

Generalised Hypergeometric Series

Manuel Eberl

July 14, 2026

Abstract

This entry provides a formalisation of the generalized hypergeometric series ${}_pF_q$, both as a formal power series and as a function in a Banach space. It is defined, for parameters $\mathbf{a} = (a_1, \dots, a_p)$ and $\mathbf{b} = (b_1, \dots, b_q)$ with $b_i \notin \mathbb{Z}_{\leq 0}$ as the exponential power series

$$F(\mathbf{a}; \mathbf{b}; z) = \sum_{n \geq 0} \frac{a_1^{\overline{n}} \cdots a_p^{\overline{n}}}{b_1^{\overline{n}} \cdots b_q^{\overline{n}}} \frac{z^n}{n!}$$

where $a^{\overline{n}} = a(a+1) \cdots (a+n-1)$ is the Pochhammer symbol.

Basic properties of ${}_pF_q$ are proven (uniform convergence, continuity, holomorphicity), as well as some important properties for specific instances, such as:

- representations of various trigonometric and hyperbolic functions in terms of ${}_0F_1$ and ${}_2F_1$
- the contiguous identities for ${}_2F_1$ and ${}_1F_1$
- the transformation identity ${}_1F_1(a; b; z) = e^z {}_1F_1(b-a; b; -z)$ for Kummer's confluent hypergeometric function
- the fact that ${}_1F_1$ is a solution of the ODE $aW(x) - (b-x)W'(x) - xW''(x) = 0$
- the fact that the error function can be expressed in terms of ${}_1F_1$ as $\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} z {}_1F_1(\frac{1}{2}; \frac{3}{2}; -z^2)$

A regularised variant of ${}_pF_q$ that is also defined if $b_i \in \mathbb{Z}_{\leq 0}$ is also provided.

Contents

1	Generalized Hypergeometric Series	2
1.1	Definition as a formal power series	2
1.2	The hypergeometric function	5
1.3	Representing trigonometric and hyperbolic functions	11
1.4	Regularised version	12
1.5	Gauß's contiguous relations for ${}_2F_1$	17
1.6	Bessel-type hypergeometric functions	20
1.7	Kummer's confluent hypergeometric function	21

theory Generalized Hypergeometric Series

```
"HOL-Complex_Analysis.Complex_Analysis"
"HOL-Real_Asymp.Real_Asymp"
"Error Function.Error Function"
```

[illegible]

Let $\mathbf{a} = (a_1, \dots, a_p)$ and $\mathbf{b} = (b_1, \dots, b_q)$. The typical notation for the hypergeometric function is

We will instead use the somewhat more compact $F(\mathbf{a}, \mathbf{b}, z)$. We will always let p and q implicitly denote the length of the vectors $\mathbf{a}$ and $\mathbf{b}$, respectively. Note that $\mathbf{b}$ must not contain non-positive integers to avoid division by zero. We first look at the hypergeometric series as an exponential generating function

where $m^{\overline{n}} = m(m+1)\cdots(m+n-1)$ is the Pochhammer symbol.

```
lemmas [simp del] = fps_hypergeo_nth
```

proof

$$\text{"fps_nth (fps_hypergeo as bs c) n} = (\prod a \leftarrow \text{as. pochhammer a n}) / (\prod b \leftarrow \text{bs. pochhammer b n}) * c ^ n / \text{fact n}"$$

⟨proof⟩

```
assumes "mset as = mset as'" "mset bs = mset bs'" "c = c'"
```

shows $"fps \text{ hypergeo } as \ bs \ c = fps \text{ hypergeo } as' \ bs' \ c'"$

2

```

lemma fps_hypergeo_singleton_Nil [simp]:
  "fps_hypergeo [a] [] c = fps_compose (fps_binomial (-a)) (-fps_const
c * fps_X)"
  <proof>

```

Parameters appearing in both of the lists can be cancelled.

```

lemma fps_hypergeo_cancel:
  assumes "a  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "fps_hypergeo (a # as) (a # bs) c = fps_hypergeo as bs c"
  <proof>

```

```

lemma fps_hypergeo_0_left [simp]:
  assumes "0  $\in$  set as"
  shows "fps_hypergeo as bs c = 1"
  <proof>

```

Next, we show a number of different expressions for the derivative of the hypergeometric series.

```

lemma fps_deriv_hypergeo1:
  fixes as bs :: "'a :: field_char_0 list"
  assumes "set bs  $\cap \mathbb{Z}_{\leq 0} = \{\}$ "
  defines "as'  $\equiv$  map ( $\lambda a. a + 1$ ) as"
  defines "bs'  $\equiv$  map ( $\lambda b. b + 1$ ) bs"
  shows "fps_deriv (fps_hypergeo as bs c) =
        fps_const (c * prod_list as) / fps_const (prod_list bs) * fps_hypergeo
as' bs' c"
  <proof>

```

```

lemma fps_deriv_hypergeo1':
  fixes as bs :: "'a :: field_char_0 list"
  assumes "0  $\notin$  set bs"
  defines "as'  $\equiv$  map ( $\lambda a. a + 1$ ) as"
  defines "bs'  $\equiv$  map ( $\lambda b. b + 1$ ) bs"
  shows "fps_const (prod_list bs) * fps_deriv (fps_hypergeo as bs c) =
        fps_const (c * prod_list as) * fps_hypergeo as' bs' c"
  <proof>

```

```

lemma fps_deriv_hypergeo2:
  fixes as bs :: "'a :: field_char_0 list" and a c :: 'a
  assumes "set bs  $\cap \mathbb{Z}_{\leq 0} = \{\}$ "
  defines "F  $\equiv$  fps_hypergeo (a # as) bs c"
  defines "G  $\equiv$  fps_hypergeo ((a + 1) # as) bs c"
  shows "fps_X * fps_deriv F + fps_const a * F = fps_const a * G"
  <proof>

```

```

lemma fps_deriv_hypergeo3:
  fixes as bs :: "'a :: field_char_0 list" and b c :: 'a
  assumes "set bs  $\cap \mathbb{Z}_{\leq 0} = \{\}$ " "b  $\notin \mathbb{Z}_{\leq 0}$ " "b  $\neq 1$ "
  defines "F  $\equiv$  fps_hypergeo as (b # bs) c"

```

```

defines "G  $\equiv$  fps_hypergeo as ((b - 1) # bs) c"
shows "fps_X * fps_deriv F + fps_const (b - 1) * F = fps_const (b - 1) * G"
<proof>

```

The radius of convergence of a hypergeometric series ${}_pF_q(\mathbf{a}; \mathbf{b}; z)$ is easy to determine: if $p \leq q$, it is ∞ . If $p = q + 1$, it is 1. If $p > q$, it is 0.

Note that the formulation here is slightly more general since, again, it talks about ${}_pF_q(\mathbf{a}; \mathbf{b}; cz)$.

```

lemma fps_conv_radius_hypergeo:
  fixes as bs :: "'a :: {banach, real_normed_field} list"
  shows "fps_conv_radius (fps_hypergeo as bs c) =
    (if length as  $\leq$  length bs  $\vee$  c = 0  $\vee$  set (as@bs)  $\cap \mathbb{Z}_{\leq 0} \neq \{\}$  then  $\infty$ 
     else if length as = length bs + 1 then 1 / ereal (norm c)
     else 0)" (is "_ = ?r")
<proof>

```

We also define the following notion, which corresponds to the ordinary generating function

$$\sum_{n \geq 0} \frac{a_1^{\overline{n}} \cdots a_p^{\overline{n}}}{b_1^{\overline{n}} \cdots b_q^{\overline{n}}} z^n$$

This is a bit easier to deal with for our convergence arguments later on.

One can express the “normal” hypergeometric series in terms of this easily by adding $a_{p+1} = 1$ to the first parameter vector.

```

definition fps_hypergeo_aux :: "'a :: field_char_0 list  $\Rightarrow$  'a list  $\Rightarrow$  'a fps" where
  "fps_hypergeo_aux as bs = Abs_fps ( $\lambda$ n. ( $\prod a \leftarrow$  as. pochhammer a n) / ( $\prod b \leftarrow$  bs. pochhammer b n))"

```

```

definition hypergeo_F_aux where
  "hypergeo_F_aux as bs = eval_fps (fps_hypergeo_aux as bs)"

```

```

lemma fps_nth_hypergeo_aux:
  "fps_nth (fps_hypergeo_aux as bs) n = ( $\prod a \leftarrow$  as. pochhammer a n) / ( $\prod b \leftarrow$  bs. pochhammer b n)"
<proof>

```

```

lemma fps_hypergeo_conv_fps_hypergeo_aux:
  "fps_hypergeo as bs c = fps_compose (fps_hypergeo_aux as (1 # bs)) (fps_const c * fps_X)"
<proof>

```

```

lemma fps_hypergeo_aux_conv_fps_hypergeo:
  "fps_hypergeo_aux as bs = fps_hypergeo (1 # as) bs 1"
<proof>

```

```

lemma fps_hypergeo_aux_split_head:
  "fps_hypergeo_aux as bs =
    1 + fps_const (prod_list as / prod_list bs) * fps_X *
      fps_hypergeo_aux (map (λa. a+1) as) (map (λb. b+1) bs)" (is
    "?lhs = ?rhs ")
  <proof>

lemma fps_conv_radius_hypergeo_aux:
  fixes as bs :: "'a :: {banach, real_normed_field} list"
  shows "fps_conv_radius (fps_hypergeo_aux as bs) =
    (if length as < length bs ∨ set (as@bs) ∩ ℤ≤0 ≠ {} then
    ∞
      else if length as = length bs then 1
      else 0)" (is "_ = ?r")
  <proof>

lemma hypergeo_F_aux_split_head:
  fixes as bs :: "'a :: {banach, real_normed_field, field_char_0} list"
  assumes "length as < length bs ∨ length as = length bs ∧ norm x <
    1"
  shows "hypergeo_F_aux as bs x =
    1 + prod_list as / prod_list bs * x * hypergeo_F_aux (map
    (λa. a+1) as) (map (λb. b+1) bs) x"
  <proof>

```

1.2 The hypergeometric function

We will now look at the case where the formal series converges to a function. The type constraints we consider here is a field extension of the reals with a complete norm defined on it. In some important lemmas (e.g. continuity), this will be further restricted to Heine–Borel spaces, which effectively means that we will only look at *finite-dimensional* real fields.

It is well-known that the only two such fields, up to isomorphism, are $\mathbb{R}$ and $\mathbb{C}$. These are the fields we care about anyway.

```

definition hypergeo_F :: "'a :: {banach, real_normed_field} list ⇒ 'a
list ⇒ 'a ⇒ 'a" where
  "hypergeo_F as bs = eval_fps (fps_hypergeo as bs 1)"

```

```

lemma hypergeo_F_conv_hypergeo_F_aux:
  "hypergeo_F as bs = hypergeo_F_aux as (1 # bs)"
  <proof>

```

```

lemma hypergeo_F_0 [simp]: "hypergeo_F as bs 0 = 1"
  <proof>

```

```

lemma hypergeo_F_Nil_Nil [simp]: "hypergeo_F [] [] = exp"
  <proof>

```

```

lemma hypergeo_F_singleton_Nil_real [simp]:
  assumes "|z| < (1::real)"
  shows "hypergeo_F [a] [] z = (1 - z) powr (-a)"
<proof>

```

```

lemma hypergeo_F_singleton_Niln_complex [simp]:
  assumes "norm (z :: complex) < 1"
  shows "hypergeo_F [a] [] z = (1 - z) powr (-a)"
<proof>

```

```

lemma hypergeo_F_cancel:
  assumes "a ∉ ℤ≤0"
  shows "hypergeo_F (a # as) (a # bs) = hypergeo_F as bs"
<proof>

```

The convergence of ${}_pF_q(\mathbf{a}; \mathbf{b}; z)$ is uniform not only in z , but also in $\mathbf{a}$ and $\mathbf{b}$.

```

lemma uniform_limit_hypergeo_F_aux:
  fixes as bs :: "('a :: {topological_space} ⇒ 'b :: {banach, real_normed_field,
field_char_0}) list"
  assumes "compact X" and "continuous_on X g" and "list_all (continuous_on
X) (as @ bs)"
  assumes norm: "length as < length bs ∨ length as = length bs ∧ (∀ x ∈ X.
norm (g x) < 1)"
  assumes "list_all (λ b. ∀ x ∈ X. b x ∉ ℤ≤0) bs"
  assumes "∧ x. A x = map (λ a. a x) as" and "∧ x. B x = map (λ b. b x)
bs"
  shows "uniform_limit X (λ N x. ∑ n < N. fps_nth (fps_hypergeo_aux (A x)
(B x)) n * (g x) ^ n)
(λ x. hypergeo_F_aux (A x) (B x) (g x)) sequentially"
<proof>

```

```

lemma sums_hypergeo_F_aux:
  fixes as bs :: "'a :: {banach, real_normed_field, field_char_0} list"
  assumes "length as < length bs ∨ length as = length bs ∧ norm z <
1"
  shows "(λ n. fps_nth (fps_hypergeo_aux as bs) n * z ^ n) sums hypergeo_F_aux
as bs z"
<proof>

```

```

lemma of_real_hypergeo_F_aux:
  assumes "length as < length bs ∨ length as = length bs ∧ norm z <
1"
  shows "(of_real (hypergeo_F_aux as bs z) :: 'a :: {banach, real_normed_field,
field_char_0}) =
hypergeo_F_aux (map of_real as) (map of_real bs) (of_real z)"
<proof>

```

```

lemma of_real_hypergeo_F:

```

```

    assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ norm
z < 1"
    shows "(of_real (hypergeo_F as bs z) :: 'a :: {banach, real_normed_field,
field_char_0}) =
    hypergeo_F (map of_real as) (map of_real bs) (of_real z)"
  <proof>

```

Due to the uniform convergence, ${}_pF_q$ is analytic and continuous in all its parameters.

```

lemma analytic_hypergeo_F_aux:
  assumes "f analytic_on X" and "list_all (λa. a analytic_on X) (as @
bs)"
  assumes norm: "length as < length bs ∨ length as = length bs ∧ (∀x∈X.
norm (f x) < 1)"
  assumes "list_all (λb. ∀x∈X. b x ∉ ℤ≤0) bs"
  assumes "∧x. A x = map (λa. a x) as" and "∧x. B x = map (λb. b x)
bs"
  shows "(λx. hypergeo_F_aux (A x) (B x) (f x)) analytic_on X"
  <proof>

```

```

lemma continuous_on_hypergeo_F_aux:
  fixes X :: "'a :: heine_borel set"
  fixes f :: "'a ⇒ 'b :: {banach, real_normed_field}"
  assumes "continuous_on X f" and "list_all (λa. continuous_on X a) (as
@ bs)"
  assumes "length as < length bs ∨ length as = length bs ∧ (∀x∈X. norm
(f x) < 1)"
  assumes "list_all (λb. ∀x∈X. b x ∉ ℤ≤0) bs"
  assumes "∧x. A x = map (λa. a x) as" and "∧x. B x = map (λb. b x)
bs"
  shows "continuous_on X (λx. hypergeo_F_aux (A x) (B x) (f x))"
  <proof>

```

We only evaluate the derivative in the variable x , not in the parameters, since there is no closed-form expression for that in general.

```

theorem has_field_derivative_hypergeo_F:
  fixes as bs
  defines "as' ≡ map (λn. n+1) as" and "bs' ≡ map (λn. n+1) bs"
  defines "C ≡ prod_list as / prod_list bs"
  assumes norm: "length as ≤ length bs ∨ length as = Suc (length bs)
∧ norm x < 1"
  assumes bs: "set bs ∩ ℤ≤0 = {}"
  shows "(hypergeo_F as bs has_field_derivative
(C * hypergeo_F as' bs' x)) (at x within A)"
  <proof>

```

```

lemma has_field_derivative_hypergeo_F' [derivative_intros]:
  fixes as bs
  defines "as' ≡ map (λn. n+1) as" and "bs' ≡ map (λn. n+1) bs"

```

```

defines "C ≡ prod_list as / prod_list bs"
assumes f: "(f has_field_derivative f') (at x within A)"
assumes "length as ≤ length bs ∨ length as = Suc (length bs) ∧ norm
(f x) < 1"
assumes "set bs ∩ ℤ≤0 = {}"
shows "((λx. hypergeo_F as bs (f x)) has_field_derivative
(C * hypergeo_F as' bs' (f x) * f')) (at x within A)"
⟨proof⟩

```

corollary deriv_hypergeo_F:

```

fixes as bs
defines "as' ≡ map (λn. n+1) as" and "bs' ≡ map (λn. n+1) bs"
defines "C ≡ prod_list as / prod_list bs"
assumes "length as ≤ length bs ∨ length as = Suc (length bs) ∧ norm
x < 1"
assumes "set bs ∩ ℤ≤0 = {}"
shows "deriv (hypergeo_F as bs) x = C * hypergeo_F as' bs' x"
⟨proof⟩

```

corollary higher_deriv_hypergeo_F:

```

fixes as bs m
defines "as' ≡ (λm. map (λn. n + of_nat m) as)" and "bs' ≡ (λm. map
(λn. n + of_nat m) bs)"
defines "C ≡ (λm. (∏ a←as. pochhammer a m) / (∏ b←bs. pochhammer
b m))"
assumes "length as ≤ length bs ∨ length as = Suc (length bs) ∧ norm
x < 1"
assumes bs: "set bs ∩ ℤ≤0 = {}"
shows "(deriv ^ m) (hypergeo_F as bs) x = C m * hypergeo_F (as' m)
(bs' m) x"
⟨proof⟩

```

theorem uniform_limit_hypergeo_F:

```

fixes as bs :: "('a :: {topological_space} ⇒ 'b :: {banach, real_normed_field,
field_char_0}) list"
assumes "compact X" and "continuous_on X g" and "list_all (continuous_on
X) (as @ bs)"
assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ (∀ x∈X.
norm (g x) < 1)"
assumes "list_all (λb. ∀ x∈X. b x ∉ ℤ≤0) bs"
assumes "∧ x. A x = map (λa. a x) as" and "∧ x. B x = map (λb. b x)
bs"
shows "uniform_limit X (λN x. ∑ n<N. fps_nth (fps_hypergeo (A x) (B
x) 1) n * (g x) ^ n)
(λx. hypergeo_F (A x) (B x) (g x)) sequentially"
⟨proof⟩

```

corollary sums_hypergeo_F:

```

fixes as bs :: "'a :: {banach, real_normed_field, field_char_0} list"
assumes "list_all (λb. b ∉ ℤ≤₀) bs" and "length as ≤ length bs ∨
length as = length bs + 1 ∧ norm z < 1"
shows "(λn. fps_nth (fps_hypergeo as bs 1) n * z ^ n) sums hypergeo_F
as bs z"
⟨proof⟩

```

```

corollary sums_hypergeo_F':
fixes as bs :: "'a :: {banach, real_normed_field, field_char_0} list"
assumes "list_all (λb. b ∉ ℤ≤₀) bs" and "length as ≤ length bs ∨
length as = length bs + 1 ∧ norm (c * z) < 1"
shows "(λn. fps_nth (fps_hypergeo as bs c) n * z ^ n) sums hypergeo_F
as bs (c * z)"
⟨proof⟩

```

The function is analytic in its variable and all of its parameters simultaneously (unsurprisingly).

```

corollary analytic_hypergeo_F:
assumes "f analytic_on X" and "list_all (λa. a analytic_on X) (as @
bs)"
assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ (∀x∈X.
norm (f x) < 1)"
assumes "list_all (λb. ∀x∈X. b x ∉ ℤ≤₀) bs"
assumes "∧x. A x = map (λa. a x) as" and "∧x. B x = map (λb. b x)
bs"
shows "(λx. hypergeo_F (A x) (B x) (f x)) analytic_on X"
⟨proof⟩

```

```

corollary analytic_hypergeo_F_simple [analytic_intros]:
assumes "f analytic_on X"
assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ (∀x∈X.
norm (f x) < 1)"
assumes "set bs ∩ ℤ≤₀ = {}"
shows "(λx. hypergeo_F as bs (f x)) analytic_on X"
⟨proof⟩

```

```

corollary holomorphic_hypergeo_F_simple [holomorphic_intros]:
assumes "f holomorphic_on X"
assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ (∀x∈X.
norm (f x) < 1)"
assumes "set bs ∩ ℤ≤₀ = {}"
shows "(λx. hypergeo_F as bs (f x)) holomorphic_on X"
⟨proof⟩

```

The continuity theorem looks a bit awkward. The natural way to express it would be as:

```

continuous_on {(as,bs,z). set bs ∩ ℤ≤₀ = {} ∧ (length as ≤ length bs
∨ length as = length bs + 1 ∧ norm x < 1)} (λ(as,bs,x). hypergeo_F as
bs x)

```

However, this is not possible since there is no topological space defined on the list type, and creating one just for this would be a lot of work.

In practice, one will typically consider lists of fixed length and derive a corresponding version of the above sketched lemma using tuples instead of lists (see example for Kummer below).

```
corollary continuous_on_hypergeo_F:
  fixes X :: "'a :: heine_borel set"
  fixes f :: "'a  $\Rightarrow$  'b :: {banach, real_normed_field}"
  assumes "continuous_on X f" and "list_all ( $\lambda$ a. continuous_on X a) (as
@ bs)"
  assumes "length as  $\leq$  length bs  $\vee$  length as = length bs + 1  $\wedge$  ( $\forall x \in X$ .
norm (f x) < 1)"
  assumes "list_all ( $\lambda$ b.  $\forall x \in X$ . b x  $\notin \mathbb{Z}_{\leq 0}$ ) bs"
  assumes " $\bigwedge x$ . A x = map ( $\lambda$ a. a x) as" and " $\bigwedge x$ . B x = map ( $\lambda$ b. b x)
bs"
  shows "continuous_on X ( $\lambda$ x. hypergeo_F (A x) (B x) (f x))"
<proof>
```

```
corollary continuous_on_hypergeo_F_simple [continuous_intros]:
  assumes "continuous_on X f"
  assumes "length as  $\leq$  length bs  $\vee$  length as = length bs + 1  $\wedge$  ( $\forall x \in X$ .
norm (f x) < 1)"
  assumes "set bs  $\cap \mathbb{Z}_{\leq 0} = \{\}$ "
  shows "continuous_on X ( $\lambda$ x. hypergeo_F as bs (f x))"
<proof>
```

```
corollary tendsto_hypergeo_F_simple [tendsto_intros]:
  assumes "(f  $\longrightarrow$  z) F"
  assumes "length as  $\leq$  length bs  $\vee$  length as = length bs + 1  $\wedge$  norm
z < 1"
  assumes "set bs  $\cap \mathbb{Z}_{\leq 0} = \{\}$ "
  shows "(( $\lambda$ x. hypergeo_F as bs (f x))  $\longrightarrow$  hypergeo_F as bs z) F"
<proof>
```

```
corollary continuous_hypergeo_F_simple [continuous_intros]:
  assumes "continuous (at z within A) f"
  assumes "length as  $\leq$  length bs  $\vee$  length as = length bs + 1  $\wedge$  norm
(f z) < 1"
  assumes "set bs  $\cap \mathbb{Z}_{\leq 0} = \{\}$ "
  shows "continuous (at z within A) ( $\lambda$ x. hypergeo_F as bs (f x))"
<proof>
```

As an example, the proof that Kummer's hypergeometric function is continuous in all variables.

```
lemma continuous_on_hypergeo_F_kummer:
  fixes A :: "('a :: {real_normed_field, banach, heine_borel})  $\times$  'a  $\times$  'a)
set"
  assumes "A  $\subseteq UNIV \times (-\mathbb{Z}_{\leq 0}) \times UNIV"$ 
```

```

shows "continuous_on A ( $\lambda(a,b,z). \text{hypergeo\_F } [a] [b] z$ )"
<proof>

lemma continuous_on_hypergeo_F_kummer':
  fixes f :: "'a :: topological_space  $\Rightarrow$  'b :: {real_normed_field, banach,
heine_borel}"
  assumes "continuous_on A f" "continuous_on A g" "continuous_on A h"
  assumes " $\bigwedge x. x \in A \implies g\ x \notin \mathbb{Z}_{\leq 0}$ "
  shows "continuous_on A ( $\lambda x. \text{hypergeo\_F } [f\ x] [g\ x] (h\ x)$ )"
<proof>

lemma has_fps_expansion_hypergeo_F [fps_expansion_intros]:
  assumes "length as  $\leq$  Suc (length bs)" "list_all ( $\lambda b. b \notin \mathbb{Z}_{\leq 0}$ ) bs"
  shows " $(\lambda x. \text{hypergeo\_F } as\ bs\ (c * x))$  has_fps_expansion fps_hypergeo
as bs c"
<proof>

lemma has_fps_expansion_hypergeo_F' [fps_expansion_intros]:
  assumes "length as  $\leq$  Suc (length bs)" "list_all ( $\lambda b. b \notin \mathbb{Z}_{\leq 0}$ ) bs"
  shows " $(\lambda x. \text{hypergeo\_F } as\ bs\ x)$  has_fps_expansion fps_hypergeo as
bs 1"
<proof>

```

1.3 Representing trigonometric and hyperbolic functions

The functions $\sin$, $\cos$, $\sinh$, $\cosh$ have simple representations in terms of ${}_0F_1$.

```

lemma pochhammer_three_halves:
  "pochhammer (3 / 2 :: 'a :: field_char_0) n =
    (2 * of_nat n + 1) * fact (2 * n) / (2 ^ (2 * n) * fact n)"
<proof>

```

```

lemma cos_conv_hypergeo_F:
  fixes z :: "'a :: {real_normed_field, field_char_0, banach}"
  shows "cos z = hypergeo_F [] [1/2] (-z2/4)"
<proof>

```

```

lemma sin_conv_hypergeo_F:
  fixes z :: "'a :: {real_normed_field, field_char_0, banach}"
  shows "sin z = z * hypergeo_F [] [3/2] (-z2/4)"
<proof>

```

```

lemma cosh_conv_hypergeo_F:
  fixes z :: "'a :: {real_normed_field, field_char_0, banach}"
  shows "cosh z = hypergeo_F [] [1/2] (z2/4)"
<proof>

```

```

lemma sinh_conv_hypergeo_F:

```

```

    fixes z :: "'a :: {real_normed_field, field_char_0, banach}"
    shows "sinh z = z * hypergeo_F [] [3/2] (z2/4)"
  <proof>

```

Similarly, arctan and artanh have representations in terms of ${}_2F_1$.

```

lemma arctan_conv_hypergeo_F_real:
  assumes "|z| < 1"
  shows "arctan z = z * hypergeo_F [1, 1/2] [3/2] (-z2)"
  <proof>

```

```

lemma arctan_conv_hypergeo_F_complex:
  assumes "norm z < 1"
  shows "Arctan z = z * hypergeo_F [1, 1/2] [3/2] (-z2)"
  <proof>

```

```

lemma artanh_conv_hypergeo_F_complex:
  assumes "norm (z :: complex) < 1"
  shows "artanh z = z * hypergeo_F [1, 1/2] [3/2] (z2)"
  <proof>

```

```

lemma artanh_conv_hypergeo_F_real:
  assumes "|z::real| < 1"
  shows "artanh z = z * hypergeo_F [1, 1/2] [3/2] (z2)"
  <proof>

```

1.4 Regularised version

The “normal” hypergeometric function is undefined if any of the parameters b_i are non-positive integers. This can be fixed by considering the following *regularised* version, which is well-defined for any combination of parameters $(a_i)_{1 \leq i \leq m}$ and $(b_i)_{1 \leq i \leq n}$:

```

definition reg_fps_hypergeo_aux :: "'a :: Gamma list ⇒ 'a list ⇒ 'a fps"
where
  "reg_fps_hypergeo_aux as bs =
    Abs_fps (λn. (∏ a←as. pochhammer a n) * (∏ b←bs. rGamma (b + of_nat
n)))"

```

```

definition reg_hypergeo_F :: "'a :: Gamma list ⇒ 'a list ⇒ 'a ⇒ 'a"
where
  "reg_hypergeo_F as bs = eval_fps (reg_fps_hypergeo_aux as (1 # bs))"

```

```

lemma reg_hypergeo_F_0: "reg_hypergeo_F as bs 0 = (∏ b←bs. rGamma b)"
  <proof>

```

```

lemma reg_fps_hypergeo_aux_conv_fps_hypergeo_aux:
  assumes "set bs ∩ ℤ≤0 = {}"
  shows "reg_fps_hypergeo_aux as bs = fps_const (∏ b←bs. rGamma b)
  * fps_hypergeo_aux as bs"

```

<proof>

lemma *reg_hypergeo_F_conv_hypergeo_F*:

assumes "set bs $\cap \mathbb{Z}_{\leq 0} = \{\}$ " "length as $\leq$ length bs $\vee$ length as = length bs + 1 $\wedge$ norm z < 1"

shows "reg_hypergeo_F as bs z = ($\prod b \leftarrow bs. rGamma\ b$) * hypergeo_F as bs z"

<proof>

lemma *fps_deriv_reg_hypergeo_aux1*:

fixes as bs :: "'a :: Gamma list"

defines "as' $\equiv$ map ($\lambda a. a + 1$) as"

defines "bs' $\equiv$ map ($\lambda b. b + 1$) bs"

shows "fps_deriv (reg_fps_hypergeo_aux as (1#bs)) =

fps_const (prod_list as) * reg_fps_hypergeo_aux as' (1#bs)'"

<proof>

lemma *fps_conv_radius_reg_hypergeo_aux*:

fixes as bs :: "'a :: Gamma list"

shows "fps_conv_radius (reg_fps_hypergeo_aux as bs) =

(if length as < length bs $\vee$ set as $\cap \mathbb{Z}_{\leq 0} \neq \{\}$ then ∞

else if length as = length bs then 1

else 0)" (is "_ = ?r")

<proof>

lemma *uniform_limit_reg_hypergeo_F_aux*:

fixes as bs :: "('a :: {topological_space} $\Rightarrow$ 'b :: Gamma) list"

assumes "compact X" and "continuous_on X g" and "list_all (continuous_on X) (as @ bs)"

assumes norm: "length as < length bs $\vee$ length as = length bs $\wedge$ ($\forall x \in X. \text{norm } (g\ x) < 1$)"

assumes " $\bigwedge x. x \in X \implies A\ x = \text{map } (\lambda a. a\ x)\ as$ " and " $\bigwedge x. x \in X \implies B\ x = \text{map } (\lambda b. b\ x)\ bs$ "

shows "uniform_limit X ($\lambda N\ x. \sum n < N. \text{fps_nth } (\text{reg_fps_hypergeo_aux } (A\ x) (B\ x))\ n * (g\ x) ^ n$)

($\lambda x. \text{eval_fps } (\text{reg_fps_hypergeo_aux } (A\ x) (B\ x))\ (g\ x)$) sequentially"

<proof>

lemma *sums_reg_hypergeo_F_aux*:

fixes as bs :: "'a :: Gamma list"

assumes "length as < length bs $\vee$ length as = length bs $\wedge$ norm z < 1"

shows "($\lambda n. \text{fps_nth } (\text{reg_fps_hypergeo_aux } as\ bs)\ n * z ^ n$) sums

eval_fps (reg_fps_hypergeo_aux as bs) z"

<proof>

lemma *complex_of_real_reg_hypergeo_F_aux*:

assumes "length as < length bs $\vee$ length as = length bs $\wedge$ norm z <

```

1"
  shows "(of_real (eval_fps (reg_fps_hypergeo_aux as bs) z) :: complex)
=
      eval_fps (reg_fps_hypergeo_aux (map of_real as) (map of_real
bs)) (of_real z)"
⟨proof⟩

lemma complex_of_real_reg_hypergeo_F:
  assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ norm
z < 1"
  shows "(complex_of_real (reg_hypergeo_F as bs z) :: complex) =
      reg_hypergeo_F (map of_real as) (map of_real bs) (of_real z)"
  ⟨proof⟩

lemma analytic_reg_hypergeo_F_aux:
  assumes "f analytic_on X" and "list_all (λa. a analytic_on X) (as @
bs)"
  assumes norm: "length as < length bs ∨ length as = length bs ∧ (∀x∈X.
norm (f x) < 1)"
  assumes "λx. A x = map (λa. a x) as" and "λx. B x = map (λb. b x)
bs"
  shows "(λx. eval_fps (reg_fps_hypergeo_aux (A x) (B x)) (f x)) analytic_on
X"
  ⟨proof⟩

lemma analytic_reg_hypergeo_F:
  assumes "f analytic_on X" and "list_all (λa. a analytic_on X) (as @
bs)"
  assumes norm: "length as ≤ length bs ∨ length as = length bs + 1 ∧
(∀x∈X. norm (f x) < 1)"
  assumes "λx. A x = map (λa. a x) as" and "λx. B x = map (λb. b x)
bs"
  shows "(λx. reg_hypergeo_F (A x) (B x) (f x)) analytic_on X"
  ⟨proof⟩

corollary analytic_reg_hypergeo_F_simple [analytic_intros]:
  assumes "f analytic_on X"
  assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ (∀x∈X.
norm (f x) < 1)"
  shows "(λx. reg_hypergeo_F as bs (f x)) analytic_on X"
  ⟨proof⟩

corollary holomorphic_reg_hypergeo_F_simple [holomorphic_intros]:
  assumes "f holomorphic_on X"
  assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ (∀x∈X.
norm (f x) < 1)"
  shows "(λx. reg_hypergeo_F as bs (f x)) holomorphic_on X"
  ⟨proof⟩

```

```

lemma continuous_on_reg_hypergeo_F_aux:
  fixes X :: "'a :: heine_borel set"
  fixes f :: "'a  $\Rightarrow$  'b :: Gamma"
  assumes "continuous_on X f" and "list_all ( $\lambda$ a. continuous_on X a) (as
@ bs)"
  assumes "length as < length bs  $\vee$  length as = length bs  $\wedge$  ( $\forall x \in X$ . norm
(f x) < 1)"
  assumes AB: " $\bigwedge x. x \in X \implies A\ x = \text{map } (\lambda a. a\ x)\ as$ " " $\bigwedge x. x \in X \implies
B\ x = \text{map } (\lambda b. b\ x)\ bs$ "
  shows "continuous_on X ( $\lambda x$ . eval_fps (reg_fps_hypergeo_aux (A x)
(B x)) (f x))"
  <proof>

```

```

corollary continuous_on_reg_hypergeo_F:
  fixes X :: "'a :: heine_borel set"
  fixes f :: "'a  $\Rightarrow$  'b :: Gamma"
  assumes "continuous_on X f" and "list_all ( $\lambda$ a. continuous_on X a) (as
@ bs)"
  assumes "length as  $\leq$  length bs  $\vee$  length as = length bs + 1  $\wedge$  ( $\forall x \in X$ .
norm (f x) < 1)"
  assumes " $\bigwedge x. A\ x = \text{map } (\lambda a. a\ x)\ as$ " and " $\bigwedge x. B\ x = \text{map } (\lambda b. b\ x)\
bs$ "
  shows "continuous_on X ( $\lambda x$ . reg_hypergeo_F (A x) (B x) (f x))"
  <proof>

```

```

theorem has_field_derivative_reg_hypergeo_F:
  fixes as bs :: "'a :: Gamma list"
  defines "as'  $\equiv$  map ( $\lambda n. n+1$ ) as" and "bs'  $\equiv$  map ( $\lambda n. n+1$ ) bs"
  assumes norm: "length as  $\leq$  length bs  $\vee$  length as = Suc (length bs)
 $\wedge$  norm x < 1"
  shows "(reg_hypergeo_F as bs has_field_derivative
(prod_list as * reg_hypergeo_F as' bs' x)) (at x within
A)"
  <proof>

```

```

lemma has_field_derivative_reg_hypergeo_F' [derivative_intros]:
  fixes as bs :: "'a :: Gamma list"
  defines "as'  $\equiv$  map ( $\lambda n. n+1$ ) as" and "bs'  $\equiv$  map ( $\lambda n. n+1$ ) bs"
  assumes f: "(f has_field_derivative f') (at x within A)"
  assumes "length as  $\leq$  length bs  $\vee$  length as = Suc (length bs)  $\wedge$  norm
(f x) < 1"
  shows "(( $\lambda x$ . reg_hypergeo_F as bs (f x)) has_field_derivative
(prod_list as * reg_hypergeo_F as' bs' (f x) * f')) (at
x within A)"
  <proof>

```

```

corollary continuous_on_reg_hypergeo_F_simple [continuous_intros]:
  assumes "continuous_on X f"

```

```

    assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ (∀ x ∈ X.
norm (f x) < 1)"
    shows "continuous_on X (λ x. reg_hypergeo_F as bs (f x))"
⟨proof⟩

```

```

corollary tendsto_reg_hypergeo_F_simple [tendsto_intros]:
  assumes "(f ⟶ z) F"
  assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ norm
z < 1"
  shows "((λ x. reg_hypergeo_F as bs (f x)) ⟶ reg_hypergeo_F as bs
z) F"
⟨proof⟩

```

```

corollary continuous_reg_hypergeo_F_simple [continuous_intros]:
  assumes "continuous (at z within A) f"
  assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ norm
(f z) < 1"
  shows "continuous (at z within A) (λ x. reg_hypergeo_F as bs (f x))"
⟨proof⟩

```

```

corollary deriv_reg_hypergeo_F:
  fixes as bs :: "'a :: Gamma list"
  defines "as' ≡ map (λ n. n+1) as" and "bs' ≡ map (λ n. n+1) bs"
  assumes "length as ≤ length bs ∨ length as = Suc (length bs) ∧ norm
x < 1"
  shows "deriv (reg_hypergeo_F as bs) x = prod_list as * reg_hypergeo_F
as' bs' x"
⟨proof⟩

```

```

corollary higher_deriv_reg_hypergeo_F:
  fixes as bs :: "'a :: Gamma list"
  defines "as' ≡ (λ m. map (λ n. n + of_nat m) as)" and "bs' ≡ (λ m. map
(λ n. n + of_nat m) bs)"
  defines "C ≡ (λ m. (∏ a ← as. pochhammer a m))"
  assumes "length as ≤ length bs ∨ length as = Suc (length bs) ∧ norm
x < 1"
  shows "(deriv ^^ m) (reg_hypergeo_F as bs) x = C m * reg_hypergeo_F
(as' m) (bs' m) x"
⟨proof⟩

```

```

theorem uniform_limit_reg_hypergeo_F:
  fixes as bs :: "('a :: {topological_space} ⇒ 'b :: Gamma) list"
  assumes "compact X" and "continuous_on X g" and "list_all (continuous_on
X) (as @ bs)"
  assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ (∀ x ∈ X.
norm (g x) < 1)"
  assumes "∧ x. A x = map (λ a. a x) as" and "∧ x. B x = map (λ b. b x)
bs"
  shows "uniform_limit X

```

```

      (λN x. ∑ n<N. (∏ a←as. pochhammer (a x) n) * (∏ b←bs. rGamma
(b x + of_nat n)) *
      (g x) ^ n / fact n)
      (λx. reg_hypergeo_F (A x) (B x) (g x)) sequentially"
⟨proof⟩

```

```

corollary sums_reg_hypergeo_F:
  fixes as bs :: "'a :: Gamma list"
  assumes "length as ≤ length bs ∨ length as = length bs + 1 ∧ norm
z < 1"
  shows "(λn. (∏ a←as. pochhammer a n) * (∏ b←bs. rGamma (b + of_nat
n)) * z ^ n / fact n)
        sums reg_hypergeo_F as bs z"
⟨proof⟩

```

```

lemma has_fps_expansion_reg_hypergeo_F [fps_expansion_intros]:
  assumes "length as ≤ Suc (length bs)" "list_all (λb. b ∉ ℤ≤0) bs"
  shows "(λx. reg_hypergeo_F as bs x) has_fps_expansion reg_fps_hypergeo_aux
as (1 # bs)"
⟨proof⟩

```

1.5 Gauß's contiguous relations for ${}_2F_1$

Two hypergeometric series are called *contiguous* if one can obtain one from the other by adding or subtracting 1 from exactly one of the parameters. There are a number of identities that relate various contiguous hypergeometric series to one another. In this section, we will derive the classic ones for ${}_2F_1$ given by Gauß.

```

context
  fixes F :: "'a :: field_char_0 ⇒ 'a ⇒ 'a ⇒ 'a fps"
  fixes F' :: "'a :: field_char_0 ⇒ 'a ⇒ 'a ⇒ 'a fls"
  defines "F ≡ (λa b c. fps_hypergeo [a, b] [c] 1)"
  defines "F' ≡ (λa b c. fps_to_fls (fps_hypergeo [a, b] [c] 1))"
begin

```

```

lemma fps_gauss_commute: "F a b c = F b a c"
⟨proof⟩

```

```

lemma fls_gauss_commute: "F' a b c = F' b a c"
⟨proof⟩

```

```

lemma gauss_contiguous1:
  assumes c: "c ∉ ℤ≤0"
  shows "fps_X * fps_deriv (F a b c) = fps_X * fps_const (a * b / c)
* F (a+1) (b+1) (c+1)"
⟨proof⟩

```

```

lemma gauss_contiguous2:

```

```

    assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
    shows "fps_X * fps_deriv (F a b c) = fps_const a * (F (a+1) b c -
F a b c)"
    <proof>

lemma gauss_contiguous3:
    assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
    shows "fps_X * fps_deriv (F a b c) = fps_const b * (F a (b+1) c -
F a b c)"
    <proof>

lemma gauss_contiguous4:
    assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ " "c  $\neq 1$ "
    shows "fps_X * fps_deriv (F a b c) = fps_const (c - 1) * (F a b (c-1)
- F a b c)"
    <proof>

lemma gauss_contiguous1':
    assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
    shows "fls_deriv (F' a b c) = fls_const (a * b / c) * F' (a+1) (b+1)
(c+1)"
    <proof>

lemma gauss_contiguous2':
    assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
    shows "fls_deriv (F' a b c) = fls_const a * (F' (a+1) b c - F' a b
c) / fls_X"
    <proof>

lemma gauss_contiguous3':
    assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
    shows "fls_deriv (F' a b c) = fls_const b * (F' a (b+1) c - F' a b
c) / fls_X"
    <proof>

lemma gauss_contiguous4':
    assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ " "c  $\neq 1$ "
    shows "fls_deriv (F' a b c) = fls_const (c - 1) * (F' a b (c-1) -
F' a b c) / fls_X"
    <proof>

lemma gauss_contiguous5_strong:
    assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
    shows "fps_X * (1 - fps_X) * fps_deriv (F a b c) =
(fps_const (c - a) * F (a-1) b c +
(fps_const (a - c) + fps_const b * fps_X) * F a b c)"
    (is "?lhs = ?rhs")
    <proof>

```

```

lemma gauss_contiguous5:
  assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "fps_X * fps_deriv (F a b c) =
    (fps_const (c - a) * F (a-1) b c +
     (fps_const (a - c) + fps_const b * fps_X) * F a b c) / (1
- fps_X)"
  <proof>

lemma gauss_contiguous6:
  assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "fps_X * fps_deriv (F a b c) =
    (fps_const (c - b) * F a (b-1) c +
     (fps_const (b - c) + fps_const a * fps_X) * F a b c) / (1
- fps_X)"
  <proof>

lemma gauss_contiguous5':
  assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "fls_deriv (F' a b c) =
    (fls_const (c - a) * F' (a-1) b c + (fls_const (a - c) + fls_const
b * fls_X) * F' a b c) /
    (fls_X * (1 - fls_X))"
  <proof>

lemma gauss_contiguous6':
  assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "fls_deriv (F' a b c) =
    (fls_const (c - b) * F' a (b-1) c +
     (fls_const (b - c) + fls_const a * fls_X) * F' a b c) / (fls_X
* (1 - fls_X))"
  <proof>

lemma gauss_contiguous7_strong:
  assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "fps_X * (1 - fps_X) * fps_deriv (F a b c) = fps_const (1/c) *
fps_X *
    (fps_const ((c-a)*(c-b)) * F a b (c+1) + fps_const (c*(a+b-c))
* F a b c)" (is "?lhs = ?rhs")
  <proof>

lemma gauss_contiguous7:
  assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "fps_X * fps_deriv (F a b c) = fps_const (1/c) * fps_X *
    (fps_const ((c-a)*(c-b)) * F a b (c+1) + fps_const (c*(a+b-c))
* F a b c) / (1 - fps_X)"
  <proof>

lemma gauss_contiguous7':
  assumes c: "c  $\notin \mathbb{Z}_{\leq 0}$ "

```

```

  shows "fls_deriv (F' a b c) = (fls_const ((c-a)*(c-b)) * F' a b (c+1)
+
                                         fls_const (c*(a+b-c)) * F' a b c) / (fls_const
c * (1 - fls_X))"
<proof>

```

```

lemma fps_deriv_gauss_hypergeo:
  assumes c: "c ∉ ℤ≤0"
  shows "fps_deriv (F a b c) = fps_const (a * b / c) * F (a+1) (b+1) (c+1)"
<proof>

```

```

lemma higher_fps_deriv_gauss_hypergeo:
  assumes c: "c ∉ ℤ≤0"
  shows "(fps_deriv ^^ n) (F a b c) =
        fps_const (pochhammer a n * pochhammer b n / pochhammer c n)
*
        F (a + of_nat n) (b + of_nat n) (c + of_nat n)"
<proof>

```

end

1.6 Bessel-type hypergeometric functions

We briefly look at the regularised hypergeometric function ${}_0F_1$, which is closely related to Bessel functions of the first kind (it is also known as the Bessel–Clifford function).

```

definition fps_bessel :: "'a :: Gamma ⇒ 'a fps" where
  "fps_bessel a = reg_fps_hypergeo_aux [] [1, a+1]"

```

```

lemma fps_conv_radius_bessel [simp]: "fps_conv_radius (fps_bessel a)
= ∞"
<proof>

```

```

lemma fps_deriv_bessel [simp]: "fps_deriv (fps_bessel a) = fps_bessel
(a+1)"
<proof>

```

The contiguous identity for these functions has the following nice form:

```

lemma fps_bessel_contiguous:
  "fps_bessel (a-1) = fps_const a * fps_bessel a + fps_X * fps_bessel
(a+1)"
<proof>

```

Together with the derivative identity, we find that ${}_0F_1(; a; z)$ is a solution of the ordinary differential equation $F = (a + 1)F' + XF'' = 0$.

```

lemma fps_bessel_ODE:

```

```

fixes a
defines "D ≡ fps_deriv"
defines "F ≡ fps_bessel a"
shows "F = fps_const (a+1) * D F + fps_X * (D ^^ 2) F"
⟨proof⟩

```

1.7 Kummer's confluent hypergeometric function

We will now look at Kummer's confluent hypergeometric function ${}_1F_1(a; b; z)$.

```

definition fps_kummer :: "'a ⇒ 'a ⇒ 'a ⇒ 'a :: field_char_0 fps"
  where "fps_kummer a b c = fps_hypergeo [a :: 'a] [b] c"

```

```

lemma fps_kummer_0_left [simp]: "fps_kummer 0 b c = 1"
  ⟨proof⟩

```

Kummer's series converges everywhere, so Kummer's function is entire.

```

lemma fps_conv_radius_kummer [simp]:
  fixes a b c :: "'a :: {banach, real_normed_field}"
  shows "fps_conv_radius (fps_kummer a b c) = ∞"
  ⟨proof⟩

```

```

lemma fps_kummer_same [simp]:
  assumes "a ∉ ℤ≤₀"
  shows "fps_kummer a a c = fps_exp c"
  ⟨proof⟩

```

```

lemma hypergeo_F_kummer_same [simp]:
  assumes "a ∉ ℤ≤₀"
  shows "hypergeo_F [a] [a] z = exp z"
  ⟨proof⟩

```

This function satisfies the identity ${}_1F_1(a; b; z) = e^z {}_1F_1(b - a; b; -z)$, also known as the *Kummer transform*.

```

theorem fps_kummer_transform:
  assumes "b ∉ ℤ≤₀"
  shows "fps_kummer a b c = fps_exp c * fps_kummer (b - a) b (-c)"
  ⟨proof⟩

```

```

lemma fps_kummer_transform':
  assumes "b ∉ ℤ≤₀"
  shows "fps_exp (-c) * fps_kummer a b c = fps_kummer (b - a) b (-c)"
  ⟨proof⟩

```

```

lemma fps_kummer_minus:
  assumes "b ∉ ℤ≤₀"
  shows "fps_kummer a b (-c) = fps_exp (-c) * fps_kummer (b - a) b c"
  ⟨proof⟩

```

```

lemma hypergeo_F_kummer_transform:
  fixes a b :: "'a :: {banach, real_normed_field}"
  assumes "b  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "hypergeo_F [a] [b] z = exp z * hypergeo_F [b - a] [b] (-z)"
<proof>

```

```

lemma fps_kummer_1_2_aux: "fps_X * fps_kummer 1 2 1 = fps_exp 1 - (1
:: 'a :: field_char_0 fps)"
<proof>

```

```

lemma fps_kummer_1_2: "fps_kummer 1 2 1 = (fps_exp 1 - (1 :: 'a :: field_char_0
fps)) / fps_X"
<proof>

```

```

lemma hypergeo_F_1_2:
  assumes "x  $\neq 0$ "
  shows "hypergeo_F [1] [2] x = (exp x - 1) / x"
<proof>

```

We derive some simple contiguous relations.

```

lemma fps_kummer_contiguous1:
  assumes "b  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "fps_deriv (fps_kummer a b c) =
        fps_const (a * c / b) * fps_kummer (a+1) (b+1) c"
<proof>

```

```

lemma fps_kummer_contiguous1':
  assumes "b  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "fps_X * fps_deriv (fps_kummer a b c) =
        fps_const (a * c / b) * fps_X * fps_kummer (a+1) (b+1) c"
<proof>

```

```

lemma fps_kummer_contiguous2:
  assumes "b  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "fps_X * fps_deriv (fps_kummer a b c) =
        fps_const a * (fps_kummer (a+1) b c - fps_kummer a b c)"
<proof>

```

```

lemma fps_kummer_contiguous3:
  assumes "b  $\notin \mathbb{Z}_{\leq 0}$ " "b  $\neq 1$ "
  shows "fps_X * fps_deriv (fps_kummer a b c) =
        fps_const (b-1) * (fps_kummer a (b-1) c - fps_kummer a b
c)"
<proof>

```

```

lemma fps_kummer_contiguous4:
  assumes "b  $\notin \mathbb{Z}_{\leq 0}$ "
  shows "fps_const b * (fps_kummer (a+1) b c - fps_kummer a b c) =
        fps_const c * fps_X * fps_kummer (a+1) (b+1) c" (is "?lhs

```

= ?rhs")
 <proof>

lemma *fps_kummer_contiguous5*:
 assumes "b $\notin \mathbb{Z}_{\leq 0}$ " "b $\neq 1$ "
 shows "fps_const (b*(b-1)) * (fps_kummer a (b - 1) c - fps_kummer a b c) =
 fps_const (a * c) * fps_X * fps_kummer (a+1) (b+1) c"
 <proof>

Kummer's function is a solution of the ODE $af(z) - (b-z)f'(z) - zf''(z) = 0$.

theorem *fps_kummer_ODE*:
 fixes a b :: "'a :: field_char_0"
 defines "W $\equiv$ fps_kummer a b 1"
 assumes b: "b $\notin \mathbb{Z}_{\leq 0}$ "
 shows "fps_const a * W - (fps_const b - fps_X) * fps_deriv W - fps_X
 * (fps_deriv ^^ 2) W = 0"
 <proof>

As an application, we show that the error function can be expression terms of the hypergeometric function ${}_1F_1(\frac{1}{2}; \frac{3}{2}; -z^2)$.

The error function is the unique function such that $\text{erf}(0) = 0$ and $\text{erf}'(z) = \frac{2}{\pi} \exp(-z^2)$. Or, in other words:

$$\text{erf}(z) = \frac{2}{\pi} \int_0^x \exp(-t^2) dt$$

This function is already available in the AFP and it is defined there using its Maclaurin series, so it is easy to prove the identity we want just by comparing coefficients.

theorem *erf_conv_hypergeo_F*:
 fixes z :: "'a :: {banach, real_normed_field}"
 shows "erf z = of_real (2 / sqrt pi) * z * hypergeo_F [1/2] [3/2] (-(z
 ^ 2))"
 <proof>

end

References

- [1] D. Duverney. *An Introduction to Hypergeometric Functions*. Springer, 2024.