J. Symbolic Computation (2002) **11**, 1–22

Telescoping in the Context of Symbolic Summation in Maple

S.A. ABRAMOV[1†] J.J. CARETTE[2‡] K.O. GEDDES[3§] AND H.Q. LE[3¶]

[1] *Russian Academy of Science, Dorodnicyn Computing Centre, Vavilova st. 40, 119991, Moscow, GSP-1, Russia*

[2] *Computing and Software, McMaster University, Hamilton, L8S 4L8, Canada*

[3] *Symbolic Computation Group, University of Waterloo, Waterloo, N2L 3G1, Canada*

(Received 31 December 2002)

This paper is an exposition of different methods for computing closed forms of definite sums. The focus is on recently-developed results on computing closed forms of definite sums of hypergeometric terms. A design and implementation of a software package which incorporates these methods into the computer algebra system Maple are described in detail.

1. Introduction

In order to compute closed forms of definite sums, one can apply one of at least three methods: the *classical telescoping method*, the *creative telescoping method*, or the *conversion method*. The classical telescoping method is based on the computation of an anti-difference of the input summand T, or on the construction of an additive decomposition of T; the conversion method uses hypergeometric series as an intermediate representation.

The creative telescoping method is principally based on Zeilberger's algorithm (Zeilberger, 1991). This method has been proven to be a very useful tool for computing closed forms of definite sums of hypergeometric terms which occur in many parts of mathematics including combinatorics, probability, number theory, and analysis of algorithms. Regardless of the extensive work on or related to Zeilberger's algorithm (see, for instance,(Wilf and Zeilberger, 1992; Chyzak and Salvy, 1998; Chyzak, 2000)), there still exist many interesting problems arising from the algorithm, and a number of them were not considered or solved in the "classics".

In addition to providing an outline of the three methods, the paper also includes a summary of some recent results (Abramov, 2002a; Abramov and Le, 2002;

† Email: abramov@ccas.ru
‡ Email: carette@mcmaster.ca
§ Email: kogeddes@scg.math.uwaterloo.ca
¶ Email: hqle@scg.math.uwaterloo.ca

0747–7171/90/000000 + 00 $03.00/0

Le, 2001a) which supply a theoretical foundation and algorithms to overcome, or at least alleviate, two key problems of Zeilberger's algorithm: (a) the limitations in the domain of applicability of Zeilberger's algorithm, and (b) the efficiency of the algorithm. The main focus of the paper, however, is on the design of a software package which provides various tools, based on the above-mentioned three methods, for computing closed forms of indefinite and definite sums. For definite sums of hypergeometric terms, the design starts with the module `Telescopers` for computing the minimal Z-pairs of hypergeometric terms (Abramov and Geddes and Le, 2002). This module forms a component of the module `Hypergeometric` (Abramov and Geddes and Le, 2001), a toolbox for working with hypergeometric terms in general, and for computing closed forms of indefinite and definite sums of hypergeometric terms in particular. The module `Hypergeometric`, together with the modules `IndefiniteSum` and `DefiniteSum`, form the main components of the module `SumTools` (Abramov and Carette and Geddes and Le, 2002), a symbolic summation toolbox in Maple (Monagan et al, 2001).

The organization of the paper is as follows. We discuss in Section 2 the classical telescoping method, and show the design of the module `IndefiniteSum` for computing the anti-differences of various classes of summands. The first part of Section 3 is essentially the work in (Abramov and Geddes and Le, 2002). It is devoted to the design of the combination of algorithms for computing the minimal Z-pairs of hypergeometric terms. An implementation based on this design results in the module `Telescopers`. A comparison between this module and other related software packages is also given. The functions in the module `Telescopers` form a component of the module `Hypergeometric` which is the focus of the second part of Section 3. In Section 4 we discuss the conversion method, and show the design of the module `DefiniteSum` for finding closed forms of definite sums. The last section, Section 5, provides the design and functional descriptions of the package `SumTools`. This package encompasses all the modules described in previous sections.

This paper provides a substantial extension to the paper as presented at ICMS 2002 (Abramov and Geddes and Le, 2002). First, the paper puts the work in (Abramov and Geddes and Le, 2002) in the context of a specific method for computing closed forms of definite sums of hypergeometric terms, namely the creative telescoping method. Secondly, the paper includes descriptions of the design and implementation of two well-known methods: the classical telescoping method, and the conversion method, as well as shows the combination of the three methods. The end result is the software package `SumTools`, a symbolic summation toolbox in Maple.

Throughout the paper, K is a field of characteristic zero, $\mathbb{C}$ is the field of complex numbers, $\mathbb{Q}$ is the field of rational numbers, $\mathbb{Z}$ and $\mathbb{N}$ denote the set of integers and nonnegative integers, respectively. The symbols E_n, E_k denote the shift operators w.r.t. n and k, respectively, defined by $E_nT(n,k) = T(n+1,k)$, and $E_kT(n,k) = T(n,k+1)$. Note that both univariate and bivariate functions will be considered.

2. Classical telescoping

For a given function $T(k)$ over K, the problem of *indefinite summation* asks if there exists a function $G(k)$ over K, or over some suitable extension of K, such that

$$(E_k - 1)\,G = T, \tag{2.1}$$

and to compute such a G, provided that it exists. The computed function G is called an *anti-difference* of T. Note that G is unique up to any function $C(k)$ such that $C(k+1) = C(k)$.

Consider the definite sum

$$\sum_{k=a}^{b} T(k), \ a \leq b, \ b-a \in \mathbb{N}. \tag{2.2}$$

If an anti-difference $G(k)$ of the summand $T(k)$ can be computed, then by writting out (2.2) in full, we have

$$\sum_{k=a}^{b} T(k) = \sum_{k=a}^{b} (G(k+1) - G(k)) = G(b+1) - G(a). \tag{2.3}$$

In this case, we have computed a closed form of (2.2) using the classical telescoping method by first computing an anti-difference $G(k)$ of the summand $T(k)$. If either the non-existence within a class of functions of an anti-difference G for the summand T is proven, or it is not known how to compute such a G, then a plausible approach is to apply an algorithm which solves the additive decomposition problem to decompose T in the form $T(k) = (E_k - 1)T_1 + T_2$ where T_2 is simpler than T in some sense. Then the application of the classical telescoping method to $(E_k - 1)T_1$ results in

$$\sum_{k=a}^{b} T(k) = T_1(b+1) - T_1(a) + \sum_{k=a}^{b} T_2(k).$$

2.1. Indefinite sums

There are different algorithms for computing the anti-differences for different classes of summands. Lafon's survey (Lafon, 1983) includes treatments for polynomials, rational functions, hypergeometric terms, and indefinite summation using extensions of function domains. In addition to the above classes, the following methods can also be included in the set of tools for solving the indefinite summation problem:

1 Koepf's extension (Koepf, 1998) of Gosper's algorithm (Gosper, 1977) to j-fold hypergeometric terms;
2 The extension of Gosper's algorithm as described in (Petkovšek and Wilf and Zeilberger, 1996, Chap. 5) to handle sums of hypergeometric terms;
3 The method of accurate summation as presented in (Abramov and Hoeij, 1999) to handle functions whose minimal annihilators can be computed.

2.2. Additive decomposition

For a given function $T(k)$, an algorithm which solves the additive decomposition problem (ADP) constructs two functions $T_1(k)$ and $T_2(k)$ such that

$$T(k) = (E_k - 1)T_1(k) + T_2(k) \tag{2.4}$$

where $T_2(k)$ is "simpler" than $T(k)$ in some sense. The functions $T_1(k)$ and $T_2(k)$ are called the *summable* and the *non-summable* parts of $T(k)$, respectively. It is important that any algorithm which solves the ADP should guarantee that if the input function $T(k)$

is summable, then the computed non-summable part $T_2(k)$ returned from the algorithm should be identically zero.

Let $T(k)$ be a *rational function* of k. Then the ADP for T was solved in (Abramov, 1975) (see also (Abramov, 1995; Paule, 1995; Pirastu and Strehl, 1995)). The characteristic property of the non-summable part $T_2(k)$ is that its denominator has the minimal degree. In this case, one can express the indefinite sum of $T_2(k)$ in terms of the digamma and polygamma functions, and the problem of computing a closed form for the indefinite sum of the input rational function $T(k)$ is solved.

Let $T(k)$ be a *hypergeometric term* in k over K (or a *term* for short). Recall that the characteristic property of a term $T(k)$ is that the ratio $T(k+1)/T(k)$ is a rational function of k over K. This rational function, denoted by $\mathcal{C}_k(T)$, is the *certificate* of $T(k)$. A term $T(n,k)$ in two variables n and k over K has two certificates $\mathcal{C}_n(T) = T(n+1,k)/T(n,k)$ and $\mathcal{C}_k(T) = T(n,k+1)/T(n,k)$. They are named the n-certificate and the k-certificate, respectively. These certificates are rational functions of n and k over K.

DEFINITION 2.1. (Abramov and Petkovšek, 2001b) *Let $R \in K(k)$ be a nonzero rational function. If there exist nonzero polynomials $f_1, f_2, v_1, v_2 \in K[k]$ such that*

(i) $R = F \cdot \dfrac{E_k V}{V}$ *where* $F = \dfrac{f_1}{f_2}$, $V = \dfrac{v_1}{v_2}$, *and* $\gcd(v_1, v_2) = 1$,

(ii) $\gcd\left(f_1, E_k^h f_2\right) = 1$ *for all* $h \in \mathbb{Z}$,

then $F \cdot \dfrac{E_k V}{V}$ *is a* rational normal form (RNF) *of* R.

For every rational function one can construct an RNF (Abramov and Petkovšek, 2001b) which in general is not unique.

As presented in (Abramov and Petkovšek, 2001a; Abramov and Petkovšek, 2002), the algorithm to solve the ADP for a term $T(k)$ constructs two terms $T_1(k)$, $T_2(k)$ such that (2.4) holds, and either T_2 vanishes or $\mathcal{C}_k(T_2)$ has an RNF

$$\frac{f_1}{f_2} \, \frac{E_k(v_1/v_2)}{(v_1/v_2)} \tag{2.5}$$

with v_2 of minimal degree. Any RNF of $\mathcal{C}_k(T_2)$ of the form (2.5) has $v_2 \in K[k]$ of the same minimal degree.

THEOREM 2.1. (Abramov and Petkovšek, 2001a) *Let $T(k)$ be a term and equality (2.4) be valid for some terms $T_1(k), T_2(k)$. Suppose that $T_2(k) \neq 0$. Let (2.5) be an RNF of $\mathcal{C}_k(T_2)$. Then (2.4) is an additive decomposition of $T(k)$ iff for each irreducible p from $K[k]$ such that $p \,|\, v_2$, the following three properties hold:*

$$\mathbf{Pa}: E_k^h p \,|\, v_2 \Rightarrow h = 0, \;\; \mathbf{Pb}: E_k^h p \,|\, f_1 \Rightarrow h < 0, \;\; \mathbf{Pc}: E_k^h p \,|\, f_2 \Rightarrow h > 0. \tag{2.6}$$

When working with terms in two variables n and k over $\mathbb{C}$, we can consider n as a parameter, and hence can construct an additive decomposition w.r.t. k:

$$T(n,k) = (E_k - 1)\, T_1(n,k) + T_2(n,k). \tag{2.7}$$

If (2.5) is an RNF w.r.t. k of $\mathcal{C}_k(T_2)$ with $f_1, f_2, v_1, v_2 \in \mathbb{C}[n,k]$, then for each irreducible factor $p \in \mathbb{C}[n,k]$ of v_2, properties (2.6) hold. Here K is $\mathbb{C}(n)$, and $K(k)$ is $\mathbb{C}(n,k)$.

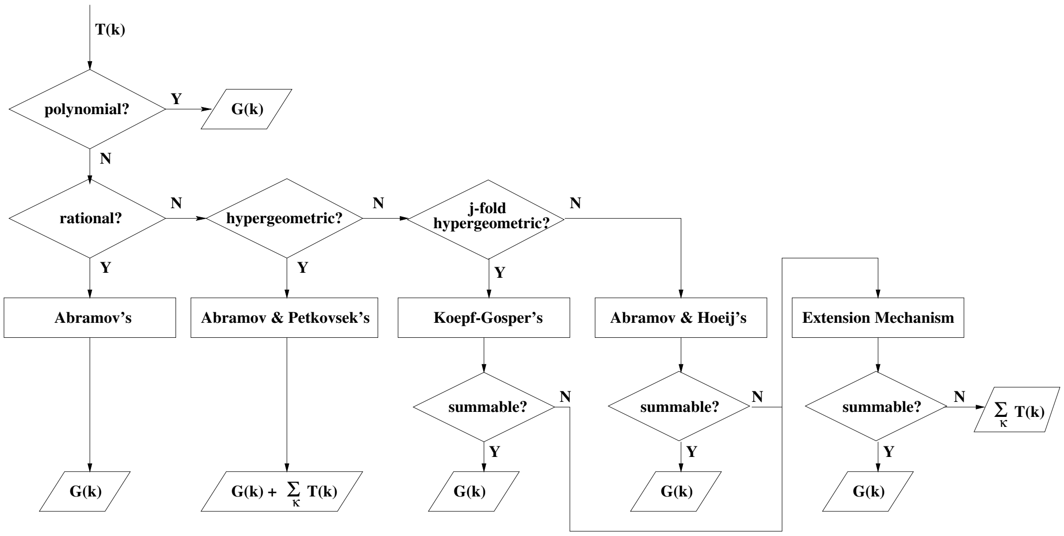
2.3. Implementation

The functions for computing indefinite sums are grouped together into the package `IndefiniteSum`:

`> eval(IndefiniteSum);`
module()
export Polynomial, Rational, Hypergeometric, AccurateSummation, Indefinite,
AddIndefiniteSum, RemoveIndefiniteSum;
description "indefinite sums";
end module

The diagram in Figure 1 provides the classes of summands the package can handle, the corresponding algorithm which handles each class, and the ordering of these algorithms. They include the classes of polynomials, rational functions, hypergeometric terms, j-fold hypergeometric terms, and functions for which their minimal annihilators can be constructed, e.g., d'Alembertian terms. The main function *Indefinite*, which computes an indefinite sum of a given input expression, is a combination of the algorithms handling these classes. 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 pattern-match approach is employed in principle). Currently the computable summands include expressions containing the harmonic function, the logarithmic function, the digamma and polygamma functions, as well as the sine, cosine and exponential functions.

Figure 1. Indefinite sum: a flowchart



EXAMPLE 2.1.

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

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

Since T is a 2-fold term in k, Koepf's extension to Gosper's algorithm is used:

```
> Sum(T,k) = Hypergeometric(T,k);
```

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

EXAMPLE 2.2.

```
> T := k^2/binomial(2*k,k)/(k^2+3*k+2);
```

$$T := \frac{k^2}{(k^2+3\,k+2)\binom{2\,k}{k}}$$

Although the term T is not summable, it is possible to apply the algorithm which solves the ADP to express the indefinite sum of T in terms of the indefinite sum of a simpler term T_2 which is the non-summable part of T:

```
> Sum(T,k) = Hypergeometric(T,k);
```

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

Note that the minimal multiplicative representation of T is

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

EXAMPLE 2.3. *(Abramov and Hoeij, 1999)*

```
> T := 1/5*((1/2+1/2*5^(1/2))^k-(1/2-1/2*5^(1/2))^k)^2;
```

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

The complete factored minimal annihilator for T can be constructed using (Abramov and Zima, 1997), and the application of the method of accurate summation (Abramov and Hoeij, 1999) provides a closed form for the indefinite sum of T:

```
> Sum(T,k) = AccurateSummation(T,k);
```

$$\sum_k \left(\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^k - \left(\frac{1-\sqrt{5}}{2}\right)^k\right)\right)^2 = \frac{1}{5}(-1)^k - \frac{1}{10}\left(1+\sqrt{5}\right)\left(\frac{1-\sqrt{5}}{2}\right)^{2\,k} - \frac{1}{10}\left(1-\sqrt{5}\right)\left(\frac{3+\sqrt{5}}{2}\right)^k$$

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

EXAMPLE 2.4. *Let*

```
> T := 2^(2*k-1)/k/(2*k+1)/binomial(2*k,k)+(k+1)^2*4^(k+1)/(k+2)/(k+3);
```

$$T := \frac{1}{k\,(2\,k+1)} 2^{2\,k-1} \binom{2\,k}{k}^{-1} + \frac{(k+1)^2}{(k+2)(k+3)} 4^{k+1}$$

Since T is a sum of terms, the extension of Gosper's algorithm described in (Petkovšek and Wilf and Zeilberger, 1996, Chap. 5) is used:

```
> Sum(T,k) = Indefinite(T,k);
```

$$\sum_k \left(\frac{1}{k\,(2\,k+1)} 2^{2\,k-1} \binom{2\,k}{k}^{-1} + \frac{(k+1)^2}{(k+2)(k+3)} 4^{k+1} \right) = -\frac{1}{k} 2^{2\,k-1} \binom{2\,k}{k}^{-1} + \frac{(k-1)}{3\,(k+2)} 4^{k+1}$$

EXAMPLE 2.5.

```
> T := sin(k)*cos(k+1)-ln(2*k);
```

$$T := \sin(k)\,\cos(k+1) - \ln(2\,k)$$

Since knowledge about the functions sin, cos, *and* ln *is known via the library extension mechanism, it is possible to compute a closed form for $\sum_k T$:*

```
> Sum(T,k) = Indefinite(T,k);
```

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

3. Creative telescoping

The method of creative telescoping can be useful when the summand T is a function of the summation index k and of a parameter n, i.e., $T = T(n,k)$. If it is not clear how to construct a function $G(n,k)$ such that $G(n,k+1) - G(n,k) = T(n,k)$, then a possible approach is to construct a *telescoper* for T, i.e., an operator

$$L = a_\rho(n) E_n^\rho + \cdots + a_1(n) E_n + a_0(n) \tag{3.1}$$

such that for the function $\tilde{T}(n,k) = LT(n,k)$, such a function $G(n,k)$ can be computed. That is,

$$LT(n,k) = G(n,k+1) - G(n,k). \tag{3.2}$$

This provides an opportunity to find closed forms of definite sums of $\tilde{T}(n,k)$, where the summation bounds, e.g., $u(n), v(n)$, can be functions which depend on n. However, we are computing the sum of $\tilde{T}(n,k)$, instead of $T(n,k)$. For the definite sum of $T(n,k)$, the application of the operator $\sum_{k=u(n)}^{v(n)}$ to both sides of (3.2) results in

$$a_\rho(n) \sum_{k=u(n)}^{v(n)} T(n+\rho,k) + \cdots + a_1(n) \sum_{k=u(n)}^{v(n)} T(n+1,k) + a_0(n) \sum_{k=u(n)}^{v(n)} T(n,k) = H(n) \tag{3.3}$$

where $H(n) = G(n, v(n)+1) - G(n, u(n))$. If $u(n), v(n)$ are polynomials of degree 1 or constants ($\pm\infty$ included), then by adding to $H(n)$ a fixed number of terms obtained from $T(n, k)$, one can transform (3.3) to a recurrence

$$a_\rho(n)f(n+\rho) + \cdots + a_1(n)f(n+1) + a_0(n)f(n) = H^*(n), \tag{3.4}$$

where $f(n) = \sum_{k=u(n)}^{v(n)} T(n, k)$. This recurrence can be used for finding $f(n)$ (if we are able to solve it), or to prove some properties of $f(n)$ by induction on n.

The theory of creative telescoping was initially designed by Zeilberger (Zeilberger, 1991) for the case when the summand $T(n, k)$ is a hypergeometric term. In this case, the operator L of the form (3.1) is an element from $\mathbb{C}[n, E_n]$, and the function $G(n, k)$ such that (3.2) holds is a hypergeometric term. The theory includes an algorithm, called Zeilberger's algorithm or $\mathcal{Z}$ for short, for computing a *Z*-pair (L, G) for T. It was later generalized to holonomic functions (Chyzak and Salvy, 1998; Chyzak, 2000). It should be noted that even for the hypergeometric case, the construction of the *Z*-pairs can be very expensive. It is therefore desirable that problems related to the efficiency of $\mathcal{Z}$ be solved.

3.1. When does Zeilberger's algorithm succeed ?

For a given term $T(n, k)$, if $\mathcal{Z}$ terminates in finite time given T as input, and succeeds in computing a *Z*-pair for T, then we say that "$\mathcal{Z}$ is *applicable* to T", or "There exists a *Z*-pair for T".

Definition 3.1. *A polynomial $\alpha(n, k) \in \mathbb{C}[n, k]$ is* integer-linear *if it has the form $a\,n + b\,k + c$ where a, $b \in \mathbb{Z}$ and $c \in \mathbb{C}$.*

Definition 3.2. (Petkovšek and Wilf and Zeilberger, 1996; Wilf and Zeilberger, 1992) *A term $T(n, k)$ is* proper *if it can be written in the form*

$$P(n, k)\frac{\prod_{i=1}^{l}\Gamma(\alpha_i(n, k))}{\prod_{i=1}^{m}\Gamma(\beta_i(n, k))}u^n v^k, \tag{3.5}$$

where $\alpha_i(n, k), \beta_i(n, k)$ are integer-linear, $l, m \in \mathbb{N}$, $u, v \in \mathbb{C}$.

The question of whether $\mathcal{Z}$ is applicable to term T was not conclusively answered for quite some time, although a sufficient condition was known via the "fundamental theorem" (Petkovšek and Wilf and Zeilberger, 1996; Wilf and Zeilberger, 1992) which states that if $T(n, k)$ is proper, then there exists a *Z*-pair for T. The following theorem provides a necessary and sufficient condition for the termination of $\mathcal{Z}$.

Theorem 3.1. (Abramov, 2002a) *Let $T(n, k)$ be a term in n and k, and (2.7) be an additive decomposition of T w.r.t. k. Let (2.5) be an RNF w.r.t. k of $\mathcal{C}_k(T_2)$ with $v_2 \in \mathbb{C}[n, k]$. Then a Z-pair for $T(n, k)$ exists iff each factor of $v_2(n, k)$ irreducible in $\mathbb{C}[n, k]$ is integer-linear.*

For a given polynomial $f(n, k) \in \mathbb{C}[n, k]$, a decision procedure for the factorability of f into integer-linear polynomials is described in (Abramov and Le, 2000). This procedure does not require a complete factorization of f into irreducible factors.

3.2. Efficient Algorithms to Compute the Minimal $\mathcal{Z}$-pairs

Let $T(n,k)$ be a term. In this section we assume that $\mathcal{Z}$ is proven applicable to T. The algorithm uses an item-by-item examination on the order ρ of the telescopers L. It starts with the value of 0 for ρ and increases ρ until it is successful in finding a Z-pair (L, G) for T. Since the computed telescoper is of minimal possible order, it is called the *minimal* telescoper, and the computed Z-pair is called the *minimal* Z-pair.

Let ρ be the order of the minimal telescoper for T, then $\mathcal{Z}$ simply wastes resources trying to compute a Z-pair where the guessed orders of the telescopers are less than ρ.

For the case where T is also a rational function of n and k (the class of rational functions is a proper subset of the class of terms), there exists a direct algorithm (Le, 2001a) which constructs the minimal Z-pair for T efficiently without using item-by-item examination. For the case where T is a non-rational term, there exists an algorithm (Abramov and Le, 2002) which computes a lower bound μ for the order of the telescopers for T. This helps avoid the time to compute a telescoper of order less than μ.

3.2.1. A direct algorithm for the rational case

Let $T(n,k) \in \mathbb{C}(n,k)$. Consider an additive decomposition of T w.r.t. k of the form (2.7). First one constructs a special form of representation for the non-summable part T_2 as stated in the following theorem.

Theorem 3.2. (Le, 2001a) *Set*

$$T_2 = \sum_{i=1}^{t} \sum_{j=1}^{m_i} \frac{r_{ij}(n)}{(a_i n + b_i k + c_i)^j}, \quad a_i, b_i \in \mathbb{Z}, b_i > 0, \gcd(a_i, b_i) = 1, \ c_i \in \mathbb{C}, \tag{3.6}$$

$r_{ij}(n) \in \mathbb{C}(n)$. *Then $T_2(n,k)$ can be represented in the form*

$$M_1 F_1 + \cdots + M_s F_s, \tag{3.7}$$

where each $M_i \in \mathbb{C}(n)[E_n, E_k, E_k^{-1}]$, each $F_i = 1/(a_i n + b_i k + c_i)^{m_i}$ is such that

$$a_i, b_i \in \mathbb{Z}, \ b_i > 0, \ \gcd(a_i, b_i) = 1, \ c_i \in \mathbb{C}, \ m_i \in \mathbb{N} \setminus \{0\}, \tag{3.8}$$

and for all $i \neq j$, at least one of the following four relations is not satisfied:

$$m_i = m_j, a_i = a_j, b_i = b_j, c_i - c_j \in \mathbb{Z} \setminus \{0\}. \tag{3.9}$$

T_2 can be written in the form (3.6) since $\mathcal{Z}$ is assumed to be applicable to T. Once the representation (3.7) is constructed, one can compute the minimal telescopers for each $M_i F_i \in \mathbb{C}(n,k)$ directly and efficiently. The minimal Z-pair for $T_2(n,k)$, and subsequently for $T(n,k)$, can then be constructed using Least Common Left Multiple computation. This direct algorithm is in general more efficient than the original $\mathcal{Z}$.

3.2.2. Computation of a lower bound for the general hypergeometric case

Let $T(n,k)$ be a non-rational term. Consider an additive decomposition of T w.r.t. k of the form (2.7). Since the minimal telescopers for T and its non-summable part T_2 are the same, the focus is shifted to computing a lower bound for the order of the telescopers for T_2. Let an RNF w.r.t. k of $\mathcal{C}_k(T_2)$ be of the form (2.5). For each irreducible p such that $p \mid v_2$, the three properties **Pa, Pb, Pc** in (2.6) hold.

DEFINITION 3.3. (Abramov and Le, 2002) *Let* $M \in \mathbb{C}[n, E_n]$ *be such that* $MT_2 \neq 0$, *and there exists an RNF* $F' \frac{E_k V'}{V'}$, $V' = \frac{v_1'}{v_2'}$ *of* $\mathcal{C}_k(MT_2)$ *such that each of the irreducible factors of* v_2' *does not have at least one of the three properties* **Pa, Pb, Pc**. *Then* M *is a* crushing operator *for* T_2. *The minimal crushing operator is a crushing operator of minimal order.*

It is simple to show that if L is a telescoper for T_2, then L is also a crushing operator for T_2. Hence, the problem of computing a lower bound for the order of the telescopers for T_2 is reduced to the problem of computing a lower bound for the order of the minimal crushing operator for T_2.

THEOREM 3.3. (Abramov and Le, 2002) *Let* $\mathcal{C}_k(T_2)$ *have an RNF w.r.t.* k $F(E_k V)/V$ *of the form (2.5),* $f_1, f_2, v_1, v_2 \in \mathbb{C}[n, k]$, *and* $D = d_1(n, k)/d_2(n, k)$, $d_1, d_2 \in \mathbb{C}[n, k]$, *be such that* $\mathcal{C}_n(T_2) = D(E_n V)/V$. *Let there exist a crushing operator for* T_2 *of order* ρ. *Then for each integer-linear factor* p *of* v_2, $\deg_k\, p = 1$, *there exists an integer* h *such that*

$$E_k^h\, p \mid E_n v_2 \cdot E_n^2 v_2 \cdots E_n^{\rho} v_2 \cdot d_2 \cdot E_n d_2 \cdots E_n^{\rho-1} d_2. \qquad (3.10)$$

As a consequence, if ρ_p *is the minimal positive value of* ρ *such that there exists an* h *satisfying (3.10), then the order of any crushing operator for* T_2 *is not less than* $\mu = \max_{p \mid v_2} \rho_p$.

Since $\mathcal{Z}$ is assumed to be applicable to the input term $T(n, k)$, it follows from Theorem 3.1 that the polynomial $v_2 \in \mathbb{C}[n, k]$ factors into integer-linear polynomials. By (Abramov and Petkovšek, 2001c), the polynomial $d_2 \in \mathbb{C}[n, k]$ in Theorem 3.3 also factors into integer-linear polynomials. An algorithm, called *LowerBound*, which realizes Theorem 3.3 is described in (Abramov and Le, 2002). Once each of the two polynomials v_2, d_2 is written as a product of integer-linear polynomials (this does require a complete factorization of monic univariate polynomials into irreducible factors, see (Le, 2001a)), the algorithm is reduced to solving bivariate linear diophantine equations, a very inexpensive operation.

3.3. IMPLEMENTATION

3.3.1. CONSTRUCTION OF THE MINIMAL Z-PAIRS

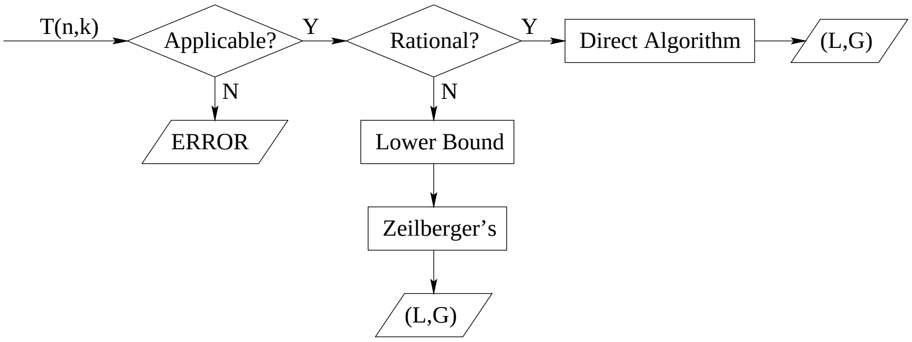
The algorithms presented in this section, when combined with the original $\mathcal{Z}$, provide us with a design of a group of functions for computing the minimal Z-pairs for terms. The diagram in Figure 2 shows a sketch of the design.

In our implementation, this group of functions forms the module `Telescopers`:

`> eval(Telescopers);`
module()
export AdditiveDecomposition, IsZApplicable, ZpairDirect, LowerBound,
Zeilberger, MinimalZpair;
option package;
description "Algorithms for computing the minimal *Z*-pairs for terms";
end module

The exported local variables indicate the functions that are accessible to users. They have the following descriptions:

Figure 2. Algorithms for computing the minimal Z-pairs



1 *AdditiveDecomposition*(T, k) computes an additive decomposition of the term T in k. The output is a list of two elements $[T_1,\ T_2]$ representing the two terms T_1, T_2 in an additive decomposition of T;
2 *IsZApplicable*(T, n, k) returns *true* if $\mathcal{Z}$ is applicable to the term $T(n, k)$, *false* otherwise;
3 *ZpairDirect*(R, n, k, E_n) computes the minimal Z-pair for the rational function $R(n, k)$ using the direct algorithm. The output is a list of two elements $[L, G]$ representing the minimal Z-pair (L, G) for R, or an error message if it is proven that $\mathcal{Z}$ is not applicable to R;
4 *LowerBound*(T, n, k) returns $\mu \in \mathbb{N}$ which is the computed lower bound for the order of the telescopers for the term $T(n, k)$, or an error message if it is proven that $\mathcal{Z}$ is not applicable to T;
5 *Zeilberger*(T, n, k, E_n) returns a list of two elements $[L, G]$ representing the minimal Z-pair (L, G) for the input term $T(n, k)$. This is an implementation of the original $\mathcal{Z}$. Note that an upper bound ρ for the order of the telescopers for $T(n, k)$ needs to be specified in advance (the default value is 6). The function returns an error message if no telescoper of order less than or equal to ρ exists.

The main function of the module is *MinimalZpair*. It has the calling sequence "*MinimalZpair*(T, n, k, E_n)" where T is a term in n and k, and E_n denotes the shift operator w.r.t. n. This function follows the design as sketched in Figure 2. For an input term $T(n, k)$, the execution steps can be described as follows.

1. determine the applicability of $\mathcal{Z}$ to T;
2. if it is proven in step 1 that a Z-pair for T does not exist, return the conclusive error message "There does not exist a Z-pair for T"; Otherwise,
   a. if T is a rational function of n and k, apply the direct algorithm to compute the minimal Z-pair for T;
   b. T is a non-rational term. First compute a lower bound μ for the order of the telescopers for T. Then compute the minimal Z-pair using the original $\mathcal{Z}$ with μ as the starting value for the guessed orders.

For case 2b, let (T_1, T_2) be an additive decomposition of T w.r.t. k. Since the non-summable part T_2 is simpler than T in some sense, we first apply $\mathcal{Z}$ to T_2 to obtain the minimal Z-pair (L, G) for T_2. It can be shown that $(L, LT_1 + G)$ is the minimal Z-pair for the input term T.

EXAMPLE 3.1. *This example is a comparison between the original $\mathcal{Z}$ and the direct algorithm (case 2a of* MinimalZpair*). The test samples are the same as those used in Example 5 in (Le, 2001a). Three set of tests (S_1,S_2,S_3), each of which consists of 20 rational functions of n and k, were randomly generated. Each rational function where the numerator and the denominator are in expanded forms is of the form (3.7). We ran* MinimalZpair, Zeilberger *on these tests, and collected resource requirements*†*. We also enforced a limit of 2,000 seconds on each input rational function in the tests. Note that we only recorded the time and space requirements for the tests that ran under this time limit.*

Table 1 shows the time and space requirements to run the three sets of tests S_1, S_2 and S_3.

Table 1. Time and space requirements for *MinimalZpair* and *Zeilberger*.

	Completed		Timing (seconds)		Memory (kilobytes)	
	MinimalZpair	*Zeilberger*	*MinimalZpair*	*Zeilberger*	*MinimalZpair*	*Zeilberger*
S_1	20	15	12.15	3127.84	54,159	8,095,930
S_2	20	18	12.43	2635.94	54,653	7,873,146
S_3	20	0	959.07	–	3,864,026	–

EXAMPLE 3.2. *Consider the term*

$$T(n, k) = \frac{1}{nk+1}\binom{2\,n}{2\,k}.$$

It takes LowerBound 0.62 *seconds and 3,045 kilobytes to return the error message "Error, (in LowerBound) Zeilberger's algorithm is not applicable". The function recognizes that the polynomial $v_2(n, k)$ in Theorem (3.1) is $(nk+1)$ which does not factor into a product of integer-linear polynomials, and returns the conclusive answer quickly. On the other hand, it takes* Zeilberger *33.95 seconds and 166,396 kilobytes to return the error message "Error, (in Zeilberger) No recurrence of order 6 was found". The function does not know if a* Z-pair *for T exists. It tries to compute one and returns an inconclusive answer. Since there does not exist a* Z-pair *for T, the higher the value of the upper bound for the order of L is set, the more time and memory are wasted.*

EXAMPLE 3.3. *For $b \in \mathbb{N} \setminus \{0\}$, $j \in \{1, 3\}$, let*

$$T_1 = \frac{1}{(nk-1)(n-bk-2)^j(2\,n+k+3)!},\ T_2 = \frac{1}{(n-bk-2)(2\,n+k+3)!}.$$

† All the reported timings were obtained on a 400Mhz SUN SPARC SOLARIS with 1Gb RAM.

Consider the term $T(n,k) = (E_k - 1)\,T_1(n,k) + T_2(n,k)$*. This example is a comparison between* Zeilberger *and case 2b of* MinimalZpair*. The computed lower bound for the order of the telescopers is* b*, while the order of the minimal telescoper is* $b+1$*. Let* $\mu \in \mathbb{N}$ *be the starting value for the guessed order of the telescopers. Recall that the function* Zeilberger *applies* $\mathcal{Z}$ *to the input term* T *with* $\mu = 0$*, while* MinimalZpair *applies* $\mathcal{Z}$ *to the non-summable term* T_2 *in the decomposition (2.7) with* $\mu = b$*. Table 2 shows the time and space requirements. As one can easily notice, as* b *and/or* j *increase, the relative performance of* Zeilberger *(compared to* MinimalZpair*) quickly worsens.*

Table 2. Time and space requirements of *MinimalZpair* and *Zeilberger*.

		Timing (seconds)		Memory (kilobytes)	
j	b	*MinimalZpair*	*Zeilberger*	*MinimalZpair*	*Zeilberger*
	1	6.49	5.35	27,838	24,702
	2	8.34	34.64	33,066	142,889
1	3	11.13	124.53	44,233	535,736
	4	14.46	570.02	56,410	1,882,730
	5	25.79	2999.22	97,506	6,536,309
	1	14.64	16.40	62,566	73,830
	2	17.24	228.59	68,304	770,529
3	3	20.15	1,286.51	78,701	3,074,051
	4	24.08	8,771.08	91,844	10,766,646
	5	38.60	77,663.68	139,823	33,423,168

3.4. A Comparison

There exist different Maple implementations of $\mathcal{Z}$ such as *Zeil* in the EKHAD package (Petkovšek and Wilf and Zeilberger, 1996), *sumrecursion* in the sumtools package (Koepf, 1998), *SummandToRec* in the HYPERG package (Gauthier, 1999). A Mathematica implementation (the function *Zb*) is described in (Paule and Schorn, 1995). Due to the non-existence of a criterion for the applicability of $\mathcal{Z}$ at the time these programs were implemented, the item-by-item examination strategy is employed (these programs are in principle equivalent to the program *Zeilberger* in the module `Telescopers`).

For the case where the input is a rational function, a program such as *Zb* "accepts an input if the irreducible factors of the denominator are integer-linear" (Paule and Schorn, 1995). This is equivalent to the condition that the input be a proper term. By Theorem 3.1, such a program prevents the computation of a *Z*-pair when such a pair exists. Note that we also implemented in the program *MinimalZpair* a direct and efficient algorithm to compute the minimal *Z*-pairs.

For the case where the input $T(n,k)$ is a non-rational term, all the aforementioned programs apply $\mathcal{Z}$ directly to T. On the other hand, *MinimalZpair* first computes a lower bound μ for the order of the telescopers (a fairly low-cost operation), and then applies $\mathcal{Z}$ to the term T_2 in the additive decomposition (2.7) using μ as the starting value for the guessed orders of the telescopers (note that the existence of a *Z*-pair is guaranteed).

The minimal Z-pair for T can then be easily obtained. Experimentation shows that this proposed approach helps expedite the construction of the minimal Z-pairs.

3.4.1. The Maple Package Hypergeometric

The package `Hypergeometric` provides tools for working with terms in general, and for finding closed forms of indefinite and definite sums of terms in particular. It includes the `Telescopers` package.

```
> eval(Hypergeometric);
```

module()
export IsHypergeometricTerm, AreSimilar, PolynomialNormalForm,
RationalCanonicalForm, ConjugateRTerm, MultiplicativeDecomposition,
AdditiveDecomposition, Gosper, ExtendedGosper, Zeilberger,
ZeilbergerRecurrence, IsZApplicable, KoepfGosper, KoepfZeilberger,
ExtendedZeilberger, ZpairDirect, LowerBound, MinimalZpair,
WZMethod, IndefiniteSum, DefiniteSum;
option package;
description "Tools for working with hypergeometric terms";
end module

The module consists of three main components.

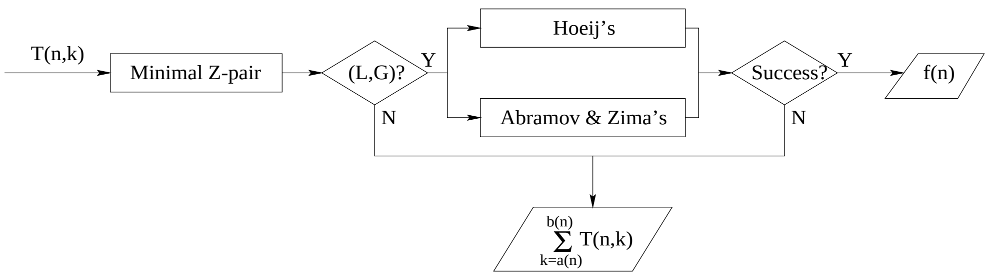
1 The first component includes functions for computing normal forms of rational functions and of terms: *PolynomialNormalForm*, *RationalCanonicalForm*, *MultiplicativeDecomposition*, *AdditiveDecomposition*. See (Abramov and Geddes and Le, 2001) for functional specifications of these functions;
2 The second component includes functions for indefinite and definite sums of terms. For indefinite sums, they are *Gosper*, *KoepfGosper*, *ExtendedGosper*, *AdditiveDecomposition*, and are described in Subsection 2.3. For definite sums, in addition to the functions as described in 3.3.1, the function *ZeilbergerRecurrence* is also included in the set of tools for definite sums of terms. *ZeilbergerRecurrence*$(T, n, k, f, l..u)$ constructs the induced recurrence for the definite sum $f(n) = \sum_{k=l}^{u} T(n, k)$ where T is a term in n and k;
3 The functions in the first two components, when combined with the existing functions of the Maple system, allow one to compute closed forms of indefinite and definite sums of terms. The two functions in the third components are *IndefiniteSum* and *DefiniteSum*. *IndefiniteSum* is described in 3.3.1. *DefiniteSum* has the calling sequence *DefiniteSum*$(T, n, k, l..u)$. The function tries to compute a closed form of the definite sum $f(n) = \sum_{k=l}^{u} T(n, k)$ where $T(n, k)$ is a term in n and k. The four types of definite sums supported are

$$\sum_{k=rn+s}^{un+v} T(n,k),\ \sum_{k=rn+s}^{\infty} T(n,k),\ \sum_{k=-\infty}^{un+v} T(n,k),\ \sum_{k=-\infty}^{\infty} T(n,k),\ r,\, s,\, u,\, v \in \mathbb{Z}.$$

The diagram in Figure 3 shows the combination of algorithms for computing closed forms of definite sums of terms.

The combination of $\mathcal{Z}$ and Petkovšek's algorithm *Hyper* (Petkovšek, 1992) plays an important role in the study of definite sums of terms. For a given term $T(n, k)$, we are

Figure 3. Definite sums of hypergeometric terms



interested in knowing if there exists a *closed form* of $\sum_{k=a(n)}^{b(n)} T(n,k)$. By closed form, we mean the sum can be expressed as a linear combination of a fixed number of terms. First, the application of $\mathcal{Z}$ to $T(n,k)$ yields a linear recurrence operator $L \in \mathbb{C}[n, E_n]$ of the form (3.1) and a term $G(n,k)$ such that relation (3.2) holds. By summing both sides of (3.2) over a specified range of k, we obtain in general an inhomogeneous linear recurrence equation with polynomial coefficients of the form (3.4). As an example, let

$$f(n) = \sum_{k=rn+s}^{un+v} T(n,k), \;\; r,s,u,v \in \mathbb{Z}.$$

Then (3.4) becomes

$$\begin{aligned} \sum_{i=0}^{\rho} a_i(n) f(n+i) &= G(n, un+v+1) - G(n, rn+s) + \\ & \sum_{i=0}^{\rho} a_i(n) \left(\sum_{k=rn+s+ri}^{rn+s-1} T(n+i,k) + \sum_{k=un+v+1}^{un+v+ui} T(n+i,k) \right). \end{aligned} \tag{3.11}$$

Hyper now comes into play (see also (Hoeij, 1999)). If the recurrence (3.4) has a solution $f(n)$ which is a linear combination of a fixed number of terms in n, then *Hyper* will find such a solution, otherwise it returns the message "No such solution exists." It is not surprising that closed forms of many sums with binomial coefficients as summands in (Gould, 1972; Riordan, 1968) can be obtained by first using $\mathcal{Z}$, and then *Hyper*.

EXAMPLE 3.4. *(Riordan, 1968, Ex. 11, p. 164) Let the hypergeometric term*

```
> T := binomial(2*n,2*k)^2;
```

$$T := \binom{2\,n}{2\,k}^2$$

Then

```
> Sum(T,k=0..n)  = DefiniteSum(T,n,k,0..n);
```

$$\sum_{k=0}^{n} \binom{2\,n}{2\,k}^2 = \frac{1}{2}\,\frac{4^n \left(\Gamma\left(2\,n+\frac{1}{2}\right)\sqrt{\pi} + (-1)^n\,\Gamma\left(n+\frac{1}{2}\right)^2\right)}{\sqrt{\pi}\Gamma\left(n+\frac{1}{2}\right)\Gamma\left(n+1\right)}$$

Note that we can *enlarge* the domain of closed forms by including d'Alembertian terms (a d'Alembertian term can be described as nested indefinite sums of hypergeometric terms, or equivalently, as a term which is annihilated by a product of first-order difference operators (see (Abramov and Zima, 1996))). The function *DefiniteSum* can handle this case as well.

4. Definite summation

In addition to the classical and the creative telescoping methods, it is a standard practice to have a front-end, principally based on the pattern-match approach, to recognize certain specific types of definite sums. We also employ another quite powerful method: the conversion method which is a combination of both algorithmic and pattern-match approaches.

4.1. The conversion method

For a given definite sum, the conversion method consists of two steps:

1. conversion of the given definite sum to an expression involving hypergeometric series. See, for example, the hypergeometric series lookup algorithm from (Petkovšek and Wilf and Zeilberger, 1996, Chap. 3),
2. conversion of the hypergeometric series produced in step (1) to standard special and elementary functions. Examples of these standard functions include Bessel functions, Legendre functions, and elliptic integrals. The process is a combination of the algorithmic approach as developed in (Roach, 1996) and the pattern-match approach from a hypergeometric database such as (Prudnikov and Brychkov and Marichev, 1990).

4.2. Implementation

The package `DefiniteSum` consists of functions for computing closed forms of definite sums:

```
> eval(SumTools:-DefiniteSum);
```

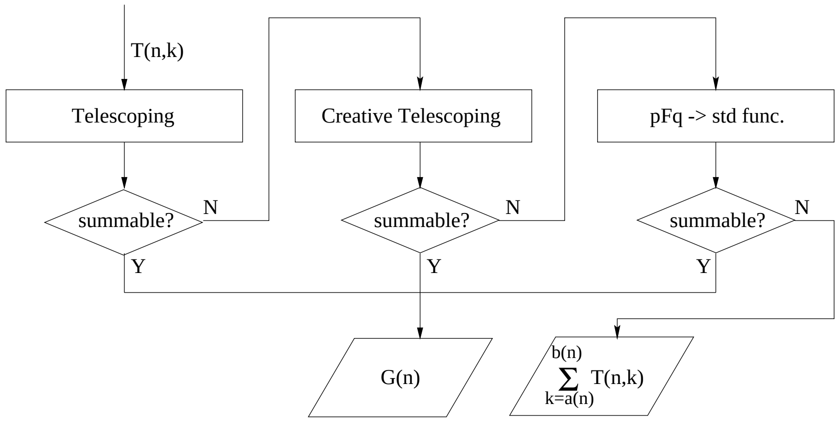
module()
export Telescoping, CreativeTelescoping, pFqToStandardFunctions, Definite;
description "definite sums";
end module

The exported functions *Telescoping*, *CreativeTelescoping*, *pFqToStandardFunctions* compute closed forms of definite sums using the classical telescoping method, the creative telescoping method, and the conversion method, respectively. The main function *Definite* is the combination of these methods with the ordering as shown in Figure 4.

Example 3.4 illustrates the use of the creative telescoping method for computing closed forms of definite sums. We now provide some examples of definite sums whose closed forms are computed using other methods.

Example 4.1. *Let*

Figure 4. Definite sum: a flowchart




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

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

Consider the problem of computing a closed form of $f(n) = \sum_{k=0}^{n} T$*. The front-end recognizes the summand is of Abel's type, and hence a closed form for* $f(n)$ *is computed:*

```
> Sum(T,k=0..n) = Definite(T,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 4.2.

```
> T := binomial(2*n-2*k,n-k)*2^(4*k)*((2*k)*(2*k+1)*binomial(2*k,k))^(-1);
```

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

Since T *is summable w.r.t.* k*, a closed form of* $\sum_{k=1}^{n} T$ *can be computed using the classical telescoping method:*

```
> Sum(T,k=1..n) = Definite(T,k=1..n);
```

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

Example 4.3. *Let*

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

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

In order to compute a closed form of $f(n) = \sum_{k=0}^{\infty} T$*, the function* Definite *uses the con-*

version method by first converting $f(n)$ to $(-\sqrt{\pi}/(\sin(\pi\, n) n)\ {}_2F_1(n, -n; 1/2; z)$, which is then converted to standard functions:

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

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

5. The SumTools package

Computing a closed form of a sum is one of the basic operations in general computer algebra systems such as Maple, Mathematica, Macsyma, MuPAD. We propose in this section a re-design of summation in Maple. The focus is on a smooth integration of independent blocks of code, and on the implementation of recently-developed algorithms. Its design is based on four requirements: *applicability*, *simplicity*, *extensibility*, and *performance.*

5.1. Non-Functional Requirements

1 **Applicability.** The package should cover a wide range of (potentially overlapping) algorithms which handle various classes of summands. If a sum is both (1) present in some form in a standard text covering summation, and (2) can be summed by a published algorithm, then this package should succeed in computing a closed form for that case.

2 **Simplicity.** The output of the main entry points for summation (*DefiniteSummation* and *IndefiniteSummation*) for the package should be as simple as possible. Simplicity is expected to be defined externally to this package, but also to be a concept compatible with summation.

3 **Extensibility.** New algorithms should be easy to incorporate into this package. As well, choosing the ordering in which to insert new algorithms should be objectively decidable. For example, assuming algorithms are known for them, it should be simple to add new code for implementing the many formulas that appear in large collections such as those in (Gould, 1972; Riordan, 1968; Prudnikov and Brychkov and Marichev, 1990).

4 **Performance.** The algorithms for any given class of summands should be the most efficient ones known. Performance benchmarks to verify that each class of summands is summed in the appropriate complexity class need to be built.

Note that a number of these requirements are opposites. For example, simplicity and performance are often incompatible. Thus compromises have to be made to balance out these requirements against one another. These natural-sounding requirements actually have some deep implications for various aspects of the implementation. For instance, extensibility and applicability imply a high level of uniform modularization of the algorithms, as well as a control structure which is quite extensible. In other words, although operationally Figure 1 and Figure 4 describe the current control flow, the actual control structure cannot be so hard-coded. Another point is that there needs to be a precise *design philosophy* carefully documented, so as to guide future developers in how to decide objectively where their new algorithms should be inserted into the existing scheme.

5.2. Functional Description

The package `SumTools` exports three functions and three sub-packages:

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

The three exported functions are *IndefiniteSummation*, *DefiniteSummation*, *Summation*. *IndefiniteSummation*(f, k) computes a closed form of an indefinite sum of f w.r.t. k; *DefiniteSummation*$(f, k = m..n)$ computes a closed form of the definite sum of f over the specified range $m..n$ of the summation index k; *Summation*(f, k) or *Summation*$(f, k = m..n)$ handles both indefinite and definite sums.

The sub-packages `IndefiniteSum`, `Hypergeometric`, and `DefiniteSum` are described in Sections 2, 3, and 4, respectively.

5.3. 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. The integrability of the package as a whole is shown by the dependency of the sub-modules: `Hypergeometric` provides functionalities, while `Tools` provides various auxiliary tools to `IndefiniteSum` and `DefiniteSum`; `Extensibility` provides a library extension mechanism to `IndefiniteSum` which in turn provides functionality to `DefiniteSum`.

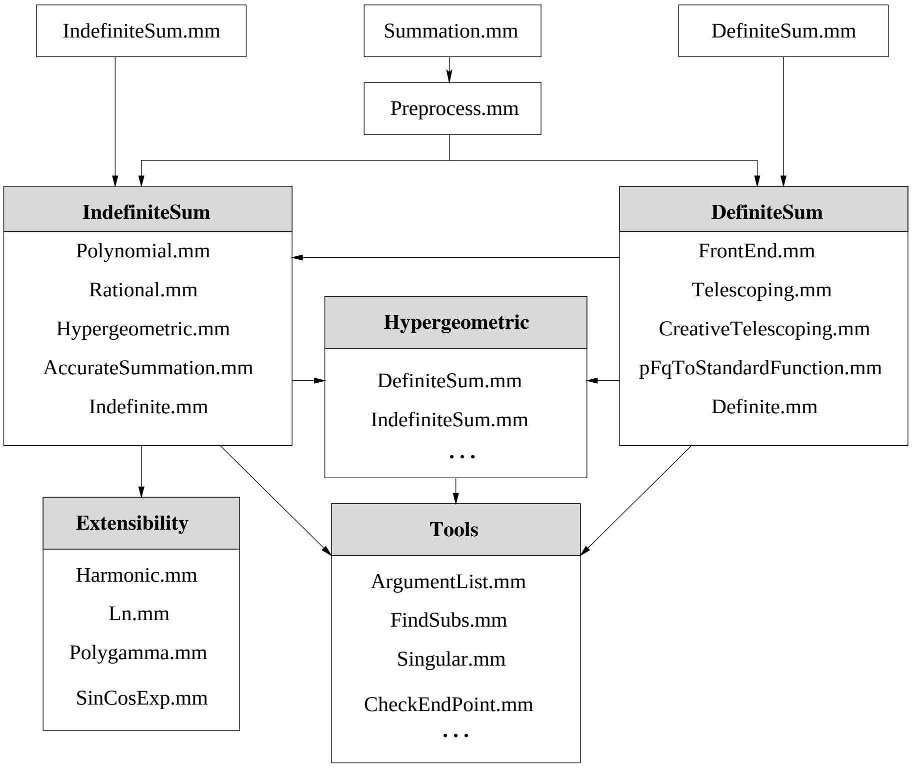
5.4. Testing

The goal is to include as many tests from different sources as possible. We have prepared a number of tests. Many of them are taken from (Gould, 1972; Riordan, 1968). 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 three main methods.

5.5. Remarks on the package

We have presented in this section a design and implementation of 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:-Hypergeometric`, and of the function *convert/StandardFunctions* (used in

Figure 5. SumTools package: code structure and code dependency



the conversion method) are included for the definite case. These algorithms are not implemented or not incorporated in 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 integrability and extensibility. This hopefully will help with the maintenance and future development.

Acknowledgements

This work is partially supported by the French-Russian Lyapunov Institute under grant 98-03, by Natural Sciences and Engineering Research Council of Canada Grant No. RGPIN8967-01, and No. CRD215442-98.

References

S.A. Abramov (1975). Rational component of the solutions of a first-order linear recurrence relation with a rational right-hand side. *USSR Comput. Math. Phys.*, Transl. from Zh. vychisl. mat. mat. fyz. **14**, 1035–1039.

S.A. Abramov (1995). Indefinite sums of rational functions. *Proc. Int. Symp. on Symbolic and Algebraic Computation (ISSAC 1995)*, Montreal, Canada, ACM Press, 303–308.

S.A. Abramov (2002). Applicability of Zeilberger's Algorithm to Hypergeometric Terms. *Proc. Int. Symp. on Symbolic and Algebraic Computation (ISSAC 2002)*, Lille, France, ACM Press, 1–7.
S.A. Abramov (2002). When does Zeilberger's Algorithm Succeed? To appear in *Advances in Applied Mathematics*.
S.A. Abramov, J.J. Carette, K.O. Geddes, H.Q. Le (2002). Symbolic Summation in Maple. *Technical Report CS-2002-32*, School of Computer Science, University of Waterloo, Ontario, Canada.
S.A. Abramov, K.O. Geddes, H.Q. Le (2002). Computer Algebra Library for the Construction of the Minimal Telescopers. In A.M. Cohen, X. Gao, N. Takayama, Eds., *International Congress of Mathematical Software*, World Scientific, 319–329.
S.A. Abramov, K.O. Geddes, H.Q. Le (2001). `HypergeometricSum`: A Maple Package for Finding Closed Forms of Indefinite and Definite Sums of Hypergeometric Type. *Technical Report CS-2001-24*, School of Computer Science, University of Waterloo, Ontario, Canada.
S.A. Abramov, M.v. Hoeij (1999). Integration of solutions of linear functional equations. *Integral Transformations and Special Functions,* **8**, No. 1-2, 3–12.
S.A. Abramov, M. Petkovšek (2001a). Minimal Decomposition of Indefinite Hypergeometric Sums. *Proc. Int. Symp. on Symbolic and Algebraic Computation (ISSAC 2001)*, London, Canada, ACM Press, 7–14.
S.A. Abramov, M. Petkovšek (2001b). Canonical Representations of Hypergeometric Terms. *Proc. Formal Power Series and Algebraic Combinatorics (FPSAC 2001)*, Arizona, U.S.A., 1-10.
S.A. Abramov, M. Petkovšek (2001c). Proof of a Conjecture of Wilf and Zeilberger. *Preprint Series of the Institute of Mathematics, Physics and Mechanics* **39**, 2001, no. 748, Ljubljana, March 9, 2001.
S.A. Abramov, M. Petkovšek (2002). Rational Normal Forms and Minimal Decompositions of Hypergeometric Terms. *J. Symb. Comput.* **33**, No. 5, 521–543.
S.A. Abramov, H.Q. Le (2002). A Lower Bound for the Order of Telescopers for a Hypergeometric Term. *Proc. Formal Power Series and Algebraic Combinatorics (FPSAC 2002)*, Sydney, Australia, CD.
S.A. Abramov, H.Q. Le (2000). Applicability of Zeilberger's Algorithm to Rational Functions. *Proc. Formal Power Series and Algebraic Combinatorics (FPSAC 2000)*, Moscow, Russia, Springer–Verlag LNCS, 91–102.
S.A. Abramov, E.V. Zima (1996). D'Alembertian Solutions of Inhomogeneous Linear Equations (differential, difference, and some other). *Proc. Int. Symp. on Symbolic and Algebraic Computation (ISSAC 1996)*, Zürich, Switzerland, ACM Press, 232–240.
S.A. Abramov, E.V. Zima (1997). Minimal Completely Factorable Annihilators. *Proc. Int. Symp. on Symbolic and Algebraic Computation (ISSAC 1997)*, Maui, Hawaii, USA, ACM Press, 290–297.
F. Chyzak, B. Salvy (1998). Non-commutative Elimination in Ore Algebras Proves Multivariate Identities. *J. Symb. Comput.* **26**, no. 2, 187–227.
F. Chyzak (2000). An Extension of Zeilberger's Fast Algorithm to General Holonomic Functions. *Discrete Mathematics* **217** no. 1-3, 115-134.
B. Gauthier (1999). HYPERG, Maple Package, User's Reference Manual. Version 1.0, `http://www-igm.univ-mlv.fr/~gauthier/HYPERG.html`.
R.W. Gosper, Jr. (1977). Decision procedure for indefinite hypergeometric summation. *Proc. Natl. Acad. Sci. USA* **75**, 40–42.
H.W. Gould (1972). *Combinatorial Identities.* Morgantown, W. Va.
M.v. Hoeij (1999). Finite Singularities and Hypergeometric Solutions of Linear Recurrence Equations. *Journal of Pure and Applied Algebra* **139**, 109-131.
W. Koepf (1998). W. Koepf, *Hypergeometric Summation: An Algorithmic Approach to Summation and Special Function Identities*, Vieweg.
J.C. Lafon (1983). Summation in finite terms. In B. Buchberger, G.E. Collins, R. Loos, Eds., *Computer Algebra: Symbolic and Algebraic Computation*, 71–77. Springer–Verlag, Wien–New York.
H.Q. Le (2001). A Direct Algorithm to Construct Zeilberger's Recurrences for Rational Functions. *Proc. Formal Power Series and Algebraic Combinatorics (FPSAC 2001)*, Arizona, U.S.A., 303–312.
M.B. Monagan, K.O. Geddes, K.M. Heal, G. Labahn, S.M. Vorkoetter, J. McCarron, P. DeMarco (2002). *Maple 8 Introductory Programming Guide.* Waterloo Maple Inc., Waterloo, Ontario, Canada.
P. Paule (1995). Greatest Factorial Factorization and Symbolic Summation. *J. Symb. Comput.* **20**, 235–268.
P. Paule, M. Schorn (1995). A Mathematica Version of Zeilberger's Algorithm for Proving Binomial Coefficient Identities. *J. Symb. Comput.* **20**, 673–698.
M. Petkovšek (1992). Hypergeometric solutions of linear recurrences with polynomial coefficients. *J. Symb. Comput.* **14**, 243–264.
M. Petkovšek, H. Wilf, D. Zeilberger (1996), *A=B*, A.K. Peters, Wellesley, Massachusetts.
R. Pirastu, V. Strehl (1995). Rational summation and Gosper-Petkovšek representation. *J. Symb. Comput.* **20**, 617–635.
A.P. Prudnikov, Yu. Brychkov, O. Marichev (1990). *Integrals and Series, Volume 3: More Special Functions.* Gordon and Breach Science Publishers.
J. Riordan (1968). *Combinatorial identities.* John Wiley & Sons.

K. Roach (1996). Hypergeometric function representations. *Proc. Int. Symp. on Symbolic and Algebraic Computation (ISSAC 1996)*, Zürich, Switzerland, ACM Press, 301–308.

H. Wilf, D. Zeilberger (1992). An Algorithmic Proof Theory for Hypergeometric (ordinary and "q") Multisum/Integral Identities, *Inventiones Mathematicae,* 108, 575–633.

D. Zeilberger (1991). The Method of Creative Telescoping, *J. Symb. Comput.* **11**, 195–204.