  'b :: comm_ring_1"
  shows "prod ( $\lambda x$ . f x - g x) A = (-1) ^ card A * prod ( $\lambda x$ . g x - f x)
A"
  <proof>

```

```

lemma prod_diff:
  fixes f :: "'a  $\Rightarrow$  'b :: field"
  assumes "finite A" "B  $\subseteq$  A" " $\bigwedge x$ . x  $\in$  B  $\implies$  f x  $\neq$  0"
  shows "prod f (A - B) = prod f A / prod f B"
  <proof>

```

```

lemma power_inject_exp':
  assumes "a  $\neq$  1" "a > 0" ("a :: linordered_semiodom)"
  shows "a ^ m = a ^ n  $\iff$  m = n"
  <proof>

```

```

lemma q_power_neq_1:
  assumes "norm (q :: 'a :: real_normed_div_algebra) < 1" "n > 0"
  shows "q ^ n  $\neq$  1"
  <proof>

```

```

lemma fls_nth_sum: "fls_nth ( $\sum_{x \in A}$ . f x) n = ( $\sum_{x \in A}$ . fls_nth (f x)
n)"
  <proof>

```

lemma *one_plus_fls_X_powi_eq*:
 "(1 + fls_X) powi n = fps_to_fls (fps_binomial (of_int n :: 'a :: field_char_0))"
 <proof>

lemma *bij_betw_imp_empty_iff*: "bij_betw f A B \implies A = {} \longleftrightarrow B = {}"
 <proof>

lemma *bij_betw_imp_Ex_iff*: "bij_betw f {x. P x} {x. Q x} \implies (\exists x. P x) \longleftrightarrow (\exists x. Q x)"
 <proof>

lemma *bij_betw_imp_Bex_iff*: "bij_betw f {x \in A. P x} {x \in B. Q x} \implies (\exists x \in A. P x) \longleftrightarrow (\exists x \in B. Q x)"
 <proof>

lemmas [derivative_intros del] = Deriv.DERIV_power_int
lemma *DERIV_power_int* [derivative_intros]:
 assumes [derivative_intros]: "(f has_field_derivative d) (at x within s)"
 and "n \geq 0 \vee f x \neq 0"
 shows "((λ x. power_int (f x) n) has_field_derivative
 (of_int n * power_int (f x) (n - 1) * d)) (at x within s)"
 <proof>

lemma *uniform_limit_compose'*:
 assumes "uniform_limit B (λ x y. f x y) (λ y. f' y) F" " \bigwedge y. y \in A \implies
 g y \in B"
 shows "uniform_limit A (λ x y. f x (g y)) (λ y. f' (g y)) F"
 <proof>

lemma *eventually_eventually_prod_filter1*:
 assumes "eventually P (F \times_F G)"
 shows "eventually (λ x. eventually (λ y. P (x, y)) G) F"
 <proof>

lemma *eventually_eventually_prod_filter2*:
 assumes "eventually P (F \times_F G)"
 shows "eventually (λ y. eventually (λ x. P (x, y)) F) G"
 <proof>

proposition *swap_uniform_limit'*:
 assumes f: " \forall_F n in F. (f n \longrightarrow g n) G"

```

assumes g: "(g  $\longrightarrow$  1) F"
assumes uc: "uniform_limit S f h F"
assumes ev: " $\forall_F$  x in G. x  $\in$  S"
assumes " $\neg$ trivial_limit F"
shows "(h  $\longrightarrow$  1) G"
<proof>

```

```

proposition swap_uniform_limit:
assumes f: " $\forall_F$  n in F. (f n  $\longrightarrow$  g n) (at x within S)"
assumes g: "(g  $\longrightarrow$  1) F"
assumes uc: "uniform_limit S f h F"
assumes nt: " $\neg$ trivial_limit F"
shows "(h  $\longrightarrow$  1) (at x within S)"
<proof>

```

Tannery's Theorem proves that, under certain boundedness conditions:

$$\lim_{x \rightarrow \bar{x}} \sum_{k=0}^{\infty} f(k, n) = \sum_{k=0}^{\infty} \lim_{x \rightarrow \bar{x}} f(k, n)$$

```

lemma tannerys_theorem:
fixes a :: "nat  $\Rightarrow$  _  $\Rightarrow$  'a :: {real_normed_algebra, banach}"
assumes limit: " $\bigwedge k$ . (( $\lambda n$ . a k n)  $\longrightarrow$  b k) F"
assumes bound: "eventually ( $\lambda(k,n)$ . norm (a k n)  $\leq$  M k) (at_top  $\times_F$  F)"
assumes "summable M"
assumes [simp]: "F  $\neq$  bot"
shows "eventually ( $\lambda n$ . summable ( $\lambda k$ . norm (a k n))) F  $\wedge$ 
summable ( $\lambda n$ . norm (b n))  $\wedge$ 
(( $\lambda n$ . suminf ( $\lambda k$ . a k n))  $\longrightarrow$  suminf b) F"
<proof>

```

end

2 q -analogues of basic combinatorial symbols

```

theory Q_Analogues
imports "HOL-Complex_Analysis.Complex_Analysis" Q_Library
begin

```

Various mathematical operations have generalisations in the form of q -analogues, usually in the sense that one recovers the original notion if we let $q \rightarrow 1$.

2.1 The q -bracket $[n]_q$

The q -bracket $[n]_q = \frac{1-q^n}{1-q}$ is the q -analogue of an integer n . The q -bracket has a removable singularity at $q = 1$ with $\lim_{q \rightarrow 1} [n]_q = n$.


```

definition qbracket :: "'a ⇒ int ⇒ 'a :: field" where
  "qbracket q n = (if q = 1 then of_int n else (1 - q powi n) / (1 - q))"

lemma qbracket_1_left [simp]: "qbracket 1 n = of_int n"
  ⟨proof⟩

lemma qbracket_0_0 [simp]: "qbracket 0 0 = 0"
  ⟨proof⟩

lemma qbracket_0_nonneg [simp]: "n ≠ 0 ⇒ qbracket 0 n = 1"
  ⟨proof⟩

lemma qbracket_0_left: "qbracket 0 n = (if n = 0 then 0 else 1)"
  ⟨proof⟩

lemma qbracket_0 [simp]: "qbracket q 0 = 0"
  ⟨proof⟩

lemma qbracket_1 [simp]: "qbracket q 1 = 1"
  ⟨proof⟩

lemma qbracket_2 [simp]: "qbracket q 2 = 1 + q"
  ⟨proof⟩

lemma qbracket_of_real: "qbracket (of_real q :: 'a :: real_field) n =
of_real (qbracket q n)"
  ⟨proof⟩

lemma qbracket_minus:
  assumes "q = 0 → n = 0"
  shows "qbracket q (-n) = -qbracket (inverse q) n / q"
  ⟨proof⟩

lemma qbracket_inverse:
  assumes "q = 0 → n = 0"
  shows "qbracket (inverse q) n = -q * qbracket q (-n)"
  ⟨proof⟩

lemma qbracket_nonneg_altdef: "n ≥ 0 ⇒ qbracket q n = (∑ k<nat n.
q ^ k)"
  ⟨proof⟩

lemma qbracket_nonpos_altdef:
  assumes n: "n ≤ 0" and [simp]: "q ≠ 0"
  shows "qbracket q n = -(q powi n * (∑ k<nat (-n). q ^ k))"
  ⟨proof⟩

lemma norm_qbracket_le:
  fixes q :: "'a :: real_normed_field"

```

```

    assumes "n ≥ 0" "norm q ≤ 1"
    shows "norm (qbracket q n) ≤ real_of_int n"
  ⟨proof⟩

lemma qbracket_add:
  assumes "q ≠ 0 ∨ (k + 1 = 0 ⟶ k = 0)"
  shows "qbracket q (k + 1) = qbracket q 1 * q powi k + qbracket q k"
  ⟨proof⟩

lemma qbracket_diff:
  assumes "q ≠ 0 ∨ (k = 1 ⟶ k = 0)"
  shows "qbracket q (k - 1) = qbracket q (-1) * q powi k + qbracket q k"
  ⟨proof⟩

lemma qbracket_diff':
  assumes "q ≠ 0 ∨ (k = 1 ⟶ k = 0)"
  shows "qbracket q (k - 1) = qbracket q k * q powi -1 + qbracket q (-1)"
  ⟨proof⟩

lemma qbracket_plus1: "q ≠ 0 ∨ n ≠ -1 ⟹ qbracket q (n + 1) = qbracket q n + q powi n"
  ⟨proof⟩

lemma qbracket_rec: "q ≠ 0 ∨ n ≠ 0 ⟹ qbracket q n = qbracket q (n-1) + q powi (n-1)"
  ⟨proof⟩

lemma qbracket_eq_0_iff:
  fixes q :: "'a :: field"
  shows "qbracket q n = 0 ⟷ (q = 1 ∧ of_int n = (0 :: 'a)) ∨ (q ≠ 1 ∧ q powi n = 1)"
  ⟨proof⟩

lemma continuous_on_qbracket [continuous_intros]:
  fixes q :: "'a::topological_space ⇒ 'b :: real_normed_field"
  assumes [continuous_intros]: "continuous_on A q"
  assumes "∧x. n < 0 ⟹ x ∈ A ⟹ q x ≠ 0"
  shows "continuous_on A (λx. qbracket (q x) n)"
  ⟨proof⟩

lemma tendsto_qbracket [tendsto_intros]:
  fixes q :: "'a::topological_space ⇒ 'b :: real_normed_field"
  assumes "(q ⟶ Q) F"
  assumes "n < 0 ⟹ Q ≠ 0"
  shows "((λx. qbracket (q x) n) ⟶ qbracket Q n) F"
  ⟨proof⟩

```

```

lemma continuous_qbracket [continuous_intros]:
  fixes q :: "'a::t2_space  $\Rightarrow$  'b :: real_normed_field"
  assumes "continuous F q"
  assumes "n < 0  $\implies$  q (netlimit F)  $\neq$  0"
  shows "continuous F ( $\lambda$ x. qbracket (q x) n)"
  <proof>

lemma has_field_derivative_qbracket_real [derivative_intros]:
  fixes q :: real
  assumes "q  $\neq$  0  $\vee$  n  $\geq$  0"
  defines "D  $\equiv$  (if q = 1 then of_int (n * (n - 1)) / 2
    else (1 - q powi n)/(1-q)^2 - of_int n * q powi (n-1)
  / (1-q))"
  shows "(( $\lambda$ q. qbracket q n) has_field_derivative D) (at q within A)"
  <proof>

lemma has_field_derivative_qbracket_complex [derivative_intros]:
  fixes q :: complex
  assumes "q  $\neq$  0  $\vee$  n  $\geq$  0"
  defines "D  $\equiv$  (if q = 1 then of_int (n * (n - 1)) / 2
    else (1 - q powi n)/(1-q)^2 - of_int n * q powi (n-1)
  / (1-q))"
  shows "(( $\lambda$ q. qbracket q n) has_field_derivative D) (at q within A)"
  <proof>

lemma holomorphic_on_qbracket [holomorphic_intros]:
  assumes "q holomorphic_on A"
  assumes " $\bigwedge$ x. n < 0  $\implies$  x  $\in$  A  $\implies$  q x  $\neq$  0"
  shows "( $\lambda$ x. qbracket (q x) n) holomorphic_on A"
  <proof>

lemma analytic_on_qbracket [analytic_intros]:
  assumes "q analytic_on A"
  assumes " $\bigwedge$ x. n < 0  $\implies$  x  $\in$  A  $\implies$  q x  $\neq$  0"
  shows "( $\lambda$ x. qbracket (q x) n) analytic_on A"
  <proof>

lemma meromorphic_on_qbracket [meromorphic_intros]:
  assumes "q meromorphic_on A"
  shows "( $\lambda$ x. qbracket (q x) n) meromorphic_on A"
  <proof>

```

2.2 The q -factorial $[n]_q!$

Since the q -bracket gives us the q -analogue of an integer n , we can use this to recursively define the q -factorial $[n]_q!$. Again, letting $q \rightarrow 1$, we recover the “normal” factorial.

definition $qfact :: "'a \Rightarrow \text{int} \Rightarrow 'a :: \text{field}"$ where
 $qfact\ q\ n = (\text{if } n < 0 \text{ then } 0 \text{ else } (\prod_{k=1..n} qbracket\ q\ k))$

```

lemma qfact_1_of_nat [simp]: "qfact 1 (int n) = fact n"
  <proof>

lemma qfact_1_nonneg [simp]: "n ≥ 0 ⇒ qfact 1 n = fact (nat n)"
  <proof>

lemma qfact_neg [simp]: "n < 0 ⇒ qfact q n = 0"
  <proof>

lemma qfact_0 [simp]: "qfact q 0 = 1"
  <proof>

lemma qfact_1 [simp]: "qfact q 1 = 1"
  <proof>

lemma qfact_2: "qfact q 2 = 1 + q"
  <proof>

lemma qfact_of_real: "qfact (of_real q :: 'a :: real_field) n = of_real
  (qfact q n)"
  <proof>

lemma qfact_plus1: "n ≠ -1 ⇒ qfact q (n + 1) = qfact q n * qbracket
  q (n + 1)"
  <proof>

lemma qfact_rec: "n > 0 ⇒ qfact q n = qbracket q n * qfact q (n - 1)"
  <proof>

lemma qfact_altdef: "q ≠ 1 ⇒ n ≥ 0 ⇒ qfact q n = (∏ k=1..n. 1 -
  q powi k) * (1 - q) powi (-n)"
  <proof>

lemma qfact_int_def: "qfact q (int n) = (∏ k=1..n. qbracket q (int k))"
  <proof>

lemma qfact_eq_0_iff:
  fixes q :: "'a :: field_char_0"
  shows "qfact q n = 0 ⟷ n < 0 ∨ (q ≠ 1 ∧ (∃ k ∈ {1..nat n}. q ^ k
  = 1))"
  <proof>

lemma qfact_eq_0_iff' [simp]:
  fixes q :: "'a :: real_normed_field"
  assumes "norm q ≠ 1"
  shows "qfact q n = 0 ⟷ n < 0"
  <proof>

```

```

lemma prod_neg_qbracket_conv_qfact:
  assumes [simp]: "q ≠ 0"
  shows "(∏ k=1..n. qbracket q (-int k)) = (-1)^n * qfact q n / q ^
((n+1) choose 2)"
⟨proof⟩

lemma norm_qfact_le:
  fixes q :: "'a :: real_normed_field"
  assumes "n ≥ 0" "norm q ≤ 1"
  shows "norm (qfact q n) ≤ fact (nat n)"
⟨proof⟩

lemma continuous_on_qfact [continuous_intros]:
  fixes q :: "'a::topological_space ⇒ 'b :: real_normed_field"
  assumes [continuous_intros]: "continuous_on A q"
  shows "continuous_on A (λx. qfact (q x) n)"
⟨proof⟩

lemma continuous_qfact [continuous_intros]:
  fixes q :: "'a::t2_space ⇒ 'b :: real_normed_field"
  assumes [continuous_intros]: "continuous F q"
  shows "continuous F (λx. qfact (q x) n)"
⟨proof⟩

lemma tendsto_qfact [tendsto_intros]:
  fixes q :: "'a::topological_space ⇒ 'b :: real_normed_field"
  assumes [tendsto_intros]: "(q ⟶ Q) F"
  shows "((λx. qfact (q x) n) ⟶ qfact Q n) F"
⟨proof⟩

lemma holomorphic_on_qfact [holomorphic_intros]:
  assumes [holomorphic_intros]: "q holomorphic_on A"
  shows "(λx. qfact (q x) n) holomorphic_on A"
⟨proof⟩

lemma analytic_on_qfact [analytic_intros]:
  assumes [analytic_intros]: "q analytic_on A"
  shows "(λx. qfact (q x) n) analytic_on A"
⟨proof⟩

lemma meromorphic_on_qfact [meromorphic_intros]:
  assumes [meromorphic_intros]: "q meromorphic_on A"
  shows "(λx. qfact (q x) n) meromorphic_on A"
⟨proof⟩

```

2.3 q -binomial coefficients $\binom{n}{k}_q$

We can also define q -binomial coefficients in such a way that we will get

$$\binom{n}{k}_q = \frac{[n]_q!}{[k]_q! [n-k]_q!}$$

and therefore recover the “normal” binomial coefficients if we let $q \rightarrow 1$.

```

fun qbinomial :: "'a  $\Rightarrow$  nat  $\Rightarrow$  nat  $\Rightarrow$  'a :: field" where
  "qbinomial q n 0 = 1"
| "qbinomial q 0 (Suc k) = 0"
| "qbinomial q (Suc n) (Suc k) = q ^ Suc k * qbinomial q n (Suc k) + qbinomial
  q n k"

```

```

lemma qbinomial_induct [case_names zero_right zero_left step]:
  "( $\bigwedge$ n. P n 0)  $\implies$  ( $\bigwedge$ k. P 0 (Suc k))  $\implies$ 
  ( $\bigwedge$ n k. P n (Suc k)  $\implies$  P n k  $\implies$  P (Suc n) (Suc k))  $\implies$  P n k"
  <proof>

```

```

lemma qbinomial_1_left [simp]: "qbinomial 1 n k = of_nat (binomial n
k)"
  <proof>

```

```

lemma qbinomial_eq_0 [simp]: "k > n  $\implies$  qbinomial q n k = 0"
  <proof>

```

```

lemma qbinomial_n_n [simp]: "qbinomial q n n = 1"
  <proof>

```

```

lemma qbinomial_0_left: "qbinomial 0 n k = (if k  $\leq$  n then 1 else 0)"
  <proof>

```

```

lemma qbinomial_0_left' [simp]: "k  $\leq$  n  $\implies$  qbinomial 0 n k = 1"
  <proof>

```

```

lemma qbinomial_0_middle: "qbinomial q 0 k = (if k = 0 then 1 else 0)"
  <proof>

```

```

lemma qbinomial_of_real: "qbinomial (of_real q :: 'a :: real_field) m
n = of_real (qbinomial q m n)"
  <proof>

```

```

lemma qbinomial_qfact_lemma:
  assumes "k  $\leq$  n"
  shows "qfact q k * qfact q (int (n - k)) * qbinomial q n k = qfact
  q n"
  <proof>

```

```

lemma qbinomial_qfact:

```

```

fixes q :: "'a :: field_char_0"
assumes "¬(∃k∈{1..n}. q ^ k = 1)"
shows "qbinomial q n k = qfact q n / (qfact q k * qfact q (int n -
int k))"
⟨proof⟩

lemma qbinomial_qfact':
fixes q :: "'a :: real_normed_field"
assumes "q = 1 ∨ norm q ≠ 1"
shows "qbinomial q n k = qfact q n / (qfact q k * qfact q (int n -
int k))"
⟨proof⟩

lemma qbinomial_symmetric:
fixes q :: "'a :: real_normed_field"
assumes "norm q ≠ 1" "k ≤ n"
shows "qbinomial q n (n - k) = qbinomial q n k"
⟨proof⟩

lemma qbinomial_rec1:
"n > 0 ⇒ k > 0 ⇒
qbinomial q n k = q ^ k * qbinomial q (n - 1) k + qbinomial q (n
- 1) (k - 1)"
⟨proof⟩

lemma qbinomial_rec2:
fixes q :: "'a :: real_normed_field"
assumes "norm q ≠ 1" "n > 0" "k < n"
shows "qbinomial q n k = (1 - q ^ n) / (1 - q ^ (n - k)) * qbinomial
q (n-1) k"
⟨proof⟩

lemma qbinomial_rec3:
fixes q :: "'a :: real_normed_field"
assumes "norm q ≠ 1" "k > 0" "k ≤ n"
shows "qbinomial q n k = (1 - q ^ n) / (1 - q ^ k) * qbinomial q (n-1)
(k-1)"
⟨proof⟩

lemma qbinomial_rec4:
fixes q :: "'a :: real_normed_field"
assumes "norm q ≠ 1" "n > 0" "k > 0" "k ≤ n"
shows "qbinomial q n k = (1 - q ^ (Suc n - k)) / (1 - q ^ k) * qbinomial
q n (k-1)"
⟨proof⟩

lemmas qbinomial_Suc_Suc [simp del] = qbinomial.simps(3)

lemma qbinomial_Suc_Suc':

```

```

fixes q :: "'a :: real_normed_field"
assumes q: "norm q ≠ 1"
shows "qbinomial q (Suc n) (Suc k) =
      qbinomial q n (Suc k) + q^(n-k) * qbinomial q n k"
⟨proof⟩

```

```

lemma continuous_on_qbinomial [continuous_intros]:
  fixes q :: "'a::topological_space ⇒ 'b :: real_normed_field"
  assumes [continuous_intros]: "continuous_on A q"
  shows "continuous_on A (λx. qbinomial (q x) m n)"
⟨proof⟩

```

```

lemma continuous_qbinomial [continuous_intros]:
  fixes q :: "'a::t2_space ⇒ 'b :: real_normed_field"
  assumes [continuous_intros]: "continuous F q"
  shows "continuous F (λx. qbinomial (q x) m n)"
⟨proof⟩

```

```

lemma tendsto_qbinomial [tendsto_intros]:
  fixes q :: "'a::topological_space ⇒ 'b :: real_normed_field"
  assumes [tendsto_intros]: "(q ⟶ Q) F"
  shows "((λx. qbinomial (q x) m n) ⟶ qbinomial Q m n) F"
⟨proof⟩

```

```

lemma holomorphic_on_qbinomial [holomorphic_intros]:
  assumes [holomorphic_intros]: "q holomorphic_on A"
  shows "(λx. qbinomial (q x) m n) holomorphic_on A"
⟨proof⟩

```

```

lemma analytic_on_qbinomial [analytic_intros]:
  assumes [analytic_intros]: "q analytic_on A"
  shows "(λx. qbinomial (q x) m n) analytic_on A"
⟨proof⟩

```

```

lemma meromorphic_on_qbinomial [meromorphic_intros]:
  assumes [meromorphic_intros]: "q meromorphic_on A"
  shows "(λx. qbinomial (q x) m n) meromorphic_on A"
⟨proof⟩

```

2.4 The Gaussian polynomials

The q -binomial coefficient $\binom{n}{k}_q$ is a polynomial of degree $k(n-k)$ in q . These polynomials are often called the *Gaussian polynomials*.

```

fun gauss_poly :: "nat ⇒ nat ⇒ 'a :: comm_semiring_1 poly" where
  "gauss_poly n 0 = 1"
| "gauss_poly 0 (Suc k) = 0"
| "gauss_poly (Suc n) (Suc k) = monom 1 (Suc k) * gauss_poly n (Suc k)

```


+ gauss_poly n k"

lemma poly_gauss_poly [simp]:

"poly (gauss_poly n k) q = qbinomial q n k"
⟨proof⟩

lemma of_nat_coeff_gauss_poly [simp]: "of_nat (coeff (gauss_poly n k) i) = coeff (gauss_poly n k) i"
⟨proof⟩

lemma of_int_coeff_gauss_poly [simp]: "of_int (coeff (gauss_poly n k) i) = coeff (gauss_poly n k) i"
⟨proof⟩

lemma norm_coeff_gauss_poly [simp]:
"norm (coeff (gauss_poly n k) i :: 'a :: {real_normed_algebra_1, comm_semiring_1}) =
coeff (gauss_poly n k) i"
⟨proof⟩

lemmas gauss_poly_Suc_Suc [simp del] = gauss_poly.simps(3)

lemma gauss_poly_eq_0 [simp]: "k > n \implies gauss_poly n k = 0"
⟨proof⟩

lemma coeff_0_gauss_poly [simp]: "k \leq n \implies coeff (gauss_poly n k) 0 = 1"
⟨proof⟩

lemma gauss_poly_eq_0_iff [simp]: "gauss_poly n k = 0 \iff k > n"
⟨proof⟩

lemma gauss_poly_n_n [simp]: "gauss_poly n n = 1"
⟨proof⟩

lemma coeff_gauss_poly_nonneg: "coeff (gauss_poly n k :: 'a :: linordered_semidom poly) i \geq 0"
⟨proof⟩

lemma coeff_gauss_poly_le:
"coeff (gauss_poly n k :: 'a :: linordered_semidom poly) i \leq of_nat (n choose k)"
⟨proof⟩

lemma degree_gauss_poly: "degree (gauss_poly n k :: 'a :: idom poly) = k * (n - k)"
⟨proof⟩

lemma norm_qbinomial_le_binomial:

```

fixes q :: "'a :: real_normed_field"
assumes "norm q < 1"
shows "norm (qbinomial q n k) ≤ real (n choose k) * (1 - norm q) ^
(k*(n-k)+1)) / (1 - norm q)"
⟨proof⟩

```

```

lemma norm_qbinomial_le_binomial':
fixes q :: "'a :: real_normed_field"
assumes "norm q < 1"
shows "norm (qbinomial q n k) ≤ real (n choose k) / (1 - norm q)"
⟨proof⟩

```

2.5 The finite Pochhammer symbol $(a; q)_n$

The definition of the q -Pochhammer symbol is a bit less obvious. Recall that the ordinary Pochhammer symbol is defined as

$$a^{\bar{n}} = a(a+1)\cdots(a+n-1).$$

The q -Pochhammer symbol is defined as

$$(a; q)_n = (1-a)(1-aq)(1-aq^2)\cdots(1-aq^{n-1})$$

for $n \geq 0$. We extend the definition to $n < 0$ such that the recurrences that hold for $n \geq 0$ carry over to the negative domain as well. Effectively, what we do is to define

$$(a; q)_{-n} = \frac{1}{(aq^{-n}; q)_n}$$

```

definition qpochhammer :: "int ⇒ 'a ⇒ 'a ⇒ 'a :: field" where
  "qpochhammer n a q =
    (if n ≥ 0 then (∏ k<nat n. (1 - a * q ^ k)) else (∏ k=1..nat (-n).
1 / (1 - a / q^k)))"

```

Seeing in which way it is an analogue of the “normal” Pochhammer symbol $a^{\bar{n}} = a(a+1)\cdots(a+n-1)$ is more involved than for the other analogues: if we simply let $q = 1$, we merely get $(1-a)^n$.

However, we do have:

$$\lim_{q \rightarrow 1} \frac{(q^a; q)_\infty}{(1-q)^n} = a^{\bar{n}}$$

```

lemma qpochhammer_tendsto_pochhammer:
  "(λq::real. qpochhammer (int n) (q powr a) q / (1 - q) ^ n) -1 → pochhammer
a n"
⟨proof⟩

```

```

lemma qpochhammer_nonneg_def: "qpochhammer (int n) a q = (∏ k<n. (1 -
a * q ^ k))"
⟨proof⟩

```

```

lemma qpochhammer_0 [simp]: "qpochhammer 0 a q = 1"
  ⟨proof⟩

lemma qpochhammer_1 [simp]: "qpochhammer 1 a q = 1 - a"
  ⟨proof⟩

lemma qpochhammer_1_right [simp]: "qpochhammer n a 1 = (1 - a) powi n"
  ⟨proof⟩

lemma qpochhammer_neg1 [simp]: "q ≠ 0 ⇒ q ≠ a ⇒ qpochhammer (-1)
a q = q / (q - a)"
  ⟨proof⟩

lemma qpochhammer_0_middle [simp]: "qpochhammer n 0 q = 1"
  ⟨proof⟩

lemma qpochhammer_0_right: "qpochhammer n a 0 = (if n > 0 then 1 - a
else 1)"
  ⟨proof⟩

lemma qpochhammer_0_right_pos [simp]: "n > 0 ⇒ qpochhammer n a 0 =
1 - a"
  and qpochhammer_0_right_nonpos [simp]: "n ≤ 0 ⇒ qpochhammer n a 0
= 1"
  ⟨proof⟩

lemma qpochhammer_nat_eq_0_iff:
  "qpochhammer (int n) a q = 0 ⟷ (∃ k < n. a * q ^ k = 1)"
  ⟨proof⟩

lemma qpochhammer_of_real:
  "qpochhammer n (of_real a :: 'a :: real_field) (of_real q) = of_real
(qpochhammer n a q)"
  ⟨proof⟩

lemma qpochhammer_eq_0_iff:
  "qpochhammer n a q = 0 ⟷ (∃ k ∈ {min n 0 .. < max n 0}. a * q powi k =
1)"
  ⟨proof⟩

lemma qpochhammer_rec:
  assumes "∧ k. int k ∈ {0 < .. -n} ⇒ q ^ k ≠ a"
  shows "qpochhammer (n + 1) a q = qpochhammer n a q * (1 - a * q powi
n)"
  ⟨proof⟩

lemma qpochhammer_plus1:
  assumes "n ≥ 0 ∨ x * q powi n ≠ 1"

```

shows "qpochhammer (n + 1) x q = qpochhammer n x q * (1 - x * q powi n)"
 <proof>

lemma qpochhammer_minus1:
 assumes "x * q powi (n - 1) ≠ 1"
 shows "qpochhammer (n - 1) x q = qpochhammer n x q / (1 - x * q powi (n - 1))"
 <proof>

lemma qpochhammer_1plus:
 assumes "n ≥ 0 ∨ x * q powi n ≠ 1"
 shows "qpochhammer (1 + n) x q = qpochhammer n x q * (1 - x * q powi n)"
 <proof>

lemma qpochhammer_nat_add:
 fixes m n :: nat
 shows "qpochhammer (int m + int n) x q = qpochhammer (int m) x q * qpochhammer n (q ^ m * x) q"
 <proof>

lemma qpochhammer_minus:
 assumes "n < 0 → q ≠ 0"
 shows "qpochhammer (-n) a q = 1 / qpochhammer n (a / q powi n) q"
 <proof>

lemma qpochhammer_add:
 assumes "∧k. k ∈ {m+min n 0..<m+max n 0} ⇒ x * q powi k ≠ 1" and
 [simp]: "q ≠ 0"
 shows "qpochhammer (m + n) x q = qpochhammer m x q * qpochhammer n (q powi m * x) q"
 <proof>

lemma qfact_conv_qpochhammer_aux:
 assumes "n < 0 → q ≠ 0"
 shows "qpochhammer n q q = qfact q n * (1 - q) powi n"
 <proof>

lemma qfact_conv_qpochhammer:
 assumes "if n ≥ 0 then q ≠ 1 else q ≠ 0"
 shows "qfact q n = qpochhammer n q q * (1 - q) powi (-n)"
 <proof>

lemma qbinomial_conv_qpochhammer:
 fixes q :: "'a :: field_char_0"
 assumes "k ≤ n"
 assumes "∧k. 0 < k ⇒ k ≤ n ⇒ q ^ k ≠ 1"
 shows "qbinomial q n k =

```

      qepochhammer (int n) q q / (qepochhammer (int k) q q * qepochhammer
(int n - int k) q q)"
⟨proof⟩

```

```

lemma norm_qepochhammer_nonneg_le:
  fixes a q :: "'a::{real_normed_field}"
  assumes "norm q ≤ 1"
  shows "norm (qepochhammer (int n) a q) ≤ (1 + norm a) ^ n"
⟨proof⟩

```

```

lemma norm_qepochhammer_nonneg_ge:
  fixes a q :: "'a::{real_normed_field}"
  assumes "norm q ≤ 1" "norm a ≤ 1"
  shows "norm (qepochhammer (int n) a q) ≥ (1 - norm a) ^ n"
⟨proof⟩

```

```

lemma qepochhammer_nonneg_nonzero:
  fixes q :: "'a :: real_normed_field"
  assumes "norm q < 1" "norm a < 1"
  shows "qepochhammer (int k) a q ≠ 0"
⟨proof⟩

```

```

lemma qbinomial_conv_qepochhammer':
  fixes q :: "'a :: {real_normed_field}"
  assumes "norm q < 1" "k ≤ n"
  shows "qbinomial q n k = qepochhammer (int k) (q ^ (n + 1 - k)) q /
qepochhammer (int k) q q"
⟨proof⟩

```

```

lemma norm_qbinomial_le:
  fixes a q :: "'a::{real_normed_field}"
  assumes "norm q < 1"
  shows "norm (qbinomial q n k) ≤ ((1 + norm q) / (1 - norm q)) ^ k"
⟨proof⟩

```

```

lemma norm_qbinomial_ge:
  fixes a q :: "'a::{real_normed_field}"
  assumes "norm q < 1" "k ≤ n"
  shows "norm (qbinomial q n k) ≥ ((1 - norm q) / (1 + norm q)) ^ k"
⟨proof⟩

```

```

lemma norm_qepochhammer_nonneg_le_qepochhammer:
  fixes q :: "'a :: real_normed_field"
  shows "norm (qepochhammer (int k) a q) ≤ qepochhammer (int k) (-norm
a) (norm q)"
⟨proof⟩

```

```

lemma norm_qepochhammer_nonneg_ge_qepochhammer:
  fixes q :: "'a :: real_normed_field"

```

```

    assumes "norm q ≤ 1" "norm a ≤ 1"
    shows "norm (qpochhammer (int k) a q) ≥ qpochhammer (int k) (norm
a) (norm q)"
⟨proof⟩

```

```

lemma qpochhammer_nonneg:
    assumes "a ≤ 1" "0 ≤ q" "q ≤ 1"
    shows "qpochhammer (int n) a (q::real) ≥ 0"
⟨proof⟩

```

```

lemma qpochhammer_pos:
    assumes "a < 1" "0 ≤ q" "q ≤ 1"
    shows "qpochhammer (int n) a (q::real) > 0"
⟨proof⟩

```

```

lemma holomorphic_qpochhammer [holomorphic_intros]:
    fixes f g :: "complex ⇒ complex"
    assumes [holomorphic_intros]: "f holomorphic_on A" "g holomorphic_on
A"
    assumes "∧x k. x ∈ A ⇒ int k ∈ {0<..-n} ⇒ f x ^ k ≠ g x" "∧x.
x ∈ A ⇒ f x ≠ 0"
    shows "(λx. qpochhammer n (g x) (f x)) holomorphic_on A"
⟨proof⟩

```

```

lemma analytic_qpochhammer [analytic_intros]:
    fixes f g :: "complex ⇒ complex"
    assumes [analytic_intros]: "f analytic_on A" "g analytic_on A"
    assumes "∧x k. x ∈ A ⇒ int k ∈ {0<..-n} ⇒ f x ^ k ≠ g x" "∧x.
x ∈ A ⇒ f x ≠ 0"
    shows "(λx. qpochhammer n (g x) (f x)) analytic_on A"
⟨proof⟩

```

```

lemma meromorphic_qpochhammer [meromorphic_intros]:
    fixes f g :: "complex ⇒ complex"
    assumes [meromorphic_intros]: "f meromorphic_on A" "g meromorphic_on
A"
    shows "(λx. qpochhammer n (g x) (f x)) meromorphic_on A"
⟨proof⟩

```

```

lemma continuous_on_qpochhammer [continuous_intros]:
    fixes f g :: "'a :: topological_space ⇒ 'b :: {real_normed_field}"
    assumes [continuous_intros]: "continuous_on A f" "continuous_on A g"
    assumes "∧x k. x ∈ A ⇒ int k ∈ {0<..-n} ⇒ f x ^ k ≠ g x" "∧x.
x ∈ A ⇒ f x ≠ 0"
    shows "continuous_on A (λx. qpochhammer n (g x) (f x))"
⟨proof⟩

```

```

lemma continuous_qpochhammer [continuous_intros]:

```

```

fixes f g :: "'a :: t2_space ⇒ 'b :: {real_normed_field}"
assumes [continuous_intros]: "continuous (at x within A) f" "continuous
(at x within A) g"
assumes "∧k. int k ∈ {0<..-n} ⇒ f x ^ k ≠ g x" "f x ≠ 0"
shows "continuous (at x within A) (λx. qpochhammer n (g x) (f x))"
⟨proof⟩

lemma tendsto_qpochhammer [tendsto_intros]:
fixes f g :: "'a ⇒ 'b :: {real_normed_field}"
assumes [tendsto_intros]: "(f ⟶ q) F" "(g ⟶ a) F"
assumes "∧k. int k ∈ {0<..-n} ⇒ q ^ k ≠ a" "q ≠ 0"
shows "((λx. qpochhammer n (g x) (f x)) ⟶ qpochhammer n a q) F"
⟨proof⟩

end

```

3 The infinite q -Pochhammer symbol $(a; q)_\infty$

```

theory Q_Pochhammer_Infinite
imports
  More_Infinite_Products
  Q_Analogues
begin

```

3.1 Definition and basic properties

```

definition qpochhammer_inf :: "'a :: {real_normed_field, banach, heine_borel}
⇒ 'a ⇒ 'a" where
  "qpochhammer_inf a q = prodinf (λk. 1 - a * q ^ k)"

```

```

bundle qpochhammer_inf_notation
begin
notation qpochhammer_inf ("'(_ ; _)∞")
end

```

```

bundle no_qpochhammer_inf_notation
begin
no_notation qpochhammer_inf ("'(_ ; _)∞")
end

```

```

lemma qpochhammer_inf_0_left [simp]: "qpochhammer_inf 0 q = 1"
⟨proof⟩

```

```

lemma qpochhammer_inf_0_right [simp]: "qpochhammer_inf a 0 = 1 - a"
⟨proof⟩

```

```

lemma abs_convergent_qpochhammer_inf:
fixes a q :: "'a :: {real_normed_div_algebra, banach}"

```

```

    assumes "norm q < 1"
    shows "abs_convergent_prod ( $\lambda n. 1 - a * q ^ n$ )"
  <proof>

```

```

lemma convergent_qpochhammer_inf:
  fixes a q :: "'a :: {real_normed_field, banach}"
  assumes "norm q < 1"
  shows "convergent_prod ( $\lambda n. 1 - a * q ^ n$ )"
  <proof>

```

```

lemma has_prod_qpochhammer_inf:
  "norm q < 1  $\implies$  ( $\lambda n. 1 - a * q ^ n$ ) has_prod qpochhammer_inf a q"
  <proof>

```

We now also see that the infinite q -Pochhammer symbol $(a; q)_\infty$ really is the limit of $(a; q)_n$ for $n \rightarrow \infty$:

```

lemma qpochhammer_tendsto_qpochhammer_inf:
  assumes q: "norm q < 1"
  shows " $(\lambda n. qpochhammer (int n) t q) \longrightarrow qpochhammer_inf t q$ "
  <proof>

```

```

lemma qpochhammer_inf_of_real:
  assumes "|q| < 1"
  shows "qpochhammer_inf (of_real a) (of_real q) = of_real (qpochhammer_inf a q)"
  <proof>

```

```

lemma qpochhammer_inf_zero_iff:
  assumes q: "norm q < 1"
  shows "qpochhammer_inf a q = 0  $\iff$  ( $\exists n. a * q ^ n = 1$ )"
  <proof>

```

```

lemma qpochhammer_inf_nonzero:
  assumes "norm q < 1" "norm a < 1"
  shows "qpochhammer_inf a q  $\neq$  0"
  <proof>

```

```

lemma qpochhammer_inf_pos:
  assumes "|q| < 1" "|a| < (1::real)"
  shows "qpochhammer_inf a q > 0"
  <proof>

```

```

lemma qpochhammer_inf_nonneg:
  assumes "|q| < 1" "|a|  $\leq$  (1::real)"
  shows "qpochhammer_inf a q  $\geq$  0"
  <proof>

```


3.2 Uniform convergence and its consequences

context

fixes $P :: \text{"nat} \Rightarrow 'a :: \{\text{real_normed_field, banach, heine_borel}\} \Rightarrow 'a \Rightarrow 'a"$

defines $"P \equiv (\lambda N a q. \prod_{n < N}. 1 - a * q ^ n)"$

begin

lemma *uniformly_convergent_qpochhammer_inf_aux*:

assumes $r: "0 \leq ra" "0 \leq rq" "rq < 1"$

shows $"\text{uniformly_convergent_on} (\text{cball } 0 \text{ ra} \times \text{cball } 0 \text{ rq}) (\lambda n (a,q). P n a q)"$

<proof>

lemma *uniformly_convergent_qpochhammer_inf*:

assumes $"\text{compact } A" "A \subseteq \text{UNIV} \times \text{ball } 0 \text{ 1}"$

shows $"\text{uniformly_convergent_on } A (\lambda n (a,q). P n a q)"$

<proof>

lemma *uniform_limit_qpochhammer_inf*:

assumes $"\text{compact } A" "A \subseteq \text{UNIV} \times \text{ball } 0 \text{ 1}"$

shows $"\text{uniform_limit } A (\lambda n (a,q). P n a q) (\lambda(a,q). \text{qpochhammer_inf } a \text{ q}) \text{ at_top}"$

<proof>

lemma *continuous_on_qpochhammer_inf* [*continuous_intros*]:

fixes $a \ q :: "'b :: \text{topological_space} \Rightarrow 'a"$

assumes [*continuous_intros*]: $"\text{continuous_on } A \ a" " \text{continuous_on } A \ q"$

assumes $"\bigwedge x. x \in A \implies \text{norm } (q \ x) < 1"$

shows $"\text{continuous_on } A (\lambda x. \text{qpochhammer_inf } (a \ x) (q \ x))"$

<proof>

lemma *continuous_qpochhammer_inf* [*continuous_intros*]:

fixes $a \ q :: "'b :: \text{t2_space} \Rightarrow 'a"$

assumes $"\text{continuous } (\text{at } x \text{ within } A) \ a" " \text{continuous } (\text{at } x \text{ within } A) \ q" " \text{norm } (q \ x) < 1"$

shows $"\text{continuous } (\text{at } x \text{ within } A) (\lambda x. \text{qpochhammer_inf } (a \ x) (q \ x))"$

<proof>

lemma *tendsto_qpochhammer_inf* [*tendsto_intros*]:

fixes $a \ q :: "'b \Rightarrow 'a"$

assumes $"(a \longrightarrow a0) \ F" "(q \longrightarrow q0) \ F" " \text{norm } q0 < 1"$

shows $"((\lambda x. \text{qpochhammer_inf } (a \ x) (q \ x)) \longrightarrow \text{qpochhammer_inf } a0 \ q0) \ F"$

<proof>

end

context

fixes $P :: \text{"nat} \Rightarrow \text{complex} \Rightarrow \text{complex} \Rightarrow \text{complex}"$

```

    defines "P ≡ (λN a q. ∏[n<N. 1 - a * q ^ n)"
begin

lemma holomorphic_qpochhammer_inf [holomorphic_intros]:
  assumes [holomorphic_intros]: "a holomorphic_on A" "q holomorphic_on
A"
  assumes "∧x. x ∈ A ⇒ norm (q x) < 1" "open A"
  shows "(λx. qpochhammer_inf (a x) (q x)) holomorphic_on A"
⟨proof⟩

lemma analytic_qpochhammer_inf [analytic_intros]:
  assumes [analytic_intros]: "a analytic_on A" "q analytic_on A"
  assumes "∧x. x ∈ A ⇒ norm (q x) < 1"
  shows "(λx. qpochhammer_inf (a x) (q x)) analytic_on A"
⟨proof⟩

lemma meromorphic_qpochhammer_inf [meromorphic_intros]:
  assumes [analytic_intros]: "a analytic_on A" "q analytic_on A"
  assumes "∧x. x ∈ A ⇒ norm (q x) < 1"
  shows "(λx. qpochhammer_inf (a x) (q x)) meromorphic_on A"
⟨proof⟩

end

```

3.3 Bounds for $(a; q)_n$ and $\binom{n}{k}_q$ in terms of $(a; q)_\infty$

```

lemma qpochhammer_le_qpochhammer_inf:
  assumes "q ≥ 0" "q < 1" "a ≤ 0"
  shows "qpochhammer (int k) a q ≤ qpochhammer_inf a (q::real)"
⟨proof⟩

lemma qpochhammer_ge_qpochhammer_inf:
  assumes "q ≥ 0" "q < 1" "a ≥ 0" "a ≤ 1"
  shows "qpochhammer (int k) a q ≥ qpochhammer_inf a (q::real)"
⟨proof⟩

lemma norm_qbinomial_le_qpochhammer_inf_strong:
  fixes q :: "'a :: {real_normed_field}"
  assumes q: "norm q < 1"
  shows "norm (qbinomial q n k) ≤
        qpochhammer_inf (-(norm q ^ (n + 1 - k))) (norm q) /
        qpochhammer_inf (norm q) (norm q)"
⟨proof⟩

lemma norm_qbinomial_le_qpochhammer_inf:
  fixes q :: "'a :: {real_normed_field}"
  assumes q: "norm q < 1"
  shows "norm (qbinomial q n k) ≤
        qpochhammer_inf (norm q) (norm q) / qpochhammer_inf (norm

```

q) (*norm* q)"
 ⟨*proof*⟩

3.4 Limits of the q -binomial coefficients

The following limit is Fact 7.7 in Andrews & Eriksson [2].

lemma *tendsto_qbinomial1*:
 fixes $q :: "'a :: \{real_normed_field, banach, heine_borel\}$ "
 assumes $q: "norm\ q < 1"$
 shows " $(\lambda n. qbinomial\ q\ n\ m) \longrightarrow 1 / qpochhammer\ m\ q\ q"$
 ⟨*proof*⟩

The following limit is a slightly stronger version of Fact 7.8 in Andrews & Eriksson [2]. Their version has $f(n) = rn + c_1$ and $g(n) = sn + c_2$ with $r > s$.

lemma *tendsto_qbinomial2*:
 fixes $q :: "'a :: \{real_normed_field, banach, heine_borel\}$ "
 assumes $q: "norm\ q < 1"$
 assumes *lim_fg*: "*filterlim* $(\lambda n. f\ n - g\ n)$ *at_top* F "
 assumes *lim_g*: "*filterlim* g *at_top* F "
 shows " $((\lambda n. qbinomial\ q\ (f\ n)\ (g\ n)) \longrightarrow 1 / qpochhammer_inf\ q\ q)$ F "
 ⟨*proof*⟩

3.5 Useful identities

The following lemmas give a recurrence for the infinite q -Pochhammer symbol similar to the one for the “normal” Pochhammer symbol.

lemma *qpochhammer_inf_mult_power_q*:
 assumes "*norm* $q < 1$ "
 shows "*qpochhammer_inf* $a\ q = qpochhammer\ (int\ n)\ a\ q * qpochhammer_inf\ (a * q ^ n)\ q"$
 ⟨*proof*⟩

One can express the finite q -Pochhammer symbol in terms of the infinite one:

$$(a; q)_n = \frac{(a; q)_\infty}{(a; q^n)_\infty}$$

lemma *qpochhammer_conv_qpochhammer_inf_nonneg*:
 assumes "*norm* $q < 1$ " " $\bigwedge m. m \geq n \implies a * q ^ m \neq 1$ "
 shows "*qpochhammer* $(int\ n)\ a\ q = qpochhammer_inf\ a\ q / qpochhammer_inf\ (a * q ^ n)\ q"$
 ⟨*proof*⟩

lemma *qpochhammer_conv_qpochhammer_inf*:
 fixes $q\ a :: "'a :: \{real_normed_field, banach, heine_borel\}$ "
 assumes $q: "norm\ q < 1"$ " $n < 0 \longrightarrow q \neq 0$ "

```

    assumes not_one: " $\bigwedge k. \text{int } k \geq n \implies a * q^k \neq 1$ "
    shows "qpochhammer n a q = qpochhammer_inf a q / qpochhammer_inf (a
* q powi n) q"
<proof>

```

```

lemma qpochhammer_inf_divide_power_q:
  assumes "norm q < 1" and [simp]: "q ≠ 0"
  shows "qpochhammer_inf (a / q^n) q = ( $\prod_{k=1..n} 1 - a / q^k$ )
* qpochhammer_inf a q"
<proof>

```

```

lemma qpochhammer_inf_mult_q:
  assumes "norm q < 1"
  shows "qpochhammer_inf a q = (1 - a) * qpochhammer_inf (a * q) q"
<proof>

```

```

lemma qpochhammer_inf_divide_q:
  assumes "norm q < 1" "q ≠ 0"
  shows "qpochhammer_inf (a / q) q = (1 - a / q) * qpochhammer_inf
a q"
<proof>

```

The following lemma allows combining a product of several q -Pochhammer symbols into one by grouping factors:

$$(a; q^m)_\infty (aq; q^m)_\infty \cdots (aq^{m-1}; q^m)_\infty = (a; q)_\infty$$

```

lemma prod_qpochhammer_group:
  assumes "norm q < 1" and "m > 0"
  shows " $(\prod_{i < m} \text{qpochhammer\_inf } (a * q^i) (q^m)) = \text{qpochhammer\_inf}$ 
a q"
<proof>

```

A product of two q -Pochhammer symbols $(\pm a; q)_\infty$ can be combined into a single q -Pochhammer symbol:

```

lemma qpochhammer_inf_square:
  assumes q: "norm q < 1"
  shows "qpochhammer_inf a q * qpochhammer_inf (-a) q = qpochhammer_inf
(a^2) (q^2)"
(is "?lhs = ?rhs")
<proof>

```

3.6 Two series expansions by Euler

The following two theorems and their proofs are taken from Bellman [3][§40]. He credits them, in their original form, to Euler. One could also deduce these relatively easily from the infinite version of the q -binomial theorem (which we will prove later), but the proves given by Bellman are so nice that I do not want to omit them from here.

The first theorem states that for any complex x, t with $|x| < 1$, we have:

$$(t; x)_\infty = \prod_{k=0}^{\infty} (1 - tx^k) = \sum_{n=0}^{\infty} \frac{x^{n(n-1)/2} t^n}{(x-1) \cdots (x^n - 1)}$$

This tells us the power series expansion for $f_x(t) = (t; x)_\infty$.

lemma

```

fixes x :: complex
assumes x: "norm x < 1"
shows sums_qpochhammer_inf_complex:
  "(λn. x^(n*(n-1) div 2) * t^n / (∏k=1..n. x^k - 1)) sums qpochhammer_inf
  t x"
and has_fps_expansion_qpochhammer_inf_complex:
  "(λt. qpochhammer_inf t x) has_fps_expansion
  Abs_fps (λn. x^(n*(n-1) div 2) / (∏k=1..n. x^k - 1))"
<proof>

```

lemma sums_qpochhammer_inf_real:

```

assumes "|x| < (1 :: real)"
shows "(λn. x^(n*(n-1) div 2) * t^n / (∏k=1..n. x^k - 1)) sums qpochhammer_inf
  t x"
<proof>

```

lemma norm_summable_qpochhammer_inf:

```

fixes x t :: "'a :: {real_normed_field}"
assumes "norm x < 1"
shows "summable (λn. norm (x^(n*(n-1) div 2) * t ^ n / (∏k=1..n.
  x^k - 1)))"
<proof>

```

The second theorem states that for any complex x, t with $|x| < 1$, $|t| < 1$, we have:

$$\frac{1}{(t; x)_\infty} = \prod_{k=0}^{\infty} \frac{1}{1 - tx^k} = \sum_{n=0}^{\infty} \frac{t^n}{(1-x) \cdots (1-x^n)}$$

This gives us the multiplicative inverse of the power series from the previous theorem.

lemma

```

fixes x :: complex
assumes x: "norm x < 1" and t: "norm t < 1"
shows sums_inverse_qpochhammer_inf_complex:
  "(λn. t^n / (∏k=1..n. 1 - x^k)) sums inverse (qpochhammer_inf
  t x)"
and has_fps_expansion_inverse_qpochhammer_inf_complex:
  "(λt. inverse (qpochhammer_inf t x)) has_fps_expansion
  Abs_fps (λn. 1 / (∏k=1..n. 1 - x^k))"
<proof>

```

```

lemma sums_inverse_qpochhammer_inf_real:
  assumes "|x| < (1 :: real)" "|t| < 1"
  shows "(λn. t^n / (∏k=1..n. 1 - x^k)) sums inverse (qpochhammer_inf
t x)"
⟨proof⟩

```

```

lemma norm_summable_inverse_qpochhammer_inf:
  fixes x t :: "'a :: {real_normed_field}"
  assumes "norm x < 1" "norm t < 1"
  shows "summable (λn. norm (t ^ n / (∏k=1..n. 1 - x^k)))"
⟨proof⟩

```

3.7 Euler's function

Euler's ϕ function is closely related to the Dedekind η function and the Jacobi ϑ nullwert functions. The q -Pochhammer symbol gives us a simple and convenient way to define it.

```

definition euler_phi :: "'a :: {real_normed_field, banach, heine_borel}
⇒ 'a" where
  "euler_phi q = qpochhammer_inf q q"

```

```

lemma euler_phi_0 [simp]: "euler_phi 0 = 1"
⟨proof⟩

```

```

lemma abs_convergent_euler_phi:
  assumes "(q :: 'a :: real_normed_div_algebra) ∈ ball 0 1"
  shows "abs_convergent_prod (λn. 1 - q ^ Suc n)"
⟨proof⟩

```

```

lemma convergent_euler_phi:
  assumes "(q :: 'a :: {real_normed_field, banach}) ∈ ball 0 1"
  shows "convergent_prod (λn. 1 - q ^ Suc n)"
⟨proof⟩

```

```

lemma has_prod_euler_phi:
  "norm q < 1 ⇒ (λn. 1 - q ^ Suc n) has_prod euler_phi q"
⟨proof⟩

```

```

lemma euler_phi_nonzero [simp]:
  assumes x: "x ∈ ball 0 1"
  shows "euler_phi x ≠ 0"
⟨proof⟩

```

```

lemma holomorphic_euler_phi [holomorphic_intros]:
  assumes [holomorphic_intros]: "f holomorphic_on A"
  assumes "∧z. z ∈ A ⇒ norm (f z) < 1"
  shows "(λz. euler_phi (f z)) holomorphic_on A"
⟨proof⟩

```

```

lemma analytic_euler_phi [analytic_intros]:
  assumes [analytic_intros]: "f analytic_on A"
  assumes "\z. z \in A \implies norm (f z) < 1"
  shows "(λz. euler_phi (f z)) analytic_on A"
  <proof>

lemma meromorphic_on_euler_phi [meromorphic_intros]:
  "f analytic_on A \implies (\z. z \in A \implies norm (f z) < 1) \implies (λz. euler_phi
(f z)) meromorphic_on A"
  <proof>

lemma continuous_on_euler_phi [continuous_intros]:
  assumes "continuous_on A f" "\z. z \in A \implies norm (f z) < 1"
  shows "continuous_on A (λz. euler_phi (f z))"
  <proof>

lemma continuous_euler_phi [continuous_intros]:
  fixes a q :: "'b :: t2_space \Rightarrow 'a :: {real_normed_field, banach, heine_borel}"
  assumes "continuous (at x within A) f" "norm (f x) < 1"
  shows "continuous (at x within A) (λx. euler_phi (f x))"
  <proof>

lemma tendsto_euler_phi [tendsto_intros]:
  assumes [tendsto_intros]: "(f \longrightarrow c) F" and "norm c < 1"
  shows "((λx. euler_phi (f x)) \longrightarrow euler_phi c) F"
  <proof>

end

```

4 q -binomial identities

```

theory Q_Binomial_Identities
  imports Q_Pochhammer_Infinite
begin

```

4.1 The q -binomial theorem

Recall the binomial theorem:

$$(1 + t)^n = \sum_{k=0}^n \binom{n}{k} t^k$$

The q -binomial numbers satisfy an analogous theorem:

$$\prod_{k=0}^{n-1} (1 + tq^k) = \sum_{k=0}^n q^{k(k-1)/2} \binom{n}{k}_q t^k$$

It can be seen easily that letting $q \rightarrow 1$ would give us the “normal” binomial theorem.

theorem `qbinomial_theorem`:

"`qpochhammer (int n) (-t) q = (∑ k ≤ n. qbinomial q n k * q ^ (k choose 2) * t ^ k)`"
 ⟨*proof*⟩

lemma `qbinomial_theorem'`:

"`qpochhammer (int n) t q = (∑ k ≤ n. qbinomial q n k * q ^ (k choose 2) * (-t) ^ k)`"
 ⟨*proof*⟩

4.2 The infinite q -binomial theorem

Taking the limit $n \rightarrow \infty$ in the q -binomial theorem and interchanging the limits with Tannery's Theorem, we obtain, for any q with $|q| < 1$:

$$\sum_{k=0}^{\infty} \frac{t^k q^{k(k-1)/2}}{[k]_q!(1-q)^k} = \prod_{k=0}^{\infty} (1 + tq^k) = (-t; q)_{\infty}$$

theorem `qbinomial_theorem_inf`:

`fixes q t :: "'a :: {real_normed_field, banach, heine_borel}"`
`assumes q: "q ∈ ball 0 1"`
`defines "S ≡ (λk. (q ^ (k choose 2) * t ^ k) / (qfact q (int k) * (1 - q) ^ k))"`
`shows "summable (λk. norm (S k))" and "(∑ k. S k) = qpochhammer_inf (-t) q"`
 ⟨*proof*⟩

4.3 The q -Vandermonde identity

The following is the q -analog of Vandermonde's identity

$$\binom{m+n}{r} = \sum_{i=0}^r \binom{m}{i} \binom{n}{r-i},$$

namely:

$$\binom{m+n}{r}_q = \sum_{i=0}^r \binom{m}{i}_q \binom{n}{r-i}_q q^{(m-i)(r-i)}$$

theorem `qvandermonde`:

`fixes m n :: nat and q :: "'a :: real_normed_field"`
`assumes "norm q ≠ 1"`
`shows "qbinomial q (m + n) r = (∑ i ≤ r. qbinomial q m i * qbinomial q n (r - i) * q ^ ((m - i) * (r - i)))"`
 ⟨*proof*⟩

We therefore also get the following identity for the central q -binomial coefficient:


```

corollary qbinomial_square_sum:
  fixes q :: "'a :: real_normed_field"
  assumes q: "norm q ≠ 1"
  shows "(∑ k ≤ n. qbinomial q n k ^ 2 * q ^ (k ^ 2)) = qbinomial q
(2 * n) n"
⟨proof⟩

end

```

References

- [1] G. Andrews, R. Askey, and R. Roy. *Special Functions*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1999.
- [2] G. Andrews and K. Eriksson. *Integer Partitions*. Cambridge University Press, 2004.
- [3] R. Bellman. *A Brief Introduction to Theta Functions*. Athena series. Holt, Rinehart and Winston, 1961.