

Combinatorial q -Analogues

Manuel Eberl

February 6, 2026

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
2	q-analogues of basic combinatorial symbols	3
2.1	The q -bracket $[n]_q$	3
2.2	The q -factorial $[n]_q!$	6
2.3	q -binomial coefficients $\binom{n}{k}_q$	8
2.4	The Gaussian polynomials	11
2.5	The finite Pochhammer symbol $(a; q)_n$	13
3	Primitive roots in an integral domain	18
4	The infinite q-Pochhammer symbol $(a; q)_\infty$	19
4.1	Definition and basic properties	19
4.2	Uniform convergence and its consequences	21
4.3	Bounds for $(a; q)_n$ and $\binom{n}{k}_q$ in terms of $(a; q)_\infty$	22
4.4	Limits of the q -binomial coefficients	23
4.5	Useful identities	23
4.6	Two series expansions by Euler	25
5	q-binomial identities	26
5.1	The q -binomial theorem	27
5.2	The infinite q -binomial theorem	27
5.3	The q -Vandermonde identity	28
6	Euler's function	28
6.1	Definition and basic properties	28
6.2	Connection to Lambert series	30
6.3	Logarithmic derivative	30

1 Auxiliary material

1.1 Additional facts about infinite products

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

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>

end
```

2 q -analogues of basic combinatorial symbols

```
theory Q_Analogues
  imports "HOL-Complex_Analysis.Complex_Analysis"
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  $\Rightarrow$  int  $\Rightarrow$  '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  $\neq$  0  $\implies$  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  $\longrightarrow$  n = 0"
  shows "qbracket q (-n) = -qbracket (inverse q) n / q"
  <proof>

lemma qbracket_inverse:
  assumes "q = 0  $\longrightarrow$  n = 0"
  shows "qbracket (inverse q) n = -q * qbracket q (-n)"
  <proof>

lemma qbracket_nonneg_altdef: "n  $\geq$  0  $\implies$  qbracket q n = ( $\sum$  k<nat n.
q ^ k)"
  <proof>

lemma qbracket_nonpos_altdef:
  assumes n: "n  $\leq$  0" and [simp]: "q  $\neq$  0"
  shows "qbracket q n = -(q powi n * ( $\sum$  k<nat (-n). q ^ k))"
  <proof>

lemma norm_qbracket_le:
  fixes q :: "'a :: real_normed_field"
  assumes "n  $\geq$  0" "norm q  $\leq$  1"
  shows "norm (qbracket q n)  $\leq$  real_of_int n"
  <proof>

lemma qbracket_add:
  assumes "q  $\neq$  0  $\vee$  (k + 1 = 0  $\longrightarrow$  k = 0)"
  shows "qbracket q (k + 1) = qbracket q 1 * q powi k + qbracket q k"
  <proof>

lemma qbracket_diff:
  assumes "q  $\neq$  0  $\vee$  (k = 1  $\longrightarrow$  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 ⇒ 'b :: real_normed_field"
 assumes "continuous F q"
 assumes "n < 0 ⇒ q (netlimit F) ≠ 0"
 shows "continuous F (λx. qbracket (q x) n)"
 <proof>

lemma has_field_derivative_qbracket_real [derivative_intros]:
 fixes q :: real
 assumes "q ≠ 0 ∨ n ≥ 0"
 defines "D ≡ (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 "(λq. 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 ≠ 0 ∨ n ≥ 0"
  defines "D ≡ (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 "(λq. qbracket q n) has_field_derivative D (at q within A)"
⟨proof⟩

```

```

lemma holomorphic_on_qbracket [holomorphic_intros]:
  assumes "q holomorphic_on A"
  assumes "∧x. n < 0 ⇒ x ∈ A ⇒ q x ≠ 0"
  shows "(λx. qbracket (q x) n) holomorphic_on A"
⟨proof⟩

```

```

lemma analytic_on_qbracket [analytic_intros]:
  assumes "q analytic_on A"
  assumes "∧x. n < 0 ⇒ x ∈ A ⇒ q x ≠ 0"
  shows "(λx. qbracket (q x) n) analytic_on A"
⟨proof⟩

```

```

lemma meromorphic_on_qbracket [meromorphic_intros]:
  assumes "q meromorphic_on A"
  shows "(λx. 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 ⇒ int ⇒ 'a :: field" where
  "qfact q n = (if n < 0 then 0 else (∏[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"

```

$\langle proof \rangle$

lemma `qfact_1 [simp]: "qfact q 1 = 1"`
 $\langle proof \rangle$

lemma `qfact_2: "qfact q 2 = 1 + q"`
 $\langle proof \rangle$

lemma `qfact_of_real: "qfact (of_real q :: 'a :: real_field) n = of_real (qfact q n)"`
 $\langle proof \rangle$

lemma `qfact_plus1: "n \neq -1 \implies qfact q (n + 1) = qfact q n * qbracket q (n + 1)"`
 $\langle proof \rangle$

lemma `qfact_rec: "n > 0 \implies qfact q n = qbracket q n * qfact q (n - 1)"`
 $\langle proof \rangle$

lemma `qfact_altdef: "q \neq 1 \implies n \geq 0 \implies qfact q n = ($\prod_{k=1..n}$ 1 - q powi k) * (1 - q) powi (-n)"`
 $\langle proof \rangle$

lemma `qfact_int_def: "qfact q (int n) = ($\prod_{k=1..n}$ qbracket q (int k))"`
 $\langle proof \rangle$

lemma `qfact_eq_0_iff:`
fixes `q :: "'a :: field_char_0"`
shows `"qfact q n = 0 \longleftrightarrow n < 0 \vee (q \neq 1 \wedge ($\exists k \in \{1..nat\ n\}$. q k = 1))"`
 $\langle proof \rangle$

lemma `qfact_eq_0_iff' [simp]:`
fixes `q :: "'a :: real_normed_field"`
assumes `"norm q \neq 1"`
shows `"qfact q n = 0 \longleftrightarrow n < 0"`
 $\langle proof \rangle$

lemma `prod_neg_qbracket_conv_qfact:`
assumes `[simp]: "q \neq 0"`
shows `"($\prod_{k=1..n}$ qbracket q (-int k)) = (-1) n * qfact q n / q $^{(n+1)}$ choose 2)"`
 $\langle proof \rangle$

lemma `norm_qfact_le:`
fixes `q :: "'a :: real_normed_field"`
assumes `"n \geq 0" "norm q \leq 1"`
shows `"norm (qfact q n) \leq fact (nat n)"`
 $\langle proof \rangle$

```

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 ⇒ nat ⇒ nat ⇒ '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]:
  "(\(\n. P n 0) \(\k. P 0 (Suc k)) \(\n k. P n (Suc k) \(\k. P n k) \(\k. P (Suc n) (Suc k)) \(\k. 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 \(\k. 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 \(\le n then 1 else 0)"
  <proof>

lemma qbinomial_0_left' [simp]: "k \(\le n \(\k. 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 \(\le 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 \(\in {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 \(\vee norm q \(\neq 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)  $\leq$  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)  $\leq$  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 \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a :: field" where
`"qpochhammer n a q =`
`(if n \geq 0 then ($\prod_{k < \text{nat } n} (1 - a * q^k)$) else ($\prod_{k=1..nat (-n)} (1 / (1 - a / q^k))$))"`

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`:
`"($\lambda q :: \text{real}.$ qpochhammer (int n) (q powr a) q / (1 - q) ^ n) \rightarrow pochhammer a n"`
`<proof>`

lemma `qpochhammer_nonneg_def`: "qpochhammer (int n) a q = ($\prod_{k < n} (1 - a * q^k)$)"
`<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]`: " $q \neq 0 \implies q \neq a \implies \text{qpochhammer } (-1) a q = q / (q - a)$ "

```

    <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  $\implies$  qpochhammer n a 0 =
1 - a"
  and qpochhammer_0_right_nonpos [simp]: "n  $\leq$  0  $\implies$  qpochhammer n a 0
= 1"
  <proof>

lemma qpochhammer_nat_eq_0_iff:
  "qpochhammer (int n) a q = 0  $\longleftrightarrow$  ( $\exists k < n$ . 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  $\longleftrightarrow$  ( $\exists k \in \{\min n 0 .. \max n 0\}$ . a * q powi k =
1)"
  <proof>

lemma qpochhammer_rec:
  assumes " $\wedge k$ . int k  $\in$  {0<.. $-n$ }  $\implies$  q ^ k  $\neq$  a"
  shows "qpochhammer (n + 1) a q = qpochhammer n a q * (1 - a * q powi
n)"
  <proof>

lemma qpochhammer_plus1:
  assumes "n  $\geq$  0  $\vee$  x * q powi n  $\neq$  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)  $\neq$  1"
  shows "qpochhammer (n - 1) x q = qpochhammer n x q / (1 - x * q powi
(n - 1))"
  <proof>

lemma qpochhammer_1plus:
  assumes "n  $\geq$  0  $\vee$  x * q powi n  $\neq$  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 =
 qpochhammer (int n) q q / (qpochhammer (int k) q q * qpochhammer (int n - int k) q q)"
 <proof>

lemma norm_qpochhammer_nonneg_le:
 fixes a q :: "'a::{real_normed_field}"
 assumes "norm q ≤ 1"
 shows "norm (qpochhammer (int n) a q) ≤ (1 + norm a) ^ n"
 <proof>

lemma norm_qpochhammer_nonneg_ge:
 fixes a q :: "'a::{real_normed_field}"

```

    assumes "norm q ≤ 1" "norm a ≤ 1"
    shows "norm (qpochhammer (int n) a q) ≥ (1 - norm a) ^ n"
  <proof>

lemma qpochhammer_nonneg_nonzero:
  fixes q :: "'a :: real_normed_field"
  assumes "norm q < 1" "norm a < 1"
  shows "qpochhammer (int k) a q ≠ 0"
  <proof>

lemma qbinomial_conv_qpochhammer':
  fixes q :: "'a :: {real_normed_field}"
  assumes "norm q < 1" "k ≤ n"
  shows "qbinomial q n k = qpochhammer (int k) (q ^ (n + 1 - k)) q /
qpochhammer (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_qpochhammer_nonneg_le_qpochhammer:
  fixes q :: "'a :: real_normed_field"
  shows "norm (qpochhammer (int k) a q) ≤ qpochhammer (int k) (-norm
a) (norm q)"
  <proof>

lemma norm_qpochhammer_nonneg_ge_qpochhammer:
  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 \Rightarrow complex"
 assumes [holomorphic_intros]: "f holomorphic_on A" "g holomorphic_on A"
 assumes " $\bigwedge x k. x \in A \implies \text{int } k \in \{0 < .. -n\} \implies f x \wedge k \neq g x$ " " $\bigwedge x. x \in A \implies f x \neq 0$ "
 shows " $(\lambda x. \text{qpochhammer } n (g x) (f x))$ holomorphic_on A"
 <proof>

lemma analytic_qpochhammer [analytic_intros]:
 fixes f g :: "complex \Rightarrow complex"
 assumes [analytic_intros]: "f analytic_on A" "g analytic_on A"
 assumes " $\bigwedge x k. x \in A \implies \text{int } k \in \{0 < .. -n\} \implies f x \wedge k \neq g x$ " " $\bigwedge x. x \in A \implies f x \neq 0$ "
 shows " $(\lambda x. \text{qpochhammer } n (g x) (f x))$ analytic_on A"
 <proof>

lemma meromorphic_qpochhammer [meromorphic_intros]:
 fixes f g :: "complex \Rightarrow complex"
 assumes [meromorphic_intros]: "f meromorphic_on A" "g meromorphic_on A"
 shows " $(\lambda x. \text{qpochhammer } n (g x) (f x))$ meromorphic_on A"
 <proof>

lemma continuous_on_qpochhammer [continuous_intros]:
 fixes f g :: "'a :: topological_space \Rightarrow 'b :: {real_normed_field}"
 assumes [continuous_intros]: "continuous_on A f" "continuous_on A g"
 assumes " $\bigwedge x k. x \in A \implies \text{int } k \in \{0 < .. -n\} \implies f x \wedge k \neq g x$ " " $\bigwedge x. x \in A \implies f x \neq 0$ "
 shows "continuous_on A $(\lambda x. \text{qpochhammer } n (g x) (f x))$ "
 <proof>

lemma continuous_qpochhammer [continuous_intros]:
 fixes f g :: "'a :: t2_space \Rightarrow 'b :: {real_normed_field}"
 assumes [continuous_intros]: "continuous (at x within A) f" "continuous (at x within A) g"
 assumes " $\bigwedge k. \text{int } k \in \{0 < .. -n\} \implies f x \wedge k \neq g x$ " "f x \neq 0"
 shows "continuous (at x within A) $(\lambda x. \text{qpochhammer } n (g x) (f x))$ "
 <proof>

lemma tendsto_qpochhammer [tendsto_intros]:
 fixes f g :: "'a \Rightarrow 'b :: {real_normed_field}"
 assumes [tendsto_intros]: "(f \longrightarrow q) F" "(g \longrightarrow a) F"
 assumes " $\bigwedge k. \text{int } k \in \{0 < .. -n\} \implies q \wedge k \neq a$ " "q \neq 0"
 shows " $(\lambda x. \text{qpochhammer } n (g x) (f x)) \longrightarrow \text{qpochhammer } n a q$ F"

<proof>

end

3 Primitive roots in an integral domain

theory *Primitive_Roots*

imports

Complex_Main

"HOL-Number_Theory.Cong"

"HOL-Computational_Algebra.Computational_Algebra"

"HOL-Library.Real_Mod"

begin

We define the notion of a primitive root in an integral domain: an n -th root of unity that is not a k -th root of unity for any $k < n$ and therefore generates all n -th roots of unity.

locale *primroot* =

fixes $n :: \text{nat}$ and $w :: "'a :: \text{idom}"$

assumes *pos_order*: " $n > 0$ "

assumes *root_of_unity*: " $w^n = 1$ "

assumes *primitive*: " $\bigwedge k. 0 < k \implies k < n \implies w^k \neq 1$ "

begin

lemma *nonzero*: " $w \neq 0$ "

<proof>

lemma *power_eq_iff*: " $w^k = w^l \iff [k = l] \pmod n$ "

<proof>

lemma *inj_on_power*: "*inj_on* ($\lambda k. w^k$) $\{..<n\}$ "

<proof>

end

locale *primroot_field* = *primroot* n w **for** n and $w :: "'a :: \text{field}"$

begin

lemma *power_int_eq_iff*: " $w^{\text{powi } k} = w^{\text{powi } l} \iff [k = l] \pmod{(\text{int } n)}$ "

<proof>

end

interpretation *primroot_1*: *primroot* 1 1

<proof>

```

interpretation primroot_neg1: primroot 2 "-1 :: 'a :: {idom, ring_char_0}"
  <proof>

locale primroot_cis =
  fixes n :: nat and k :: int
  assumes coprime: "coprime k n"
  assumes pos_order': "n > 0"
begin

sublocale primroot n "cis (2 * pi * k / n)"
  <proof>

end

interpretation primroot_ii: primroot 4 i
  <proof>

lemma (in primroot) cyclotomic_poly_conv_prod_unity_root:
  "Polynomial.monom 1 n - 1 = (∏ k<n. [:- (w ^ k), 1:])" (is "?lhs = ?rhs")
  <proof>

lemma (in primroot) cyclotomic_poly_conv_prod_unity_root':
  "1 - Polynomial.monom 1 n = (∏ k<n. [:1, -(w ^ k):])" (is "?lhs = ?rhs")
  <proof>

end

```

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

```

theory Q_Pochhammer_Infinite
imports
  More_Infinite_Products
  Q_Analogues
  Primitive_Roots
begin

```

4.1 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 (λn. 1 - a * q ^ n)"
  <proof>
```

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

```
lemma has_prod_qpochhammer_inf:
  "norm q < 1 ⇒ (λn. 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 "(λn. qpochhammer (int n) t q) ⟶ 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 ⟷ (∃n. a * q ^ n = 1)"
  <proof>
```

```
lemma qpochhammer_inf_nonzero:
  assumes "norm q < 1" "norm a < 1"
  shows "qpochhammer_inf a q ≠ 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| ≤ (1::real)"
  shows "qpochhammer_inf a q ≥ 0"
  <proof>

```

4.2 Uniform convergence and its consequences

```

context
  fixes P :: "nat ⇒ 'a :: {real_normed_field, banach, heine_borel} ⇒
'a ⇒ 'a"
  defines "P ≡ (λN a q. ∏n<N. 1 - a * q ^ n)"
begin

```

```

lemma uniformly_convergent_qpochhammer_inf_aux:
  assumes r: "0 ≤ ra" "0 ≤ rq" "rq < 1"
  shows "uniformly_convergent_on (cball 0 ra × cball 0 rq) (λn (a,q).
P n a q)"
  <proof>

```

```

lemma uniformly_convergent_qpochhammer_inf:
  assumes "compact A" "A ⊆ UNIV × ball 0 1"
  shows "uniformly_convergent_on A (λn (a,q). P n a q)"
  <proof>

```

```

lemma uniform_limit_qpochhammer_inf:
  assumes "compact A" "A ⊆ UNIV × ball 0 1"
  shows "uniform_limit A (λn (a,q). P n a q) (λ(a,q). qpochhammer_inf
a q) at_top"
  <proof>

```

```

lemma continuous_on_qpochhammer_inf [continuous_intros]:
  fixes a q :: "'b :: topological_space ⇒ 'a"
  assumes [continuous_intros]: "continuous_on A a" "continuous_on A q"
  assumes "∧x. x ∈ A ⇒ norm (q x) < 1"
  shows "continuous_on A (λx. qpochhammer_inf (a x) (q x))"
  <proof>

```

```

lemma continuous_qpochhammer_inf [continuous_intros]:
  fixes a q :: "'b :: t2_space ⇒ 'a"
  assumes "continuous (at x within A) a" "continuous (at x within A)
q" "norm (q x) < 1"
  shows "continuous (at x within A) (λx. 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" "norm q0 < 1"
  shows "(( $\lambda$ x. qpochhammer_inf (a x) (q x))  $\longrightarrow$  qpochhammer_inf a0
q0) F"
<proof>

```

end

context

```

  fixes P :: "nat  $\Rightarrow$  complex  $\Rightarrow$  complex  $\Rightarrow$  complex"
  defines "P  $\equiv$  ( $\lambda$ N a q.  $\prod_{n < N}. 1 - a * q ^ n$ )"
begin

```

```

lemma holomorphic_qpochhammer_inf [holomorphic_intros]:
  assumes [holomorphic_intros]: "a holomorphic_on A" "q holomorphic_on
A"
  assumes " $\bigwedge$ x. x  $\in$  A  $\implies$  norm (q x) < 1" "open A"
  shows "(( $\lambda$ x. 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 " $\bigwedge$ x. x  $\in$  A  $\implies$  norm (q x) < 1"
  shows "(( $\lambda$ x. 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 " $\bigwedge$ x. x  $\in$  A  $\implies$  norm (q x) < 1"
  shows "(( $\lambda$ x. qpochhammer_inf (a x) (q x)) meromorphic_on A"
<proof>

```

end

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

```

lemma qpochhammer_le_qpochhammer_inf:
  assumes "q  $\geq$  0" "q < 1" "a  $\leq$  0"
  shows "qpochhammer (int k) a q  $\leq$  qpochhammer_inf a (q::real)"
<proof>

```

```

lemma qpochhammer_ge_qpochhammer_inf:
  assumes "q  $\geq$  0" "q < 1" "a  $\geq$  0" "a  $\leq$  1"
  shows "qpochhammer (int k) a q  $\geq$  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⟩

```

4.4 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 "(λn. qbinomial q n m) → 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 (λn. f n - g n) at_top F"
  assumes lim_g: "filterlim g at_top F"
  shows "((λn. qbinomial q (f n) (g n)) → 1 / qpochhammer_inf q
q) F"
⟨proof⟩

```

4.5 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" "\m. m ≥ n ⇒ a * q ^ m ≠ 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 → q ≠ 0"
  assumes not_one: "\k. int k ≥ n ⇒ a * q ^ k ≠ 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 = (∏ 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_inf_group:
  assumes "norm q < 1" and "m > 0"
  shows "(∏ i < m. qpochhammer_inf (a * q ^ i) (q ^ m)) = qpochhammer_inf
a q"
⟨proof⟩
```

In a similar fashion, let w be a primitive n -th root of unity. Then:

$$(a; q)_\infty (aw; q)_\infty (aw^2; q)_\infty \cdots (aw^{m-1}; q)_\infty = (a^m; q^m)_\infty$$

```

lemma prod_qpochhammer_group_primroot:
  assumes "norm q < 1"
  assumes "primroot m w"
  shows "(∏ k<m. qpochhammer_inf (w ^ k * a) q) = qpochhammer_inf (a ^ m)
(q ^ m)"
⟨proof⟩

```

```

lemma (in primroot_cis) prod_qpochhammer_group_cis:
  assumes "norm q < 1"
  defines "w ≡ (λj. cis (2 * pi * j * k / n))"
  shows "(∏ j<n. qpochhammer_inf (w j * a) q) = qpochhammer_inf (a ^ n)
(q ^ n)"
⟨proof⟩

```

The particular instance of the above for $m = 2$ is the following: 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⟩

```

4.6 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⟩

```

end

5 q -binomial identities

theory $Q_Binomial_Identities$

```

imports Q_Pochhammer_Infinite
begin

```

5.1 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⟩

```

5.2 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⟩

```

5.3 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

6 Euler's function

theory *Euler_Function*

```

imports Q_Pochhammer_Infinite "Lambert_Series.Lambert_Series"
begin

```

6.1 Definition and basic properties

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 euler_phi_of_real: " $|x| < 1 \implies \text{euler\_phi (of\_real } x) = \text{of\_real (euler\_phi } x)$ "
  <proof>

lemma abs_convergent_euler_phi:
  assumes "(q :: 'a :: real_normed_div_algebra) ∈ ball 0 1"
  shows "abs_convergent_prod ( $\lambda n. 1 - q ^ \text{Suc } n$ )"
  <proof>

lemma convergent_euler_phi:
  assumes "(q :: 'a :: {real_normed_field, banach}) ∈ ball 0 1"
  shows "convergent_prod ( $\lambda n. 1 - q ^ \text{Suc } n$ )"
  <proof>

lemma has_prod_euler_phi:
  " $\text{norm } q < 1 \implies (\lambda n. 1 - q ^ \text{Suc } n) \text{ has\_prod euler\_phi } q$ "
  <proof>

lemma euler_phi_nonzero [simp]:
  assumes x: " $x \in \text{ball } 0 1$ "
  shows "euler_phi x  $\neq$  0"
  <proof>

lemma euler_phi_pos_real:
  assumes x: " $|x::\text{real}| < 1$ "
  shows "euler_phi x  $>$  0"
  <proof>

lemma holomorphic_euler_phi [holomorphic_intros]:
  assumes [holomorphic_intros]: "f holomorphic_on A"
  assumes " $\bigwedge z. z \in A \implies \text{norm (f } z) < 1$ "
  shows " $(\lambda z. \text{euler\_phi (f } z)) \text{ holomorphic\_on } A$ "
  <proof>

lemma analytic_euler_phi [analytic_intros]:
  assumes [analytic_intros]: "f analytic_on A"
  assumes " $\bigwedge z. z \in A \implies \text{norm (f } z) < 1$ "
  shows " $(\lambda z. \text{euler\_phi (f } z)) \text{ analytic\_on } A$ "
  <proof>

lemma meromorphic_on_euler_phi [meromorphic_intros]:
  " $\text{f analytic\_on } A \implies (\bigwedge z. z \in A \implies \text{norm (f } z) < 1) \implies (\lambda z. \text{euler\_phi (f } z)) \text{ meromorphic\_on } A$ "
  <proof>

lemma continuous_on_euler_phi [continuous_intros]:
  assumes "continuous_on A f" " $\bigwedge z. z \in A \implies \text{norm (f } z) < 1$ "
  shows "continuous_on A ( $\lambda z. \text{euler\_phi (f } z)$ )"

```

<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>

lemma *uniform_limit_euler_phi*:
fixes A :: "complex set"
assumes A: "compact A" "A \subseteq ball 0 1"
shows "uniform_limit A (λn q. $\prod_{k < n} 1 - q \wedge \text{Suc } k$) euler_phi sequentially"
<proof>

6.2 Connection to Lambert series

The Lambert series $\sum_{k=1}^{\infty} \frac{1}{k} \frac{q^k}{1-q^k}$ is a logarithm of $\frac{1}{\phi(q)}$:

lemma *euler_phi_conv_lambert_complex'*:
fixes q :: complex
assumes q: "norm q < 1"
shows "euler_phi q = exp (-lambert (λn . 1 / of_nat n) q)"
<proof>

lemma *euler_phi_conv_lambert_real'*:
fixes q :: real
assumes q: "|q| < 1"
shows "euler_phi q = exp (-lambert (λn . 1 / real n) q)"
<proof>

As an application of this, we obtain the useful inequality $\phi(q) \geq -\frac{\pi^2 q}{6(1-q)}$ for real $q \in (0, 1)$.

theorem *ln_euler_phi_ge*:
fixes x :: real
assumes x: "x \in {0<.. <1 }"
shows "ln (euler_phi x) \geq -pi² / 6 * x / (1 - x)"
<proof>

6.3 Logarithmic derivative

The logarithmic derivative of ϕ is given by the following Lambert-style series:

$$\frac{\phi'(q)}{\phi(q)} = \sum_{k=0}^{\infty} (k+1) \frac{q^k}{q^{k+1} - 1}$$

```

lemma sums_logderiv_euler_phi:
  fixes q :: complex
  assumes q: "norm q < 1"
  shows "( $\lambda k. \text{of\_nat } (\text{Suc } k) * q ^ k / (q ^ \text{Suc } k - 1)) \text{ sums } (\text{deriv } \text{euler\_phi } q / \text{euler\_phi } q)"
  \langle proof \rangle

lemma deriv_euler_phi_aux:
  fixes q :: complex
  assumes q: "norm q < 1"
  shows "q * deriv euler_phi q = -lambert of_nat q * euler_phi q"
  \langle proof \rangle

theorem deriv_euler_phi:
  fixes q :: complex
  assumes q: "norm q < 1" "q  $\neq$  0"
  shows "deriv euler_phi q = -lambert of_nat q * euler_phi q / q"
  \langle proof \rangle

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.