

Discrete Summation

Florian Haftmann
with contributions by Amine Chaieb

March 8, 2026

Abstract

These theories introduce basic concepts and proofs about discrete summation: shifts, formal summation, falling factorials and stirling numbers. As proof of concept, a simple summation conversion is provided.

1 Falling factorials

theory *Factorials*

imports *Complex-Main HOL-Combinatorics.Stirling*
begin

lemma *pochhammer-0* [*simp*]: — TODO move
 $\text{pochhammer } 0 \ n = (0::\text{nat}) \text{ if } n > 0$
{*proof*}

definition *ffact* :: $\text{nat} \Rightarrow 'a::\text{comm-semiring-1-cancel} \Rightarrow 'a$
where $\text{ffact } n \ a = \text{pochhammer } (a + 1 - \text{of-nat } n) \ n$

lemma *ffact-0* [*simp*]:
 $\text{ffact } 0 = (\lambda x. 1)$
{*proof*}

lemma *ffact-Suc*:
 $\text{ffact } (\text{Suc } n) \ a = a * \text{ffact } n \ (a - 1)$
for $a :: 'a :: \text{comm-ring-1}$
{*proof*}

lemma *ffact-Suc-rev*:
 $\text{ffact } (\text{Suc } n) \ m = (m - \text{of-nat } n) * \text{ffact } n \ m$
for $m :: 'a :: \{\text{comm-semiring-1-cancel}, \text{ab-group-add}\}$
{*proof*}

lemma *ffact-nat-triv*:
 $\text{ffact } n \ m = 0 \text{ if } m < n$

<proof>

lemma *ffact-Suc-nat*:

$ffact (Suc n) m = m * ffact n (m - 1)$
for $m :: nat$

<proof>

lemma *ffact-Suc-rev-nat*:

$ffact (Suc n) m = (m - n) * ffact n m$
<proof>

lemma *fact-div-fact-ffact*:

$fact n \text{ div } fact m = ffact (n - m) n$ **if** $m \leq n$
<proof>

lemma *fact-div-fact-ffact-nat*:

$fact n \text{ div } fact (n - k) = ffact k n$ **if** $k \leq n$
<proof>

lemma *ffact-fact*:

$ffact n (of-nat n) = (of-nat (fact n)) :: 'a :: comm-ring-1$
<proof>

lemma *ffact-add-diff-assoc*:

$(a - of-nat n) * ffact n a + of-nat n * ffact n a = a * ffact n a$
for $a :: 'a :: comm-ring-1$
<proof>

lemma *mult-ffact*:

$a * ffact n a = ffact (Suc n) a + of-nat n * ffact n a$
for $a :: 'a :: comm-ring-1$
<proof>

lemma *prod-ffact*:

fixes $m :: 'a :: \{ord, ring-1, comm-monoid-mult, comm-semiring-1-cancel\}$
shows $(\prod i = 0..<n. m - of-nat i) = ffact n m$
<proof>

lemma *prod-ffact-nat*:

fixes $m :: nat$
shows $(\prod i = 0..<n. m - i) = ffact n m$
<proof>

lemma *prod-rev-ffact*:

fixes $m :: 'a :: \{ord, ring-1, comm-monoid-mult, comm-semiring-1-cancel\}$
shows $(\prod i = 1..n. m - of-nat n + of-nat i) = ffact n m$
<proof>

lemma *prod-rev-ffact-nat*:
fixes $m :: nat$
assumes $n \leq m$
shows $(\prod_{i=1..n} m - n + i) = \text{ffact } n \ m$
 $\langle \text{proof} \rangle$

lemma *prod-rev-ffact-nat'*:
fixes $m :: nat$
assumes $n \leq m$
shows $\prod \{m - n + 1..m\} = \text{ffact } n \ m$
 $\langle \text{proof} \rangle$

lemma *ffact-eq-fact-mult-binomial*:
 $\text{ffact } k \ n = \text{fact } k * (n \text{ choose } k)$
 $\langle \text{proof} \rangle$

lemma *of-nat-ffact*:
 $\text{of-nat } (\text{ffact } n \ m) = \text{ffact } n \ (\text{of-nat } m :: 'a :: \text{comm-ring-1})$
 $\langle \text{proof} \rangle$

lemma *of-int-ffact*:
 $\text{of-int } (\text{ffact } n \ k) = \text{ffact } n \ (\text{of-int } k :: 'a :: \text{comm-ring-1})$
 $\langle \text{proof} \rangle$

lemma *ffact-minus*:
fixes $x :: 'a :: \text{comm-ring-1}$
shows $\text{ffact } n \ (-x) = (-1) ^ n * \text{pochhammer } x \ n$
 $\langle \text{proof} \rangle$

Conversion of natural potences into falling factorials and back

lemma *monomial-ffact*:
 $a ^ n = (\sum_{k=0..n} \text{of-nat } (\text{Stirling } n \ k) * \text{ffact } k \ a)$
for $a :: 'a :: \text{comm-ring-1}$
 $\langle \text{proof} \rangle$

lemma *ffact-monomial*:
 $\text{ffact } n \ a = (\sum_{k=0..n} (-1) ^ (n - k) * \text{of-nat } (\text{stirling } n \ k) * a ^ k)$
for $a :: 'a :: \text{comm-ring-1}$
 $\langle \text{proof} \rangle$

end

2 Some basic facts about discrete summation

theory *Discrete-Summation*
imports *Main*
begin

2.1 Auxiliary

lemma *add-sum-orient*:

$$\text{sum } f \{k..<j\} + \text{sum } f \{l..<k\} = \text{sum } f \{l..<k\} + \text{sum } f \{k..<j\}$$

<proof>

lemma *add-sum-int*:

fixes $j \ k \ l :: \text{int}$

shows $j < k \implies k < l \implies$

$$\text{sum } f \{j..<k\} + \text{sum } f \{k..<l\} = \text{sum } f \{j..<l\}$$

<proof>

2.2 The shift operator

definition $\Delta :: ('b::\text{ring-1} \Rightarrow 'a::\text{ab-group-add}) \Rightarrow \text{int} \Rightarrow 'a$

where

$$\Delta f \ k = f (\text{of-int } (k + 1)) - f (\text{of-int } k)$$

lemma *Δ -shift*:

$$\Delta (\lambda k. l + f \ k) = \Delta f$$

<proof>

lemma *Δ -same-shift*:

assumes $\Delta f = \Delta g$

shows $\exists l. \text{plus } l \circ f \circ \text{of-int} = g \circ \text{of-int}$

<proof>

lemma *Δ -add*:

$$\Delta (\lambda k. f \ k + g \ k) \ k = \Delta f \ k + \Delta g \ k$$

<proof>

lemma *Δ -factor*:

$$\Delta (\lambda k. c * k) \ k = c$$

<proof>

2.3 The formal sum operator

definition $\Sigma :: ('b::\text{ring-1} \Rightarrow 'a::\text{ab-group-add}) \Rightarrow \text{int} \Rightarrow \text{int} \Rightarrow 'a$

where

$$\Sigma f \ j \ l = (\text{if } j < l \text{ then } \text{sum } (f \circ \text{of-int}) \{j..<l\} \\ \text{else if } j > l \text{ then } - \text{sum } (f \circ \text{of-int}) \{l..<j\} \\ \text{else } 0)$$

lemma *Σ -same [simp]*:

$$\Sigma f \ j \ j = 0$$

<proof>

lemma *Σ -positive*:

$$j < l \implies \Sigma f \ j \ l = \text{sum } (f \circ \text{of-int}) \{j..<l\}$$

<proof>

lemma Σ -negative:

$$j > l \implies \Sigma f j l = - \Sigma f l j$$

<proof>

lemma Σ -comp-of-int:

$$\Sigma (f \circ \text{of-int}) = \Sigma f$$

<proof>

lemma Σ -const:

$$\Sigma (\lambda k. c) j l = \text{of-int } (l - j) * c$$

<proof>

lemma Σ -add:

$$\Sigma (\lambda k. f k + g k) j l = \Sigma f j l + \Sigma g j l$$

<proof>

lemma Σ -factor:

$$\Sigma (\lambda k. c * f k) j l = (c::'a::ring) * \Sigma (\lambda k. f k) j l$$

<proof>

lemma Σ -concat:

$$\Sigma f j k + \Sigma f k l = \Sigma f j l$$

<proof>

lemma Σ -incr-upper:

$$\Sigma f j (l + 1) = \Sigma f j l + f (\text{of-int } l)$$

<proof>

2.4 Fundamental lemmas: The relation between Δ and Σ

lemma Δ - Σ :

$$\Delta (\Sigma f j) = f$$

<proof>

lemma Σ - Δ :

$$\Sigma (\Delta f) j l = f (\text{of-int } l) - f (\text{of-int } j)$$

<proof>

end

3 A barebone conversion for discrete summation

theory *Summation-Conversion*

imports *Factorials Discrete-Summation*

begin

Extensible theorem collection for solving summation problems

named-theorems *summation rules for solving summation problems*

declare

Σ -const [summation] Σ -add [summation]
 Σ -factor [summation] monomial-ffact [summation]

lemma *intervall-simps* [summation]:

$(\sum k::nat = 0..0. f k) = f 0$
 $(\sum k::nat = 0..Suc n. f k) = f (Suc n) + (\sum k::nat = 0..n. f k)$
<proof>

lemma Δ -ffact:

$\Delta (ffact (Suc n)) k = of-nat (Suc n) * ffact n (of-int k :: 'a :: comm-ring-1)$
<proof>

lemma Σ -ffact-divide [summation]:

$\Sigma (ffact n) j l =$
 $(ffact (Suc n) (of-int l :: 'a :: \{idom-divide, semiring-char-0\}) - ffact (Suc n)$
 $(of-int j)) \text{ div } of-nat (Suc n)$
<proof>

Various other rules

lemma *of-int-coeff*:

$(of-int l :: 'a::comm-ring-1) * numeral k = of-int (l * numeral k)$
<proof>

lemmas *nat-simps* =

add-0-left add-0-right add-Suc add-Suc-right
mult-Suc mult-Suc-right mult-zero-left mult-zero-right
One-nat-def of-nat-simps

lemmas *of-int-pull-out* =

of-int-add [symmetric] of-int-diff [symmetric] of-int-mult [symmetric]
of-int-coeff

lemma *of-nat-coeff*:

$(of-nat n :: 'a::comm-semiring-1) * numeral m = of-nat (n * numeral m)$
<proof>

lemmas *of-nat-pull-out* =

of-nat-add [symmetric] of-nat-mult [symmetric] of-nat-coeff

lemmas *nat-pull-in* =

nat-int-add

lemmas *of-int-pull-in* =

of-int-pull-out [symmetric] add-divide-distrib diff-divide-distrib of-int-power
of-int-numeral of-int-neg-numeral times-divide-eq-left [symmetric]

Special for *nat*

definition *lift-nat* :: $(nat \Rightarrow nat) \Rightarrow int \Rightarrow int$

where

$$\text{lift-nat } f = \text{int} \circ f \circ \text{nat}$$

definition $\Sigma\text{-nat} :: (\text{nat} \Rightarrow \text{nat}) \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat} \langle \Sigma_{\mathbb{N}} \rangle$

where

$$[\text{summation}]: \Sigma_{\mathbb{N}} f m n = \text{nat} (\Sigma (\text{lift-nat } f) (\text{int } m) (\text{int } n))$$

definition $\text{pos-id} :: \text{int} \Rightarrow \text{int}$

where

$$\text{pos-id } k = (\text{if } k < 0 \text{ then } 0 \text{ else } k)$$

lemma $\Sigma\text{-pos-id}$ $[\text{summation}]$:

$$0 \leq k \implies 0 \leq l \implies \Sigma (\lambda r. f (\text{pos-id } r)) k l = \Sigma f k l$$

$\langle \text{proof} \rangle$

lemma $[\text{summation}]$:

$$\begin{aligned} (0::\text{int}) &\leq 0 \\ (0::\text{int}) &\leq 1 \\ (0::\text{int}) &\leq \text{numeral } m \\ (0::\text{int}) &\leq \text{int } n \end{aligned}$$

$\langle \text{proof} \rangle$

lemma $[\text{summation}]$:

$$\text{lift-nat } (\lambda n. m) = (\lambda k. \text{int } m)$$

$\langle \text{proof} \rangle$

lemma $[\text{summation}]$:

$$\text{lift-nat } (\lambda n. n) = \text{pos-id}$$

$\langle \text{proof} \rangle$

lemma $[\text{summation}]$:

$$\text{lift-nat } (\lambda n. f n + g n) = (\lambda k. \text{lift-nat } f k + \text{lift-nat } g k)$$

$\langle \text{proof} \rangle$

lemma $[\text{summation}]$:

$$\text{lift-nat } (\lambda n. m * f n) = (\lambda k. \text{int } m * \text{lift-nat } f k)$$

$\langle \text{proof} \rangle$

lemma $[\text{summation}]$:

$$\text{lift-nat } (\lambda n. f n * m) = (\lambda k. \text{lift-nat } f k * \text{int } m)$$

$\langle \text{proof} \rangle$

lemma $[\text{summation}]$:

$$\text{lift-nat } (\lambda n. f n \hat{=} m) = (\lambda k. \text{lift-nat } f k \hat{=} m)$$

$\langle \text{proof} \rangle$

Generic conversion

$\langle ML \rangle$

```
hide-fact (open) nat-simps of-int-pull-out of-int-pull-in  
end
```

4 Simple examples

```
theory Examples  
imports Summation-Conversion  
begin  
  
⟨ML⟩  
  
end
```