Bessel Functions of the First Kind

Manuel Eberl

July 14, 2026

Abstract

This entry provides a formalisation of the Bessel function of the first kind J_a and the modified Bessel function of the first kind I_a as well as several related notions such as Bessel polynomials and the spherical Bessel functions.

Properties that are proven about I_a and J_a include:

- the definition as a hypergeometric series
$$\sum_{n\geq 0}(-1)^n(z/2)^{a+2n}/(n!\Gamma(1+a+n))$$
- monotonicity and convexity of I_a on the reals
- the contiguous relations, e.g. $2aJ_a(z) = z(J_{a-1}(z) + J_{a+1}(z))$
- the derivatives, e.g. $J'_a = \frac{1}{2}(J_{a-1} - J_{a+1})$
- closed form expressions for the half-integer case $a = n + \frac{1}{2}$ in terms of trigonometric and hyperbolic functions
- the representation of J_a as contour integral of the form $\int_{c-i\infty}^{c+i\infty}$

Contents

1 Bessel functions

```
theory Bessel
  imports "Incomplete_Gamma.Incomplete_Gamma"
begin
```

1.1 Auxiliary material

```
lemma rGamma_real_nonneg [simp]:
  assumes "x ≥ (0::real)"
  shows   "rGamma x ≥ 0"
  ⟨proof⟩


lemma sinh_of_real: "sinh (of_real x :: 'a :: {real_normed_field, banach})
= of_real (sinh x)"
  ⟨proof⟩


lemma cosh_of_real: "cosh (of_real x :: 'a :: {real_normed_field, banach})
= of_real (cosh x)"
  ⟨proof⟩


lemma plus_of_nat_in_nonpos_IntsD:
  assumes "x + of_nat n ∈ ℤ≤0"
  shows   "x ∈ ℤ≤0"
⟨proof⟩


lemma plus_of_nat_notin_nonpos_Ints:
  assumes "x ∉ ℤ≤0"
  shows   "x + of_nat n ∉ ℤ≤0"
  ⟨proof⟩


lemma plus1_notin_nonpos_Ints:
  assumes "x ∉ ℤ≤0"
  shows   "x + 1 ∉ ℤ≤0"
  ⟨proof⟩


lemma plus_numeral_notin_nonpos_Ints:
  assumes "x ∉ ℤ≤0"
  shows   "x + numeral n ∉ ℤ≤0"
  ⟨proof⟩
```

1.2 Hyperbolic sine and cosine as formal power series

```
definition fps_sinh :: "'a :: {inverse, ring_1} ⇒ 'a fps" where
```

"fps_sinh c = fps_const (1/2) * (fps_exp c - fps_exp (-c))"

definition *fps_cosh :: "'a :: {inverse, ring_1} ⇒ 'a fps"* **where**
"fps_cosh c = fps_const (1/2) * (fps_exp c + fps_exp (-c))"

lemma *fps_nth_cosh:*
fixes *c :: "'a :: field_char_0"*
shows *"fps_nth (fps_cosh c) n = (if even n then c ^ n / of_nat (fact n) else 0)"*
⟨*proof*⟩

lemma *has_fps_expansion_sinh [fps_expansion_intros]:*
fixes *c :: "'a :: {banach, real_normed_field, field_char_0}"*
shows *"(λx. sinh (c * x)) has_fps_expansion fps_sinh c"*
⟨*proof*⟩

lemma *has_fps_expansion_sinh' [fps_expansion_intros]:*
"(λx::'a :: {banach, real_normed_field}. sinh x) has_fps_expansion fps_sinh 1"
⟨*proof*⟩

lemma *has_fps_expansion_cosh [fps_expansion_intros]:*
fixes *c :: "'a :: {banach, real_normed_field, field_char_0}"*
shows *"(λx. cosh (c * x)) has_fps_expansion fps_cosh c"*
⟨*proof*⟩

lemma *has_fps_expansion_cosh' [fps_expansion_intros]:*
"(λx::'a :: {banach, real_normed_field}. cosh x) has_fps_expansion fps_cosh 1"
⟨*proof*⟩

1.3 The cardinal hyperbolic sine

definition *sinch :: "'a ⇒ 'a :: {banach, real_normed_field, field_char_0}"* **where**
"sinch z = hypergeo_F [] [3 / 2] (z^2 / 4)"

lemma *sinch_altdef: "sinch z = (if z = 0 then 1 else sinh z / z)"*
⟨*proof*⟩

lemma *sinch_0 [simp]: "sinch 0 = 1"*
⟨*proof*⟩

lemma *sinch_of_real: "sinch (of_real x) = of_real (sinch x)"*
⟨*proof*⟩

lemma *sums_sinch': "(λn. x ^ (2*n) / fact (2 * n + 1)) sums sinch x"*
⟨*proof*⟩

lemma *sums_sinch: "(λn. if even n then x ^ n / fact (n+1) else 0) sums sinch x"*
⟨*proof*⟩

lemma *has_field_derivative_sinch:*
assumes *"x ≠ 0"*
shows *"(sinch has_field_derivative (cosh x / x - sinh x / x ^ 2)) (at x within A)"*
⟨*proof*⟩

lemma *analytic_on_sinch [analytic_intros]:*
"f analytic_on A ⟹ (λz. sinch (f z)) analytic_on A"
⟨*proof*⟩

lemma *holomorphic_on_sinch [holomorphic_intros]:*
"f holomorphic_on A ⟹ (λz. sinch (f z)) holomorphic_on A"
⟨*proof*⟩

lemma *continuous_on_sinch [continuous_intros]:*
"continuous_on A f ⟹ continuous_on A (λz. sinch (f z))"
⟨*proof*⟩

lemma *subdegree_fps_sinh [simp]: "(c :: 'a :: field_char_0) ≠ 0 ⟹ subdegree (fps_sinh c) = 1"*
⟨*proof*⟩

lemma *has_fps_expansion_sinch [fps_expansion_intros]:*
assumes *[simp]: "c ≠ 0"*
shows *"(λz. sinch (c * z)) has_fps_expansion fps_sinh c / (fps_const c * fps_X)"*
⟨*proof*⟩

lemma *has_fps_expansion_sinch' [fps_expansion_intros]:*
"sinch has_fps_expansion fps_sinh 1 / fps_X"
⟨*proof*⟩

lemma *has_laurent_expansion_sinch [laurent_expansion_intros]:*
assumes *[simp]: "c ≠ 0"*
shows *"(λz. sinch (c * z)) has_laurent_expansion fps_to_fls (fps_sinh c) / (fls_const c * fls_X)"*
⟨*proof*⟩

lemma *has_laurent_expansion_sinch' [laurent_expansion_intros]:*

```
"sinch has_laurent_expansion fps_to_fls (fps_sinh 1) / fls_X"
⟨proof⟩
```

1.4 Bessel polynomials

The Bessel polynomials $y_n(X)$ are defined by the following recurrence:

$$\begin{aligned} y_0(X) &= 1 \\ y_1(X) &= 1 + X \\ y_n(X) &= (2n-1)Xy_{n-1}(X) + y_{n-2}(X) \end{aligned}$$

Later, we will additionally show the following alternative recurrence:

$$y_n(X) = (1 + nX)y_{n-1}(X) + X^2 y'_{n-1}(X)$$

```
fun bessel_poly :: "nat ⇒ 'a :: semidom poly" where
  "bessel_poly 0 = 1"
| "bessel_poly (Suc 0) = [:1, 1:]"
| "bessel_poly (Suc (Suc n)) = monom (of_nat (2*n+3)) 1 * bessel_poly
(Suc n) + bessel_poly n"

lemma coeff_bessel_poly_eq_0_aux:
  "m > n ⟹ coeff (bessel_poly n) m = 0"
  ⟨proof⟩

lemma bessel_poly_conv_bessel_poly_of_nat:
  "bessel_poly n = map_poly of_nat (bessel_poly n)"
⟨proof⟩
```

Their coefficients ([**?**, A001498] on OEIS) have the following closed form:

$$[X^m]y_m(X) = \frac{(n+m)!}{2^m m!(n-m)!}$$

```
lemma coeff_bessel_poly_conv_fact_aux1:
  "m ≤ n ⟹ real (coeff (bessel_poly n) m) = fact (n+m) / (fact m *
fact (n-m) * 2^m)"
⟨proof⟩

lemma degree_bessel_poly: "degree (bessel_poly n :: 'a :: {semidom, semiring_char_0}
poly) = n"
⟨proof⟩
```

The coefficients of the Bessel polynomials also satisfy the following recurrence:

```
fun bessel_poly_coeff :: "nat ⇒ nat ⇒ nat" where
  "bessel_poly_coeff n 0 = 1"
```

```
| "bessel_poly_coeff 0 (Suc m) = 0"
| "bessel_poly_coeff (Suc n) (Suc m) = bessel_poly_coeff n (Suc m) + (n
+ m + 1) * bessel_poly_coeff n m"
```

lemma *bessel_poly_coeff_eq_0 [simp]: "m > n ⟹ bessel_poly_coeff n m = 0"*
⟨*proof*⟩

lemma *bessel_poly_coeff_pos: "m ≤ n ⟹ bessel_poly_coeff n m > 0"*
⟨*proof*⟩

lemma *bessel_poly_coeff_eq_0_iff: "bessel_poly_coeff n m = 0 ⟷ m > n"*
⟨*proof*⟩

lemma *bessel_poly_coeff_0_left: "m > 0 ⟹ bessel_poly_coeff 0 m = 0"*
⟨*proof*⟩

lemma *bessel_poly_coeff_conv_fact_aux2:*
*"m ≤ n ⟹ real (bessel_poly_coeff n m) = fact (n+m) / (fact m * fact (n-m) * 2^m)"*
⟨*proof*⟩

lemma *bessel_poly_coeff_conv_fact:*
assumes *"m ≤ n"*
shows *"bessel_poly_coeff n m = fact (n+m) div (fact m * fact (n-m) * 2^m)"*
⟨*proof*⟩

lemma *bessel_poly_coeff_conv_fact':*
*"bessel_poly_coeff n m = (if m ≤ n then fact (n+m) div (fact m * fact (n-m) * 2^m) else 0)"*
⟨*proof*⟩

lemma *coeff_bessel_poly: "coeff (bessel_poly n) m = of_nat (bessel_poly_coeff n m)"*
⟨*proof*⟩

The Bessel polynomials also satisfy the following recurrence with respect to their derivative:

$$y_n(X) = (1 + nX)y_{n-1}(X) + X^2 y'_{n-1}(X)$$

lemma *bessel_poly_Suc_conv_pderiv:*
*"bessel_poly (Suc n) = [:1, of_nat n + 1:] * bessel_poly n + [:0,0,1:] * pderiv (bessel_poly n)"*
⟨*proof*⟩

1.5 Spherical Bessel and Hankel functions

1.5.1 The spherical Bessel function of the first kind

The spherical Bessel functions of the first kind $j_n(z)$ are defined by letting $j_0(z) = \sin z/z$ and $j_{-1} = \cos z/z$ and then extending the definition to all integers n via the following recurrence:

$$(2n + 1)j_n(z) = z(j_{n+1}(z) + j_{n-1}(z))$$

```
function SBessel_J :: "int ⇒ 'a :: {banach, real_normed_field} ⇒ 'a"
where
  "SBessel_J 0 = (λz. sin z / z)"
| "SBessel_J (-1) = (λz. cos z / z)"
| "n > 0 ⟹ SBessel_J n = (λz. (2 * of_int n - 1) / z * SBessel_J (n-1)
z - SBessel_J (n-2) z)"
| "n < -1 ⟹ SBessel_J n = (λz. (2 * of_int n + 3) / z * SBessel_J (n+1)
z - SBessel_J (n+2) z)"
  ⟨proof⟩
termination
  ⟨proof⟩

lemmas [simp del] = SBessel_J.simps(3,4)

lemma SBessel_J_induct:
  "P 0 ⟹ P (-1) ⟹
    (⋀n. 0 < n ⟹ (P (n - 1)) ⟹ (P (n - 2)) ⟹ P n) ⟹
    (⋀n. n < - 1 ⟹ (P (n + 1)) ⟹ (P (n + 2)) ⟹ P n) ⟹
  P (n :: int)"
  ⟨proof⟩

lemma SBessel_J_0_right [simp]: "SBessel_J n 0 = 0"
  ⟨proof⟩

lemma SBessel_J_of_real: "SBessel_J n (of_real x) = of_real (SBessel_J
n x)"
  ⟨proof⟩

lemma SBessel_J_minus_right: "SBessel_J n (-z) = (-1) powi n * SBessel_J
n z"
  ⟨proof⟩

lemma SBessel_J_contiguous:
  assumes "z ≠ 0"
  shows   "(2 * of_int n + 1) * SBessel_J n z = z * (SBessel_J (n+1) z
+ SBessel_J (n-1) z)"
⟨proof⟩
```

1.5.2 The spherical Bessel function of the second kind

For convenience, we also define the closely related spherical Bessel functions of the second kind $y_n(z)$. They satisfy the same recurrence, but with the initial conditions $y_0(z) = -\cos z/z$ and $y_{-1}(z) = \sin z/z$. They can easily be expressed in terms of $j_n(z)$ and vice versa.

definition *SBessel_Y :: "int ⇒ 'a :: {real_normed_field,banach} ⇒ 'a"*
  where *"SBessel_Y n z = (-1) powi (n+1) * SBessel_J (-n-1) z"*

lemma *SBessel_Y_0_right [simp]: "SBessel_Y n 0 = 0"*
  ⟨*proof*⟩

lemma *SBessel_Y_conv_J: "SBessel_Y n z = (-1) powi (n+1) * SBessel_J (-n-1) z"*
  ⟨*proof*⟩

lemma *SBessel_J_conv_Y: "SBessel_J n z = (-1) powi n * SBessel_Y (-n-1) z"*
  ⟨*proof*⟩

lemma *SBessel_Y_minus_left: "SBessel_Y (-n) z = (-1) powi (n+1) * SBessel_J (n - 1) z"*
  ⟨*proof*⟩

lemma *SBessel_J_minus_left: "SBessel_J (-n) z = (-1) powi n * SBessel_Y (n - 1) z"*
  ⟨*proof*⟩

lemma *SBessel_Y_minus_right: "SBessel_Y n (-z) = (-1) powi (n+1) * SBessel_Y n z"*
  ⟨*proof*⟩

lemma *SBessel_Y_of_real: "SBessel_Y n (of_real x) = of_real (SBessel_Y n x)"*
  ⟨*proof*⟩

lemma *SBessel_Y_contiguous:*
  assumes *"z ≠ 0"*
  shows *"(2 * of_int n + 1) * SBessel_Y n z = z * (SBessel_Y (n - 1) z + SBessel_Y (n + 1) z)"*
⟨*proof*⟩

1.5.3 The modified spherical Bessel function of the first kind

The modified spherical Bessel functions of the first kind are the hyperbolic versions of $j_n(z)$. They satisfy a slightly different recurrence and, in the complex case, can easily be written in terms of j_n and vice versa.

function *SBessel_I :: "int ⇒ 'a :: {banach, real_normed_field} ⇒ 'a"*

```
where
  "SBessel_I 0 = (λz. sinh z / z)"
| "SBessel_I (-1) = (λz. cosh z / z)"
| "n > 0 ⟹ SBessel_I n = (λz. -(2 * of_int n - 1) / z * SBessel_I (n-1)
z + SBessel_I (n-2) z)"
| "n < -1 ⟹ SBessel_I n = (λz. (2 * of_int n + 3) / z * SBessel_I (n+1)
z + SBessel_I (n+2) z)"
  ⟨proof⟩
termination
  ⟨proof⟩

lemmas [simp del] = SBessel_I.simps(3,4)

lemma SBessel_I_0_right [simp]: "SBessel_I n 0 = 0"
  ⟨proof⟩

lemma SBessel_I_of_real: "SBessel_I n (of_real x) = of_real (SBessel_I
n x)"
  ⟨proof⟩

lemma SBessel_I_minus_right: "SBessel_I n (-z) = (-1) powi n * SBessel_I
n z"
  ⟨proof⟩

lemma SBessel_I_contiguous:
  assumes "z ≠ 0"
  shows   "(2 * of_int n + 1) * SBessel_I n z = z * (SBessel_I (n-1) z
- SBessel_I (n+1) z)"
⟨proof⟩

lemma SBessel_I_conv_J: "SBessel_I n z = (-i) powi n * SBessel_J n (i
* z)"
⟨proof⟩

lemma SBessel_J_conv_I: "SBessel_J n z = i powi n * SBessel_I n (-i *
z)"
  ⟨proof⟩
```

1.5.4 Spherical Hankel functions

The spherical Hankel functions are complex functions that can be written as linear combinations of the spherical Bessel functions. Therefore, they also satisfy the same contiguous relations.

```
definition SHankel_1 :: "int ⇒ complex ⇒ complex"
  where "SHankel_1 n z = SBessel_J n z + i * SBessel_Y n z"

definition SHankel_2 :: "int ⇒ complex ⇒ complex"
  where "SHankel_2 n z = SBessel_J n z - i * SBessel_Y n z"
```

lemma *SHankel_1_minus_left: "SHankel_1 (-n) z = -i * (-1) powi n * SHankel_1 (n-1) z"*
⟨*proof*⟩

lemma *SHankel_2_minus_left: "SHankel_2 (-n) z = i * (-1) powi n * SHankel_2 (n-1) z"*
⟨*proof*⟩

lemma *SHankel_1_minus_right: "SHankel_1 n (-z) = (-1) powi n * SHankel_2 n z"*
⟨*proof*⟩

lemma *SHankel_2_minus_right: "SHankel_2 n (-z) = (-1) powi n * SHankel_1 n z"*
⟨*proof*⟩

lemma *SBessel_J_conv_SHankel: "SBessel_J n z = (SHankel_1 n z + SHankel_2 n z) / 2"*
⟨*proof*⟩

lemma *SBessel_Y_conv_SHankel: "SBessel_Y n z = (SHankel_1 n z - SHankel_2 n z) / (2 * i)"*
⟨*proof*⟩

lemma *SHankel_1_contiguous:*
assumes *"z ≠ 0"*
shows *"(2 * of_int n + 1) * SHankel_1 n z = z * (SHankel_1 (n - 1) z + SHankel_1 (n + 1) z)"*
⟨*proof*⟩

lemma *SHankel_2_contiguous:*
assumes *"z ≠ 0"*
shows *"(2 * of_int n + 1) * SHankel_2 n z = z * (SHankel_2 (n - 1) z + SHankel_2 (n + 1) z)"*
⟨*proof*⟩

The spherical Hankel functions have a simple closed form in terms of the exponential and the Bessel polynomials.

lemma *SHankel_1_nonneg_eq:*
assumes *"z ≠ 0"*
shows *"SHankel_1 (int n) z = (-i)^(n+1) * exp (i*z) * poly (bessel_poly n) (i/z) / z"*
⟨*proof*⟩

lemma *SHankel_2_nonneg_eq:*
assumes *"z ≠ 0"*
shows *"SHankel_2 (int n) z = i^(n+1) * exp (-i*z) * poly (bessel_poly n) (-i/z) / z"*

⟨*proof*⟩

lemma *SHankel_1_neg_eq:*
  assumes *"z ≠ 0"*
  shows *"SHankel_1 (-int (n+1)) z = i ^ n * exp (i * z) * poly (bessel_poly n) (i / z) / z"*
⟨*proof*⟩

lemma *SHankel_2_neg_eq:*
  assumes *"z ≠ 0"*
  shows *"SHankel_2 (-int (n+1)) z = (-i) ^ n * exp (-i * z) * poly (bessel_poly n) (-i / z) / z"*
⟨*proof*⟩

1.5.5 Closed form expressions

We can now give closed-form expressions for all the spherical Bessel functions in terms of sin, cos, sinh, and cosh.

lemma *SBessel_J_conv_sin_cos_nonneg_complex:*
  fixes *z :: complex* **and** *n :: nat*
  shows *"SBessel_J (int n) z =
            (∑k≤n. (-1)^((n+k+1) div 2 + k) * of_nat (bessel_poly_coeff n k) / z^(k+1) *
                  (if even (n + k) then sin z else cos z))"*
⟨*proof*⟩

lemma *SBessel_Y_conv_sin_cos_nonneg_complex:*
  fixes *z :: complex* **and** *n :: nat*
  shows *"SBessel_Y (int n) z =
            (∑k≤n. (-1)^((n+k) div 2 + k + 1) * of_nat (bessel_poly_coeff n k) / z^(k+1) *
                  (if even (n + k) then cos z else sin z))"*
⟨*proof*⟩

lemma *SBessel_J_conv_sin_cos_neg_complex:*
  fixes *z :: complex* **and** *n :: nat*
  shows *"SBessel_J (-int (n+1)) z =
            (∑k≤n. (-1)^(n + k + (n+k) div 2) * of_nat (bessel_poly_coeff n k) / z^(k+1) *
                  (if even (n + k) then cos z else sin z))"*
⟨*proof*⟩

lemma *SBessel_Y_conv_sin_cos_neg_complex:*
  fixes *z :: complex* **and** *n :: nat*
  shows *"SBessel_Y (-int (n+1)) z =
            (∑k≤n. (-1)^(n + k + (n+k+1) div 2) * of_nat (bessel_poly_coeff n k) / z^(k+1) *
                  (if even (n + k) then sin z else cos z))"*
⟨*proof*⟩

```
lemma SBessel_I_conv_sinh_cosh_nonneg_complex:
  fixes z :: complex and n :: nat
  shows "SBessel_I (int n) z =
           (∑k≤n. (-1)^k * of_nat (bessel_poly_coeff n k) / z^(k+1)
*
                      (if even (n + k) then sinh z else cosh z))"
⟨proof⟩
```

```
lemma SBessel_I_conv_sinh_cosh_neg_complex:
  fixes z :: complex and n :: nat
  shows "SBessel_I (-int (n+1)) z =
           (∑k≤n. (-1)^k * of_nat (bessel_poly_coeff n k) / z^(k+1)
*
                      (if even (n + k) then cosh z else sinh z))"
⟨proof⟩
```

Variants of the above for the real-valued functions:

```
lemma SBessel_J_conv_sin_cos_nonneg_real:
  fixes z :: real and n :: nat
  shows "SBessel_J (int n) z =
           (∑k≤n. (-1)^((n+k+1) div 2 + k) * of_nat (bessel_poly_coeff
n k) / z^(k+1) *
                      (if even (n + k) then sin z else cos z))" (is "_
= ?rhs")
⟨proof⟩
```

```
lemma SBessel_Y_conv_sin_cos_nonneg_real:
  fixes z :: real and n :: nat
  shows "SBessel_Y (int n) z =
           (∑k≤n. (-1)^((n+k) div 2 + k + 1) * of_nat (bessel_poly_coeff
n k) / z^(k+1) *
                      (if even (n + k) then cos z else sin z))" (is "_
= ?rhs")
⟨proof⟩
```

```
lemma SBessel_J_conv_sin_cos_neg_real:
  fixes z :: real and n :: nat
  shows "SBessel_J (-int (n+1)) z =
            (∑k≤n. (-1)^(n + k + (n+k) div 2) * of_nat (bessel_poly_coeff
n k) / z^(k+1) *
                      (if even (n + k) then cos z else sin z))" (is "_
= ?rhs")
⟨proof⟩
```

```
lemma SBessel_Y_conv_sin_cos_neg_real:
  fixes z :: real and n :: nat
  shows "SBessel_Y (-int (n+1)) z =
           (∑k≤n. (-1)^(n + k + (n+k+1) div 2) * of_nat (bessel_poly_coeff
```

```
n k) / z^(k+1) *
                        (if even (n + k) then sin z else cos z))" (is "_
= ?rhs")
```
⟨proof⟩

lemma *SBessel_I_conv_sinh_cosh_nonneg_real:*
fixes *z :: real* **and** *n :: nat*
shows *"SBessel_I (int n) z =
(∑k≤n. (-1)^k * of_nat (bessel_poly_coeff n k) / z^(k+1)
*
(if even (n + k) then sinh z else cosh z))"* (**is** *"_
= ?rhs"*)
⟨proof⟩

lemma *SBessel_I_conv_sinh_cosh_neg_real:*
fixes *z :: real* **and** *n :: nat*
shows *"SBessel_I (-int (n+1)) z =
(∑k≤n. (-1)^k * of_nat (bessel_poly_coeff n k) / z^(k+1)
*
(if even (n + k) then cosh z else sinh z))"* (**is** *"_
= ?rhs"*)
⟨proof⟩

experiment
begin

As an example: the close form of $j_3(z)$:

lemma *"SBessel_J 3 (z :: complex) =
15 * sin z / z ^ 4 + cos z / z - 6 * sin z / z ^ 2 - 15 * cos
z / z ^ 3"*
⟨proof⟩

end

1.6 The Bessel–Clifford function

The Bessel–Clifford function C_a is a useful building block to construct the Bessel functions. It is a holomorphic function in two variables and therefore very well-behaved. It can be defined most easily via the regularised hypergeometric function.

Two important properties are that it is is that it satisfies $\frac{d}{dz}C_a(z) = C_{a+1}(z)$ and the contiguous relation $C_a(z) = (a+1)C_{a+1}(z) + zC_{a+2}(z)$, which already hints at its connection to the Bessel functions.

A direct consequence of this (which we do not show here) is that it is a solution of the ODE $y = (a+1)y' + xy''$.

We get these properties mostly for free using the development on hypergeometric functions.

definition *Bessel_Clifford :: "'a :: Gamma ⇒ 'a ⇒ 'a"* **where**
"Bessel_Clifford a = reg_hypergeo_F [] [a+1]"

lemma *Bessel_Clifford_altdef: "Bessel_Clifford a = eval_fps (fps_bessel a)"*
⟨*proof*⟩

lemma *Bessel_Clifford_conv_hypergeo_F:*
*"a + 1 ∉ $\mathbb{Z}_{\leq 0}$ ⟹ Bessel_Clifford a z = rGamma (a+1) * hypergeo_F [] [a+1] z"*
⟨*proof*⟩

lemma *Bessel_Clifford_complex_of_real:*
"Bessel_Clifford (of_real a) (of_real z) = complex_of_real (Bessel_Clifford a z)"
⟨*proof*⟩

lemma *sums_Bessel_Clifford:*
*"(λn. rGamma (a + of_nat (Suc n)) * z ^ n / fact n) sums Bessel_Clifford a z"*
⟨*proof*⟩

lemma *Bessel_Clifford_contiguous:*
*"Bessel_Clifford a z = (a+1) * Bessel_Clifford (a+1) z + z * Bessel_Clifford (a+2) z"*
⟨*proof*⟩

lemma *Bessel_Clifford_0_right [simp]: "Bessel_Clifford a 0 = rGamma (a + 1)"*
⟨*proof*⟩

lemma *Bessel_Clifford_minus_of_nat:*
*"Bessel_Clifford (-of_nat n) z = z ^ n * Bessel_Clifford (of_nat n) z"*
⟨*proof*⟩

lemma *Bessel_Clifford_minus_of_int:*
assumes *"n ≥ 0 ∨ z ≠ 0"*
shows *"Bessel_Clifford (-of_int n) z = z powi n * Bessel_Clifford (of_int n) z"*
⟨*proof*⟩

lemma *analytic_Bessel_Clifford [analytic_intros]:*
assumes *"a analytic_on A" "b analytic_on A"*
shows *"(λx. Bessel_Clifford (a x) (b x)) analytic_on A"*
⟨*proof*⟩

lemma *has_field_derivative_Bessel_Clifford [derivative_intros]:*

assumes *"(f has_field_derivative f') (at x within A)"*
shows *"((λx. Bessel_Clifford a (f x)) has_field_derivative (Bessel_Clifford (a+1) (f x) * f')) (at x within A)"*
⟨*proof*⟩

lemma *Bessel_Clifford_real_mono:*
assumes *xy: "0 ≤ (x::real)" "x ≤ y" "a ≥ -1"*
shows *"Bessel_Clifford a x ≤ Bessel_Clifford a y"*
⟨*proof*⟩

lemma *Bessel_Clifford_real_pos [simp]:*
assumes *"x ≥ (0::real)" "a > -1"*
shows *"Bessel_Clifford a x > 0"*
⟨*proof*⟩

lemma *Bessel_Clifford_real_nonneg [simp]: "x ≥ (0::real) ⟹ a ≥ -1 ⟹ Bessel_Clifford a x ≥ 0"*
⟨*proof*⟩

lemma *Bessel_Clifford_convex_real:*
assumes *"a ≥ -3"*
shows *"convex_on {0..} (Bessel_Clifford (a::real))"*
⟨*proof*⟩

1.7 The Bessel function of the first kind

The Bessel function of the first kind J_a(z) can now easily be derived from the Bessel–Clifford function with the change of variables $z \mapsto -z^2/4$ and multiplication with $(z/2)^a$ (from which it gets its branch cut). All the basic properties follow directly from the ones we have for the Bessel–Clifford function.

definition *Bessel_J :: "'a :: {Gamma, ln} ⇒ 'a ⇒ 'a"* **where**
*"Bessel_J a z = (z / 2) powr' a * Bessel_Clifford a (-(z²/4))"*

lemma *Bessel_J_0_0 [simp]: "Bessel_J 0 0 = 1"*
⟨*proof*⟩

lemma *Bessel_J_0_right [simp]: "a ≠ 0 ⟹ Bessel_J a 0 = 0"*
⟨*proof*⟩

lemma *Bessel_J_0_right': "Bessel_J a 0 = (if a = 0 then 1 else 0)"*
⟨*proof*⟩

lemma *Bessel_J_complex_of_real:*
assumes *"a ∈ ℤ ∨ z ≥ 0"*
shows *"Bessel_J (complex_of_real a) (of_real z) = of_real (Bessel_J a z)"*
⟨*proof*⟩

```
lemma Bessel_J_contiguous_complex:
  fixes a z :: complex
  shows "2 * a * Bessel_J a z = z * (Bessel_J (a-1) z + Bessel_J (a+1)
z)"
⟨proof⟩

lemma Bessel_J_contiguous_real:
  fixes a z :: real
  assumes "z ≥ 0 ∨ a ∈ ℤ"
  shows   "2 * a * Bessel_J a z = z * (Bessel_J (a-1) z + Bessel_J (a+1)
z)"
⟨proof⟩

lemma Bessel_J_minus_of_nat_complex:
  "Bessel_J (-of_nat n :: complex) z = (-1) ^ n * Bessel_J (of_nat n)
z"
⟨proof⟩

lemma Bessel_J_minus_of_int_complex:
  "Bessel_J (-of_int n :: complex) z = (-1) powi n * Bessel_J (of_int
n) z"
⟨proof⟩

lemma Bessel_J_minus_of_int_real:
  "Bessel_J (-of_int n :: real) z = (-1) powi n * Bessel_J (of_int n)
z"
⟨proof⟩

lemma Bessel_J_minus_of_nat_real:
  "Bessel_J (-of_nat n :: real) z = (-1) powi n * Bessel_J (of_nat n)
z"
  ⟨proof⟩

lemma sums_Bessel_J:
  fixes a z :: "'a :: {Gamma, ln}"
  shows  "(λn. (-1)^n * (z/2) powr' a * (z/2) ^ (2*n) / (fact n * Gamma
(1 + a + of_nat n))) sums
            Bessel_J a z"
⟨proof⟩

lemma sums_Bessel_J_complex:
  fixes a z :: complex
  assumes "z ≠ 0 ∨ a ∉ ℤ≤0"
  shows  "(λn. (-1)^n * (z/2) powr' (a + 2 * of_nat n) / (fact n * Gamma
(1 + a + of_nat n))) sums
            Bessel_J a z"
⟨proof⟩
```

lemma *sums_Bessel_J_real:*
fixes *a z :: real*
assumes *"a ∈ ℤ ∨ z ≥ 0" "z ≠ 0 ∨ a ∉ $\mathbb{Z}_{\leq 0}$"*
shows *"(λn. (-1)^n * (z/2) powr' (a + 2 * of_nat n) / (fact n * Gamma (1 + a + of_nat n))) sums Bessel_J a z"*
⟨*proof*⟩

lemma *has_field_derivative_Bessel_J_complex:*
assumes *"a ∈ ℤ ∨ (z::complex) ∉ $\mathbb{R}_{\leq 0}$"*
shows *"(Bessel_J a has_field_derivative ((Bessel_J (a-1) z - Bessel_J (a+1) z) / 2)) (at z within A)"*
⟨*proof*⟩

lemma *has_field_derivative_Bessel_J_complex' [derivative_intros]:*
assumes *"(f has_field_derivative f') (at x within A)" "a ∈ ℤ ∨ (f x :: complex) ∉ $\mathbb{R}_{\leq 0}$"*
shows *"((λx. Bessel_J a (f x)) has_field_derivative (f' * (Bessel_J (a-1) (f x) - Bessel_J (a+1) (f x)) / 2)) (at x within A)"*
⟨*proof*⟩

lemma *has_field_derivative_Bessel_J_real:*
assumes *"a ∈ ℤ ∨ (z :: real) > 0"*
shows *"(Bessel_J a has_field_derivative ((Bessel_J (a-1) z - Bessel_J (a+1) z) / 2)) (at z within A)"*
⟨*proof*⟩

lemma *has_field_derivative_Bessel_J_real' [derivative_intros]:*
assumes *"(f has_field_derivative f') (at x within A)" "a ∈ ℤ ∨ f x > (0 :: real)"*
shows *"((λx. Bessel_J a (f x)) has_field_derivative ((Bessel_J (a-1) (f x) - Bessel_J (a+1) (f x)) / 2) * f') (at x within A)"*
⟨*proof*⟩

lemma *analytic_Bessel_J [analytic_intros]:*
assumes *"a analytic_on A" "b analytic_on A" "⋀x. x ∈ A ⟹ b x ∉ $\mathbb{R}_{\leq 0}$"*
shows *"(λx. Bessel_J (a x) (b x)) analytic_on A"*
⟨*proof*⟩

lemma *analytic_Bessel_J_Ints:*
assumes *"b analytic_on A" "a ∈ ℤ"*
shows *"(λx. Bessel_J a (b x)) analytic_on A"*
⟨*proof*⟩

The half-integer case $J_{n+\frac{1}{2}}$ can be expressed in terms of spherical Bessel functions of the first kind j_n and therefore has a closed form.

```
theorem Bessel_J_conv_SBessel_J_complex:
  assumes z: "z ≠ (0 :: complex)"
  defines "C ≡ sqrt (2 / pi) * csqrt z"
  shows   "Bessel_J (of_int n + 1 / 2) z = C * SBessel_J n z"
⟨proof⟩

lemma Bessel_J_conv_SBessel_J_real:
  assumes z: "z > (0 :: real)"
  shows   "Bessel_J (of_int n + 1 / 2) z = sqrt (2 * z / pi) * SBessel_J
n z"
⟨proof⟩
```

1.8 The modified Bessel function of the first kind

Again, the modified Bessel function of the first kind I_a is essentially the hyperbolic version of J_a. In the complex case, it can also easily be written in terms of I_a and vice versa.

```
definition Bessel_I :: "'a :: {Gamma, ln} ⇒ 'a ⇒ 'a" where
  "Bessel_I a z = (z / 2) powr' a * Bessel_Clifford a (z²/4)"

lemma Bessel_I_0_0 [simp]: "Bessel_I 0 0 = 1"
  ⟨proof⟩

lemma Bessel_I_0_right [simp]: "a ≠ 0 ⟹ Bessel_I a 0 = 0"
  ⟨proof⟩

lemma Bessel_I_0_right': "Bessel_I a 0 = (if a = 0 then 1 else 0)"
  ⟨proof⟩

lemma Bessel_I_complex_of_real:
  assumes "a ∈ ℤ ∨ z ≥ 0"
  shows   "Bessel_I (complex_of_real a) (of_real z) = of_real (Bessel_I
a z)"
  ⟨proof⟩

lemma Bessel_I_contiguous_complex:
  fixes a z :: complex
  shows "z * Bessel_I a z = 2 * (a + 1) * Bessel_I (a+1) z + z * Bessel_I
(a+2) z"
⟨proof⟩

lemma Bessel_I_contiguous_real:
  fixes a z :: real
  assumes "z ≥ 0 ∨ a ∈ ℤ"
  shows   "z * Bessel_I a z = 2 * (a + 1) * Bessel_I (a+1) z + z * Bessel_I
(a+2) z"
```

⟨proof⟩

lemma *Bessel_I_minus_of_nat_complex:*
  "Bessel_I (-of_nat n :: complex) z = Bessel_I (of_nat n) z"
⟨proof⟩

lemma *Bessel_I_minus_of_int_complex:*
  "Bessel_I (-of_int n :: complex) z = Bessel_I (of_int n) z"
⟨proof⟩

lemma *Bessel_I_minus_of_int_real:*
  "Bessel_I (-of_int n :: real) z = Bessel_I (of_int n) z"
⟨proof⟩

lemma *Bessel_I_minus_of_nat_real:*
  "Bessel_I (-of_nat n :: real) z = Bessel_I (of_nat n) z"
  ⟨proof⟩

lemma *Bessel_I_conv_J:*
  assumes *"a ∈ ℤ ∨ Re z ≥ 0 ∨ Im z < 0"*
  shows *"Bessel_I a z = i powr (-a) * Bessel_J a (i * z)"*
⟨proof⟩

lemma *Bessel_J_conv_I:*
  assumes *"a ∈ ℤ ∨ Re x > 0 ∨ Im x ≥ 0"*
  shows *"Bessel_J a x = i powr a * Bessel_I a (-i * x)"*
  ⟨proof⟩

lemma *sums_Bessel_I:*
  fixes *a z :: "'a :: {Gamma, ln}"*
  shows *"(λn. (z/2) powr' a * (z/2) ^ (2*n) / (fact n * Gamma (1 + a + of_nat n))) sums*
            Bessel_I a z"
⟨proof⟩

lemma *sums_Bessel_I_complex:*
  fixes *a z :: complex*
  assumes *"z ≠ 0 ∨ a ∉ $\mathbb{Z}_{\leq 0}$"*
  shows *"(λn. (z/2) powr' (a + 2 * of_nat n) / (fact n * Gamma (1 + a + of_nat n))) sums*
            Bessel_I a z"
⟨proof⟩

lemma *sums_Bessel_I_real:*
  fixes *a z :: real*
  assumes *"a ∈ ℤ ∨ z ≥ 0" "z ≠ 0 ∨ a ∉ $\mathbb{Z}_{\leq 0}$"*
  shows *"(λn. (z/2) powr' (a + 2 * of_nat n) / (fact n * Gamma (1 + a + of_nat n))) sums*
            Bessel_I a z"

⟨proof⟩

lemma *has_field_derivative_Bessel_I_complex:*
  assumes *"a ∈ ℤ ∨ (z::complex) ∉ $\mathbb{R}_{\leq 0}$"*
  shows *"(Bessel_I a has_field_derivative*
  ((Bessel_I (a-1) z + Bessel_I (a+1) z) / 2)) (at z within A)"
⟨proof⟩

lemma *has_field_derivative_Bessel_I_complex' [derivative_intros]:*
  assumes *"(f has_field_derivative f') (at x within A)" "a ∈ ℤ ∨ (f x :: complex) ∉ $\mathbb{R}_{\leq 0}$"*
  shows *"((λx. Bessel_I a (f x)) has_field_derivative*
  *(f' * (Bessel_I (a-1) (f x) + Bessel_I (a+1) (f x)) / 2)) (at x within A)"*
  ⟨proof⟩

lemma *has_field_derivative_Bessel_I_real:*
  assumes *"a ∈ ℤ ∨ (z :: real) > 0"*
  shows *"(Bessel_I a has_field_derivative*
  ((Bessel_I (a-1) z + Bessel_I (a+1) z) / 2)) (at z within A)"
⟨proof⟩

lemma *has_field_derivative_Bessel_I_real' [derivative_intros]:*
  assumes *"(f has_field_derivative f') (at x within A)" "a ∈ ℤ ∨ f x > (0 :: real)"*
  shows *"((λx. Bessel_I a (f x)) has_field_derivative*
  *((Bessel_I (a-1) (f x) + Bessel_I (a+1) (f x)) / 2) * f') (at x within A)"*
  ⟨proof⟩

lemma *analytic_Bessel_I [analytic_intros]:*
  assumes *"a analytic_on A" "b analytic_on A" "⋀x. x ∈ A ⟹ b x ∉ $\mathbb{R}_{\leq 0}$"*
  shows *"(λx. Bessel_I (a x) (b x)) analytic_on A"*
  ⟨proof⟩

lemma *analytic_Bessel_I_Ints:*
  assumes *"b analytic_on A" "a ∈ ℤ"*
  shows *"(λx. Bessel_I a (b x)) analytic_on A"*
⟨proof⟩

theorem *Bessel_I_conv_SBessel_I_complex:*
  assumes *z: "z ≠ (0 :: complex)"*
  defines *"C ≡ sqrt (2 / pi) * csqrt z"*
  shows *"Bessel_I (of_int n + 1 / 2) z = C * SBessel_I n z"*
⟨proof⟩

```
lemma Bessel_I_conv_SBessel_I_real:
  assumes z: "z > (0 :: real)"
  shows   "Bessel_I (of_int n + 1 / 2) z = sqrt (2 * z / pi) * SBessel_I
n z"
⟨proof⟩

lemma Bessel_I_pos_real:
  assumes "(x :: real) > 0" "a > -1"
  shows   "Bessel_I a x > 0"
⟨proof⟩

lemma Bessel_I_strict_mono_real:
  assumes "0 ≤ x" "x < (y :: real)" "a > 0"
  shows   "Bessel_I a x < Bessel_I a y"
⟨proof⟩

lemma Bessel_I_mono_real:
  assumes "0 ≤ x" "x ≤ (y :: real)" "a > 0"
  shows   "Bessel_I a x ≤ Bessel_I a y"
  ⟨proof⟩


lemma convex_on_realI_strong:
  assumes "connected A"
    and "⋀x. x ∈ A ⟹ (f has_real_derivative f' x) (at x within A)"
    and "⋀x y. x ∈ A ⟹ y ∈ A ⟹ x ≤ y ⟹ f' x ≤ f' y"
  shows "convex_on A f"
⟨proof⟩


lemma Bessel_I_convex_real:
  assumes "a > 1"
  shows   "convex_on {0<..} (Bessel_I a)"
⟨proof⟩
```

The Bessel function of the second kind Y_a and its modified version K_a are easy to derive from J_a and I_a whenever $a \notin \mathbb{Z}$, but the case where $a \in \mathbb{Z}$ is more tedious to do, so we leave that as future work.

end

2 A Hankel-style integral formula for the Bessel I function

```
theory Bessel_Hankel_Integral
imports
  "Bessel.Bessel"
  "Path_Automation.Path_Automation"
  "Prime_Number_Theorem.Prime_Number_Theorem_Library"
```

"Incomplete_Gamma.More_Beta"
begin

2.1 Auxiliary integrals

Later on, we will need the integral

$$\int_{-\infty}^{\infty} (c^2 + t^2)^{-a}\,\mathrm{d}t = \frac{1}{2}c^{1-2a}B(\frac{1}{2}, a - \frac{1}{2})$$

for $c > 0$, $a > \frac{1}{2}$. We first show the $\int_0^\infty$ version. To this end, we perform the substitution $x = (t/c)^2$ to get

$$\frac{1}{2}c^{1-2e}\int_0^\infty x^{-\frac{1}{2}}(1+x)^{-e}\,\mathrm{d}x\ ,$$

which is one of the various equivalent integral definitions of the Beta function.

lemma *Bessel_I_aux_integral1:*
  fixes *c e :: real*
  assumes *c: "c > 0"* **and** *e: "e > 1/2"*
  shows *"(λt. (c² + t²) powr -e) absolutely_integrable_on {0<..}"*
    *"integral {0<..} (λt. (c² + t²) powr -e) = c powr (1 - 2 * e) / 2 * Beta (1/2) (e - 1/2)"*
⟨*proof*⟩

The $\int_{-\infty}^\infty$ version then easily follows by symmetry.

lemma *Bessel_I_aux_integral2:*
  fixes *c e :: real*
  assumes *c: "c > 0"* **and** *e: "e > 1/2"*
  shows *"integrable lebesgue (λt. (c² + t²) powr -e)"*
  and *"lebesgue_integral lebesgue (λt. (c² + t²) powr -e) = c powr (1-2*e) * Beta (1/2) (e - 1/2)"*
⟨*proof*⟩

2.2 A Hankel-style representation for $1/\Gamma(s)$

The biggest piece of work in this section will be to derive the following Hankel-style contour integral formula for Γ:

$$\frac{1}{\Gamma(s)} = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} e^z z^{-s}\,\mathrm{d}z$$

For $\mathrm{Re}(s) > 1$, the integral clearly converges absolutely as a Lebesgue integral. For $\mathrm{Re}(s) \in (0, 1]$ it has to be interpreted as a Cauchy principal value, i.e. $\int_{c-i\infty}^{c+i\infty} = \lim_{T\to\infty}\int_{c-iT}^{c+iT}$.

We first show the existence of the integral as a Lebesgue integral if $\mathrm{Re}(s) > 1$. This is fairly easy to see by a comparison test, since it is $O(|t|^{-\mathrm{Re}(s)})$ uniformly for all t.

```
lemma rGamma_hankel_integrable:
  assumes c: "c > 0" and s: "Re s > 1"
  shows   "((λz. exp z * z powr (-s)) ∘ (λt. Complex c t)) absolutely_integrable_on
UNIV"
⟨proof⟩
```

Next, we show the actual integral formula in the case that $\mathrm{Re}(s) > 0$. The proof works is modelled after Lemma 2.3 by Kong and Teo [2] and goes roughly like this:

- We consider the Hankel-style integration contour sketched in Figure 1.
- Since two of the horizontal lines lie directly on the branch cut, this integral is very awkward to handle. We instead only consider the part of the contour in the upper half plane, adding a horizontal line on the positive real axis to connect the part we cut off, and then also move the branch cut out of the way to the negative real axis by using the alternative branch of the logarithm $\widehat{\ln} z = \ln(-iz) + \frac{1}{2}i\pi$.
- We apply the Cauchy integral theorem to this contour and note that the integrand has no singularities inside, so its value must be 0.
- We do the same for the portion of the contour that is in the lower half of the complex plane and add the two values, noting that the horizontal lines we added cancel out. To save some work, we note that we can get this result for the lower half plane from that for the upper half plane (which we already have) by complex conjugation due to the fact that the integrand is symmetric under conjugation.
- Denote the vertical integral at $\mathrm{Re}(z) = c$ as $\mathrm{lhs}(T, s)$ and the sum of the other ones as $\mathrm{rhs}(r, T, s)$. The above shows that $\mathrm{lhs}(T, s) = \mathrm{rhs}(r, T, s)$ holds for all s with $\mathrm{Re}(s) > 0$.
- This immediately means that the value of $\mathrm{rhs}(r, T, s)$ is actually independent of r.
- Note also that $\lim_{T\to\infty} \mathrm{rhs}(r, T, s)$ exists for any s with $\mathrm{Re}(s) > 0$, and of course this limit is also independent of r. Write $\mathrm{RHS}(r, s)$ for this limit. In particular, this already shows that $\mathrm{lhs}(T, s)$ also has a limit for $T \to \infty$ and its value is $\mathrm{RHS}(r, s)$.
- In fact, $\mathrm{RHS}(r, s)$ can be expressed in terms of the incomplete Gamma function $\Gamma(1 - s, r)$ and some error terms corresponding to the small circle around the origin.

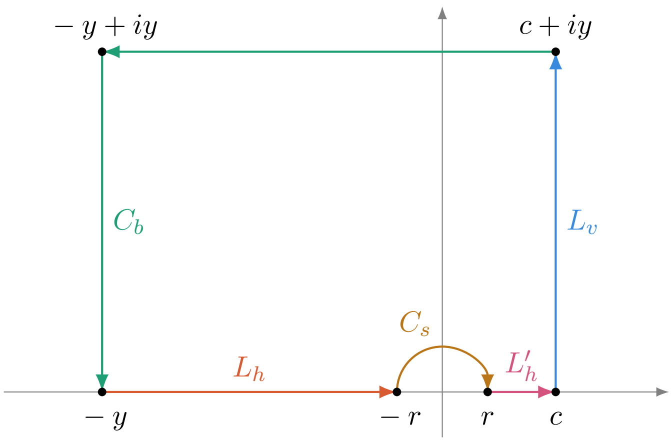

Figure 1: The Hankel-style integration contour used in the proof of the $\int_{c-i\infty}^{c+i\infty}$-style integral representation for $1/\Gamma(s)$.

- Thus, if $\mathrm{Re}(s) < 1$, we can let $r \to 0$ and find that the error terms in $\mathrm{RHS}(r, s)$ vanish and $\lim_{r\to 0} \Gamma(1-s, r) = \Gamma(1-s)$. Together with the reflection formula for Γ, we obtain $\mathrm{RHS}(s) = 2\pi/\Gamma(s)$, exactly as we wanted. This establishes the result we wanted in the strip $0 < \mathrm{Re}(s) < 1$.
- Moreover, both $\Gamma(1-s, r)$ and the error terms and other factors occurring in $\mathrm{RHS}(r, s)$ are holomorphic functions in s for $\mathrm{Re}(s) > 0$, so we can use analytic continuation to lift the above equality to all s with $\mathrm{Re}(s) > 0$ and we are done.

```
theorem rGamma_hankel_strong:
  assumes c: "c > 0" and s: "Re s > 0"
  shows   "((λT. integral {-T..T} ((λz. exp z * z powr (-s)) ∘ (λt. Complex
c t)))
              ⟶ (2 * of_real pi * rGamma s)) at_top"
⟨proof⟩
```

If $\mathrm{Re}(s) > 1$, the integral exists not only as an improper/Cauchy principal value style integral, but as a Lebesgue integral.

```
corollary rGamma_hankel:
  assumes c: "c > 0" and s: "Re s > 1"
  shows   "(((λz. exp z * z powr (-s)) ∘ (λt. Complex c t))
               has_integral (2 * of_real pi * rGamma s)) UNIV"
⟨proof⟩
```

2.3 The integral for the Bessel function

We now show the main result of this section: A Hankel-style contour-integral representation for the modified Bessel function of the first kind $I(\nu, s)$, valid for all s and ν with $\mathrm{Re}(\nu) > 0$.

The integral is of the form $\int_{c-i\infty}^{c+i\infty}$ for any real $c > 0$ and converges absolutely. The proof is modelled after Proposition 2.4 by Kong and Teo [2] and works by taking the similar integral formula for $1/\Gamma(s)$, summing over it, and exchanging the order of summation and integration.

The proof is fairly quick since the main work was already done in other lemmas. The main bit of work that we still have to do here is to justify exchanging summation and integration. This is done by showing that the integrals over the absolute value exist for every summand, and then the sum over these integrals also exists.

The integrals over the summand can be brought into the form

$$\int_{-\infty}^{\infty} (c^2 + t^2)^a \,\mathrm{d}t$$

for some suitable $a > \frac{1}{2}$, which we showed earlier to have a closed form in terms of the Beta function.

```
theorem
  fixes c :: real and s v :: complex
  assumes v: "Re v > 0" and c: "c > 0"
  shows   has_integral_Bessel_I_complex:
            "(((λz. 1 / (2*pi) * (s/2) powr v * exp (z + s²/(4*z)) / z
powr (v+1)) ∘ Complex c)
                 has_integral Bessel_I v s) UNIV"
    and   absolutely_integrable_Bessel_I_complex:
            "((λz. 1 / (2*pi) * (s/2) powr v * exp (z + s²/(4*z)) / z
powr (v+1)) ∘ Complex c)
               absolutely_integrable_on UNIV"
⟨proof⟩

lemma has_integral_Bessel_I_complex':
  fixes x :: complex and c :: real and v :: complex and f :: "complex
⇒ complex"
  defines "f ≡ (λt. exp (t + x² / t) / t powr v)"
  assumes c: "c > 0" and v: "Re v > 1" and x: "x ≠ 0"
  shows   "((f ∘ (λt. Complex c t)) has_integral
             (2 * pi * Bessel_I (v-1) (2*x)) / (x powr (v-1))) UNIV"
⟨proof⟩

lemma absolutely_integrable_Bessel_I_complex':
  fixes c :: real and s v :: complex
  assumes v: "Re v > 1" and c: "c > 0"
```

```
  shows "((λz. exp (z + s/z) / z powr v) ∘ Complex c) absolutely_integrable_on
UNIV"
```
⟨*proof*⟩

end

References

[1] NIST Digital Library of Mathematical Functions. https://dlmf.nist.gov/, Release 1.2.4 of 2025-03-15. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds.

[2] Z.-Y. Kong and L.-P. Teo. Rademacher's formula for the partition function, 2023.