Symbolic Summation in Maple

S. A. Abramov[*]
Russian Academy of Sciences
Dorodnicyn Computing Centre
Vavilova st. 40, 119991, Moscow, GSP-1, Russia
abramov@ccas.ru

J. J. Carette
Computing and Software
McMaster University
Hamilton, L8S 4L8, Canada
carette@mcmaster.ca

K. O. Geddes[†]
Symbolic Computation Group
University of Waterloo
Waterloo, N2L 3G1, Canada
kogeddes@scg.math.uwaterloo.ca

H. Q. Le[‡]
Symbolic Computation Group
University of Waterloo
Waterloo, N2L 3G1, Canada
hqle@scg.math.uwaterloo.ca

Abstract

We describe the design and implementation of the Maple toolbox `SumTools`, a package for computing closed forms of indefinite and definite sums.

1 Introduction

This document discusses the design of the `SumTools` package which is used to finding *closed forms* of indefinite and definite sums.

For a given function f of k, one can pose the following two problems related to summation:

1. **indefinite sum:** compute $g(k) = \sum_k f(k)$, i.e., find a function $g(k)$ such that $g(k+1) - g(k) = f(k)$.
2. **definite sum:** compute $g(m,n) = \sum_{k=m}^{n} f(k)$, i.e., the definite sum of $f(k)$ over the given range $m..n$.

Summation has been an active research topic. There exist many algorithms, which cover various classes of summands, for computing closed forms of indefinite and definite sums.

Computing a closed form of a sum is one of the very basic operations. This operation exists in all general computer algebra systems such as Maple, Mathematica, Macsyma, MuPAD. In this document, we propose a re-design of summation in Maple. The focus is on a smooth integration of independent blocks of code, and on the development of recently-developed algorithms. Its design is based on four criteria: *functionality*, *integrability*, *extensibility*, and *performance*.

2 Design Criteria

1. **Functionality.** The package should cover a wide range of algorithms which handle various classes of summands. Not only the focus should be placed on computing a closed form of a given sum successfully, but also on the *ordering* of the algorithms implemented so that the "*simplest*" possible output can be obtained. Note that there is a trade-off between simplicity of output and performance.

[*]Partially supported by the French-Russian Lyapunov Institute under grant 98-03.
[†]Partially supported by Natural Sciences and Engineering Research Council of Canada Grant No. RGPIN8967-01.
[‡]Partially supported by Natural Sciences and Engineering Research Council of Canada Grant No. CRD215442-98.

2. **Integrability.** The package should be designed to allow the integration of different algorithms in a simple fashion. This implies code modularization (logical level) and appropriate code structure (physical level).
3. **Extensibility.** In order to incorporate newly-designed algorithms and to implement formulas of sums which exist in large collections of database such as those in [13, 23, 21], the package should be structured to allow easy addition of new code development.
4. **Performance.** The selection of algorithms to be implemented should also focus on efficiency. This also includes implementation of new, efficient algorithms.

3 Architecture Design

3.1 Indefinite Sums

The diagram in Figure 1 provides the classes of summands the package can handle, and the corresponding algorithms. These include the classes of polynomials, rational functions [1], hypergeometric terms [7], j-fold hypergeometric terms [12, 15], functions for which their minimal annihilators can be constructed [4], e.g., d'Alembertian terms [9]. A library extension mechanism is also used to include sums for which an algorithmic approach to finding closed forms does not yet exist (i.e., the pattern match approach is employed in principal). Currently the computable summands include expressions containing the harmonic function; the logarithmic function; the digamma and polygamma functions; the sine, cosine and exponential functions.

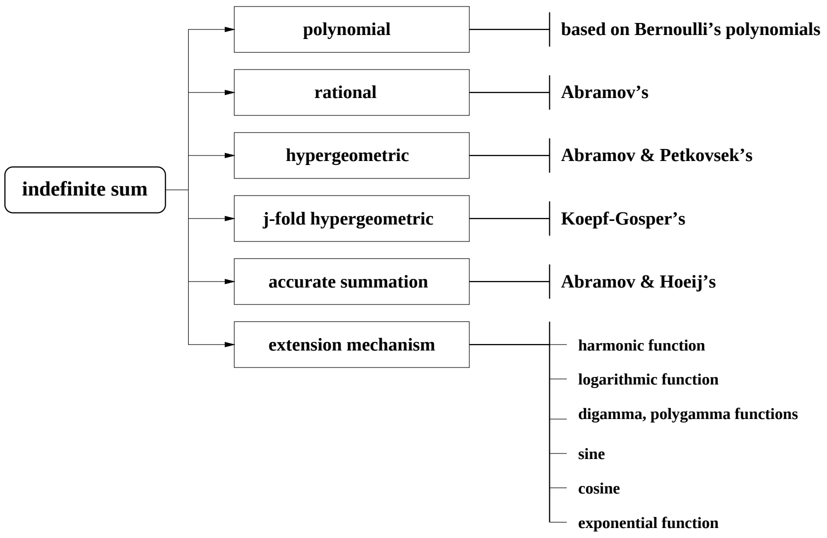

Figure 1: Indefinite sum: classification

Figure 2 is a flow chart illustrating the use of various algorithms each of which is applicable to a specific class of summands, as well as the ordering of these algorithms. Since each class is handled by a specific algorithm, the ordering is clear-cut for the case of indefinite sums.

3.2 Definite Sums

Figure 3 shows a diagram of different algorithms which help compute closed forms of definite sums. In addition to the method of first computing a closed form of the corresponding *indefinite* sum as described in 3.1, there exist other well-known algorithms which allow the computation of a closed form of a given definite sum "directly" (in some sense). They include the method of integral representation [11], the method of hypergeometric sums [26, 14], and the "conversion" method which first converts the given definite sum to a hypergeometric function [22], and then converts this hypergeometric function to "standard functions" [24, 21].

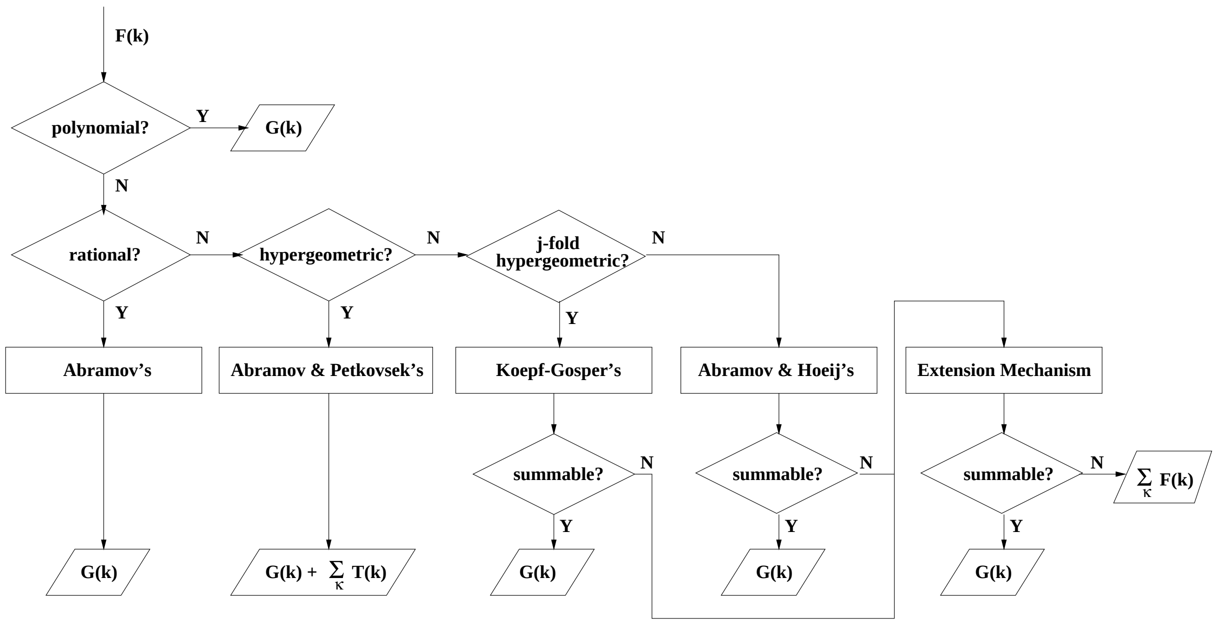

Figure 2: Indefinite sum: a flow chart

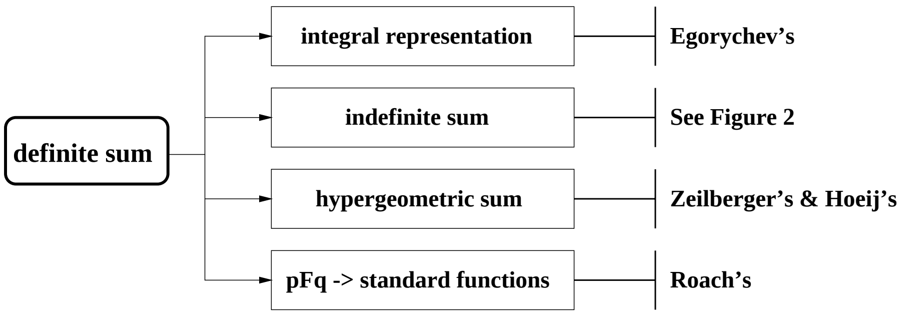

Figure 3: Definite sum: algorithms

The flow chart in Figure 4 shows the use of different algorithms which attempt to find closed forms of definite sums, and the ordering of the algorithms used. Except for the case where the summand is a hypergeometric term and there exist algorithms that specifically handles this case, the input summand used in other algorithms can be any arbitrary expression in general. Experimentation has been done on a large set of samples (see Section 7) in order to determine the ordering of the algorithms (recall the trade-off between simplicity and performance).

For the method "hypergeometric sum", the implementation also includes new results related to the applicability of Zeilberger's algorithm [2], and efficient algorithms to compute Zeilberger's recurrences [5, 16]. An implementation of these algorithms is discussed in details in [3, 18].

4 Functional Specification

For the remainder of the document, we denote $poly$, rat, hg_j, $hg_{i,j}$ for a polynomial, a rational function, a j-fold, and an (i,j)-fold hypergeometric term, respectively; psi for an expression involving the digamma and polygamma functions; ma for an arbitrary function where its minimal annihilator can be constructed; and $expr$ for a general expression.

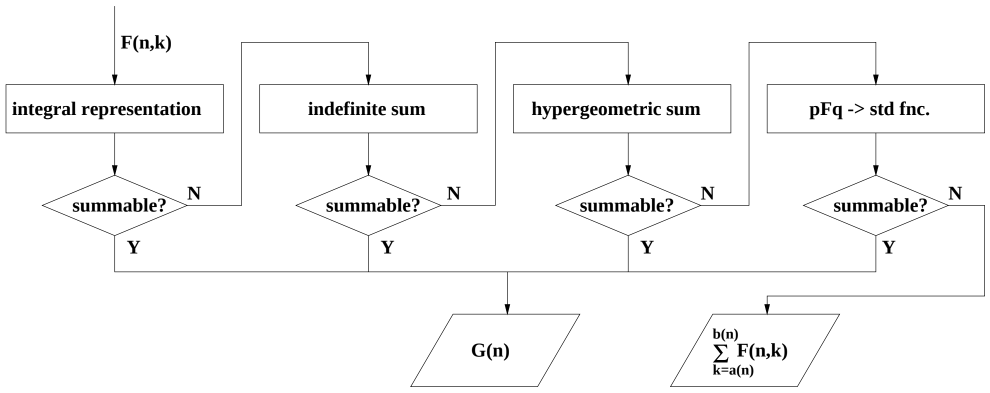

Figure 4: Definite sum: a flow chart

4.1 Main Functionalities

The package `SumTools` exports three functions and three sub-packages (or sub-modules):

`> eval(SumTools);`
module()
export HypergeometricTerm, IndefiniteSum, DefiniteSum,
IndefiniteSummation, DefiniteSummation, Summation;
local Preprocess, Tools, LimitRootOf, Floats;
option package;
description "summation tools";
end module

Let f be a function in k. The three exported functions are

calling sequence	description
$IndefiniteSummation(f, k)$;	compute an indefinite sum of $f(k)$
$DefiniteSummation(f, k = m..n)$;	compute the definite sum of $f(k)$ over the specified range $m..n$
$Summation(f, \ldots)$;	handle both indefinite and definite sums

They have the following signatures:

$$\begin{array}{lll} IndefiniteSummation: & (f\text{::expr}, k\text{::name}) & \longrightarrow \quad g\text{::expr} \\ DefiniteSummation: & (f\text{:expr}, k\text{::name} = m\text{::expr..}\ n\text{::expr}) & \longrightarrow \quad g\text{::expr} \\ Summation: & (f\text{::expr}, k\text{::name}) & \longrightarrow \quad g\text{::expr} \\ Summation: & (f\text{:expr}, k\text{::name} = m\text{::expr..}\ n\text{::expr}) & \longrightarrow \quad g\text{::expr} \end{array}$$

See Subsection 4.3 for the signatures of the two special cases of $DefiniteSummation$.

4.2 The Sub-package `IndefiniteSum`

The sub-package `IndefiniteSum` consists of functions to compute indefinite sums of various classes of summands:

`> eval(SumTools:-IndefiniteSum);`
module()
export Polynomial, Rational, Hypergeometric, AccurateSummation, Indefinite,
AddIndefiniteSum, RemoveIndefiniteSum;
local functab, Polygamma, Remindef, HasIndefiniteSum, GetIndefiniteSum, GetIndefiniteSums,
Geometric, Harmonic, Ln, Stirling2, Cos, Sin, Exp;
option package;

description "indefinite sums";
end module

The exported functions have the following signatures:

Polynomial:	(p::poly,k::name)	$\longrightarrow$	g::poly
Rational:	(r::rat,k::name)	$\longrightarrow$	s::rat + t::psi
Hypergeometric:	(t :: hg$_j$,k::name)	$\longrightarrow$	$\sum_i g_i$::hg$_1$
AccurateSummation:	(f::ma,k::name)	$\longrightarrow$	g::expr
Indefinite:	(f::expr,k::name)	$\longrightarrow$	g::expr
AddIndefiniteSum:	(*fname*::name,*impl*::procedure)	$\longrightarrow$	NULL
RemoveIndefiniteSum:	(*fname*::name)	$\longrightarrow$	NULL

The exported functions *Polynomial*, *Rational*, *Hypergeometric*, *AccurateSummation* cover the classes of summands the sub-package can handle (see Figure 1). The main function *Indefinite*, which computes an indefinite sum of a given input expression, is a combination of the algorithms handling various classes of summands. The two functions *AddIndefiniteSum*, *RemoveIndefiniteSum* provide a library extension mechanism which allows the addition and removal of closed forms of indefinite sums which the existing algorithms cannot yet handle; the argument *fname* is the name of the function which appears in the summand, e.g., sin, cos; and the argument *impl* is a procedure for handling summands containing *fname*.

4.3 The Sub-package `DefiniteSum`

The sub-package consists of only one function for computing closed forms of definite sums:

```
> eval(SumTools:-DefiniteSum);
```
module()
export Definite;
local Frontend, Combinatorial, Hypergeometric, IndefFirst, ToHypergeometric, SelectVar;
option package;
description "definite sums";
end module

The exported function *Definite* has the following signatures

Definite:	(f::expr,k::name = m::expr .. n::expr)	$\longrightarrow$	g::expr
Definite:	(f::expr,k::name = r::*RootOf*(expr))	$\longrightarrow$	g::expr
Definite:	(f::expr,k::name = m::expr)	$\longrightarrow$	g::expr

The last two signatures are special cases of the first one, and are used for notational convenience, i.e., users can enter them directly. The second signature is simply for a definite sum over the index of a RootOf; and the third one is for function evaluation at a given point.

The local procedures *Combinatorial*, *Hypergeometric*, *IndefFirst*, *ToHypergeometricFunction* are implementations of the four methods as shown in Figure 3; *Frontend* is used as the front-end of *Definite* for handling *simple* definite sums quickly.

4.4 The Sub-package `HypergeometricTerm`

In addition to providing functions for computing closed forms of indefinite and definite sums of hypergeometric terms, the sub-package `HypergeometricTerm` also includes functions to construct normal forms of rational functions and of hypergeometric terms. These forms have various applications in computer algebra.

```
> eval(SumTools:-Hypergeometric);
```
module()
export PolynomialNormalForm, RationalCanonicalForm, MultiplicativeDecomposition, SumDecomposition, CanonicalRepresentation, IsHypergeometricTerm, AreSimilar, IsHolonomic, IsProperHypergeometricTerm, Gosper, ExtendedGosper, KoepfGosper, Zeilberger, ZeilbergerRecurrence, ExtendedZeilberger, KoepfZeilberger, IsZApplicable, ZpairDirect,

LowerBound, MinimalZpair, WZMethod, IndefiniteSum, DefiniteSum;
local ZeroAndPole, SelectRoots, GosperStep3, ApplyLtoF, MakeMonic;
option package;
description "tools for handling hypergeometric terms";
end module

The functions to construct normal and canonical forms include

calling sequence	description
$PolynomialNormalForm(R$::rat,k::name);	the polynomial normal form of $R(k)$ [12, 20]
$RationalCanonicalForm(R$::rat,k::name);	the two rational canonical forms of $R(k)$ [6]
$MultiplicativeDecomposition(T$::hg$_1$,$k$::name);	the two multiplicative decompositions of $T(k)$ [7]
$SumDecomposition(T$::hg$_1$,k::name);	an additive decomposition of $T(k)$ [7]
$CanonicalRepresentation(F$::hg$_{1,1}$,$n$::name,$k$::name);	the two canonical representations of $F(n,k)$ [8]

The functions related to finding closed forms of indefinite and definite sums of hypergeometric terms are

calling sequence	description
$IsHypergeometricTerm(T$::expr,k::name);	check if $T(k)$ is a hypergeometric term in k
$AreSimilar(T_1$::hg$_1$,T_2::hg$_1$,k::name);	check if the terms $T_1(k)$ and $T_2(k)$ are similar [22]
$IsHolonomic(F$::hg$_{1,1}$,n::name,k::name);	check if $F(n,k)$ is holonomic [8]
$IsProperHypergeometricTerm(F$::hg$_{1,1}$, n::name,k::name);	check if $F(n,k)$ is a proper hypergeometric term [8]
$Gosper(T$::hg$_1$,k::name);	indefinite hypergeomeric summation [12]
$ExtendedGosper(\{T_1$::hg$_1$,T_2::hg$_1$,...$\}$, k::name);	extended Gosper's algorithm [22]
$KoepfGosper(T$::hg$_j$,k::name);	extension of Gosper's algorithm to j-fold terms[15]
$Zeilberger(F$::hg$_{1,1}$,n::name,k::name);	perform Zeilberger's algorithm [26]
$ZeilbergerRecurrence(F$::hg$_{1,1}$, n::name,k::name,m::expr..n::expr);	construct Zeilberger's recurrence for $\sum_{k=m}^{n} F(n,k)$ [26]
$ExtendedZeilberger(V$,n::name,k::name);	extension of Zeilberger's for $V = \sum_{k=m}^{n} F(m,k)$ [17]
$KoepfZeilberger(G$::hg$_{i,j}$, n::name,k::name);	extension of Zeilberger's to (i,j)-fold terms [15]
$IsZApplicable(F$::hg$_{1,1}$,n::name,k::name);	applicability of Zeilberger's algorithm to $F(n,k)$ [2]
$ZpairDirect(R$::rat,n::name,k::name);	direct method to construct the Z-pair for $R(n,k)$ [16]
$LowerBound(F$::hg$_{1,1}$,n::name,k::name);	a lower bound for the order of the telescopers for $F(n,k)$ [5]
$MinimalZpair(F$::hg$_{1,1}$,n::name,k::name);	compute the minimal Z-pair for $F(n,k)$
$WZMethod(f$::expr,r::expr, n::name,k::name);	perform Wilf-Zeilberger's algorithm [25]
$IndefiniteSum(T$::expr,k::name);	indefinite sum of hypergeometric terms
$DefiniteSum(F$::expr,n::name, k::name $= m$::expr..n::expr);	definite sum of hypergeometric terms

See [3, 18] for a detailed discussion of the implementation of this sub-package.

5 Implementation Details

5.1 Code Structure and Dependency

Figure 5 shows code structure and code dependency of the package `SumTools`. The `Preprocess` function classifies the given sum into one of the two types (indefinite or definite). Each type is handled by the corresponding independent sub-module. This allows easy extensibility of functionalities for each sub-module. The integrability of the package as a whole is shown by the dependency of the sub-modules: `HypergeometricTerm` provides functionalities, while `Tools` provides various auxiliary tools to `IndefiniteSum` and `DefiniteSum`;

`Extensibility` provides a library extension mechanism to `IndefiniteSum` which in turns provides functionality to `DefiniteSum`.

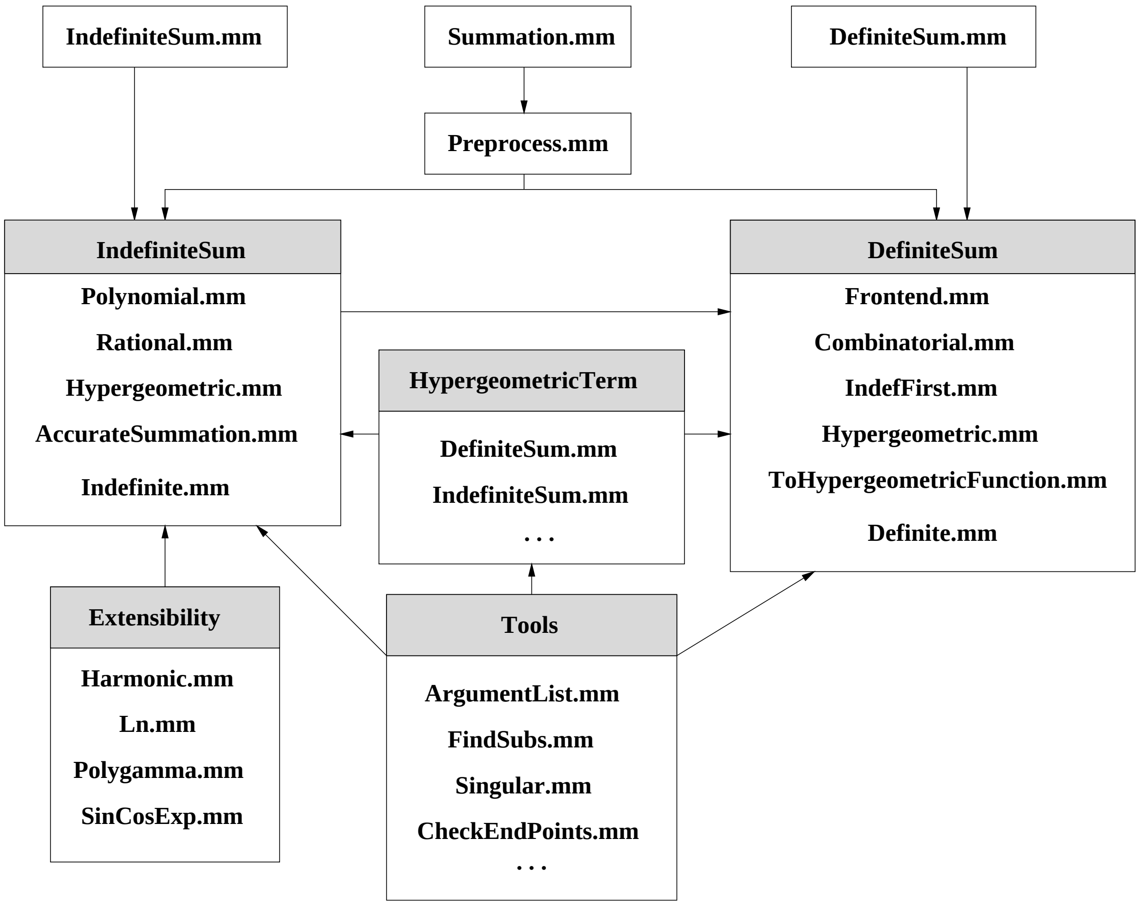

Figure 5: SumTools package: code structure and code dependency

5.2 Library Extension Mechanism

For the case of indefinite sum, a library extension mechanism [19, Chap. 6] is provided. This is done via the two functions *AddIndefiniteSum* and *RemoveIndefiniteSum* of the `IndefiniteSum` sub-package:

```
> with(SumTools:-IndefiniteSum);
```

$$[\text{AccurateSummation}, \text{Hypergeometric}, \text{Indefinite}, \text{Polynomial}, \text{Rational}, \text{AddIndefiniteSum}, \text{RemoveIndefiniteSum}]$$

Let f be a user-defined procedure which handles summands containing a function g. To add this knowledge into the function `Indefinite`, one does

$$AddIndefiniteSum(g, eval(f, 1));$$

Similarly, to remove the knowledge of g from `Indefinite`, one does

$$RemoveIndefiniteSum(g);$$

5.3 Handling Floats

For a given summand which contains floating-point numbers, the current implementation simply converts these numbers into exact rational numbers before the corresponding function f which handles the summand is invoked. Numerical evaluation is then applied to the output from f.

5.4 Handling Composite Structures

If the input is a composite structure (e.g., list, set, matrix, vector) of summands, then the corresponding solver will map itself into the elements of this structure.

6 Examples

6.1 Indefinite Sums

```
> with(SumTools:-IndefiniteSum):
```

Example 1

```
> F := 1/(n^2+sqrt 5 *n-1);
```

$$F := \frac{1}{n^2 + \sqrt{5}\, n - 1}$$

Since F is a rational function of n, the algorithm [1] is used:

```
> Sum(F,n) = Rational(F,n);
```

$$\sum_n \frac{1}{n^2 + \sqrt{5}\, n - 1} = -\frac{4\,(6\,n^2 + 6\,\sqrt{5}\, n - 6\,n + 7 - 3\,\sqrt{5})}{3\,\,(2\,n + \sqrt{5} + 1)\,(2\,n + \sqrt{5} - 1)\,(2\,n + \sqrt{5} - 3)}$$

The result shows that F is rational summable.

Example 2

```
> F := binomial(n/2+k,k)*2^(-k);
```

$$F := 2^{-k}\binom{n/2 + k}{k}$$

Since F is a 2-fold hypergeometric term w.r.t. n, W. Koepf's extension to Gosper's algorithm [15] is used:

```
> Sum(F,n) = Hypergeometric(F,n);
```

$$\sum_n 2^{-k}\binom{n/2 + k}{k} = \frac{1}{2(k+1)}\,2^{-k}\left(n\binom{n/2 + k}{k} + (n+1)\binom{n/2 + 1/2 + k}{k}\right)$$

Example 3

```
> F := n^2/binomial(2*n,n)/(n^2+3*n+2);
```

$$F := \frac{n^2}{(n^2 + 3\,n + 2)\binom{2\,n}{n}}$$

Although the hypergeometric term F is not hypergeometric summable, it is possible to re-write F as $\Delta_n F_1 + F_2$ where F_1 and F_2 are hypergeometric terms and F_2 is *simpler* than F [7]. Hence,

```
> Sum(F,n) = Hypergeometric(F,n);
```

$$\sum_n \frac{n^2}{(n^2 + 3\,n + 2)\binom{2\,n}{n}} = -\frac{6\,n^2 - 11\,n - 125}{9\,(n+1)}\prod_{i=1}^{n-1}\frac{i}{4\,i + 2} + \sum_n \frac{457\,n + 250}{54\,(n+1)}\prod_{i=1}^{n-1}\frac{i}{4\,i + 6}$$

Note that a multiplicative representation of F is

$$\frac{n^2}{2\,(n+1)(n+2)}\prod_{i=1}^{n-1}\frac{i+1}{2\,(2\,i + 1)}.$$

Example 4 [4]
```
> F := 1/5*((1/2+1/2*5^(1/2))^n-(1/2-1/2*5^(1/2))^n)^2;
```

$$F := \left(\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)\right)^2$$

The complete factored minimal annihilator for F can be constructed using [10], and the application of the method of accurate summation [4] provides a closed form for the indefinite sum of F :
```
> Sum(F,n) = AccurateSummation(F,n);
```

$$\sum_n \left(\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)\right)^2 = \frac{1}{5}(-1)^n - \frac{1}{10}(1+\sqrt{5})\left(\frac{1-\sqrt{5}}{2}\right)^{2n} - \frac{1}{10}(1-\sqrt{5})\left(\frac{3+\sqrt{5}}{2}\right)^n$$

Note that instead of calling a specific routine corresponding to the given class of summands as shown in the above four examples, calling the general routine *Indefinite* should yield the same answers.

Example 5
```
> F := sin(x)*cos(x+1)-ln(2*x);
```

$$F := \sin(x)\,\cos(x+1) - \ln(2\,x)$$

It is shown in Figure 1 that knowledge about the functions sin, cos, and ln is known via the library extension mechanism. Hence, it is possible to compute a closed form for $\sum_x F$:
```
> Sum(F,n) = Indefinite(F,x);
```

$$\sum_n \sin(x)\,\cos(x+1) - \ln(2\,x) = -\frac{1}{2}\,\frac{x - x\cos(1)^2 + \cos(x)^2 + 2\,x\ln(2)\sin(1) + 2\,\ln(\Gamma(x))\sin(1)}{\sin(1)}$$

6.2 Definite Sums

```
> with(SumTools:-Definite):
```

Example 6
```
> F := (2+k)^(k-2)*(1+n-k)^(n-k)/(k!*(n-k)!);
```

$$F := \frac{(2+k)^{k-2}\,(1+n-k)^{n-k}}{k!\,(n-k)!}$$

The method [11] recognizes the summand is Abel's type, and is able to find a closed form for $\sum_{k=0}^n F$:
```
> Sum(F,k=0..n) = Definite(F,k=0..n);
```

$$\sum_{k=0}^{n} \frac{(2+k)^{k-2}\,(1+n-k)^{n-k}}{k!\,(n-k)!} = \frac{1}{4}\,\frac{(3+n)^n}{n!} - \frac{1}{6}\,\frac{(3+n)^{n-1}}{(n-1)!}$$

Example 7 [23, Ex. 10, p.121]
```
> F := binomial(2*n-2*k,n-k)*2^(4*k)*((2*k)*(2*k+1)*binomial(2*k,k))^(-1);
```

$$F := \frac{1}{2}\,\frac{\binom{2n-2k}{n-k}2^{4k}}{k\,(2\,k+1)\binom{2k}{k}}$$

Since F is hypergeometric summable w.r.t. k, a closed form of $\sum_{k=1}^n F$ can be computed:
```
> Sum(F,k=1..n) = Definite(F,k=1..n);
```

$$\sum_{k=1}^{n} \frac{1}{2}\,\frac{\binom{2n-2k}{n-k}2^{4k}}{k\,(2\,k+1)\binom{2k}{k}} = 4\,\frac{(2\,n-1)\binom{2n-2}{n-1}}{2\,n+1}$$

Example 8 [23, Ex. 11, p.164]

```
> F := binomial(2*n+1,2*k)^2;
```

$$\binom{2\,n+1}{2\,k}^2$$

Zeilberger's algorithm is applicable to the hypergeometric term F [2]. The combination of this algorithm and the one in [14] helps find a closed form of $\sum_{k=0}^{n} F$:

```
> Sum(F,k=0..n) = Definite(F,k=0..n);
```

$$\sum_{k=0}^{n} \binom{2\,n+1}{2\,k}^2 = \frac{4^n\,\Gamma\,(2\,n+3/2)}{\Gamma\,(n+1)\,\Gamma\,(n+3/2)}$$

Example 9

```
> F := 2^(2*k)/Pi^(1/2)*GAMMA(k-n)*GAMMA(k+n)/GAMMA(2*k+1)*z^k;
```

$$F := \frac{2^{2\,k}\,\Gamma\,(k-n)\,\Gamma\,(k+n)\,z^k}{\sqrt{\pi}\,\Gamma\,(2\,k+1)}$$

The "conversion" method is used in this example. It first converts $\sum_{k=0}^{\infty} F$ to hypergeometric function $(-\sqrt{\pi}/(\sin(\pi\,n)n)\ {}_2F_1(n,-n;1/2;z)$ which is then converted to "standard functions":

```
> Sum(F,k=0..infinity) = Definite(F,k=0..infinity);
```

$$\sum_{k=0}^{\infty} \frac{2^{2\,k}\,\Gamma\,(k-n)\,\Gamma\,(k+n)\,z^k}{\sqrt{\pi}\,\Gamma\,(2\,k+1)} = -\frac{\sqrt{\pi}\cos\left(2\,n\arcsin\left(\sqrt{z}\right)\right)\csc\left(\pi\,n\right)}{n}$$

Example 10

Recall that the *RootOf* structure in Maple is a place holder for representing all the roots of a given expression in one variable. In particular, it is the standard representation for algebraic numbers, algebraic functions.

```
> F := (t^2+1)/(t^3-5*t+2);
```

$$F := \frac{t^2+1}{t^3-5\,t+2}$$

```
> Sum(F,t=RootOf(x^5+x+1)) = Definite(F,t=RootOf(x^5+x+1));
```

$$\sum_{t=RootOf(\_Z^5+\_Z+1)} \frac{t^2+1}{t^3-5\,t+2} = -\frac{269}{833}$$

7 Testing

The goal is to include as many tests from different sources as possible. At this moment, we have prepared a number of tests. Many of them are taken from [13, 23]. For the indefinite case, 618 summands are tested: 30 polynomials, 60 rational functions, 477 hypergeometric terms, and 51 used for accurate summation. As for the definite case, 177 summands are used to test the four main methods.

It is necessary that all the calls to *sum* in the Maple test suite be tested. This is planned for near future.

8 Concluding Remarks

We have presented in this document a design for the `SumTools` package. When the package is "completed", the function *Summation* is expected to replace the current command *sum* in Maple. In terms of functionalities, the package includes the incorporation of accurate summation and of additive decomposition of hypergeometric terms to the indefinite case; the integration of the sub-package `SumTools:-HypergeometricTerm`, and of the function `convert/StandardFunctions` (in the "conversion" method) are included for the definite case. These algorithms are not implemented or not incorporated to the current *sum*.

Although the code structure is new, we should stress that we re-use good pieces of code written by various Maple developers throughout many years. Hence, this work is a collective contribution of the Maple developers. Of equal importance, the design also focuses on the integrability and on the extensibility. This hopefully will help the maintenance and future development.

References

[1] S.A. Abramov, Indefinite sums of rational functions, *Proc. ISSAC'95*, 1995, 303–308.

[2] S.A. Abramov, Applicability of Zeilberger's algorithm to hypergeometric terms, *Proc. ISSAC'02*, ACM Press, 2002, 1–7.

[3] S.A. Abramov, K.O. Geddes, H.Q. Le, Computer algebra library for the construction of the minimal telescopers, *Proc. ICMS'02*, 2002, 319–329.

[4] S.A. Abramov, M.v. Hoeij, Integration of solutions of linear functional equations, *Integral Transformations and Special Functions,* 1999, Vol.8, No. 1-2, 1999, 3–12.

[5] S.A. Abramov, H.Q. Le, A lower bound for the order of telescopers for a hypergeometric term, *Proc. FPSAC'02*, 2002, on CD.

[6] S.A. Abramov, M. Petkovšek, Canonical representations of hypergeometric terms. *Proc. FPSAC'2001*, 2001, 1–10.

[7] S.A. Abramov, M. Petkovšek, Optimal decomposition of indefinite hypergeometric sums, *Proc. ISSAC'01*, ACM Press, 2001, 7–14.

[8] S. Abramov and M. Petkovšek, Proof of a conjecture of Wilf and Zeilberger, *University of Ljubljana, Preprint series*, 39, 2001.

[9] S.A. Abramov, E.V. Zima, D'Alembertian solutions of inhomogeneous linear equations (differential, difference, and some other). *Proc. ISSAC'96*,ACM Press, 1996, 232–240.

[10] S.A. Abramov, E.V. Zima, Minimal Completely Factorable Annihilators, *Proc. ISSAC'97,* 1997.

[11] G.P. Egorychev, Integral representation and the computation of combinatorial sums. Novosibirsk, Nauka, 1977 (in Russian); English: Transl. of Math. Monographs, Vol. **59**, AMS, 1984, 2-nd Ed. in 1989.

[12] R.W. Gosper, Jr., Decision procedure for indefinite hypergeometric summation. *Proc. Natl. Acad. Sci. USA* **75**, 1977, 40–42.

[13] H.W. Gould, *Combinatorial Identities,* Morgantown, W. Va. 1972.

[14] M. van Hoeij, Finite singularities and hypergeometric solutions of linear recurrence equations. *J. Pure Appl. Algebra*, **139**, 1999, 109-131.

[15] W. Koepf, *Hypergeometric summation: an algorithmic approach to summation and special function identities*, Vieweg, 1998.

[16] H.Q. Le, A direct algorithm to construct Zeilberger's recurrences for rational functions, *Proc. FPSAC'01*, 2001, 303–312.

[17] H.Q. Le, Computing the minimal telescoper for sums of hypergeometric terms, SIGSAM Bulletin, v. 35, no. 3, September, 2–10.

[18] H.Q. Le, S.A. Abramov, K.O. Geddes, HypergeometricSum: A Maple package for finding closed forms of indefinite and definite sums of hypergeometric type, *Technical report CS-2001-24*, Department of Computer Science, University of Waterloo, Ontario, Canada.

[19] M.B. Monagan, K.O. Geddes, K.M. Heal, G. Labahn, S.M. Vorkoetter, J. McCarron, P. DeMarco, *Maple 7 Programming Guide,* Toronto: Waterloo Maple Inc., 2001.

[20] M. Petkovšek, Hypergeometric solutions of linear recurrences with polynomial coefficients, *J. Symb. Comput.* **14**, 1992, 243–264.

[21] A.P. Prudnikov, Yu. Brychkov, O. Marichev, *Integrals and Series, Volume 3: More Special Functions.* Gordon and Breach Science Publishers, 1990.

[22] M. Petkovšek, H. Wilf, D. Zeilberger, *A=B*, A.K.Peters, Wellesley, Massachusetts, 1996.

[23] J. Riordan, *Combinatorial identities*, John Wiley & Sons, 1968.

[24] K. Roach, Hypergeometric function representations. Proc. ISSAC'96, ACM Press, New York, 1996, 301–308.

[25] H. Wilf, D. Zeilberger, *Rational functions certify combinatorial identities*, J. Amer. Math. Soc. **3**, 1990, 147–158.

[26] D. Zeilberger, The method of creative telescoping, *J. Symb. Comput.* **11**, 1991, 195–204.