Herleitung hypergeometrischer Identitäten

Masterarbeit

April 2017

von

Mustafa Aktan
Matrikelnummer: 33106227

UNIKASSEL
VERSITÄT

Universität Kassel
Fachbereich 10 - Mathematik und Naturwissenschaften
Institut Mathematik

Inhaltsverzeichnis

1. Einführung

In der Mathematik wird man oft mit endlichen oder unendlichen Summen konfrontiert und es wird versucht Identitäten zu beweisen, die Summen beinhalten.

In den letzten Jahrzehnten hat die Computeralgebra, dank der Fortschritte in der Computertechnik, immer mehr an Bedeutung gewonnen. Hierdurch öffneten sich der Mathematik ganz neue Türen, wie zum Beispiel das effizientere Lösen von Summen.

Gosper gelang es den Grundstein für die schnelle Berechnung von geschlossenen Ausdrücken für die unbestimmte Summation zu legen ([GOS78]). Diese fand er im Jahr 1978 während der Mitentwicklung des Computeralgebrasystems Macsyma, welches eines der ersten Algebrasysteme war und im Labor für künstliche Intelligenz des Massachusetts Institute of Technology {MIT} entwickelt wurde. Mathematikern ist es heute möglich Computeralgebrasysteme wie Maple, Mathematica oder Mupad für die Berechnung komplexer algorithmischer Fragestellungen einzusetzen.

Die vorliegende Arbeit beschäftigt sich mit der Frage, wie aus Summen eine geschlossene Form hergeleitet werden kann, mit denen effizienter und schneller gerechnet werden kann und stellt das Thema der Summation von hypergeometrischen Termen vor. Hierfür werden einige effiziente Algorithmen vorgestellt, die zum schnellen Lösen von Summen genutzt werden können. Zunächst betrachten wir die unbestimmte Summe

$$s_n = \sum_{k=m}^{n} a_k,$$

und unser Ziel wird es sein, hierzu eine geschlossene Form zu finden. Diese geschlossene Form stellt eine einfachere Darstellung der Summe s_n als eine Funktion mit der Variablen n dar, welche weder das Summationszeichen $\sum$ noch die Summationslaufvariable k enthält und lediglich aus einer Linearkombination von endlich vielen hypergeometrischen Termen besteht. In dieser Arbeit setzen wir voraus, dass der Summand a_k ein hypergeometrischer Term bezüglich k ist, das heißt, dass der Quotient

$$\frac{a_{k+1}}{a_k} \in \mathbb{Q}(k) \tag{1.1}$$

einer rationalen Funktion bezüglich k entspricht. Mit dem in Kapitel 2 vorgestellten *simpcomb* Algorithmus kann die Bedingung (1.1) überprüft werden. Durch diese Annahme vereinfachen sich die Rechnungen, da wir nur noch Polynome betrachten.

Neben der unbestimmten Summation wird auch die bestimmte Summation behandelt. Bei diesen Summen läuft der Index k über ein festes Intervall aus $\mathbb{Z}$. Es ist durchaus möglich, dass das Intervall $]-\infty, \infty[$ sein kann.

Die ersten Schritte zur algorithmischen Summation gehen zurück bis zu Schwester Celine Fasenmyer. Bereits im Jahr 1945 veröffentlichte sie den Fasenmyer-Algorithmus zur bestimmten Summation, der jedoch lange Zeit aufgrund fehlender Computertechnik ignoriert wurde. Der Algorithmus findet eine holonome Rekursionsgleichung für einen hypergeometrischen Term. Erst viele Jahre später konnten Gosper und Zeilberger mit der Einführung von Computeralgebrasystemen die algorithmische Summation vorantreiben. Gosper entwickelte einen nach ihm benannten Algorithmus für die unbestimmte Summation. Dieser findet eine diskrete Stammfunktion s_k mit

$$a_k = s_{k+1} - s_k,$$

wobei der gegebene Term a_k hypergeometrisch ist. Der Term a_k wird dabei als gospersummierbar bezeichnet, wenn der Gosper-Algorithmus eine zu ihm gehörige diskrete Stammfunktion findet. Wir betrachten die bestimmte Summe in der Form

$$s_n = \sum_{k \in \mathbb{Z}} F(n,k), \tag{1.2}$$

wobei wir auch dieses Mal voraussetzen, dass der Summand $F(n,k)$ hypergeometrisch ist, diesmal sogar bezüglich k und n. Das heißt, dass

$$\frac{F(n+1,k)}{F(n,k)} \in \mathbb{Q}(n,k) \quad \text{und} \quad \frac{F(n,k+1)}{F(n,k)} \in \mathbb{Q}(n,k)$$

jeweils rationale Funktionen sind. Der Gosper-Algorithmus ist nicht in der Lage die Summe s_n aus (1.2) in eine geschlossene Form zu überführen. Daher formulierte Zeilberger im Jahr 1990 einen Algorithmus, welcher eine angepasste Form des Gosper-Algorithmus nutzt ([ZEI91]). Dieser wendet den Gosper-Algorithmus nicht auf $F(n,k)$ an, sondern auf den Ausdruck

$$a_k = F(n,k) + \sum_{j=1}^{J} \sigma_j(n) F(n+j,k)$$

mit rationalen Funktionen $\sigma_j(n)$ $(j = 1, ..., J)$ und einem geeigneten $J \in \mathbb{N}_0$. Der Algorithmus liefert eine holonome Rekursionsgleichung für die Summe s_n, sofern eine existiert. In der vorliegenden Arbeit setzen wir voraus, dass der Summand $F(n,k)$ einen endlichen Träger hat. Das bedeutet, dass zu jedem $n \in \mathbb{Z}$ nur endlich viele $F(n,k) \neq 0$ existieren. Diese Forderung ist notwendig, um eine geschlossene Form für die Summe s_n finden zu können.

Ist die vom Zeilberger-Algorithmus erhaltene Rekursionsgleichung erster Ordnung, so ist es einfach eine geschlossene Form für s_n zu finden. Denn aus der Rekursionsgleichung erster Ordnung kann aus dem hypergeometrischen Quotienten $\frac{s_{n+1}}{s_n}$ die gesuchte Lösung s_n bestimmt werden.

Ein Problem vom Zeilberger-Algorithmus ist, dass er nicht immer die Rekursionsgleichung niedrigster Ordnung findet. Ist die Ordnung der erhaltenen Rekursionsgleichung höher als 1 kann der Zeilberger-Algorithmus keine geschlossene Form finden, sodass ein weiterer Algorithmus, der Petkovsek-Algorithmus, herangezogen werden muss. Dieser findet zu einer gegebenen holonomen Rekursionsgleichung der Ordnung J

$$\sum_{j=0}^{J} P_j(n) s_{n+j} = 0$$

mit polynomiellen Funktionen $P_j(n)$ alle existierenden hypergeometrischen Lösungen und wandelt mithilfe dieser Menge die Summe s_n in eine geschlossene Form um, sofern eine existiert ([KOE14]).

1.1 Ziel und Aufbau der Arbeit

Das Ziel der vorliegenden Arbeit wird sein eine Maplet-Anwendung zur Verfügung zu stellen, welche eine gegebene hypergeometrische Summe in eine geschlossene Form umwandelt. Die Maplet-Anwendung soll zusätzlich eine Implementierung des Petkovsek-Algorithmus beinhalten, welcher in dem aktuellen hsum-Paket nicht enthalten ist. Mit dieser geschlossenen Form soll es möglich sein Summen einfacher und schneller auszurechnen. Um dieses Ziel zu erreichen werden zunächst in Kapitel 2 die Grundlagen vorgestellt, die zum Verständnis der Arbeit notwendig sind. In Kapitel 3 wird der Gosper-Algorithmus eingeführt, welcher eine Einbettung für den in Kapitel 4 vorgestellten
Zeilberger-Algorithmus ist. In diesem Kapitel wird neben dem Zeilberger-Algorithmus auch der Petkovsek-Algorithmus beschrieben, der zum Einsatz kommt, wenn der Zeilberger-Algorithmus keine geschlossene Form liefern kann. Schließlich wird in Kapitel 5 die von mir entwickelte Maplet-Anwendung dokumentiert und einige Identitäten bewiesen, welche jeweils mit und auch ohne die Maplet-Anwendung gelöst werden.

2. Hypergeometrische Funktionen und Vereinfachungen

In diesem Kapitel werden die mathematischen Grundlagen für das weitere Verständnis der Arbeit aufgezeigt, in Anlehnung an [KOE14]. Für die gesamte Arbeit sei $\mathbb{K}$ der Körper der rationalen Zahlen $\mathbb{Q}$ und der transzendenten Körpererweiterungen, also gelte $\mathbb{K} = \mathbb{Q}(a, b, ...)$. Nach [KOE06] führen wir die Polynome ein:

Definition 2.1 *Seien $a_0, ..., a_n \in \mathbb{K}$ und $n \in \mathbb{N}_0$. Dann bezeichnen wir die Abbildung*

$$p : \mathbb{K} \to \mathbb{K}, \quad k \mapsto \sum_{i=0}^{n} a_i k^i$$

als Polynom in k über $\mathbb{K}$. $a_0, ..., a_n$ heißen Koeffizienten von p. Mit $\mathbb{K}[k]$ definieren wir den Polynomring über $\mathbb{K}$ bezüglich k. Für $p \in \mathbb{K}[k]$ sei der Grad des Polynoms p in k mit $\deg(p) := \max\{i \in \mathbb{N}_0 | a_i \neq 0\}$ gegeben. Ist $a_n \neq 0$, so gilt $\deg(p) = n$. Für $p(k) = 0$ sei $\deg(p) = -\infty$. Es sei zudem $\mathbb{K}(k)$ der Körper der rationalen Funktionen bezüglich k über dem Körper $\mathbb{K}$.

Definition 2.2 *Ein Term a_k ist hypergeometrisch über dem Körper $\mathbb{K}$, wenn*

$$\frac{a_{k+1}}{a_k} \in \mathbb{K}(k)$$

gilt. Das heißt, dass der Quotient eine rationale Funktion bezüglich der Variablen k ist.

Definition 2.3 *Für eine beliebige natürliche Zahl k ist die Fakultät $k!$ definiert als*

$$k! := \prod_{i=1}^{k} i = 1 \cdot 2 \cdots (k-1) \cdot k. \tag{2.1}$$

Es gelte zusätzlich $0! := 1$.

Die Klasse der Fakultäten gehören der Klasse der hypergeometrischen Terme an, denn mit $a_k = k!$ gilt für das Termverhältnis nach Definition 2.2:

$$\frac{a_{k+1}}{a_k} = \frac{(k+1)!}{k!} = k + 1 \in \mathbb{K}(k).$$

Definition 2.4 *Für $n \in \mathbb{N}_0$ und $0 \leq k \leq n$ ist der Binomialkoeffizient („n über k“) definiert als*

$$\binom{n}{k} := \frac{n!}{k!(n-k)!}. \tag{2.2}$$

Die Klasse der Binomialkoeffizienten gehören auch der Klasse der hypergeometrischen Terme an, denn mit $a_k = \binom{n}{k}$ und Definition 2.2 gilt:

$$\frac{a_{k+1}}{a_k} = \frac{\binom{n}{k+1}}{\binom{n}{k}} = \frac{n!}{(k+1)!(n-k-1)!}\frac{k!(n-k)!}{n!} = -\frac{k-n}{k+1} \in \mathbb{K}(k).$$

Definition 2.5 *Sei $z, k \in \mathbb{N}_0$. Wir bezeichnen das Pochhammer-Symbol als*

$$(z)_k := \begin{cases} z(z+1)\cdots(z+k-1), & \text{falls } k \in \mathbb{N} \\ 1, & \text{falls } k = 0 \end{cases}. \tag{2.3}$$

Die Klasse der Pochhammer-Symbole sind ebenfalls hypergeometrisch. Dies kann einfach mit $a_k = (z)_k$ und der Definition 2.2 wie folgt gezeigt werden:

$$\frac{a_{k+1}}{a_k} = \frac{(z)_{k+1}}{(z)_k} = \frac{z(z+1)\cdots(z+k)}{z(z+1)\cdots(z+k-1)} = k + z \in \mathbb{K}(k).$$

Definition 2.6 *Die Eulersche Gamma-Funktion ist durch*

$$\Gamma(z) := \int_0^\infty t^{z-1}e^{-t}dt \tag{2.4}$$

definiert.

Das uneigentliche Riemann-Integral (2.4) existiert für jedes $z \in \mathbb{C}$ mit $Re(z) > 0$. Es kann in [VAR13] mehr über die Gamma-Funktion erfahren sowie der Beweis zur Existenz nachgeschlagen werden. Nach Anwendung der partiellen Integration auf $\Gamma(z+1)$ ergibt sich die Gleichheit

$$\Gamma(z+1) := \int_0^\infty t^z e^{-t}dt = z\int_0^\infty t^{z-1}e^{-t}dt = z\Gamma(z).$$

Wir stellen nun folgende Behauptung auf:

$$\text{Für jedes } k \in \mathbb{N}_0 \text{ gilt: } \Gamma(k+1) \stackrel{!}{=} k! \tag{2.5}$$

und beweisen diese durch vollständige Induktion:

$$\begin{aligned} k = 0: &\quad \Gamma(1) = \int_0^\infty e^{-t}dt = 1 = 0! \\ k \to k+1: &\quad \text{Sei } k \in \mathbb{N} \text{ und gelte } \Gamma(k+1) = k!. \text{ Dann folgt} \\ &\quad \Gamma(k+2) = (k+1)\Gamma(k+1) = (k+1)k! = (k+1)!. \end{aligned}$$

Nachdem wir nun die Fakultät (2.1) als Gammafunktion dargestellt haben, werden wir im Folgenden den Binomialkoeffizienten (2.2) und das Pochhammersymbol (2.3) mit Gammatermen darstellen. Hierzu erhalten wir für den Binomialkoeffizienten folgende Darstellung:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{\Gamma(n+1)}{\Gamma(k+1)\Gamma(n-k+1)} \tag{2.6}$$

und für das Pochhammersymbol:

$$(z)_k = z(z+1)\cdots(z+k-1) = \frac{(z+k-1)!}{(z-1)!} = \frac{\Gamma(z+k)}{\Gamma(z)}. \tag{2.7}$$

Wir werden nun einen Algorithmus einführen, der in allen weiteren, in dieser Arbeit aufgeführten Algorithmen Verwendung findet.

Algorithmus 2.7 (simpcomb) *Die Aufgabe des Algorithmus besteht darin einen Term* $\frac{a_{k+1}}{a_k}$ *auf dessen Rationalität zu überprüfen.*
Gegeben: $\frac{a_{k+1}}{a_k}$.
Gesucht: $\frac{a_{k+1}}{a_k} = \frac{u_k}{v_k}$ *mit* u_k, $v_k \in \mathbb{K}[k]$.

1. *Eingabe:* $\frac{a_{k+1}}{a_k}$, *wobei* $a_k \neq 0$ *ein Produkt bestehend aus rationalen Funktionen, Potenzen, Fakultäten, Binomialkoeffizienten, Pochhammersymbolen und Gammafunktionen ist.*

2. *Bilde* $\frac{a_{k+1}}{a_k}$ *und schreibe alle auftretenden Terme in Gammafunktionen nach den Regeln* (2.5), (2.6), (2.7) *um. Es muss darauf geachtet werden, dass negative Argumente vermieden werden. Hierfür stellt die folgende Rechenregel für den Binomialkoeffizienten eine Hilfe:*

$$\binom{n}{k} \to \begin{cases} (-1)^k \dfrac{\Gamma(k-n)}{\Gamma(k+1)\,\Gamma(-n)}, & \text{falls } n \in \mathbb{Z},\ n < 0 \\ 0, & \text{falls } n-k \in \mathbb{Z},\ n-k < 0 \\ \dfrac{\Gamma(n+1)}{\Gamma(n+1)\,\Gamma(n-k+1)}, & \text{sonst} \end{cases}.$$

3. *Vereinfache und kürze die erhaltenen Ausdrücke mit Hilfe der Regel:*

$$\Gamma(k+j) = (k)_j\Gamma(k) = k(k+1)\cdots(k+j-1)\Gamma(k),$$

immer wenn $j \in \mathbb{N}$ *gilt.*

4. *Vereinfache und kürze die Exponententerme des erhaltenen Ausdrucks für* $j \in \mathbb{N}$ *mit folgender Hilfe:*

$$b^{n+j} = b^n b^j.$$

5. *Der Ausdruck* $\frac{a_{k+1}}{a_k}$ *ist genau dann rational, wenn der im letzten Schritt erhaltene Ausdruck* $\frac{u_k}{v_k}$ *rational ist, das bedeutet, dass* u_k, $v_k \in \mathbb{K}[k]$ *gilt.*

6. *Ausgabe:* u_k, v_k.

Zum besseren Verständnis des Algorithmus soll das folgende Beispiel dienen:

Beispiel 2.8 *Gegeben sei die Potenzreihendarstellung der Kosinus-Funktion (siehe zum Beispiel in [RZ16])*

$$\cos(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$$

und gesucht sei die hypergeometrische Darstellung. Zunächst definieren wir den Summanden mit $a_k := (-1)^k \frac{x^{2k}}{(2k)!}$ *und wenden den Algorithmus 2.7 an. Dafür berechnen wir das Verhältnis* $\frac{a_{k+1}}{a_k}$*. Die dabei über den mathematischen Symbolen auftretenden Ziffern geben den durchgeführten Schritt des Algorithmus an:*

$$\begin{aligned}
\frac{a_{k+1}}{a_k} &\overset{1.}{=} \frac{(-1)^{k+1}x^{2k+2}}{(2k+2)!}\frac{(2k)!}{(-1)^k x^{2k}} \overset{2.}{=} \frac{(-1)^{k+1}x^{2k+2}}{\Gamma(2k+3)}\frac{\Gamma(2k+1)}{(-1)^k x^{2k}} \\
&\overset{3.}{=} \frac{(-1)^{k+1}x^{2k+2}}{(2k+2)(2k+1)}\frac{1}{(-1)^k x^{2k}} \\
&\overset{4.}{=} \frac{(-1)x^2}{(2k+2)(2k+1)} \overset{5.}{\in} \mathbb{K}(k) \\
&\overset{6.}{=} -\frac{1}{2}\frac{x^2}{(k+1)(2k+1)}.
\end{aligned}$$

Der Algorithmus liefert eine Ausgabe, nämlich $\frac{u_k}{v_k} = -\frac{1}{2}\frac{x^2}{(k+1)(2k+1)}$*, da* $\frac{u_k}{v_k}$ *eine rationale Funktion in k ist. Für das Computeralgebrasystem Maple (siehe in [MPG15]) gibt es das hsum-Paket, welches zu dem Buch [KOE14] gehört. In diesem Paket existiert eine Maple-Implementierung des Algorithmus 2.7, mit dem das Verhältnis* $\frac{a_{k+1}}{a_k}$ *folgendermaßen berechnet werden kann:*

```
> a_k := (-1)^k x^(2k) / (2k)! :
> simpcomb( subs(k = k+1, a_k) / a_k );
```

$$-\frac{1}{2}\frac{x^2}{(k+1)(2k+1)}$$

Mit dem Befehl „ratio(a_k, k)*;“ bekommt man dasselbe Resultat.*

Definition 2.9 *Die allgemeine hypergeometrische Funktion* ${}_pF_q$ *ist durch*

$${}_pF_q\left(\left.\begin{matrix}\alpha_1, \alpha_2, \ldots, \alpha_p \\ \beta_1, \beta_2, \ldots, \beta_q\end{matrix}\right| x\right) := \sum_{k=0}^{\infty} A_k x^k = \sum_{k=0}^{\infty} \frac{(\alpha_1)_k (\alpha_2)_k \cdots (\alpha_p)_k}{(\beta_1)_k (\beta_2)_k \cdots (\beta_q)_k} \frac{x^k}{k!} \tag{2.8}$$

gegeben, wobei $\alpha_i, \beta_j \in \mathbb{K}$ für $i = 1, ..., p$ und $j = 1, ..., q$ gilt. Die Argumente α_i werden als „obere" Parameter bezeichnet, wohingegen die Argumente β_j die „unteren" Parameter darstellen. Die rechte Seite von (2.8) *wird als allgemeine hypergeometrische Reihe bezeichnet.*

Mit $a_k = A_k x^k$ und (2.3) erhalten wir folgendes Verhältnis:

$$\frac{a_{k+1}}{a_k} = \frac{A_{k+1}x^{k+1}}{A_k x^k} = \frac{(k+\alpha_1)(k+\alpha_2)\cdots(k+\alpha_p)}{(k+\beta_1)(k+\beta_2)\cdots(k+\beta_q)}\frac{x}{k+1} \qquad (k \in \mathbb{N}_0).$$

Diese Darstellung dient dazu künftig eine Erleichterung bei der Bestimmung der oberen und unteren Parameter zu bekommen. Daraus ergibt sich für A_k folgende Rekursionsgleichung erster Ordnung:

$$(k+\beta_1)(k+\beta_2)\cdots(k+\beta_q)(k+1)A_{k+1} - (k+\alpha_1)(k+\alpha_2)\cdots(k+\alpha_p)A_k = 0.$$

Bemerkung 2.10 *Die Funktion ${}_pF_q$ ist wohldefiniert, wenn kein unterer Parameter $\beta_j = 0$ ist, da $(0)_k = 0$ gilt, und kein unterer Parameter eine negative ganze Zahl ist, da $\frac{1}{\Gamma(-z)} = 0$ für $z \in \mathbb{N}_0$ gilt. Des Weiteren ist ${}_pF_q$ eine konvergente Reihe, wenn im Fall a) $p \leq q$ oder im Fall b) $p = q+1$ und $|x| < 1$ gilt.*

Die Fälle *a*) und *b*) sind leicht zu beweisen:
Zu *a*): Sei $p \leq q$. Dann gilt

$$\lim_{k\to\infty}\frac{a_{k+1}}{a_k} = \lim_{k\to\infty}\frac{(k+\alpha_1)(k+\alpha_2)\cdots(k+\alpha_p)}{(k+\beta_1)(k+\beta_2)\cdots(k+\beta_q)}\frac{x}{k+1} = 0.$$

Zu *b*): Seien $p = q+1$ und $|x| < 1$. Dann gilt

$$\lim_{k\to\infty}\frac{a_{k+1}}{a_k} = x.$$

Nach dem Quotientenkonvergenzkriterium konvergiert somit ${}_pF_q$ für die Fälle *a*) und *b*).

Nun führen wir einen Algorithmus ein, welcher eine Hilfe zum Finden von ${}_pF_q$ bietet. Hat man solch eine Darstellung gefunden, kann in hypergeometrischen Datenbanken nach entsprechenden Identitäten gesucht werden. Hierzu wird auf das Kapitel 3 von [KOE14] verwiesen.

Algorithmus 2.11 *Der Algorithmus wandelt eine gegebene hypergeometrische Summe in eine hypergeometrische Funktion ${}_pF_q$ um.*
Gegeben: $\sum_{k\in\mathbb{Z}} a_k$.
Gesucht: $\sum_{k\in\mathbb{Z}} a_k = b_0\ {}_pF_q\left(\begin{matrix}\alpha_1, \alpha_2, \ldots, \alpha_p \\ \beta_1, \beta_2, \ldots, \beta_q\end{matrix}\middle| x\right)$.
Voraussetzung: Die Summanden a_k außerhalb des oberen Summationsbereiches entfallen.

1. *Eingabe:* a_k, *wobei* $a_k \neq 0$ *ein Produkt von Quotienten von rationalen Funktionen, Potenzen, Fakultäten, Binomialkoeffizienten, Pochhammersymbolen und Gammafunktionen ist.*

2. *Falls nötig, verschiebe den Index, sodass die Summe bei einem* $k \geq 0$ *beginnt.*

3. *Bilde* $\frac{a_{k+1}}{a_k}$. *Durch Nutzung des Algorithmus 2.7 werden* u_k, $v_k \in \mathbb{K}[k]$ *generiert, sodass* $\frac{a_{k+1}}{a_k} = \frac{u_k}{v_k}$ *gilt. Falls dies nicht möglich ist, existiert keine allgemeine hypergeometrische Funktion.*

4. *Faktorisiere* u_k, v_k. *Falls keine linearen Faktoren entstehen, bricht der Algorithmus ab und gibt aus, dass keine rationale Faktorisierung gefunden wurde. Bei erfolgreicher Faktorisierung liefert der Algorithmus folgende Darstellung für* u_k *und* v_k*:*

$$\begin{aligned} u_k &= A(k+\alpha_1)(k+\alpha_2)\cdots(k+\alpha_p), \\ v_k &= B(k+\beta_1)(k+\beta_2)\cdots(k+\beta_{q+1}). \end{aligned}$$

5. *Ist einer der unteren Parameter* β_j *eine ganze Zahl, so wählt man die kleinste ganze Zahl* $\beta_j = m$ *und verschiebt alle oberen und unteren Parameter um* $K := -m+1$.

6. *Falls keiner der verschobenen unteren Parameter eine 1 ist, existiert keine hypergeometrische Darstellung.*

7. *Berechne den Anfangswert* $b_0 = a_K$. *Wurde im fünften Schritt keine Verschiebung durchgeführt, setze* $K = 0$. *Weiterhin wird* upper $:= \{\alpha_1, \alpha_2, ..., \alpha_p\}$, lower $:= \{\beta_1, \beta_2, ..., \beta_q, 1\}$ *und* $x := \frac{A}{B}$ *gesetzt.*

8. *Ausgabe:* $b_0 \; {}_pF_q\left(\begin{matrix}\text{upper}\\ \text{lower}\end{matrix}\middle| x\right)$.

Mit den folgenden Beispielen soll der Umgang mit dem Algorithmus 2.11 veranschaulicht werden.

Beispiel 2.12 *In diesem Beispiel werden wir die Kosinus-Funktion aus dem Beispiel 2.8 in eine hypergeometrische Funktion umwandeln. Aus dem bereits bekannten Termverhältnis*

$$\frac{a_{k+1}}{a_k} = -\frac{1}{2}\frac{x^2}{(k+1)(2k+1)}$$

und dem Anfangswert $b_0 = a_0 = 1$ *erhalten wir die hypergeometrische Funktion*

$$\cos(x) = {}_0F_1\left(\begin{matrix}-\\ \frac{1}{2}\end{matrix}\middle| -\frac{x^2}{4}\right).$$

Beispiel 2.13 *Nun werden wir die Summe*

$$\sum_{k=0}^{n} \binom{n}{k}\binom{s}{t+k} \tag{2.9}$$

in eine hypergeometrische Funktion umwandeln. Wir definieren zunächst den Summanden $a_k := \binom{n}{k}\binom{s}{t+k}$ *und berechnen das Verhältnis*

$$\frac{a_{k+1}}{a_k} = \frac{(k-n)(k+t-s)}{(k+t+1)(k+1)}.$$

Mit dem Anfangswert $b_0 = \frac{\Gamma(s+1)}{\Gamma(s+1-t)\Gamma(t+1)} = \binom{s}{t}$ *liefert uns Algorithmus 2.11 für die Summe* (2.9) *folgende Darstellung der hypergeometrischen Funktion:*

$$\sum_{k=0}^{n} \binom{n}{k}\binom{s}{t+k} = b_0 \cdot {}_2F_1\left(\begin{matrix}-n, t-s \\ t+1\end{matrix}\middle| 1\right). \tag{2.10}$$

Im nächsten Schritt kann in einer hypergeometrischen Datenbank nach dieser Darstellung gesucht und so die entsprechende Identität gefunden werden. An dieser Stelle sei gesagt, dass (2.10) *der Chu-Vandermonde-Funktion entspricht, wodurch folgende Identität abgeleitet werden kann:*

$$\begin{aligned} b_0 \cdot {}_2F_1\left(\begin{matrix}-n, t-s \\ t+1\end{matrix}\middle| 1\right) &= \frac{\Gamma(s+1)}{\Gamma(s+1-t)\Gamma(t+1)}\frac{(s+1)_n}{(t+1)_n} \\ &= \frac{\Gamma(s+1)}{\Gamma(s+1-t)\Gamma(t+1)}\frac{\Gamma(n+s+1)\Gamma(t+1)}{\Gamma(s+1)\Gamma(n+t+1)} \\ &= \frac{\Gamma(n+s+1)}{\Gamma(s+1-t)\Gamma(n+t+1)} = \frac{(n+s)!}{(s-t)!(n+t)!} = \binom{n+s}{n+t}. \end{aligned}$$

In Maple kann mit dem Befehl „Sumtohyper“ direkt die Darstellung der hypergeometrischen Funktion (2.10) *wie folgt ausgegeben werden:*

```
> a_k := binomial(n, k) binomial(s, t + k) :
> convert(Sumtohyper(a_k, k), binomial);
```

$$\binom{s}{t}\ \mathrm{Hypergeom}([-n, t-s], [t+1], 1)$$

Im weiteren Verlauf der Arbeit werden wir uns nicht mehr mit der Darstellung der hypergeometrischen Funktion beschäftigen. Vielmehr werden wir für die gegebenen Summen versuchen ohne Hilfe der erwähnten Datenbank eine geschlossene Form zu finden. Hierfür formulieren wir folgende Definitionen:

Definition 2.14 *Ein Term $F(n,k)$, welcher abhängig von n und der Summationsvariablen k ist, wird als hypergeometrisch bezüglich n und k bezeichnet, wenn*

$$\frac{F(n+1,k)}{F(n,k)} \in \mathbb{K}(n,k) \quad \textit{und} \quad \frac{F(n,k+1)}{F(n,k)} \in \mathbb{K}(n,k)$$

gilt.

Definition 2.15 *Eine Summe der Form $s_n = \sum_{k=-\infty}^{\infty} F(n,k)$ liegt in geschlossener Form vor, wenn sie als Linearkombination von endlich vielen hypergeometrischen Termen darstellbar ist, in der das Summationszeichen $\sum_k$ und die Laufvariable k nicht mehr vorkommen.*

Im bisherigen Verlauf der Arbeit haben wir die Summe in die hypergeometrische Notation überführt und anschließend die Möglichkeit genannt, in der hypergeometrischen Datenbank eine Identität nachzuschlagen. Im Folgenden wird der Fasenmyer-Algorithmus vorgestellt, welcher versucht eine holonome Rekursionsgleichung für hypergeometrische Summen zu finden. Hierzu führen wir zunächst folgende Definition ein:

Definition 2.16 *Eine Rekursionsgleichung heißt holonom, wenn sie linear und homogen ist und zudem polynomielle Koeffizienten hat.*

Algorithmus 2.17 (Fasenmyer) *Der Algorithmus sucht eine holonome Rekursionsgleichung für die hypergeometrische Summe s_n.*
Gegeben: $s_n = \sum_{k\in\mathbb{Z}} F(n,k)$.
Gesucht: holonome Rekursionsgleichung für s_n.

1. *Eingabe: $F(n,k) \neq 0$, wobei F ein hypergeometrischer Term nach Definition 2.14 ist. Gebe zusätzlich die obere Grenze J für die Ordnung der gesuchten Rekursionsgleichung vor.*

2. *Finde eine k-freie Rekursionsgleichung für $F(n,k)$, gehe dazu die folgenden Schritte durch:*

   (a) *Setze $I = J$ und mache folgenden Ansatz*

   $$\sum_{i=0}^{I}\sum_{j=0}^{J} a_{ij} F(n+j,k+i) = 0, \tag{2.11}$$

   wobei mit $a_{ij} = a_{ij}(n)$ k-freie Polynome gemeint sind.

   (b) *Setze gegebenes $F(n,k)$ in den Ansatz (2.11) ein.*

   (c) *Dividiere die erhaltene Gleichung durch $F(n,k)$ und vereinfache anschließend durch Anwenden des Algorithmus 2.7, um einen rationalen Ausdruck zu erhalten.*

(d) Bringe den erhaltenen Ausdruck in Normalform, indem die Summanden auf einen gemeinsamen Nenner gebracht werden. Multipliziere im Anschluss daran mit dem gemeinsamen Nenner. Die erhaltene Gleichung kann nun als Polynom in k betrachtet werden.

(e) Vergleiche die Koeffizienten von Potenzen von k, die alle 0 ergeben sollen. Es ergibt sich ein lineares Gleichungssystem für die Unbekannten a_{ij}. Löse a_{ij}, $i = 0, ..., I$, $j = 0, ..., J$.

(f) Falls die Lösung trivial ist, das heißt für alle $a_{ij} = 0$ gilt, dann existiert keine k-freie Rekursionsgleichung. Sonst setze die Lösung in den Ansatz (2.11) ein, vereinfache den resultierenden Ausdruck und erhalte so die k-freie Rekursionsgleichung.

3. Falls Schritt 2 erfolgreich war, dann ersetze in der k-freien Rekursionsgleichung $F(n + j, k + i)$ symbolisch durch s_{n+j}.

4. Falls die erhaltene Rekursionsgleichung trivial ist, bricht der Algorithmus ab. Ansonsten:

5. Ausgabe: Holonome Rekursionsgleichung für s_n.

Beweis: Nach Schritt 2.(c) erhält man einen Ausdruck bestehend aus rationalen Ausdrücken. Dieser Ausdruck kann nur Null ergeben genau dann, wenn der Nenner das Nullpolynom bezüglich k ist. Wenn das Gleichungssystem aus Schritt 2.(e) eine nichttriviale Lösung besitzt, ist es notwendig als auch hinreichend, dass eine nichttriviale k-freie Rekursion mit polynomiellen Koeffizienten existiert. Also ist der 2. Schritt im Algorithmus, dass eine k-freie Rekursionsgleichung gefunden wird, sofern eine existiert, trivial. Summiert man den Ansatz aus Schritt 2.(a) über $k = -\infty, ..., \infty$, so ergibt sich

$$\begin{aligned} 0 &= \sum_{k=-\infty}^{\infty} \sum_{i=0}^{I} \sum_{j=0}^{J} a_{ij}(n) F(n+j, k+i) \\ &= \sum_{i=0}^{I} \sum_{j=0}^{J} a_{ij}(n) \left(\sum_{k=-\infty}^{\infty} F(n+j, k+i) \right) \\ &= \sum_{i=0}^{I} \sum_{j=0}^{J} a_{ij}(n) s_{n+j} \\ &= \sum_{j=0}^{J} \left(\sum_{i=0}^{I} a_{ij}(n) \right) s_{n+j}. \end{aligned}$$

Damit wurde Schritt 3 im Algorithmus gezeigt. Denn falls eine nichttriviale k-freie Rekursionsgleichung für $F(n, k)$ gefunden wird, erhält man eine holonome Rekursionsgleichung für s_n. □

Anhand eines Beispiels soll der Fasenmyer-Algorithmus demonstriert werden:

Beispiel 2.18 *Wir betrachten die Summe*

$$s_n := \sum_{k=-\infty}^{\infty} (-1)^k k \binom{n}{k} \tag{2.12}$$

mit dem hypergeometrischen Summanden $F(n,k) := (-1)^k k\binom{n}{k}$ *und möchten* (2.12) *als Rekursionsgleichung aufschreiben. Es wird* $I = J = 1$ *gewählt und versucht durch diese Wahl geeignete* a_{ij} *zu finden. Mit* (2.11) *ergibt sich folgender Ansatz für* $F(n,k)$*:*

$$a_{00}F(n,k) + a_{01}F(n+1,k) + a_{10}F(n,k+1) + a_{11}F(n+1,k+1) = 0. \tag{2.13}$$

Wir teilen durch $F(n,k)$ *und erhalten*

$$a_{00} + a_{01}\frac{F(n+1,k)}{F(n,k)} + a_{10}\frac{F(n,k+1)}{F(n,k)} + a_{11}\frac{F(n+1,k+1)}{F(n,k)} = 0. \tag{2.14}$$

Nun setzen wir $F(n,k) = (-1)^k k\binom{k}{n}$ *in* (2.14) *ein, vereinfachen und bekommen so*

$$a_{00} + a_{01}\frac{-n-1}{-n-1+k} + a_{10}\frac{-n+k}{k} + a_{11}\frac{-n-1}{k} = 0.$$

Wir bringen nun die Gleichung auf einen gemeinsamen Nenner, multiplizieren im Anschluss daran mit dem Nenner, und sammeln Koeffizienten von gleichen Potenzen in k*. Dies liefert*

$$\begin{aligned}(a_{00}+a_{10})k^2 + (-na_{00} - na_{01} - 2na_{10} - na_{11} - a_{00} - a_{01} - a_{10} - a_{11})k \\ +n^2a_{10} + n^2a_{11} + na_{10} + 2na_{11} + a_{11} = 0.\end{aligned} \tag{2.15}$$

Da die Gleichung (2.15) *ein Polynom in* k *ist, sind die Koeffizienten* a_{ij} k*-frei. Wir erhalten durch Koeffizientenvergleich das lineare Gleichungssystem*

$$\begin{aligned}0 &= a_{00} + a_{10} \\ 0 &= -na_{00} - na_{01} - 2na_{10} - na_{11} - a_{00} - a_{01} - a_{10} - a_{11} \\ 0 &= n^2a_{10} + n^2a_{11} + na_{10} + 2na_{11} + a_{11}\end{aligned}$$

und erhalten mit $a_{00} = 1$ *als Lösung*

$$\begin{aligned}a_{01} &= 0, \\ a_{10} &= -1, \\ a_{11} &= \frac{n}{n+1}.\end{aligned}$$

Wir setzen nun die berechneten a_{ij} in den Ansatz (2.13) *ein und erhalten so die Rekursionsgleichung*

$$F(n,k) - F(n,k+1) + \frac{n}{n+1}F(n+1,k+1) = 0. \tag{2.16}$$

Schließlich ergibt sich mit Schritt 3 des Fasenmyer-Algorithmus für die hypergeometrische Summe s_n die Rekursionsgleichung

$$\frac{n}{n+1}s_{n+1} = 0. \tag{2.17}$$

In Maple kann die gesuchte Rekursionsgleichung mit dem hsum-Paket (2.17) *durch*

```
> F := (-1)^k k binomial(n,k):
> fasenmyer(F,k,s(n),1);
```

$$s(n+1) = 0$$

oder mit

```
> sumrecursion(F,k,s(n),1);
```

$$n(nb_0 - n - b_0)s(n+1) + (n+1)(n-1)s(n) = 0$$

ermittelt werden.

Wir wissen nun, dass eine Rekursionsgleichung einer hypergeometrischen Summe alle Informationen enthält, um eine hypergeometrische Identität zu beweisen beziehungsweise zu erzeugen. In Kapitel 4 werden Algorithmen vorgestellt, die eine geschlossene Form mithilfe einer Rekursionsgleichung finden, wobei wir im nächsten Kapitel erst einmal den Gosper-Algorithmus einführen werden. Bei diesem addieren sich die mittleren Terme durch sogenanntes Teleskopieren jeweils paarweise zu 0, wodurch eine definite Summe entsteht.

3. Gosper-Algorithmus

In diesem Kapitel wird der Gosper-Algorithmus beschrieben, welcher zum Summanden a_k der unbestimmten Summe

$$\sum_{k=m}^{n} a_k \tag{3.1}$$

eine diskrete Stammfunktion s_k berechnet, wobei der gegebene Term a_k hypergeometrisch ist. Die diskrete Stammfunktion s_k, auch Antidifferenz genannt, ergibt sich aus

$$a_k = s_{k+1} - s_k. \tag{3.2}$$

Zunächst einmal werfen wir einen kurzen Blick auf die Integralrechnung und übertragen anschließend das zu zeigende Problem auf die unbestimmte Summation. Hierzu führen wir zunächst den Hauptsatz der Differential- und Integralrechnung ein:

Satz 3.1 *Sei $f : [a, b] \to \mathbb{R}$ eine Funktion mit der Eigenschaft $f \in C([a, b])$ und F eine Stammfunktion zu f auf $[a, b]$ mit $F'(x) = f(x)$. Dann gilt:*

$$\int_a^b f(x)dx = F(b) - F(a).$$

Der Beweis zum Satz kann in [PFO93] nachgelesen werden. Mit dem Satz 3.1 und der Kenntnis der Stammfunktion F kann jedes beliebige bestimmte Integral von f ausgerechnet werden. Übertragen wir nun das Problem auf die unbestimmte Summe (3.1), so möchten wir mithilfe einer Antidifferenz s_k für ein gegebenes a_k, für welches (3.2) gilt, eine geschlossene Form nach Definition 2.15 finden. Der Ausdruck s_k soll dabei hypergeometrisch sein, also soll s_k der Eigenschaft

$$\frac{s_{k+1}}{s_k} \in \mathbb{K}(k) \tag{3.3}$$

genügen. Wurde ein s_k mit der Eigenschaft (3.2) gefunden, so kann (3.1) genau wie bei Integralen an den Grenzen ausgewertet werden. Dies liefert die Betrachtung der Teleskopsumme

$$\sum_{k=m}^{n} a_k = (s_{n+1} - s_n) + (s_n - s_{n-1}) + \cdots + (s_{m+1} - s_m) = s_{n+1} - s_m. \tag{3.4}$$

Definition 3.2 *Ein Term a_k heißt gospersummierbar, wenn zu ihm eine hypergeometrische Antidifferenz s_k existiert.*

Ist s_k ein hypergeometrischer Term, so folgt, dass auch der Summand a_k hypergeometrisch ist. Dies ergibt sich mit (3.2) und (3.3) durch folgende Umformung:

$$\frac{a_{k+1}}{a_k} = \frac{s_{k+2} - s_{k+1}}{s_{k+1} - s_k} = \frac{s_{k+1}}{s_k} \frac{\frac{s_{k+2}}{s_{k+1}} - 1}{\frac{s_{k+1}}{s_k} - 1} = \frac{u_k}{v_k} \in \mathbb{K}(k).$$

Die Polynome $u_k, v_k \in \mathbb{K}[k]$ können dabei mithilfe des Algorithmus 2.7 ermittelt werden.

Im Folgenden sollen die einzelnen Schritte des Gosper-Algorithmus beschrieben und anschließend bewiesen werden. Dabei orientiere ich mich an [KOE14].

Algorithmus 3.3 (Gosper) *Der Algorithmus bestimmt für einen hypergeometrischen Term a_k eine hypergeometrische diskrete Stammfunktion s_k, falls diese existiert.*
Gegeben: a_k.
Gesucht: s_k.

1. *Eingabe: hypergeometrischer Term $a_k \neq 0$.*

2. *Bilde $\frac{a_{k+1}}{a_k}$. Durch Nutzung des Algorithmus 2.7 werden u_k, $v_k \in \mathbb{K}[k]$ generiert, sodass*

$$\frac{a_{k+1}}{a_k} = \frac{p_{k+1}}{p_k} \frac{q_{k+1}}{r_{k+1}} = \frac{u_k}{v_k} \tag{3.5}$$

   gilt. Die Funktionen p_k, q_k und r_k stellen dabei Polynome dar.

3. *Berechne p_k, q_k und $r_k \in \mathbb{K}[k]$ so, dass die Eigenschaft*

$$\gcd(q_k, r_{k+j}) = 1 \qquad \forall j \in \mathbb{N}_0 \tag{3.6}$$

   gilt. Als gcd[1] *wird der größte gemeinsame Teiler bezeichnet. Bei der Berechnung der Polynome wird zunächst $p_k := 1$ definiert und mit den daraus resultierenden q_k und r_k überprüft, ob die Bedingung erfüllt ist. Falls dies nicht der Fall ist, müssen p_k, q_k und r_k mit einer Ersetzungsregel solange aktualisiert werden bis die Bedingung erfüllt ist. Auf die Ersetzungsregel wird in diesem Kapitel noch eingegangen.*

4. *Definiere eine Funktion f_k durch*

$$f_k := \frac{s_{k+1}}{a_{k+1}} \frac{p_{k+1}}{r_{k+1}} \qquad \textit{oder} \tag{3.7}$$

$$s_k = \frac{r_k}{p_k} f_{k-1} a_k. \tag{3.8}$$

[1] **g**reatest **c**ommon **d**ivisor.

Mit $a_k = s_{k+1} - s_k$ ist f_k rational, denn es gilt:

$$f_k = \frac{s_{k+1}}{a_{k+1}}\frac{p_{k+1}}{r_{k+1}} = \frac{s_{k+1}}{s_{k+2} - s_{k+1}}\frac{p_{k+1}}{r_{k+1}} = \frac{1}{\frac{s_{k+2}}{s_{k+1}} - 1}\frac{p_{k+1}}{r_{k+1}}. \tag{3.9}$$

Mithilfe von (3.6) *konnte Gosper zeigen, dass f_k ein Polynom ist ([PWZ96]). Wir erhalten für a_k mit* (3.8) *folgende Darstellung:*

$$a_k = s_{k+1} - s_k = \frac{r_{k+1}}{p_{k+1}} f_k a_{k+1} - \frac{a_k}{p_k} f_{k-1} a_k. \tag{3.10}$$

Wird die Gleichung (3.10) *mit $\frac{p_k}{a_k}$ multipliziert, so erhalten wir eine lineare inhomogene Rekursionsgleichung der Form*

$$p_k = q_{k+1} f_k - r_k f_{k-1}. \tag{3.11}$$

5. *Bestimme die obere Gradschranke für die Lösungsfunktion f_k von* (3.11). *Sei M gegeben als der Grad von f_k. Falls $M < 0$ gilt, dann existiert nach dem Algorithmus keine hypergeometrische Antidifferenz für den Eingabeterm und der Algorithmus bricht ab. Ansonsten setze*

$$f_k = \sum_{l=0}^{M} b_l k^l \tag{3.12}$$

in die Gleichung (3.11) *ein und löse aus dem erhaltenen linearen Gleichungssystem die Unbekannten b_l, $l = 0, ..., M$. Setze diese anschließend in* (3.12) *ein.*

6. *Falls der Gosper-Algorithmus kein f_k findet, existiert keine hypergeometrische Antidifferenz s_k von a_k und der Algorithmus bricht ab. Ansonsten setze f_k in* (3.8) *ein.*

7. *Ausgabe: s_k.*

Mit einem einfachen Beispiel soll der Algorithmus veranschaulicht werden:

Beispiel 3.4 *Wir betrachten die unbestimmte Summe $\sum_{k=m}^{n} a_k$ mit $a_k := \frac{k!}{(k+2)!}$ und wollen eine hypergeometrische Antidifferenz s_k für den hypergeometrischen Term a_k finden. Wir berechnen zunächst das Termverhältnis*

$$\frac{a_{k+1}}{a_k} = \frac{k+1}{k+3},$$

definieren dann $p_k := 1$ und leiten die Funktionen q_k und r_k daraus ab. Es ergibt sich $q_k = k$ und $r_k = k + 2$. Wir können die Polynome p_k, q_k und r_k verwenden, da sie die Eigenschaft (3.6) *im dritten Schritt des Algorithmus erfüllen. Sei nun der Grad von f_k*

mit $M = 1$ vorgegeben. So erhalten wir nach (3.12) *$f_k = b_1 k + b_0$, wobei f_k mit $b_0, b_1 \in \mathbb{R}$ eine lineare Funktion ist. Nun setzen wir f_k in die Gleichung* (3.11) *ein und berechnen die Unbekannten b_0 und b_1:*

$$\begin{aligned} p_k &= q_{k+1} f_k - r_k f_{k-1} \\ 0 &= -b_0 + 2b_1 - 1. \end{aligned}$$

Mit $b_0 = -1$ und $b_1 = 0$ erhalten wir $f_k = -1$. Wir setzen f_k in (3.8) *ein und erhalten die hypergeometrische Antidifferenz*

$$s_k = \frac{r_k}{p_k} f_{k-1} a_k = -\frac{1}{k+1}. \tag{3.13}$$

Mit (3.13) *können wir nun die unbestimmte Summe an den Grenzen wie folgt auswerten*

$$\begin{aligned} \sum_{k=m}^{n} a_k &= s_{n+1} - s_m \\ \sum_{k=m}^{n} \frac{k!}{(k+2)!} &= \frac{n-m+1}{(n+2)(m+1)}. \end{aligned}$$

Da die Summe nicht über alle k durchlaufen wird, kann dadurch für große n und kleine m viel Rechenzeit gespart werden.

Im Folgenden werden wir den Gosper-Algorithmus beweisen. Hierzu führen wir zunächst folgendes Lemma ein:

Lemma 3.5 *Die Funktionen p_k, q_k und r_k in* (3.5) *können so gewählt werden, dass*

$$\gcd(q_k, r_{k+j}) = 1 \qquad \forall j \in \mathbb{N}_0 \tag{3.14}$$

gilt.

Beweis: Sei $p_k := 1$, $q_k := u_{k-1}$ und $r_k := v_{k-1}$ und sei die endliche Dispersionsmenge J durch

$$J := \{j \in \mathbb{N}_0 | \gcd(q_k, r_{k+j}) \neq 1\} \tag{3.15}$$

gegeben. Falls $J = \{\}$ ist, dann ist die Eigenschaft (3.14) erfüllt und der Beweis ist erbracht. Ansonsten existiert ein festes $j \in \mathbb{N}_0$ und $g_k \in \mathbb{K}[k]$, sodass

$$\gcd(q_k, r_{k+j}) = g_k^{(j)} \neq 1 \tag{3.16}$$

gilt. Der gemeinsame Teiler $g_k^{(j)}$ kann für jedes $j \in J$ durch Einführen von neuen Funktionen p'_k, q'_k und r'_k eliminiert werden. Wir definieren

$$p'_k := p_k g_k^{(j)} g_{k-1}^{(j)} \cdots g_{k-j+1}^{(j)}, \quad q'_k := \frac{q_k}{g_k^{(j)}} \text{ und } r'_k := \frac{r_k}{g_{k-j}^{(j)}}. \tag{3.17}$$

Dann gilt:

$$\frac{p'_{k+1}}{p'_k}\frac{q'_{k+1}}{r'_{k+1}} = \frac{p_{k+1}g^{(j)}_{k+1}g^{(j)}_k \cdots g^{(j)}_{k-j+2}}{p_k g^{(j)}_k g^{(j)}_{k-1} \cdots g^{(j)}_{k-j+1}} \frac{q_{k+1}}{g^{(j)}_{k+1}} \frac{g^{(j)}_{k-j+1}}{r_{k+1}} = \frac{p_{k+1}}{p_k}\frac{q_{k+1}}{r_{k+1}}.$$

Es folgt mit (3.16) für $j \in J$:

$$\gcd(q'_k, r'_{k+j}) = \gcd\left(\frac{q_k}{g^{(j)}_k}, \frac{r_{k+j}}{g^{(j)}_k}\right) = 1.$$

Es werden $p_k = p'_k$, $q_k = q'_k$ und $r_k = r'_k$ gesetzt und die Menge J aktualisiert. Die Menge J und die Ersetzungsregel (3.17) werden solange aktualisiert und durchgeführt bis J leer ist. □

Im folgenden Abschnitt der Arbeit werden wir uns mit der Dispersionsmenge J, definiert in (3.15), beschäftigen. Für die Polynome q_k und r_k sei

$$\mathrm{D} := \max\{J\}$$

die Dispersion von q_k und r_k. Zur vollständigen Notation sei $\max\{\emptyset\} = -\infty$. Die im vorigen Beweis beschriebene Dispersionsmenge kann mit verschiedenen Methoden bestimmt werden. Die im folgenden vorgestellte algorithmische Herangehensweise arbeitet mit der rationalen Faktorisierung von Polynomen. Dieser Algorithmus ist effizient und schneller als viele andere Varianten.

Algorithmus 3.6 *Der Algorithmus bestimmt die Dispersionsmenge J von zwei Polynomen q_k und r_k nach* (3.15).

1. *Eingabe:* $q_k, r_k \in \mathbb{K}[k]$.

2. *Faktorisiere q_k und r_k über $\mathbb{K}$ durch*

$$q_k = s^{(1)}_k s^{(2)}_k \cdots s^{(n)}_k,$$
$$r_k = t^{(1)}_k t^{(2)}_k \cdots t^{(m)}_k,$$

   wobei $s^{(i)}_k$ mit $i = 1, ..., n$ und $t^{(j)}_k$ mit $j = 1, ..., m$ irreduzible Faktoren sind.

3. *Setze $J := \emptyset$. Für jedes Paar der Faktoren $s^{(i)}_k$ von q_k und $t^{(j)}_k$ von r_k bestimme die Dispersion D zweier irreduzibler Polynome ($D := \mathrm{primedispersion}(s^{(i)}, t^{(j)}, k)$) durch folgende vier Schritte:*

   (a) *Sei M der Grad von $s^{(i)}_k$ und N der Grad von $t^{(j)}_k$. Falls $M \neq N$ gilt, dann setze $D = \{\}$ und gib D aus.*

(b) Falls M=N gilt, dann sei
a der Koeffizient von k^N des Polynoms $s_k^{(i)}$,
b der Koeffizient von k^{N-1} von $s_k^{(i)}$,
c der Koeffizient von k^N des Polynoms $t_k^{(j)}$ und
d der Koeffizient von k^{N-1} des Polynoms $t_k^{(j)}$.

(c) Falls $l := \frac{bc-ad}{acN} \notin \mathbb{N}_0$, dann setze $D = \{\}$ und gebe D aus.

(d) Falls $cs_k^{(i)} - at_{k+l}^{(j)} = 0$ gilt, dann setze $D = \{j\}$ und gebe D aus.

Aktualisiere $J = J \cup D$.

4. Ausgabe: J.

Beweis: Wenn s_k und t_k nicht den gleichen Grad haben, dann kann keine Dispersion gefunden werden, so erhalten wir den Fall 3.(a). Die Faktoren s_k und t_k sind irreduzible Polynome und wir nehmen an, dass sie eine Dispersion $j \geq 0$ haben. Dann gilt

$$\gcd(s_k, t_{k+j}) = g_k^{(j)} \neq 1,$$

wobei s_k und t_{k+j} ein rationales Vielfaches von $g_k^{(j)}$ sind. Da s_k und t_k irreduzible Polynome sind, haben s_k, t_k und $g_k^{(j)}$ denselben Grad und können wie folgt definiert werden:

$$\begin{aligned} s_k &:= ak^N + bk^{N-1} + \cdots, \\ t_k &:= ck^N + dk^{N-1} + \cdots. \end{aligned}$$

Die beiden Polynome haben eine Dispersion $j \in \mathbb{N}_0$ genau dann, wenn

$$\gcd(s_k, t_{k+j}) \neq 1$$

gilt. Eine äquivalente Forderung ist, dass folgende Gleichheit gelte:

$$\begin{aligned} \frac{c}{a} s_k &= t_{k+j} && (3.18) \\ &= c(k+j)^N + d(k+j)^{N-1} + \cdots. && (3.19) \end{aligned}$$

Mit dem binomischen Lehrsatz können wir Gleichung (3.19) wie folgt umformen:

$$\begin{aligned} t_{k+j} &= c\left(\sum_{i=0}^{N} \binom{N}{i} k^{N-i} j^i\right) + d\left(\sum_{i=0}^{N-1} \binom{N-1}{i} k^{N-1-i} j^i\right) + \cdots \\ &= ck^N + (cjN + d)k^{N-1} + \cdots. \end{aligned}$$

Es ergibt sich somit für Gleichung (3.18)

$$\begin{aligned} \frac{c}{a} s_k &= t_{k+j} \\ ck^N + \frac{bc}{a} k^{N-1} + \cdots &= ck^N + (cjN + d)k^{N-1} + \cdots. \end{aligned}$$

Diese Gleichheit ist erfüllt, falls die Koeffizienten von k^{N-1} auf beiden Seiten der Gleichung gleich sind. Dies ist der Fall, wenn

$$\frac{bc}{a} = (cjN + d) \quad \text{oder} \quad j = \frac{bc - ad}{acN}$$

gelten. Nach der Definition der Dispersionsmenge muss $j \in \mathbb{N}_0$ gelten. Ist jedoch $j \in -\mathbb{N}$, so kann keine Dispersion existieren und der Schritt 3.(c) des Algorithmus ist gezeigt. Ansonsten gilt: $j \in \mathbb{N}_0$, t_{k+j} kann bestimmt werden und die Gleichung (3.18) kann überprüft werden. Die Multiplikation mit a liefert Schritt 3.(d) mit

$$cs_k - t_{k+j} = 0.$$

Der Algorithmus ermittelt die Dispersion j zu jedem Paar irreduzibler Faktoren $s_k^{(i)}$ und $t_k^{(j)}$ und sammelt diese in der Dispersionsmenge J. □

Wir haben f_k durch (3.7) definiert und haben in (3.9) gesehen, dass f_k eine rationale Funktion ist. Nun wollen wir die Behauptung im dritten Schritt des Algorithmus 3.3 „f_k ist ein Polynom“ durch den Beweis des folgenden Theorems zeigen. Beim Theorem sowie beim Beweis orientiere ich mich an [PWZ96].

Theorem 3.7 *Seien p_k, q_k und r_k Polynome in k, sodass die Eigenschaft* (3.14) *erfüllt ist. Wenn f_k, gegeben durch* (3.7)*, eine rationale Funktion in k ist, welche* (3.11) *erfüllt, dann ist f_k ein Polynom in k.*

Beweis: Sei $f_k := \frac{c_k}{d_k}$, wobei c_k und d_k Polynome sind, die nach der Faktorisierung keine gemeinsamen Faktoren besitzen. Es gilt daher

$$\gcd(c_k, d_k) = 1 = \gcd(c_{k-1}, d_{k-1}).$$

Wir schreiben mit $f_k = \frac{c_k}{d_k}$ die Rekursionsgleichung (3.11) um zu

$$q_{k+1}\frac{c_k}{d_k} - r_k\frac{c_{k-1}}{d_{k-1}} = p_k. \tag{3.20}$$

Nun multiplizieren wir die Gleichung (3.20) mit $d_k d_{k-1}$ und erhalten

$$q_{k+1}c_k d_{k-1} - r_k c_{k-1} d_k = d_k d_{k-1} p_k. \tag{3.21}$$

Nun nehmen wir an, dass die Behauptung des Theorems nicht stimmt. Dann ist d_k ein nicht konstantes Polynom mit $\deg(d_k) > 0$. Sei $N \geq 0$ die größte ganze Zahl, sodass

$$\gcd(d_{k-1}, d_{k+N-1}) = g_{k-1} \neq 1$$

ein nicht konstantes irreduzibles Polynom g_{k-1} ergibt. Anders ausgedrückt, N ist die Dispersion von d_{k-1} mit sich selbst.

Da g_{k-N-1} Teiler von d_{k-1} ist, folgt mit (3.21), dass $r_k c_{k-1} d_k$ durch g_{k-N-1} teilbar ist. Es ist aber zu erwähnen, dass g_{k-N-1} nicht c_{k-1} teilt, da es d_{k-1} teilt, denn nach Annahme haben d_{k-1} und c_{k-1} keine gemeinsamen Faktoren.
Zudem ist g_{k-N-1} kein Teiler von d_k, denn ansonsten wäre g_{k-1} ein nicht konstanter gemeinsamer Teiler von d_{k-1} und d_{k+N} mit

$$\gcd(d_{k-1}, d_{k+N}) = g_{k-1} \neq 1.$$

Dies führt jedoch zum Widerspruch zur Wahl von N.
Deswegen ist r_k teilbar durch g_{k-N-1} und daher ist auch r_{k+N} teilbar durch g_{k-1}.

Nach (3.21) ist g_k ein Teiler von $q_{k+1} c_k d_{k-1}$. Nach Annahme teilt g_k nicht c_k. Zudem teilt g_k nicht d_{k-1}. Denn ansonsten wäre g_{k-1} ein nicht konstanter gemeinsamer Faktor von d_{k-2} und d_{k+N-1} mit

$$\gcd(d_{k-2}, d_{k+N+1}) = g_{k-1} \neq 1,$$

was erneut zum Widerspruch zur Wahl von N führt.
Es folgt daraus, dass q_{k+1} teilbar durch g_k ist beziehungsweise q_k teilbar durch g_{k-1} ist. Aber dann wäre g_{k-1} ein nicht konstanter gemeinsamer Teiler von q_k und r_{k+N}. Dies ist ein Widerspruch zu (3.14). Dieser Widerspruch zeigt, dass d_k konstant ist und somit f_k ein Polynom. □

Im Folgenden wollen wir den fünften Schritt des Gosper-Algorithmus beweisen und so den Beweis des Algorithmus abschließen. Dieser Schritt des Algorithmus dient dazu eine obere Gradschranke für das Polynom f_k zu suchen. Nachdem eine obere Gradschranke gefunden wurde, kann f_k als allgemeines Polynom vom berechneten Grad in die Rekursionsgleichung (3.11) eingesetzt werden. Im Anschluss daran kann durch Koeffizientenvergleich f_k berechnet werden. Ist ein f_k nicht zu finden, so kann schlussgefolgert werden, dass der Eingabeterm nicht gospersummierbar ist. Die Berechnung der oberen Gradschranke von f_k erfolgt durch das folgende Lemma:

Lemma 3.8 *Eine obere Gradschranke für den Grad des Polynoms f_k ist bestimmt durch den folgenden Algorithmus:*

1. *Falls* $\deg(q_{k+1} + r_k) \leq \deg(q_{k+1} - r_k)$, *dann gilt*

$$\deg(f_k) = \deg(p_k) - \deg(q_{k+1} - r_k).$$

2. *Falls* $n := \deg(q_{k+1} + r_k) > \deg(q_{k+1} - r_k)$, *dann sei a der Koeffizient von k^n des Polynoms $q_{k+1} + r_k$, und b der Koeffizient von k^{n-1} des Polynoms $q_{k+1} - r_k$.*
Eine Fallunterscheidung wird betrachtet:

(a) Falls $-\frac{2b}{a} \in -\mathbb{N}$*, dann gilt*

$$\deg(f_k) = \deg(p_k) - n + 1.$$

(b) Falls $-\frac{2b}{a} \in \mathbb{N}_0$*, dann gilt*

$$\deg(f_k) \leq \max\left\{-\frac{2b}{a}, \deg(p_k) - n + 1\right\}.$$

Beweis: Wir betrachten die Rekursionsgleichung (3.11) und schreiben diese wie folgt um

$$\begin{aligned} p_k &= q_{k+1}f_k - r_k f_{k-1} \\ &= q_{k+1}\frac{f_k + f_{k-1} + f_k - f_{k-1}}{2} - r_k\frac{f_k + f_{k-1} - f_k + f_{k-1}}{2} \\ &= q_{k+1}\frac{f_k + f_{k-1}}{2} - r_k\frac{f_k + f_{k-1}}{2} + q_{k+1}\frac{f_k - f_{k-1}}{2} + r_k\frac{f_k - f_{k-1}}{2} \\ &= (q_{k+1} - r_k)\frac{f_k + f_{k-1}}{2} + (q_{k+1} + r_k)\frac{f_k - f_{k-1}}{2}. \end{aligned} \tag{3.22}$$

Sei f_k verschieden vom Nullpolynom, dann gilt folgende Gradbeziehung

$$\deg(f_k - f_{k-1}) = \deg(f_k + f_{k-1}) - 1. \tag{3.23}$$

Dies ist damit zu begründen, dass der Koeffizient von k mit dem höchsten Exponenten bei $f_k - f_{k-1}$ verschwindet, wohingegen er bei $f_k + f_{k-1}$ bestehen bleibt. Sei $\deg(0) := -1$, dann gilt (3.23) auch für von 0 verschiedene, konstante f_k. Falls
$\deg(q_{k+1} + r_k) \leq \deg(q_{k+1} - r_k)$ gilt, dann hat der zweite Summand von (3.22) einen kleineren Grad als der erste Summand, was die erste Aussage liefert. Gelte
$\deg(q_{k+1} + r_k) > \deg(q_{k+1} - r_k)$ und sei m der Grad von f_k und $f_k = ck^m + \cdots$. Dann erhalten wir für die Gleichung (3.22):

$$\begin{aligned} p_k = (bk^{n-1} + \cdots)&\left(\frac{\left(ck^m + \cdots\right) + \left(c(k-1)^m + \cdots\right)}{2}\right) + \\ (ak^n + bk^{n-1} + \cdots)&\left(\frac{\left(ck^m + \cdots\right) - \left(c(k-1)^m + \cdots\right)}{2}\right) \\ \Leftrightarrow p_k = (bk^{n-1} + \cdots)&\left(\frac{\left(ck^m + \cdots\right) + \left((ck^m + cmk^{n-1}\cdots) + \cdots\right)}{2}\right) + \\ (ak^n + bk^{n-1} + \cdots)&\left(\frac{\left(ck^m + \cdots\right) - \left(ck^m + cmk^{m-1} + \cdots\right)}{2}\right). \end{aligned}$$

Wir klammern ck^{m+n-1} aus und erhalten folgende Darstellung:

$$\left(b + \frac{am}{2}\right) ck^{m+n-1} + \text{ Terme niedrigeren Grades.}$$

Der Koeffizient von k^{n+m-1} ist gleich 0, wenn $m = -\frac{2b}{a}$ ist. Ist $m \in -\mathbb{N}$, so erhalten wir den Fall 2.(a), sonst Fall 2.(b).

□

Der Gosper-Algorithmus wurde nun vorgestellt und bewiesen. Jetzt werden Beispielaufgaben mit dem Gosper-Algorithmus behandelt, wodurch die Nutzung des Algorithmus verdeutlicht werden soll. Die nächsten beiden Beispiele 3.9 und 3.10 sind aus [KOE14] entnommen.

Beispiel 3.9 (Polynome) *Polynome sind gospersummierbar, da jedes Polynom a_k eine polynomielle Antidifferenz s_k besitzt. Zunächst einmal bilden wir $\frac{a_{k+1}}{a_k}$ und wählen $p_k := 1$, $q_k := a_k$ und $r_k := a_{k-1}$. Da die Bedingung* (3.14) *von Lemma 3.5 für $j = 1$ nicht erfüllt ist, ist die 1 in der Dispersionsmenge enthalten. Mit der im Beweis von Lemma 3.5 beschriebenen Ersetzungsregel aktualisieren wir p_k, q_k und r_k durch $p_k = a_k$, $q_k = 1$ und $r_k = 1$. Durch diese Wahl der Funktionen ist nun die Bedingung* (3.14) *erfüllt. Es ergibt sich für die Rekursionsgleichung*

$$a_k = f_k - f_{k-1}.$$

Nach (3.7) *gilt $f_k = s_{k+1}$. Da $\deg(q_{k+1} + r_k) = 0$ und $\deg(q_{k+1} - r_k) = -1$ gilt, betrachten wir Fall 2 von Lemma 3.8. Mit $a = 2$ und $b = 0$ ergibt sich mit* 2.(b) *die obere Gradschranke $\deg(f_k) = \deg(a_k) + 1$. Die Koeffizienten von s_k können durch das Lösen des linearen Gleichungssystem mit $\deg(a_k) + 1$ Unbekannten bestimmt werden.*

Beispiel 3.10 (Rationale Funktionen) *Wir betrachten den Term $a_k = \frac{1}{k}$ und suchen hierfür eine hypergeometrische Antidifferenz, sofern diese existiert. Wir wählen $p_k := 1$, $q_k := k - 1$ und $r_k := k$. Durch diese Wahl ist die Bedingung* (3.14) *erfüllt. Nach Lemma 3.8 erhalten wir für die Gradschranke von f_k die 0. Mit $f_k = b_0$ erhalten wir die Rekursionsgleichung*

$$1 = b_0 k - b_0 k = 0.$$

Daraus können wir schlussfolgern, dass keine Lösung existiert und somit nicht jede rationale Funktion eine hypergeometrische Antidifferenz besitzt. Mit diesem Beispiel wurde zusätzlich gezeigt, dass die harmonischen Zahlen

$$\sum_{j=1}^{k} \frac{1}{j}$$

keinen hypergeometrischen Term bilden. Eine rationale Funktion ist gospersummierbar, wenn sie eine rationale Antidifferenz besitzt. Tritt dieser Fall ein, so kann s_k mit dem Gosper-Algorithmus bestimmt werden. Als nächstes Beispiel betrachten wir die rationale Funktion $a_k = \frac{1}{k(k+1)}$. Es gilt

$$\frac{a_{k+1}}{a_k} = \frac{k}{k+2}.$$

Nach Lemma 3.5 können wir $p_k := 1$, $q_k := k-1$ und $r_k := k+1$ wählen und nach Lemma 3.8 ergibt sich mit dem Fall 2.(b) für die obere Gradschranke $\deg(f_k) = 1$. *Wir setzen $f_k = b_1 k + b_0$ und erhalten die Rekursionsgleichung*

$$1 = k(b_1 k + b_0) - (k+1)(b_1(k-1) + b_0) = b_1 - b_0.$$

Da wir den Grad von f_k so gering wie möglich halten wollen, setzen wir $b_1 = 0$ und erhalten so $f_k = -1$. Die Antidifferenz s_k ist somit nach (3.8) *durch*

$$s_k = -\frac{1}{k}$$

gegeben.

Beispiel 3.11 *In diesem Beispiel werden wir überprüfen, ob der Summand $a_k := \binom{n}{k}\left(\frac{n}{2} - k\right)$ der Summe $\sum_{k=0}^{m}\binom{n}{k}\left(\frac{n}{2} - k\right)$ gospersummierbar ist. Wir bilden*

$$\frac{a_{k+1}}{a_k} = -\frac{(-n+2k+2)(-n+k)}{(k+1)(-n+2k)}$$

und wählen $p_k := 1$, $q_k := -(-n+2k)(-n+k-1)$ und $r_k := k(-n+2k-2)$, welche der Bedingung (3.14) *nicht genügen. Daher aktualisieren wir die Funktionen mit der Ersetzungsregel zu*

$$\begin{aligned} p_k &= -n + 2k, \\ q_k &= n + 1 - k, \\ r_k &= k. \end{aligned}$$

Mit dieser Wahl der Funktionen ist die Dispersionsmenge leer und die Bedingung (3.14) *erfüllt. Für die obere Gradschranke von f_k liefert Fall 1 in Lemma 3.8* $\deg(f_k) = 0$. *Mit $f_k = b_0$ erhalten wir die Rekursionsgleichung*

$$-n + 2k = (n-k)b_0 - kb_0$$

und daraus ergibt sich mit $b_0 = -1$ die Funktion $f_k = -1$. Nachdem wir f_k in die Gleichung (3.8) *einsetzen, bekommen wir die gesuchte Antidifferenz*

$$s_k = \frac{1}{2}\binom{n}{k}k.$$

Da das Intervall der gegebenen Summe durch $[0, m]$ *gegeben war, erhalten wir mit*

$$\begin{aligned} s_0 &= 0 \quad \textit{und} \\ s_{m+1} &= \frac{m+1}{2}\binom{n}{m+1} \end{aligned}$$

die Identität

$$\sum_{k=0}^{m}\binom{n}{k}\left(\frac{n}{2}-k\right) = \frac{m+1}{2}\binom{n}{m+1}.$$

Die Aufgabenstellung des nächsten Beispiels ist aus [KAP05] entnommen und zeigt die Mächtigkeit des Gosper-Algorithmus bei einer komplexeren Aufgabe:

Beispiel 3.12 *Wir betrachten den Term* $a_k := \frac{k^3}{\prod_{j=1}^{k}(j^3+1)}$ *und wollen eine dazugehörige Antidifferenz finden. Es gilt*

$$\frac{a_{k+1}}{a_k} = \frac{\left[\prod_{j=1}^{k}(j^3+1)\right](k+1)^3}{\left[\prod_{j=1}^{k+1}(j^3+1)\right]k^3}.$$

Nach Vereinfachung erhalten wir

$$\frac{a_{k+1}}{a_k} = \frac{(k+1)^3}{k^3}\frac{1}{(k+1)^3+1}.$$

Nun setzen wir $p_k := k^3$, $q_k := 1$ *und* $r_k := k^3+1$, *womit die Bedingung* (3.14) *erfüllt ist. Nach Lemma 3.8 erhalten wir nach Fall 1 als Gradschranke* $\deg(f_k) = 0$. *Mit* $f_k = b_0$ *und der Rekursionsgleichung*

$$\begin{aligned} k^3 &= b_0 - (k^3+1)b_0 \\ k^3 &= -\,k^3 b_0 \end{aligned}$$

ergibt sich für $f_k = -1$ *die Antidifferenz*

$$-\frac{k^3+1}{\prod_{j=1}^{k}(j^3+1)} = -\frac{1}{\prod_{j=1}^{k-1}(j^3+1)}.$$

Im letzten Beispiel dieses Kapitels ist die Aufgabenstellung aus dem Buch [BCL82] entnommen.

Beispiel 3.13 *Wir wollen die Identität*

$$\sum_{k=0}^{n} kx^k = \frac{(nx-n-1)x^{n+1}-x}{(x-1)^2}$$

mit dem Gosper-Algorithmus beweisen. Hierzu definieren wir $a_k := kx^k$ *und berechnen das Verhältnis*

$$\frac{a_{k+1}}{a_k} = \frac{(k+1)x}{k}.$$

Nun setzen wir $p_k := 1$, $q_k := kx$ *und* $r_k := k-1$, *womit die Bedingung* (3.14) *nicht erfüllt ist, denn die Dispersionsmenge ist mit* $J = \{1\}$ *nicht leer. Mit der Ersetzungsregel aus dem Beweis von Lemma 3.5 aktualisieren wir* p_k, q_k *und* r_k *durch* $p_k = k$, $q_k = x$ *und* $r_k = 1$. *Durch diese Wahl ist die Bedingung* (3.14) *nun erfüllt. Mit Lemma 3.8 bekommen wir als obere Gradschranke mithilfe von Fall 1* $\deg(f_k) = 1$. *Wir setzen* $f_k = b_1k + b_0$ *in die Rekursionsgleichung*

$$k = x(b_1k + b_0) - (b_1(k-1) + b_0)$$

ein und leiten daraus durch Koeffizientenvergleich das lineare Gleichungssystem

$$\begin{aligned} 0 &= b_1x - b_1 - 1 \\ 0 &= b_0x + b_1 - b_0 \end{aligned}$$

ab. Wir lösen nach den Unbekannten auf und erhalten mit $b_0 = -\frac{1}{(x-1)^2}$ *und* $b_1 = \frac{1}{x-1}$ *die Funktion*

$$f_k = \frac{k}{x-1} - \frac{1}{(x-1)^2}.$$

Daraus ergibt sich die gesuchte Antidifferenz

$$s_k = \frac{((k-1)x-k)x^k}{(x-1)^2}.$$

Mit den Randwerten

$$\begin{aligned} s_0 &= -\frac{x}{(x-1)^2} \qquad \textit{und} \\ s_{n+1} &= \frac{(nx-n-1)x^{n+1}}{(x-1)^2} \end{aligned}$$

ist die Identität

$$\sum_{k=0}^{n} kx^k = \frac{(nx-n-1)x^{n+1} - x}{(x-1)^2}$$

damit bewiesen worden.

Nachdem wir nun den Gosper-Algorithmus im Detail kennengelernt haben, können wir uns dem Zeilberger-Algorithmus widmen. Durch die Einbettung des Gosper-Algorithmus in den Zeilberger-Algorithmus ist dieser von großer Bedeutung und für das Verständnis des Zeilberger-Algorithmus wichtig.

4. Zeilberger-Algorithmus und Petkovsek-Algorithmus

Im zweiten Kapitel haben wir den Fasenmyer-Algorithmus vorgestellt. Dieser findet eine Rekursionsgleichung für eine Summe der Form

$$s_n = \sum_{k=-\infty}^{\infty} F(n,k), \tag{4.1}$$

wobei $F(n,k)$ ein hypergeometrischer Term nach Definition 2.14 ist. Der Fasenmyer-Algorithmus hat einige Nachteile, die wichtigsten sollen kurz angerissen werden. Der Algorithmus liefert keinerlei Aussage zur Existenz einer Lösung. Daher weiß man vor einer bestimmten Eingabe nicht, ob eine Lösung gefunden werden kann. Zudem hat die Wahl der beiden Summationsgrenzen willkürlichen Charakter. Es ist nicht klar, wie I und J gewählt werden sollen. Liefert die Wahl von J keine Lösung, so kann durch Erhöhung der Ordnung der Rekursion die Lösbarkeit erreicht werden. Ein Problem ist jedoch, dass die Information zu der benötigten Ordnung nicht bekannt ist. Des Weiteren ist nicht sicher, dass der Algorithmus die holonome Rekursionsgleichung niedrigster Ordnung findet. Es kann nur gewährleistet werden, dass eine Rekursion gefunden wird, sofern eine existiert. Das größte Problem des Fasenmyer-Algorithmus ist die Komplexität, denn bei jedem Durchgang muss ein lineares Gleichungssystem der Ordnung $(I+1)(J+1)$ gelöst werden.

Im nächsten Kapitel haben wir mit dem Gosper-Algorithmus für einen gegebenen hypergeometrischen Term $F(k)$ eine diskrete Stammfunktion $G(k)$ gesucht, welche

$$F(k) = G(k+1) - G(k)$$

erfüllt. Falls solch ein $G(k)$ existiert, so ist der Term $F(k)$ unbestimmt gospersummierbar. Es gibt jedoch viele Summanden, die nicht unbestimmt, sondern bestimmt summierbar sind. In diesen Fällen ist der Summand nicht gospersummierbar und daher findet der Gosper-Algorithmus keine geschlossene Form für die Summe s_n aus (4.1), obwohl eine geschlossene Form existiert.

4.1 Zeilberger-Algorithmus

In diesem Kapitel wird der Zeilberger-Algorithmus vorgestellt, welcher ein Zusammenspiel beider Algorithmen ist. Der Zeilberger-Algorithmus sucht genau wie der Fasenmyer-Algorithmus eine holonome Rekursionsgleichung und funktioniert im Grunde wie der

Gosper-Algorithmus, mit dem Unterschied, dass sich der Algorithmus mit der bestimmten Summation beschäftigt. Wir setzen für dieses Kapitel voraus, dass der Summand $F(n,k)$ von (4.1) einen endlichen Träger hat, das heißt, dass die Summe für jedes $n \in \mathbb{Z}$ nur für endlich viele k Summanden $F(n,k)$ ungleich 0 ist. Das Ziel, um das wir uns bemühen, ist eine geschlossene Form beziehungsweise eine Rekursionsgleichung für die Summe s_n zu finden. Die Sätze und Beweise aus diesem Kapitel sind aus [KOE14] entnommen.

Wir wollen zunächst zeigen, dass die Anwendung des Gosper-Algorithmus auf die bestimmte Summe (4.1) nicht zum Ziel führt, weil für einen hypergeometrischen Term $F(n,k)$ keine geschlossene Form $s_n \neq 0$ gefunden werden kann. Hierzu führen wir folgenden Satz ein:

Satz 4.1 *Sei $F(n,k)$ für alle $n \in \mathbb{N}_0$ wohldefiniert und habe einen endlichen Träger. Zusätzlich sei $F(n,k)$ ein hypergeometrischer Term nach Definition 2.14 und gospersummierbar bezüglich k mit einer hypergeometrischen diskreten Stammfunktion $G(n,k) = R(n,k)F(n,k)$, die für alle $k \in \mathbb{Z}$ endlich sei. Dann gilt*

$$\sum_{k=-\infty}^{\infty} F(n,k) = 0$$

für alle $n \in \mathbb{N}_0$, für welche der Nenner der rationalen Funktionen $R(n,k) \in \mathbb{K}(n,k)$ nicht 0 ist.

Beweis: Da $F(n,k)$ gospersummierbar bezüglich k ist, existiert zu F eine hypergeometrische diskrete Stammfunktion $G(n,k)$, für die

$$F(n,k) = G(n,k+1) - G(n,k) \tag{4.2}$$

gilt. Wird (4.2) über alle $k \in \mathbb{Z}$ summiert, so erhalten wir

$$\sum_{k=-\infty}^{\infty} F(n,k) = \sum_{k=-\infty}^{\infty} \Big(G(n,k+1) - G(n,k)\Big) = 0.$$

Die Summe ergibt in der Tat 0, da sie eine Teleskopsumme ist und daher die Auswertungen an den Grenzen gegenseitig wegfallen. Da $F(n,k)$ einen endlichen Träger hat, erhalten wir eine endliche Summe der Form

$$\sum_{k=-\infty}^{\infty} F(n,k) = \sum_{k=a}^{b} F(n,k),$$

wobei a und b die natürlichen Grenzen von $F(n,k)$ darstellen. Es ist möglich, dass G für bestimmte $n \in \mathbb{N}_0$ Singularitäten besitzt. Diese sind dann die Pole von R, da G ein rationales Vielfaches von F ist. □

Der Satz 4.1 besagt also, dass die direkte Anwendung des Gosper-Algorithmus auf $F(n,k)$ keine hypergeometrische diskrete Stammfunktion $G(n,k)$ liefert, sodass $F(n,k) = G(n,k+1) - G(n,k)$ gilt. Die Idee von Zeilberger ist nun, den Gosper-Algorithmus nicht auf $F(n,k)$, sondern auf den Ausdruck

$$a_k := F(n,k) + \sum_{j=1}^{J} \sigma_j(n) F(n+j,k) \tag{4.3}$$

mit einem geeigneten $J \in \mathbb{N}_0$ anzuwenden. Die unbekannten Variablen σ_j, $j = 1, ..., J$ sollen nicht von k sondern nur von n abhängig sein. Wir wollen nun überprüfen, ob (4.3) ein zulässiger Eingabeterm für den Gosper-Algorithmus ist. Hierzu reicht es zu zeigen, dass a_k ein hypergeometrischer Ausdruck ist. Das Verhältnis

$$\begin{aligned} \frac{a_{k+1}}{a_k} &= \frac{F(n,k+1) + \sum_{j=1}^{J} \sigma_j(n) F(n+j,k+1)}{F(n,k) + \sum_{j=1}^{J} \sigma_j(n) F(n+j,k)} \\ &= \frac{F(n,k+1)}{F(n,k)} \frac{1 + \sum_{j=1}^{J} \sigma_j(n) \frac{F(n+j,k+1)}{F(n,k+1)}}{1 + \sum_{j=1}^{J} \sigma_j(n) \frac{F(n+j,k)}{F(n,k)}} = \frac{u_k}{v_k} \in \mathbb{K}(n,k) \end{aligned} \tag{4.4}$$

ist rational bezüglich n und k und folglich ist a_k ein hypergeometrischer Term. Im Folgenden werden wir nun die einzelnen Schritte des Gosper-Algorithmus auf den hypergeometrischen Eingabeterm (4.3) anwenden. Hierzu bestimmen wir zunächst das Verhältnis $\frac{a_{k+1}}{a_k}$ gemäß (4.4) und suchen die Funktionen p_k, q_k und r_k, welche die Bedingung (3.14) erfüllen sollen. Wir definieren $p_k := 1$, $q_k := u_{k-1}$ und $r_k := v_{k-1}$ und stellen fest, dass für $j = 1$ die Ungleichheit

$$\gcd(q_k, r_{k+1}) = q_k \neq 1$$

gilt und deshalb die Bedingung (3.14) nicht erfüllt ist. Folglich ist die 1 zu Beginn immer in der Dispersionsmenge enthalten. Um die 1 aus der Dispersionsmenge zu eliminieren, werden p_k, q_k und r_k mit der Ersetzungsregel aus Lemma 3.5 aktualisiert, woraufhin p_k die unbekannten Variablen $\sigma_j(n)$, $j = 1, ..., J$ in linearer Form enthält. Nachdem wir die Polynome p_k, q_k und r_k aktualisiert haben, erfüllen sie die Bedingung (3.14) und wir können nun im nächsten Schritt die Gradschranke M für f_k mit Lemma 3.8 bestimmen. Dabei wird M so gesucht, dass die Gradschranke für alle n gültig ist. Ist M gefunden, so wird das allgemeine Polynom

$$f_k := b_0 + b_1 k + \cdots + b_M k^M$$

definiert und anschließend in die Rekursionsgleichung gemäß

$$p_k = q_{k+1} f_k - r_k f_{k-1}$$

eingesetzt. Durch Koeffizientenvergleich können nun die unbekannten Koeffizienten b_l, $l = 0, ..., M$ von f_k und die unbekannten Variablen σ_j, $j = 1, ..., J$ von a_k gleichzeitig bestimmt werden. Nach der erfolgreichen Bestimmung der Koeffizienten erhalten wir mit (3.8) eine Antidifferenz

$$G(n,k) = \frac{r_k}{p_k} f_{k-1} a_k$$

zu a_k, wobei a_k (4.3) entspricht. Zudem wird eine Menge rationaler Funktionen $\sigma_j(n) \in \mathbb{K}(n)$ gefunden, so dass

$$G(n,k+1) - G(n,k) = a_k = F(n,k) + \sum_{j=1}^{J} \sigma_j(n) F(n+j,k)$$

gilt. Wird nun a_k über $k \in \mathbb{Z}$ summiert, so ergibt sich

$$\begin{aligned} \sum_{k=-\infty}^{\infty} a_k &= \sum_{k=-\infty}^{\infty} \Big(F(n,k) + \sum_{j=1}^{J} \sigma_j(n) F(n+j,k) \Big) \\ &= s_n + \sum_{j=1}^{J} \sigma_j(n) s_{n+j} = \sum_{k=-\infty}^{\infty} \Big(G(n,k+1) - G(n,k) \Big) = 0. \end{aligned}$$

Auch hier ist die Summe eine Teleskopsumme und folglich 0. Multiplizieren wir mit dem Hauptnenner, so bekommen wir die gesuchte holonome Rekursionsgleichung der Ordnung J für die Summe s_n in der Form

$$\sum_{j=0}^{J} \sigma_j(n) s_{n+j} = 0.$$

Der beschriebene Algorithmus von Zeilberger wird nun an einem Beispiel veranschaulicht.

Beispiel 4.2 *Gegeben sei die hypergeometrische Summe*

$$s_n := \sum_{k=0}^{n} (-1)^n \binom{n}{k}^2, \quad n \in 2\mathbb{N}_0$$

und gesucht ist eine holonome Rekursionsgleichung erster Ordnung. Wir definieren zunächst

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

und stellen den Ansatz für den Zeilberger-Algorithmus wie folgt auf:

$$a_k := F(n,k) + \sigma_1(n) F(n+1,k) = (-1)^n \binom{n}{k}^2 + \sigma_1(n) (-1)^{n+1} \binom{n+1}{k}^2.$$

Im ersten Schritt des Algorithmus berechnen wir das Verhältnis

$$\frac{a_{k+1}}{a_k} = \frac{(-n-1+k)^2(-n^2\sigma_1 + k^2 - 2kn + n^2 - 2n\sigma_1 - \sigma_1)}{(-n^2\sigma_1 + k^2 - 2kn + n^2 - 2n\sigma_1 - 2k + 2n - \sigma_1 + 1)(k+1)^2}$$

und leiten daraus die Funktionen p_k, q_k und r_k ab. Da die Dispersionsmenge die Zahl 1 enthält, muss diese erst einmal durch Umschreiben der Funktionen eliminiert werden. Es ergeben sich

$$\begin{aligned} p_k &= -n^2\sigma_1 + k^2 - 2kn + n^2 - 2n\sigma_1 - 2k + 2n - \sigma_1 + 1, \\ q_k &= (-n-2+k)^2, \\ r_k &= k^2, \end{aligned}$$

welche folglich die Bedingung (3.14) *erfüllen. Lemma 3.8 liefert die obere Gradschranke 1 für f_k, wodurch wir die Funktion $f_k = b_1 k + b_0$ erhalten. Anschließend setzen wir f_k in die Rekursionsgleichung* (3.11) *ein und erhalten die Gleichung*

$$\begin{aligned} &-n^2\sigma_1 + k^2 - 2kn + n^2 - 2n\sigma_1 - 2k + 2n - \sigma_1 + 1 \\ &= (-n-1+k)^2(b_1 k + b_0) - k^2(b_1(k-1) + b_0), \end{aligned} \tag{4.5}$$

aus der wir durch Koeffizientenvergleich bezüglich k folgendes lineare Gleichungssystem

$$\begin{aligned} (-2n-2)b_1 + b_1 - 1 &= 0 \\ (-n-1)^2 b_1 + (-2n-2)b_0 + 2n + 2 &= 0 \\ (-n-1)^2 b_0 + n^2\sigma_1 - n^2 + 2n\sigma_1 - 2n + \sigma_1 - 1 &= 0 \end{aligned}$$

aufstellen. Aus diesem Gleichungssystem erhalten wir die unbekannten Variablen

$$b_0 = \frac{1}{2}\frac{3n+1}{2n+1}, \quad b_1 = -\frac{1}{2n+1}, \quad \sigma_1(n) = \frac{1}{2}\frac{n+1}{2n+1}.$$

Somit resultiert die Rekursionsgleichung

$$s_n + \frac{1}{2}\frac{n+1}{2n+1}s_{n+1} = 0. \tag{4.6}$$

Wir werden nun zeigen, dass diese Rekursionsgleichung explizit lösbar ist. Wir stellen hierfür die Behauptung

$$s_n \stackrel{!}{=} (-1)^n \frac{(2n)!}{(n!)^2} \tag{4.7}$$

auf und beweisen diese per vollständige Induktion:

$$n = 0: \qquad s_0 = \sum_{k=0}^{0} (-1)^n \binom{n}{k}^2 = (-1)^n \binom{0}{k}^2 = 1.$$

$n \to n+1$: *Sei* $n \in \mathbb{N}$ *und gelte* $s_n = (-1)^n \frac{(2n)!}{(n!)^2}$. *Dann folgt mit Gleichung* (4.6) :

$$s_{n+1} = -\frac{2(2n+1)}{(n+1)} (-1)^n \frac{(2n)!}{(n!)^2} = (-1)^{n+1} \frac{(2(n+1))!}{(n+1)!^2}.$$

Die Behauptung (4.7) *ist damit bewiesen.*

Nachdem nun die Vorgehensweise vom Zeilberger-Algorithmus an einem Beispiel erläutert wurde, können wir diesen Algorithmus folgendermaßen als Pseudo-Code aufschreiben:

Algorithmus 4.3 (Zeilberger) *Der Algorithmus sucht eine holonome Rekursionsgleichung für die hypergeometrische Summe* s_n.
Gegeben: $s_n = \sum_{k \in \mathbb{Z}} F(n,k)$.
Gesucht: holonome Rekursionsgleichung für s_n.

1. *Eingabe:* $F(n,k) \neq 0$, *wobei* F *ein hypergeometrischer Term nach Definition 2.14 ist.*

2. *Setze für die Ordnung der gesuchten Rekursionsgleichung* $J := 1$.

3. *Mache den folgenden Ansatz*

$$a_k := F(n,k) + \sum_{j=1}^{J} \sigma_j F(n+j,k),$$

   wobei die Funktionen σ_j *nur von* n *abhängen, das heißt* $\sigma_j = \sigma_j(n)$.

4. *Wende den angepassten Gosper-Algorithmus auf* a_k *an. Löse dabei im fünften Schritt des Gosper-Algorithmus aus dem linearen Gleichungssystem gleichzeitig die unbekannten Koeffizienten* b_l, $l = 0, ..., M$ *von* f_k *und die unbekannten Variablen* σ_j, $j = 1, ..., J$.

5. *Falls der Algorithmus erfolgreich ist, wird eine Antidifferenz* $G(n,k)$ *mit der Eigenschaft*

$$a_k = G(n, k+1) - G(n,k)$$

   gefunden.
   Es ist zu berücksichtigen, dass ganzzahlige Nullstellen im Nenner bezüglich n *auftreten können. In diesen Fällen ist das rationale Zertifikat* $\tilde{R}(n,k) = \frac{G(n,k)}{a_k}$ *für die Rekursionsgleichung möglicherweise nicht gültig. Falls der Algorithmus in Schritt 4 nicht erfolgreich war, dann erhöhe* J *um 1 und gehe zu Schritt 3 zurück.*

6. Ausgabe: holonome Rekursionsgleichung der Form

$$s_n + \sum_{j=1}^{J} \sigma_j(n) s_{n+j} = 0$$

für s_n.

Wir wissen, dass in Schritt 5 des Algorithmus die Ordnung J der gesuchten Rekursionsgleichung um 1 erhöht wird, falls der Algorithmus nicht erfolgreich war. Darüber hinaus wissen wir, dass der Algorithmus nicht immer die Rekursionsgleichung mit der niedrigsten Ordnung findet. Wir werden uns nun mit der Frage beschäftigen, ob sichergestellt werden kann, dass der Algorithmus wirklich terminiert. Hierfür werden wir zunächst einen zulässigen hypergeometrischen Eingabeterm $F(n,k)$ definieren. Dieser hat einen endlichen Träger und kann in der Form $F(n,k) = P(n,k)\frac{Q(n,k)}{R(n,k)} w^n z^k$ geschrieben werden, wobei $P(n,k)$ den polynomiellen Anteil und $Q(n,k)$ sowie $R(n,k)$ den faktoriellen Anteil darstellen. $Q(n,k)$ und $R(n,k)$ bestehen aus Produkten von Gammatermen, deren Argumente zudem ganzzahlig oder rational linear sind. Dann werden wir zeigen, dass der Zeilberger-Algorithmus für den zulässigen hypergeometrischen Term terminiert.

Definition 4.4 *Sei $F(n,k)$ ein zulässiger hypergeometrischer Term. Dann hat F einen endlichen Träger und die Form*

$$F(n,k) = P(n,k)\frac{\Gamma(\alpha_1 k + \beta_1 n + c_1)\cdots\Gamma(\alpha_p k + \beta_p n + c_p)}{\Gamma(\gamma_1 k + \delta_1 n + d_1)\cdots\Gamma(\gamma_q k + \delta_q n + d_q)} w^n z^k, \tag{4.8}$$

wobei $P(n,k) \in \mathbb{K}[n,k]$, α_l, β_l, γ_l, $\delta_l \in \mathbb{Z}$, und c_l, d_l, w, $z \in \mathbb{Q}$ gelten.

Für einen zulässigen hypergeometrischen Term gilt folgender Satz:

Satz 4.5 *Sei $F(n,k)$ ein zulässiger hypergeometrischer Term. Dann existiert eine k-freie Rekursionsgleichung der Form*

$$\sum_{i=0}^{I}\sum_{j=0}^{J} a_{ij}(n) F(n+j, k+i) = 0 \tag{4.9}$$

mit Polynomen $a_{ij}(n)$ für I und J groß genug. Genauer: Für die Existenz dieser Rekursionsgleichung sind die Bedingungen

$$J \geq J_0 := \sum_{l=1}^{p} |\alpha_l| + \sum_{l=1}^{q} |\gamma_l| \quad \textit{und} \quad I \geq \left(\sum_{l=1}^{p} |\beta_l| + \sum_{l=1}^{q} |\delta_l| - 1\right) \cdot J_0 + \deg_k(P(n,k)) \tag{4.10}$$

hinreichend, aber nicht notwendig.

Beweis: Sei N der Vorwärtsoperator bezüglich n und K der Vorwärtsoperator bezüglich k, gegeben durch

$$N^j F(n,k) = F(n+j,k) \quad \text{und}$$
$$K^i F(n,k) = F(n,k+i).$$

Dann können wir mit $K^i N^j F(n,k) = F(n+j,k+i)$ die Rekursionsgleichung (4.9) umschreiben zu

$$\underbrace{\sum_{i=0}^{I}\sum_{j=0}^{J} a_{ij}(n) K^i N^j}_{=:H(N,K,n)} F(n,k) = 0. \tag{4.11}$$

Da jedes $a_{ij}(n)$ ein Polynom in n ist, stellt $H(N,K,n)$ ein Polynom bezüglich der Variablen N, K und n dar. Wir nehmen nun an, dass $F(n,k)$ ein zulässiger hypergeometrischer Term ist und wollen zeigen, dass $H(N,K,n)F(n,k)$ ebenfalls ein zulässiger hypergeometrischer Term ist. Hierzu sei $F(n,k)$ wie in (4.8) und H durch

$$H(N,K,n) = \sum_{i=0}^{I}\sum_{j=0}^{J} a_{ij}(n) K^i N^j$$

gegeben. Wir definieren

$$\tilde{H}(n,k) := \frac{\prod_{l=1}^{p} \Gamma(\alpha_l k + \beta_l n + s(-\alpha_l)\alpha_l I + s(-\beta_l)\beta_l J + c_l)}{\prod_{l=1}^{p} \Gamma(\gamma_l k + \delta_l n + s(-\gamma_l)\gamma_l I + s(-\delta_l)\delta_l J + d_l)} w^n z^k,$$

wobei die Funktion $s(x)$ als Heaviside-Funktion bezeichnet wird und durch

$$s(x) := \begin{cases} 1, \text{ falls } x > 0 \\ 0, \text{ sonst} \end{cases}$$

definiert ist. Wir wissen, dass für alle $i = 0, ..., I$ und $j = 0, ..., J$ der Ausdruck $a_{ij} K^i N^j F(n,k)$ ein polynomielles Vielfaches von $\tilde{H}(n,k)$ ist und somit

$$a_{ij} K^i N^j F(n,k) = p_{ij}(n,k)\tilde{H}(n,k), \quad p_{ij}(n,k) \in \mathbb{Q}[n,k] \tag{4.12}$$

gilt. Da eine endliche Summe von Polynomen ebenfalls ein Polynom ist, folgt, dass $H(N,K,n)F(n,k)$ ein zulässiger hypergeometrischer Term ist. Der polynomielle Faktor $p_{ij} \in \mathbb{Q}[n,k]$ aus Gleichung (4.12) hat bezüglich k höchstens den Grad

$$D := \deg_k P(n,k) + \Big(\sum_{l=1}^{p} |\alpha_l| + \sum_{l=1}^{q} |\gamma_l|\Big) I + \Big(\sum_{l=1}^{p} |\beta_l| + \sum_{l=1}^{q} |\delta_l|\Big) J.$$

Wenn wir nun die Gleichung (4.12) über $i = 0, ..., I$ und $j = 0, ..., J$ summieren, erhalten wir

$$\sum_{i=0}^{I}\sum_{j=0}^{J} a_{ij}K^iN^jF(n,k) = \sum_{i=0}^{I}\sum_{j=0}^{J} p_{ij}(n,k)\tilde{H}(n,k)$$
$$H(N,K,n)F(n,k) = \sum_{i=0}^{I}\sum_{j=0}^{J} p_{ij}(n,k)\tilde{H}(n,k).$$

Nach Gleichung (4.11) muss

$$\sum_{i=0}^{I}\sum_{j=0}^{J} p_{ij}(n,k)\tilde{H}(n,k) = 0$$

gelten, und zwar für alle $k \in \mathbb{Z}$. Dies ist genau dann erfüllt, wenn

$$p(n,k) := \sum_{i=0}^{I}\sum_{j=0}^{J} p_{ij}(n,k) = 0$$

für alle $k \in \mathbb{Z}$ gilt. Nach Koeffizientenvergleich erhalten wir höchstens $D+1$ lineare Gleichungen mit $(I+1)(J+1)$ Variablen. Das resultierende lineare Gleichungssystem besitzt eine nichttriviale Lösung, wenn die Anzahl der Variablen höher ist als die Anzahl der Gleichungen, also wenn (4.10) gilt. □

Es gilt das folgende Korollar:

Korollar 4.6 *Sei $F(n,k)$ ein zulässiger hypergeometrischer Term und existiere eine Rekursionsgleichung der Ordnung J. Dann existieren Polynome $\sigma_j \in \mathbb{Q}[n]$, $j = 0, ..., J$, die nicht alle 0 sind, und ein zulässiger hypergeometrischer Term $G(n,k) = R(n,k)F(n,k)$ mit einer rationalen Funktion $R(n,k)$, sodass*

$$\sum_{j=0}^{J} \sigma_j(n)F(n+j,k) = G(n.k+1) - G(n,k)$$

gilt.

Beweis: Sei für einen zulässigen hypergeometrischen Term $F(n,k)$ die Rekursionsgleichung in der Form

$$\sum_{i=0}^{I}\sum_{j=0}^{J} \sigma_{ij}(n)F(n+j,k+i) \tag{4.13}$$

für I, $J \in \mathbb{N}_0$ gegeben. Mit den Shiftoperatoren N und K, wie im vorigen Beweis, können wir die Gleichung (4.13) umschreiben zu

$$H(N,K,n)F(n,k) = 0.$$

$H(N, K, n)$ ist ein Polynom, welches wir zu

$$H(N, K, n) = H(N, 1, n) + (1 - K)V(N, K, n)$$

umformulieren können, wobei V ein Polynom ist. Wenn wir dies nun auf die Gleichung (4.13) anwenden, erhalten wir

$$0 = H(N, K, n)F(n, k) = H(N, 1, n)F(n, k) + (1 - K)V(N, K, n)F(n, k)$$

und es folgt

$$H(N, 1, n)F(n, k) = (K - 1)V(N, K, n)F(n, k). \tag{4.14}$$

Wir setzen $G(n, k) := V(N, K, n)F(n, k)$ und erhalten für (4.14)

$$H(N, 1, n)F(n, k) = G(n, k + 1) - G(n, k),$$

welche eine Darstellung der Form (4.13) entspricht. Weiterhin ist $G(n, k)$ ein rationales Vielfaches von $F(n, k)$, da $V(N, K, n)F(n, k)$ eine Linearkombination mit polynomiellen Koeffizienten ist. Da F von der Form (4.8) ist, ist jedes $F(n + j, k + i)$ ein rationales Vielfaches von $F(n, k)$. □

Dies führt schließlich zu dem folgenden Korollar:

Korollar 4.7 *Für Summen mit einem zulässigen hypergeometrischen Term terminiert der Zeilberger-Algorithmus.*

Beweis: Mit dem Beweis zum Korollar 4.6 haben wir gezeigt, dass eine Rekursionsgleichung für einen zulässigen hypergeometrischen Term nach Definition 4.4 existiert. Das bedeutet, dass Zeilbergers Ansatz

$$\sum_{j=0}^{J} \sigma_j(n)F(n + j, k)$$

eine zulässige hypergeometrische Antidifferenz $G(n, k)$ für geeignete $\sigma_j(n)$, $j = 0, ..., J$ und $J \in \mathbb{N}_0$ besitzt. Dies ist damit zu begründen, dass der Gosper-Algorithmus ein Entscheidungsalgorithmus ist und somit ein $G(n, k)$ findet. Falls $F(n, k)$ einen endlichen Träger hat, wird die gewünschte holonome Rekursionsgleichung gemäß Schritt 6 des Zeilberger-Algorithmus gefunden. □

Der Zeilberger-Algorithmus terminiert für alle zulässigen hypergeometrischen Terme. Dies impliziert jedoch nicht, dass der Algorithmus für alle hypergeometrischen Terme terminiert.

Nun werden zwei Beispiele vorgerechnet, für die der Zeilberger-Algorithmus terminiert und eine holonome Rekursionsgleichung findet. Die Aufgabenstellungen sind aus [GKP94] entnommen.

Beispiel 4.8 *Wir betrachten die Summe*

$$s_n(z) = \sum_{k=0}^{n} \binom{n+k}{k} z^k$$

und suchen hierfür eine holonome Rekursionsgleichung. Mit $F(n,k) := \binom{n+k}{k} z^k$ *definieren wir zunächst den Ansatz*

$$a_k := F(n,k) + \sigma_1(n) F(n+1,k) = \binom{n+k}{k} z^k + \sigma_1(n) \binom{n+k+1}{k} z^k \tag{4.15}$$

und berechnen das Termverhältnis

$$\begin{aligned} \frac{a_{k+1}}{a_k} &= \frac{\binom{n+k+1}{k+1} z^{k+1} + \sigma_1(n) \binom{n+k+2}{k+1} z^{k+1}}{\binom{n+k}{k} z^k + \sigma_1(n) \binom{n+k+1}{k} z^k} \\ &= \frac{(n+k+1) z (k\sigma_1 + n\sigma_1 + n + 2\sigma_1 + 1)}{(k\sigma_1 + n\sigma_1 + n + \sigma_1 + 1)(k+1)}. \end{aligned}$$

Hieraus leiten wir $p_k = 1$, $q_k = (n+k) z \big((k-1)\sigma_1 + n\sigma_1 + n + 2\sigma_1 + 1 \big)$ *und* $r_k = \big((k-1)\sigma_1 + n\sigma_1 + n + \sigma_1 + 1 \big)$ *ab. Die Dispersionsmenge enthält die 1, sodass die Bedingung* (3.14) *nicht erfüllt ist. Mit der bekannten Ersetzungsregel aktualisieren wir* p_k, q_k *und* r_k *zu*

$$\begin{aligned} p_k &= k\sigma_1 + n\sigma_1 + n + \sigma_1 + 1, \\ q_k &= z(n+k), \\ r_k &= k. \end{aligned}$$

Mit Fall 1 vom Lemma 3.8 erhalten wir für f_k *die obere Gradschranke* $\deg(f_k) = 0$. *Wir setzen* $f_k = b_0$ *in die Rekursionsgleichung* (3.11) *ein und erhalten*

$$\begin{aligned} k\sigma_1 + n\sigma_1 + n + \sigma_1 + 1 &= z(n+k+1) b_0 + k b_0 \\ 0 &= k(\sigma_1 - z b_0 + b_0) + n\sigma_1 + n + \sigma_1 + z n b_0 + z b_0 + 1. \end{aligned}$$

Mit einem Koeffizientenvergleich bezüglich k *ergibt sich folgendes Gleichungssystem*

$$\begin{aligned} z b_0 - b_0 - \sigma_1 &= 0 \\ z(n+1) b_0 - \sigma_1 n - n - \sigma_1 - 1 &= 0, \end{aligned}$$

aus dem wir die unbekannten Variablen

$$b_0 = 1, \quad \sigma_1(n) = z - 1.$$

berechnen. Somit ergibt sich für unseren Ansatz (4.15)

$$F(n,k) + (z-1)F(n+1,k) = G(n,k+1) - G(n,k),$$

wobei

$$G(n,k) = \frac{r_k}{p_k}\frac{F(n,k)}{n+1} = \binom{n+k}{k-1} z^k$$

gilt. Summieren wir nun über $0 \leq k \leq n+1$*, so erhalten wir*

$$\begin{aligned} s_n(z) + F(n,n+1) + (z-1)s_{n+1}(z) &= G(n,n+2) - G(n,0) \\ &= \binom{2n+2}{n+1} z^{n+2} = 2\binom{2n+2}{n} z^{n+2}. \end{aligned} \tag{4.16}$$

Wir wissen, dass

$$F(n,n+1) = \binom{2n+1}{n+1} z^{n+1} = \binom{2n+1}{n} z^{n+1}$$

ist und folglich die Umstellung der Gleichung (4.16) *nach* s_{n+1} *folgende Gleichung*

$$s_{n+1}(z) = \frac{1}{1-z}\left(s_n(z) + (1-2z)\binom{2n+1}{n} z^{n+1}\right) \tag{4.17}$$

liefert. Es ist zu erkennen, dass für $z = \frac{1}{2}$ *ein Spezialfall vorliegt, denn es resultiert die Rekursionsgleichung*

$$s_{n+1}\left(\frac{1}{2}\right) = 2s_n\left(\frac{1}{2}\right).$$

Multiplizieren wir nun beide Seiten der Gleichung (4.17) *mit dem Faktor* $(1-z)^{n+1}$*, so erhalten wir die Identität*

$$(1-z)^n \sum_{k=0}^{n} \binom{n+k}{k} z^k = 1 + \frac{1-2z}{2-2z} \sum_{k=1}^{n} \binom{2k}{k} \big(z(1-z)\big)^k.$$

Beispiel 4.9 *Gegeben sei die Summe*

$$s_n(z) = \sum_{k=0}^{n} F(n,k)$$

mit $F(n,k) = \binom{n-k}{k} z^k$*. Wir suchen wie im vorigen Beispiel eine Rekursionsgleichung der Ordnung* $J = 1$*. Wir wählen daher folgende Ansatzfunktion*

$$a_k := F(n,k) + \sigma_1(n)F(n+1,k) = \binom{n-k}{k} z^k + \sigma_1 \binom{n+1-k}{k} z^k.$$

Das Verhältnis ergibt

$$\frac{a_{k+1}}{a_k} = -\frac{(-n-1+2k)(-n+2k)z(k\sigma_1 - n\sigma_1 + 2k - n + 1)}{(k\sigma_1 - n\sigma_1 + 2k - n - \sigma_1 - 1)(-n+k)(k+1)}.$$

Daraus folgen $p_k = 1$, $q_k = -(-n-3+2k)(-n+2k-2)z\Big(\sigma_1(k-1) - n\sigma_1 + 2k - 1 - n\Big)$ *und* $r_k = \Big(\sigma_1(k-1) - n\sigma_1 + 2k - 3 - n - \sigma_1\Big)(-n-1+k)k$. *Da die Dispersionsmenge nicht leer ist, aktualisieren wir die Funktionen und erhalten*

$$\begin{aligned} p_k &= k\sigma_1 - n\sigma_1 + 2k - n - \sigma_1 - 1, \\ q_k &= -z(-n+2k-2)(-n-3+2k), \\ r_k &= (-n-1+k)k. \end{aligned}$$

Da Lemma 3.8 $\deg(f_k) = -1$ *liefert, existiert nach dem Zeilberger-Algorithmus keine Rekursionsgleichung erster Ordnung. Wir werden nun die Ordnung der gesuchten Rekursionsgleichung um 1 erhöhen und den Algorithmus 4.3 ab dem dritten Schritt erneut durchlaufen. Wir wählen jetzt den Ansatz*

$$\begin{aligned} a_k &:= F(n,k) + \sigma_1(n)F(n+1,k) + \sigma_2(n)F(n+2,k) \\ &= \binom{n-k}{k} z^k + \sigma_1 \binom{n+1-k}{k} z^k + \sigma_2 \binom{n+2-k}{k} z^k. \end{aligned}$$

Für das Verhältnis erhalten wir

$$\frac{a_{k+1}}{a_k} = \frac{\begin{array}{c}(2k^2\sigma_1 + k^2\sigma_2 - 3kn\sigma_1 - 2kn\sigma_2 + n^2\sigma_1 + n^2\sigma_2 + 4k^2 - 4kn \\ - k\sigma_2 + n^2 + n\sigma_2 + 2k - n)z(-n-1+2k)(-n+2k-2)\end{array}}{\begin{array}{c}(2k^2\sigma_1 + k^2\sigma_2 - 3kn\sigma_1 - 2kn\sigma_2 + n^2\sigma_1 + n^2\sigma_2 + 4k^2 - 4kn - 4k\sigma_1 \\ - 3k\sigma_2 + n^2 + 3n\sigma_1 + 3n\sigma_2 - 6k + 3n + 2\sigma_1 + 2\sigma_2 + 2)(-n+k)(k+1)\end{array}}.$$

Diese Darstellung liefert uns

$$\begin{aligned} p_k &= 1, \\ q_k &= -(2(k-1)^2\sigma_1 + (k-1)^2\sigma_2 - (3(k-1))n\sigma_1 - (2(k-1))n\sigma_2 + n^2\sigma_1 + n^2\sigma_2 + 4(k-1)^2 \\ &\quad - (4(k-1))n - (k-1)\sigma_2 + n^2 + n\sigma_2 + 2k - 2 - n)z(-n-3+2k)(-n+2k-4), \\ r_k &= (2(k-1)^2\sigma_1 + (k-1)^2\sigma_2 - (3(k-1))n\sigma_1 - (2(k-1))n\sigma_2 + n^2\sigma_1 + n^2\sigma_2 + 4(k-1)^2 \\ &\quad - (4(k-1))n - (4(k-1))\sigma_1 - (3(k-1))\sigma_2 + n^2 + 3n\sigma_1 + 3n\sigma_2 - 6k + 8 + 3n + 2\sigma_1 \\ &\quad + 2\sigma_2)(-n-1+k)k. \end{aligned}$$

Die Funktionen müssen aktualisiert werden, da die 1 in der Dispersionsmenge enthalten ist. Nach der bekannten Ersetzungsregel erhalten wir nun die Funktionen

$$\begin{aligned} p_k &= 2k^2\sigma_1 + k^2\sigma_2 - 3kn\sigma_1 - 2kn\sigma_2 + n^2\sigma_1 + n^2\sigma_2 + 4k^2 - 4kn - 4k\sigma_1 - 3k\sigma_2 + n^2 + 3n\sigma_1 \\ &\quad + 3n\sigma_2 - 6k + 3n + 2\sigma_1 + 2\sigma_2 + 2, \\ q_k &= -(-n+2k-4)(-n-3+2k)z, \\ r_k &= (-n-1+k)k. \end{aligned}$$

Lemma 3.8 liefert für die gesuchte Funktion f_k die obere Gradschranke 0. Nachdem wir $f_k = b_0$ in die Rekursionsgleichung (3.11) *eingesetzt haben und einen Koeffizientenvergleich bezüglich k durchgeführt haben, erhalten wir das lineare Gleichungssystem*

$$\begin{aligned}
0 =& -4b_0z - b_0 - 2\sigma_1 - \sigma_2 - 4\\
0 =& -(-4n-6)zb_0 - (-n-1)b_0 + 3n\sigma_1 + 2n\sigma_2 + 4n + 4\sigma_1 + 3\sigma_2 + 6\\
0 =& (-n-2)(-n-1)zb_0 - n^2\sigma_1 - n^2\sigma_2 - n^2 - 3n\sigma_1 - 3n\sigma_2 - 3n - 2\sigma_1 - 2\sigma_2 - 2.
\end{aligned}$$

Aus diesem können wir nach den Unbekannten $\{b_0, \sigma_1(n), \sigma_2(n)\}$ auflösen. Als Ergebnis bekommen wir

$$b_0 = -\frac{1}{z}, \quad \sigma_1(n) = \frac{1}{z}, \quad \sigma_2(n) = -\frac{1}{z}.$$

Es ergibt sich gemäß Schritt 6 des Zeilberger-Algorithmus folgende Rekursionsgleichung zweiter Ordnung

$$\begin{aligned}
s_n(z) + \frac{1}{z}s_{n+1}(z) - \frac{1}{z}s_{n+2}(z) = 0, &\quad n \geq 0\\
\Leftrightarrow s_{n+2}(z) = zs_n(z) + s_{n+1}(z), &\quad n \geq 0.
\end{aligned}$$

Im bisherigen Verlauf der Arbeit wurden Algorithmen vorgestellt, mit denen wir für die bestimmte Summation eine Rekursionsgleichung finden können. Dazu zählen unter anderem der Fasenmyer-Algorithmus und der Zeilberger-Algorithmus. Im Vergleich ist der Zeilberger-Algorithmus deutlich effizienter, jedoch ist der Ordnungsbereich für das Finden der geschlossenen Form beschränkt. Bisher können wir lediglich eine Rekursionsgleichung erster Ordnung in eine geschlossene Form bringen. In diesem Kapitel haben wir auch gesehen, dass beim Zeilberger-Algorithmus die Ordnung der gesuchten Rekursionsgleichung um 1 erhöht wird, falls der Algorithmus nicht erfolgreich war. Mit einer Forderung an den Eingabeterm, nämlich, dass der Eingabeterm $F(n, k)$ zulässig nach Definition 4.4 ist, können wir sicherstellen, dass der Algorithmus terminiert. Die obere Gradschranke kann hierfür bestimmt werden. Das heißt, dass der Zeilberger-Algorithmus theoretisch auch Rekursionsgleichungen höherer Ordnung bestimmen kann, es aber hierbei meist zu Problemen führt. Denn bei einer höheren Rekursionsordnung($J > 5$) werden die Ausdrücke so komplex, dass die effiziente algorithmische Berechnung sehr zeitaufwendig und die erhaltenen Rekursionen sehr komplex werden. Ist die Rekursionsordnung also in einer höheren Ordnung ($J \geq 2$), so stellt sich die Frage, wie wir eine geschlossene Form finden können. Im folgenden Abschnitt werden wir einen Algorithmus vorstellen, der dieses Problem löst.

4.2 Petkovsek-Algorithmus

Der Petkovsek-Algorithmus findet für beliebige holonome Rekursionsgleichungen eine Lösung, falls solch eine existiert. Dieser ist ein Entscheidungsalgorithmus, welcher für

eine gegebene Rekursionsgleichung entscheidet, ob hypergeometrische Lösungen existieren. Falls hypergeometrische Lösungen gefunden werden können, sind wir in der Lage die gewünschte geschlossene Form auszugeben. Der Petkovsek-Algorithmus beinhaltet zwei Algorithmen. Zuerst werden für eine gegebene holonome Rekursionsgleichung Polynomlösungen gefunden und anschließend nach hypergeometrischen Lösungen gesucht. Die in diesem Kapitel aufgeführten Algorithmen sind aus [KOE14] entnommen.

Algorithmus 4.10 (Polynomlösungen holonomer Rekursionsgleichungen)
Der Algorithmus findet alle Polynomlösungen einer gegebenen holonomen Rekursionsgleichung, sofern solche existieren.
Gegeben: holonome Rekursionsgleichung.
Gesucht: polynomielle Lösungen der gegebenen holonomen Rekursionsgleichung.

1. *Eingabe: holonome Rekursionsgleichung der Ordnung J der Form*

$$\sum_{j=0}^{J} P_j(n) s_{n+j} = 0$$

mit Polynomen

$$P_j(n) = \sum_{l=0}^{M} \alpha_{j,l} n^{M-l} \quad \in \mathbb{K}[n],$$

wobei für den Koeffizienten von n^M für mindestens ein $j \in \{0, ..., J\}$ $\alpha_{j,0} \neq 0$ gilt.

2. *Setze $m := 0$.*

3. *Berechne für jedes $l \in \{0, ..., m\}$*

$$b_{lm} := \sum_{j=0}^{J} j^l \alpha_{j,m-l}.$$

Falls für alle $b_{lm} = 0$ gilt, dann erhöhe m um 1 und wiederhole Schritt 3.

4. *Sei $\mathcal{N}$ definiert durch*

$$\mathcal{N} := \left\{ N \in \mathbb{N}_0 \middle| 0 = \sum_{l=0}^{m} \binom{N}{l} b_{lm} \right\},$$

das heißt die Menge $\mathcal{N}$ beinhaltet alle nichtnegativen Nullstellen von $\sum_{l=0}^{m} \binom{N}{l} b_{lm}$.

5. *Falls $\mathcal{N} = \emptyset$ gilt, dann existiert keine polynomielle Lösung.*

6. *Ansonsten setze den Grad* $N := \max\{\mathcal{N}\}$ *in die allgemeine Polynomlösung*

$$s_n = n^N + \delta_1 n^{N-1} + \cdots + \delta_N$$

ein und führe einen Koeffizientenvergleich bezüglich n *durch. Löse das erhaltene lineare Gleichungssystem nach den Unbekannten* $\{\delta_1, ..., \delta_N\}$.

7. *Ausgabe: Die polynomielle Lösung für* s_n.

Der beschriebene Algorithmus ist eine Unterroutine, die wir für die Bestimmung aller hypergeometrischen Lösungen einer holonomen Rekursionsgleichung benötigen. Bevor wir jedoch den Hauptalgorithmus von Petkovsek vorstellen, führen wir zuvor das folgende Lemma ein:

Lemma 4.11 (Darstellung rationaler Funktionen nach Gosper-Petkovsek) *Jede beliebige rationale Funktion* $t_k \in \mathbb{K}(k)\backslash\{0\}$ *besitzt eine Darstellung der Form*

$$t_k = C\frac{p_{k+1}}{p_k}\frac{q_{k+1}}{r_{k+1}},$$

wobei die Polynome $p_k, q_k, r_k \in \mathbb{K}[k]$ *monisch*[2] *sind und* $C \in \mathbb{K}$ *gilt. Dann sind zusätzlich folgende Eigenschaften erfüllt:*

1. $\gcd(q_k, r_{k+j}) = 1 \quad \forall j \in \mathbb{N}_0$,
2. $\gcd(p_k, q_{k+1}) = 1$,
3. $\gcd(p_k, r_k) = 1$.

Das Lemma 4.11 wird an dieser Stelle nicht bewiesen, kann jedoch entweder in [KOE14] oder in [PWZ96] nachgelesen werden.

Nun können wir den Algorithmus von Petkovsek zur Bestimmung hypergeometrischer Termlösungen einführen:

Algorithmus 4.12 (Lösungen holonomer Rekursionsgleichungen)
Der Algorithmus findet alle hypergeometrischen Lösungen einer gegebenen holonomen Rekursionsgleichung, sofern solche existieren.
Gegeben: holonome Rekursionsgleichung.
Gesucht: hypergeometrische Lösungen der gegebenen holonomen Rekursionsgleichung.

1. *Eingabe: holonome Rekursionsgleichung der Ordnung* J *der Form*

$$\sum_{j=0}^{J} P_j(n) s_{n+j} = 0$$

mit Polynomen $P_j(n) \in \mathbb{K}[n]$.

[2]Polynome, bei denen der Leitkoeffizient gleich 1 ist.

2. *Setze* $L := \{\}$.

3. *Führe für alle Kombinationen der monischen Faktoren* q_n *von* $P_0(n-1)$ *und* r_n *von* $P_J(n-J)$ *folgende Schritte durch:*

   (a) *Definiere für* $j = 0, ..., J$ *die Funktionen*

   $$h_j(n) := P_j(n) \prod_{l=1}^{j} q_{n+l} \prod_{l=j+1}^{J} r_{n+l}.$$

   (b) *Sei* $M := \max\Big\{ \deg(h_0(n)), ..., \deg(h_J(n)) \Big\}$ *und seien* α_j *für* $j = 0, ..., J$ *die Koeffizienten von* n^M *in* $h_j(n)$.

   (c) *Berechne zunächst die Nullstellen* $C \in \mathbb{K}$ *der Polynomgleichung*

   $$\sum_{j=0}^{J} \alpha_j C^j = 0.$$

   Finde nun alle Polynomlösungen p_n *und wende hierzu für jedes* C *den Algorithmus 4.10 für die Rekursionsgleichung*

   $$\sum_{j=0}^{J} C^j h_j(n) p_{n+j} = 0$$

   an. Falls eine Polynomlösung p_n *existiert, füge die Lösung*

   $$\frac{t_{n+1}}{t_n} = C \frac{p_{n+1}}{p_n} \frac{q_{n+1}}{r_{n+1}} \tag{4.18}$$

   der Menge L *hinzu.*

4. *Ausgabe: die Menge* L*. Diese enthält alle rationalen hypergeometrischen Lösungen* (4.18) *der Rekursionsgleichung.*

Wir sind nun in der Lage mit dem Petkovsek-Algorithmus eine Menge L von hypergeometrischen Quotienten der Form

$$\frac{t_{n+1}^{(i)}}{t_n^{(i)}}, \qquad i = 1, ..., l \tag{4.19}$$

zu finden. l stellt die Anzahl der Lösungen der Menge L dar. Wir wissen jedoch noch nicht, wie wir aus dieser Menge eine geschlossene Form bilden können. Hierzu führen wir den folgenden Algorithmus ein:

Algorithmus 4.13 *Der Algorithmus wandelt eine Menge von hypergeometrischen Quotienten der Form* (4.19) *in eine geschlossene Form um, wenn eine existiert.*
Gegeben: Summe s_n und die Menge L.
Gesucht: Geschlossene Form $s_n = \alpha_1 t_n^{(1)} + ... + \alpha_l t_n^{(l)}$.

1. *Eingabe: Eine Menge $L = \left\{ \frac{t_{n+1}^{(1)}}{t_n^{(1)}}, ..., \frac{t_{n+1}^{(l)}}{t_n^{(l)}} \right\}$.*

2. *Bestimme $t_n^{(i)}$ aus $\frac{t_{n+1}^{(i)}}{t_n^{(i)}}$ für alle $i = 1, ..., l$ mit folgenden Schritten:*

   (a) *Setze $\frac{t_{n+1}^{(i)}}{t_n^{(i)}} = \frac{u_n}{v_n}$.*

   (b) *Faktorisiere u_n, v_n gemäß Algorithmus 2.11.*

   (c) *Schreibe den erhaltenen Ausdruck $t_n^{(i)}$ in Form von Pochhammersymbolen und Potenzfunktionen.*

3. *Bestimme J Anfangswerte $s_0, ..., s_J$ für die Summe s_n. J ist die Ordnung der gegebenen Rekursionsgleichung.*

4. *Löse das Gleichungssystem*

$$s_n = \alpha_1 t_n^{(1)} + ... + \alpha_l t_n^{(l)}, \quad n = 0, ..., J-1$$

   nach den Unbekannten α_1,...,α_l.

5. *Ausgabe: Die geschlossene Form $s_n = \alpha_1 t_n^{(1)} + ... + \alpha_l t_n^{(l)}$.*

Es wurden in diesem Kapitel zwei nützliche Algorithmen vorgestellt, die sich mit holonomen Rekursionsgleichungen beschäftigen. Mit dem Zeilberger-Algorithmus können solche gefunden werden, sofern sie existieren. Anschließend kann die Rekursionsgleichung in eine geschlossene Form umgewandelt werden, falls die Rekursion höchstens 1. Ordnung ist. Bei einer Rekursionsordnung größer als 1, kann zum Finden der geschlossenen Form der Petkovsek-Algorithmus eingesetzt werden. Mit dieser Kenntnis sind wir in der Lage Identitäten zu beweisen, die hypergeometrische Reihen enthalten. Und zusätzlich können wir hypergeometrische Reihen in eine geschlossene Form überführen, sofern sie existieren.

5. Herleitung hypergeometrischer Identitäten

In diesem Kapitel der Arbeit wird die Maplet-Anwendung vorgestellt, indem auf die Anwendungsweise und Implementierung eingegangen wird. Zudem werden einige hypergeometrische Identitäten mit und ohne Maplet-Anwendung hergeleitet.

5.1 Dokumentation der Maplet-Anwendung

Bevor die Maplet-Anwendung beschrieben wird und ihre Vorteile aufgezeigt werden, gehe ich kurz auf die aktuelle Vorgehensweise ohne Maplet-Anwendung ein:
Sucht man zu einer hypergeometrischen Reihe eine geschlossene Form, so muss zunächst die Maple-Anwendung gestartet, das hsum-Paket importiert und ein Algorithmus gefunden werden, welcher die gewünschte geschlossene Form liefert. Dabei ist es durchaus möglich, dass erst nach Anwenden einiger Algorithmen der Erfolg in Form einer geschlossenen Form erreicht wird. Wenn der Gosper- und Zeilberger-Algorithmus keine geschlossene Form finden können, muss der Petkovsek-Algorithmus angewendet werden. Da keine Implementierung des Petkovsek-Algorithmus in dem hsum-Paket enthalten ist, ist es sehr aufwendig den Petkovsek-Algorithmus in Maple anzuwenden. Denn es müssen alle Schritte des Algorithmus als Maple-Befehle eingegeben werden.
Im Vergleich zu diesem Vorgehen bietet die von mir implementierte Maplet-Anwendung den Vorteil, dass durch die Benutzung das Finden einer geschlossenen Form deutlich einfacher, angenehmer und schneller geschieht. Dies bedeutet auch, dass die Anwendung von Benutzern verwendet werden kann, die keinen einzigen Maple-Befehl kennen. Denn man muss lediglich eine hypergeometrische Reihe eingeben und eine Schaltfläche tätigen.
Daraufhin wird automatisch versucht eine geschlossene Form zu finden und im Erfolgsfall wird diese ausgegeben. Die Maplet-Anwendung wandelt eine beliebige hypergeometrische Reihe in eine geschlossene Form um, sofern diese mithilfe der Algorithmen von Gosper, Zeilberger oder Petkovsek gefunden werden kann. Dabei enthält die Maplet-Anwendung die Implementierung des Petkovsek-Algorithmus, auf welche in diesem Kapitel noch eingegangen wird.

Im Folgenden wird die Anleitung zur Benutzung der Maplet-Anwendung beschrieben und anschließend die Implementierung der Maplet-Anwendung erklärt.

Anleitung 5.1 (Benutzung der Maplet-Anwendung)

1. *Starten Sie die Maplet-Anwendung „Masterarbeit-Herleitung hypergeometrischer Identitäten“. Es öffnet sich das Hauptfenster der Anwendung wie in Abbildung 5.1 zu sehen ist.*

2. *Klicken Sie unter „Import“ den Menüpunkt „Import Paket“. Drücken Sie nun auf die Schaltfläche „Import“ und es wird ein Auswahlfenster erscheinen. Wählen Sie aus Ihrem Verzeichnis ein hsum-Paket aus und drücken Sie anschließend auf „open“. Beachten Sie, dass nur Dateien mit der Endung „mpl“ importiert werden dürfen. Bei einem erfolgreichen Import erscheint im Textfeld der Dateipfad der ausgewählten Datei, wie beispielsweise in der Abbildung 5.2 zu sehen ist.*

3. *Es wurde nun das hsum-Paket importiert und gelesen. Mit einem Klick auf „Schließen“ können Sie zur Hauptansicht zurückkehren.*

4. *Geben Sie einen beliebigen hypergeometrischen Term ein, beispielsweise* $F := \text{binomial}(n, k)$*, und drücken Sie „Berechne“. Es werden die leeren Felder „Antidifferenz/Rekursionsgleichung“, „geschlossene Form“ und „Algorithmus“ überschrieben, wenn das jeweilige Ergebnis gefunden wird. Für den Beispieleingabeterm* $F := \text{binomial}(n, k)$ *ergibt sich die Ausgabe in Abbildung 5.3.*

5. *Ist bei einer Eingabe sowohl der Gosper als auch der Zeilberger-Algorithmus beim Finden einer geschlossenen Form nicht erfolgreich, so wird der Petkovsek-Algorithmus angewendet. Beachten Sie beim Petkovsek-Algorithmus, dass dieser zum Eingabeterm zusätzlich die Summengrenzen benötigt, siehe hierfür Schritt 3 des Algorithmus 4.13.*

6. *Wenn Sie in der Symbolleiste auf „Algorithmen“ klicken, erscheinen die Menüpunkte „Gosper“, „Zeilberger“ und „Petkovsek“. Mit einem Klick auf einen Menüpunkt, öffnet sich die jeweilige Beschreibung des Algorithmus in einem neuen Fenster, siehe Abbildung 5.4.*

7. *Durch Betätigen von „Lösche Eingaben“ beziehungsweise „Lösche Ausgaben“ werden alle Eingabefelder beziehungsweise Ausgabefelder gelöscht.*

8. *Drücken Sie auf „Schließen“, um die Anwendung zu beenden.*

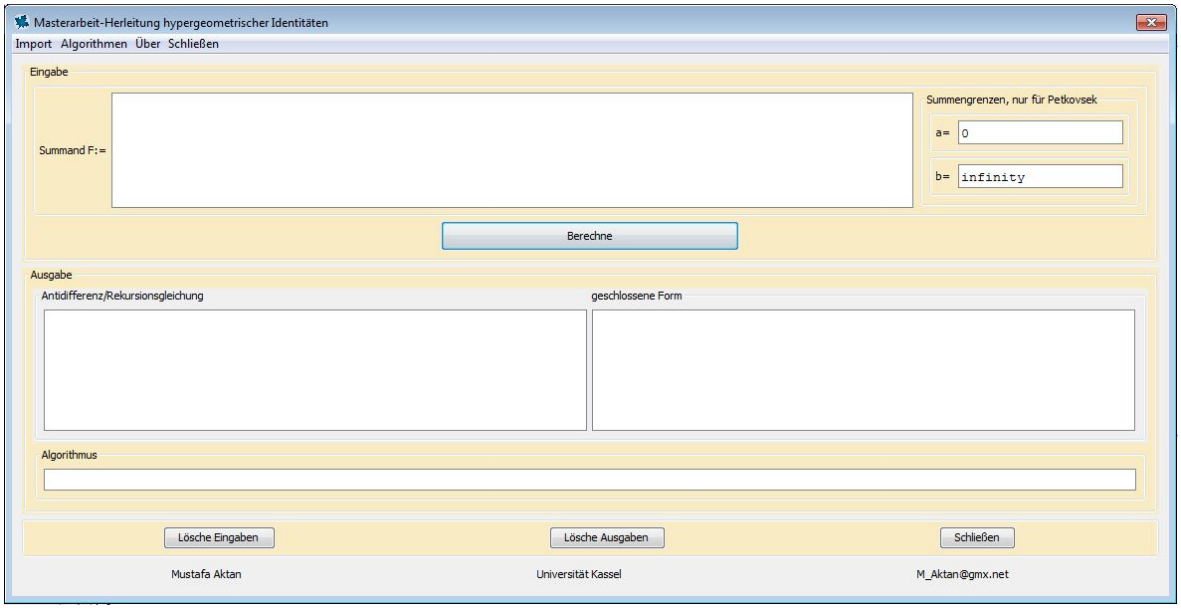

Abbildung 5.1: Hauptansicht der Maplet-Anwendung

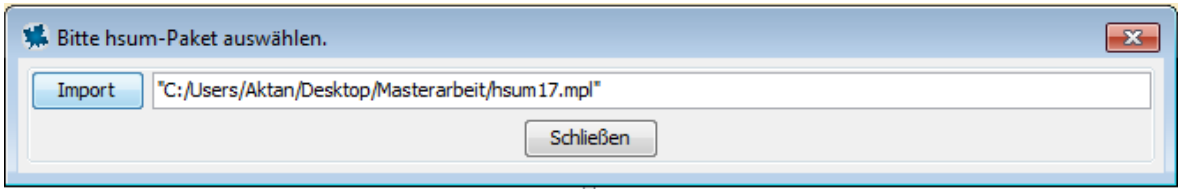

Abbildung 5.2: Importansicht

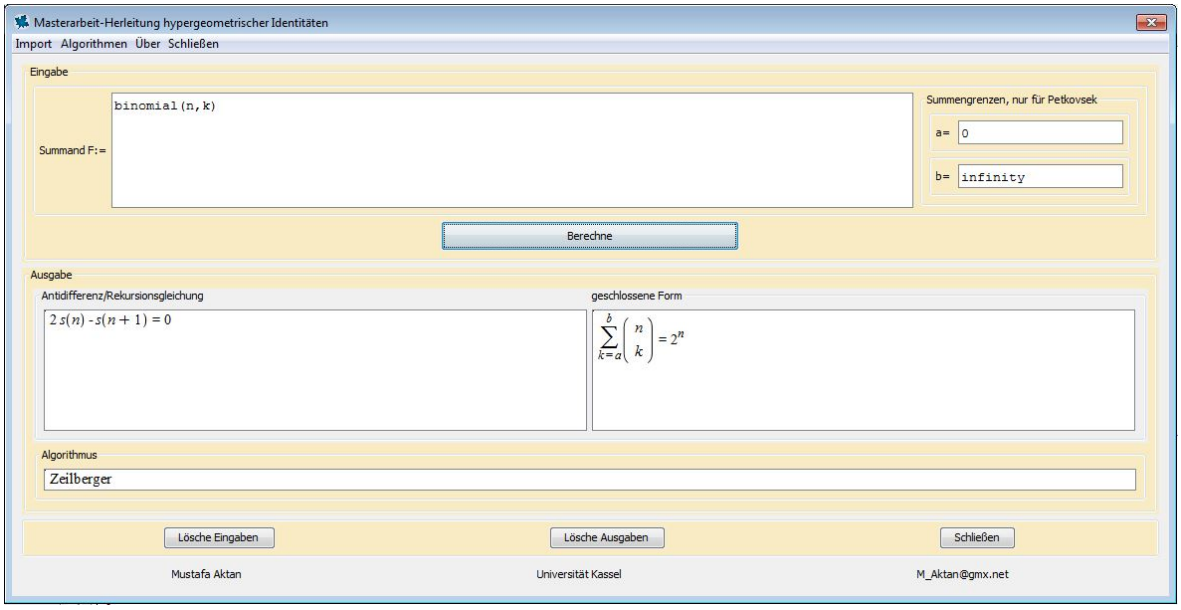

Abbildung 5.3: Ausgabeansicht für den Eingabeterm

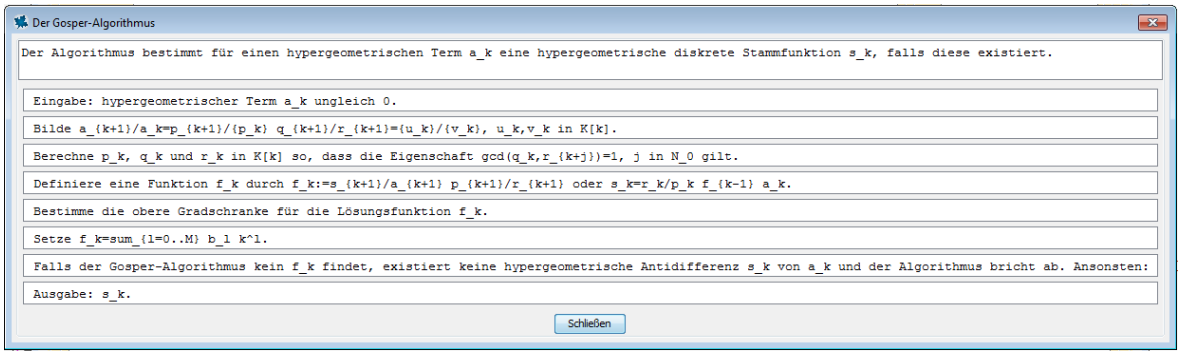

Abbildung 5.4: Beschreibung des Gosper-Algorithmus

Die nachfolgende Abbildung 5.5 beschreibt die Ablauflogik der Algorithmen in der Maplet-Anwendung.

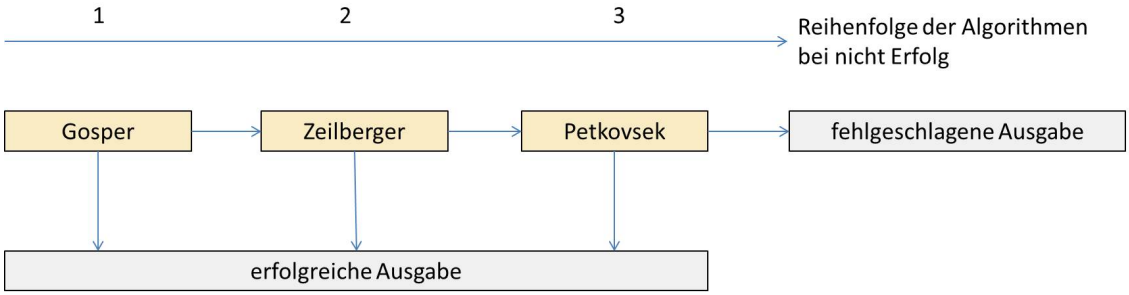

Abbildung 5.5: Ablauflogik der Algorithmen

Nachdem ein hypergeometrischer Term eingegeben und die Schaltfläche „Berechne“ betätigt wurde, wird auf den Eingabeterm zunächst der Gosper-Algorithmus angewandt. Ist dieser erfolgreich, so wird eine Antidifferenz berechnet und ausgegeben. Zudem wird eine geschlossene Form nach Gleichung (3.4) ausgegeben und in das Feld „Algorithmus“ Gosper eingetragen. Im Falle, dass der Gosper-Algorithmus nicht erfolgreich ist und somit keine geschlossene Form findet, wird auf den Eingabeterm der Zeilberger-Algorithmus angewendet. In den meisten Fällen kann der Zeilberger-Algorithmus eine Rekursionsgleichung ausgeben, wobei eine geschlossene Form nur dann gefunden wird, wenn die Rekursionsordnung höchstens erster Ordnung ist. Dann werden die Rekursionsgleichung und die geschlossene Form ausgegeben und in das dritte Ausgabefeld Zeilberger geschrieben. Falls der Zeilberger-Algorithmus keine geschlossene Form finden kann, wird schließlich der Petkovsek-Algorithmus angewendet. Diesen habe ich nach dem Algorithmus 4.13 implementiert. Er beinhaltet zusätzlich den Algorithmus 4.12, da die Menge L notwendig für

das Finden der geschlossenen Form ist. In Schritt 2 des Algorithmus 4.13 wird $t_n^{(i)}$ aus $\frac{t_{n+1}^{(i)}}{t_n^{(i)}}$ für alle $i = 1, ..., l$ aus L bestimmt. Für diese Umwandlung ist keine Funktion vorhanden. Daher habe ich diese implementiert und der interessierte Leser kann die Implementierung dieser Funktion sowie die Implementierung des Petkovsek-Algorithmus im Anhang als Maplet-Anwendung ansehen. Ist der Petkovsek-Algorithmus erfolgreich, so wird die Rekursionsgleichung und die geschlossene Form in den entsprechenden Feldern ausgegeben und zusätzlich Petkovsek in das Feld „Algorithmus“ eingetragen. Wenn jedoch die Ausgabe „Es konnte kein Algorithmus eine geschlossene Form finden“ lautet, dann bedeutet dies, dass kein Algorithmus in der Lage war eine geschlossene Form zu finden.

5.2 Beispiele hypergeometrischer Identitäten

In diesem Abschnitt der Arbeit wollen wir einige Identitäten herleiten beziehungsweise beweisen. Die Identiäten sind aus [PBM90] entnommen. Auf den ersten Blick können diese sehr komplex wirken, sind jedoch mit den vorgestellten Algorithmen leicht und schnell zu berechnen. Den Anfang macht die hypergeometrische Funktion ${}_4F_3$ im folgenden Satz:

Satz 5.2 *Es gelte die Identität*

$$s_n = {}_4F_3\left(\left.\begin{matrix}-n, a, a+\frac{1}{2}, b \\ 2a+1, \frac{b-n}{2}, \frac{b-n+1}{2}\end{matrix}\right| 1\right) := \sum_{k=0}^{\infty} \frac{(-n)_k (a)_k (a+\frac{1}{2})_k (b)_k}{(2a+1)_k (\frac{b-n}{2})_k (\frac{b-n+1}{2})_k} \frac{1}{k!}$$
$$= \frac{(2a-b+1)_n}{(1-b)_n}. \tag{5.1}$$

Beweis: Wir werden die hypergeometrische Funktion ${}_4F_3$ mithilfe der bisher vorgestellten Algorithmen in eine geschlossene Form überführen, wobei wir zum jetzigen Zeitpunkt nicht wissen welcher Algorithmus die gewünschte Darstellung liefern wird. Daher werden wir zunächst den Gosper-Algorithmus auf (5.1) anwenden. Existiert eine hypergeometrische Antidifferenz, so können wir die geschlossene Form auch ohne Anwendung weiterer Algorithmen liefern. Für den Gosper-Algorithmus berechnen wir zum Summanden

$$a_k := \frac{(-n)_k (a)_k (a+\frac{1}{2})_k (b)_k}{(2a+1)_k (\frac{b-n}{2})_k (\frac{b-n+1}{2})_k} \frac{1}{k!}$$

zu aller erst das Termverhältnis

$$\frac{a_{k+1}}{a_k} = \frac{2(k-n)(k+a)(2k+1+2a)(k+b)}{(k+2a+1)(2k+b-n)(2k+b+1-n)(k+1)}. \tag{5.2}$$

Aus (5.2) resultieren die Funktionen

$$p_k = 1,$$
$$q_k = 2(-n+k-1)(a+k-1)(2a-1+2k)(b+k-1),$$
$$r_k = (2a+k)(b-n+2k-2)(b-n-1+2k)k.$$

Diese erfüllen die Bedingung (3.14) und somit müssen wir die Ersetzungsregel nicht anwenden. Wir erhalten durch das Lemma 3.8 mithilfe von Fall 1 für den polynomiellen Ansatz die Gradschranke $\deg(f_k) = -3$. Da die Gradschranke für f_k negativ ist, kann der Gosper-Algorithmus keine hypergeometrische Antidifferenz finden und somit auch keine geschlossene Form für (5.1) ausgeben. Das heißt, dass der Gosper-Algorithmus an dieser Stelle für die Herleitung der gewünschten Identität nicht geeignet ist. Deswegen ziehen wir den nächsten Algorithmus heran, nämlich den Zeilberger-Algorithmus, und versuchen hiermit eine holonome Rekursionsgleichung für (5.1) zu suchen, um daraus eine geschlossene Form abzuleiten. Hierfür definieren wir

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

und wählen mit $J = 1$ die folgende Ansatzfunktion

$$a_k := F(n,k) + \sigma_1(n) F(n+1,k).$$

Das Verhältnis ergibt

$$\frac{a_{k+1}}{a_k} = \frac{\begin{array}{c}\Big(2(-bn\sigma_1 - 2kn\sigma_1 + n^2\sigma_1 + bk - bn - b\sigma_1 - kn - 2k\sigma_1 \\ + n^2 - k + n - \sigma_1)(b+k)(2a+1+2k)(a+k)(-n-1+k)\Big)\end{array}}{\begin{array}{c}\Big((-bn\sigma_1 - 2kn\sigma_1 + n^2\sigma_1 + bk - bn - b\sigma_1 - kn - 2k\sigma_1 + n^2 + 2n\sigma_1 - b \\ - k + 2n + \sigma_1 + 1)(k+1)(b-n+1+2k)(b-n+2k)(2a+1+k)\Big)\end{array}}.$$

Daraus resultieren die Funktionen

$$\begin{aligned}
p_k &= 1, \\
q_k &= 2(-bn\sigma_1 - (2(k-1))n\sigma_1 + n^2\sigma_1 + (k-1)b - bn - b\sigma_1 - (k-1)n - (2(k-1))\sigma_1 \\
&\quad + n^2 - k + 1 + n - \sigma_1)(b+k-1)(2a-1+2k)(a+k-1)(-n-2+k), \\
r_k &= (-bn\sigma_1 - (2(k-1))n\sigma_1 + n^2\sigma_1 + (k-1)b - bn - b\sigma_1 - (k-1)n - (2(k-1))\sigma_1 \\
&\quad + n^2 + 2n\sigma_1 - b - k + 2 + 2n + \sigma_1)k(b-n-1+2k)(b-n+2k-2)(2a+k).
\end{aligned}$$

Die Dispersionsmenge ist nicht leer, da sie die 1 enthält. Daher aktualisieren wir die Funktionen und erhalten

$$\begin{aligned}
p_k &= -bn\sigma_1 - 2kn\sigma_1 + n^2\sigma_1 + bk - bn - b\sigma_1 - kn - 2k\sigma_1 + n^2 \\
&\quad + 2n\sigma_1 - b - k + 2n + \sigma_1 + 1, \\
q_k &= 2(-n-2+k)(a+k-1)(2a-1+2k)(b+k-1), \\
r_k &= k(b-n-1+2k)(b-n+2k-2)(2a+k).
\end{aligned}$$

Nun ist die Dispersionsmenge leer und eine weitere Aktualisierung der Funktionen p_k, q_k und r_k ist nicht erforderlich. Das Lemma 3.8 liefert für die Ansatzfunktion die Gradschranke $\deg(f_k) = -2$. Da die Gradschranke eine negative Zahl ist, existiert nach dem Zeilberger-Algorithmus keine Rekursionsgleichung erster Ordnung. Wir werden nun die Ordnung der gesuchten Rekursionsgleichung um 1 erhöhen und den Algorithmus 4.3 ab dem dritten Schritt erneut durchlaufen. Wir wählen jetzt den Ansatz

$$a_k := F(n,k) + \sigma_1(n)F(n+1,k) + \sigma_2(n)F(n+2,k).$$

Für das Verhältnis erhalten wir

$$\frac{a_{k+1}}{a_k} = \frac{\begin{aligned}&\Big(2(-b^2kn\sigma_1 + b^2n^2\sigma_1 + b^2n^2\sigma_2 - 2bk^2n\sigma_1 + 4bkn^2\sigma_1 + 4bkn^2\sigma_2 - 2bn^3\sigma_1 - 2bn^3\sigma_2\\ &+ 2k^2n^2\sigma_1 + 4k^2n^2\sigma_2 - 3kn^3\sigma_1 - 4kn^3\sigma_2 + n^4\sigma_1 + n^4\sigma_2 + b^2k^2 - 2b^2kn - b^2k\sigma_1\\ &+ b^2n^2 + 2b^2n\sigma_1 + 3b^2n\sigma_2 - 2bk^2n - 2bk^2\sigma_1 + 4bkn^2 + 7bkn\sigma_1 + 12bkn\sigma_2 - 2bn^3\\ &- 5bn^2\sigma_1 - 5bn^2\sigma_2 + k^2n^2 + 6k^2n\sigma_1 + 12k^2n\sigma_2 - 2kn^3 - 10kn^2\sigma_1 - 10kn^2\sigma_2 + n^4\\ &+ 3n^3\sigma_1 + 2n^3\sigma_2 - b^2k + b^2n + b^2\sigma_1 + 2b^2\sigma_2 - 3bk^2 + 8bkn + 3bk\sigma_1 + 8bk\sigma_2\\ &- 5bn^2 - 4bn\sigma_1 - bn\sigma_2 + 3k^2n + 4k^2\sigma_1 + 8k^2\sigma_2 - 7kn^2 - 9kn\sigma_1 - 2kn\sigma_2 + 4n^3\\ &+ n^2\sigma_1 - n^2\sigma_2 + 3bk - 3bn - b\sigma_1 + 2b\sigma_2 + 2k^2 - 7kn - 2k\sigma_1 + 4k\sigma_2 + 5n^2 - 3n\sigma_1\\ &- 2n\sigma_2 - 2k + 2n - 2\sigma_1))(b+k)(2a+1+2k)(a+k)(-n+k-2)\Big)\end{aligned}}{\begin{aligned}&\Big((-b^2kn\sigma_1 + b^2n^2\sigma_1 + b^2n^2\sigma_2 - 2bk^2n\sigma_1 + 4bkn^2\sigma_1 + 4bkn^2\sigma_2 - 2bn^3\sigma_1 - 2bn^3\sigma_2\\ &+ 2k^2n^2\sigma_1 + 4k^2n^2\sigma_2 - 3kn^3\sigma_1 - 4kn^3\sigma_2 + n^4\sigma_1 + n^4\sigma_2 + b^2k^2 - 2b^2kn - b^2k\sigma_1\\ &+ b^2n^2 + 3b^2n\sigma_1 + 3b^2n\sigma_2 - 2bk^2n - 2bk^2\sigma_1 + 4bkn^2 + 11bkn\sigma_1 + 12bkn\sigma_2 - 2bn^3\\ &- 9bn^2\sigma_1 - 9bn^2\sigma_2 + k^2n^2 + 6k^2n\sigma_1 + 12k^2n\sigma_2 - 2kn^3 - 14kn^2\sigma_1 - 18kn^2\sigma_2 + n^4\\ &+ 6n^3\sigma_1 + 6n^3\sigma_2 - 3b^2k + 3b^2n + 2b^2\sigma_1 + 2b^2\sigma_2 - 3bk^2 + 12bkn + 7bk\sigma_1 + 8bk\sigma_2\\ &- 9bn^2 - 13bn\sigma_1 - 13bn\sigma_2 + 3k^2n + 4k^2\sigma_1 + 8k^2\sigma_2 - 9kn^2 - 21kn\sigma_1 - 26kn\sigma_2\\ &+ 6n^3 + 13n^2\sigma_1 + 13n^2\sigma_2 + 2b^2 + 9bk - 13bn - 6b\sigma_1 - 6b\sigma_2 + 2k^2 - 13kn - 10k\sigma_1\\ &- 12k\sigma_2 + 13n^2 + 12n\sigma_1 + 12n\sigma_2 - 6b - 6k + 12n + 4\sigma_1\\ &+ 4\sigma_2 + 4)(k+1)(b-n+1+2k)(b-n+2k)(2a+1+k)\Big)\end{aligned}}.$$

Daraus leiten wir die Funktionen

$$
\begin{aligned}
p_k &= 1,\\
q_k &= (2(2+2n-2k+n^4\sigma_1+n^4\sigma_2+b^2n^2-2bn^3+3n^3\sigma_1+2n^3\sigma_2+b^2n+b^2\sigma_1+2b^2\sigma_2\\
&\quad -5bn^2+n^2\sigma_1-n^2\sigma_2-3bn-b\sigma_1+2b\sigma_2-3n\sigma_1-2n\sigma_2+b^2n^2\sigma_2-5bn^2\sigma_1-2bn^3\sigma_1\\
&\quad -bn\sigma_2+b^2n^2\sigma_1-5bn^2\sigma_2-2bn^3\sigma_2+3b^2n\sigma_2+2b^2n\sigma_1-4bn\sigma_1+4b(k-1)n^2\\
&\quad +4(k-1)^2n^2\sigma_2+12(k-1)^2n\sigma_2-b^2(k-1)\sigma_1+2(k-1)^2n^2\sigma_1-b^2(k-1)\\
&\quad +b^2(k-1)^2+(k-1)^2n^2+n^4+4n^3+5n^2-3b(k-1)^2+3b(k-1)+3(k-1)^2n\\
&\quad +4(k-1)^2\sigma_1+8(k-1)^2\sigma_2-(2(k-1))n^3-2\sigma_1-(10(k-1))n^2\sigma_2\\
&\quad -(10(k-1))n^2\sigma_1-(3(k-1))n^3\sigma_1-(7(k-1))n^2-b^2(k-1)n\sigma_1-2b(k-1)^2n\sigma_1\\
&\quad +4b(k-1)n^2\sigma_1+4b(k-1)n^2\sigma_2+7b(k-1)n\sigma_1+12b(k-1)n\sigma_2-(4(k-1))n^3\sigma_2\\
&\quad +(4(k-1))\sigma_2-(2(k-1))n\sigma_2-(9(k-1))n\sigma_1-(7(k-1))n-(2(k-1))\sigma_1\\
&\quad +3b(k-1)\sigma_1+6(k-1)^2n\sigma_1+8b(k-1)\sigma_2+8b(k-1)n-2b^2(k-1)n\\
&\quad -2b(k-1)^2n-2b(k-1)^2\sigma_1\\
&\quad +2(k-1)^2))(b+k-1)(2a-1+2k)(a+k-1)(-n-3+k),\\
r_k &= (10-(21(k-1))n\sigma_1-(26(k-1))n\sigma_2+12n-6b-6k+(k-1)^2n^2-3b(k-1)^2\\
&\quad +9b(k-1)+3(k-1)^2n-(14(k-1))n^2\sigma_1-(3(k-1))n^3\sigma_1-(4(k-1))n^3\sigma_2\\
&\quad -(18(k-1))n^2\sigma_2+n^4\sigma_1+n^4\sigma_2+b^2n^2-2bn^3+6n^3\sigma_1+6n^3\sigma_2+3b^2n+2b^2\sigma_1\\
&\quad +2b^2\sigma_2-9bn^2+13n^2\sigma_1+13n^2\sigma_2-13bn-6b\sigma_1-6b\sigma_2+12n\sigma_1+12n\sigma_2\\
&\quad +4(k-1)^2\sigma_1+8(k-1)^2\sigma_2+b^2n^2\sigma_2-9bn^2\sigma_1-2bn^3\sigma_1-13bn\sigma_2+b^2n^2\sigma_1\\
&\quad -9bn^2\sigma_2-2bn^3\sigma_2+3b^2n\sigma_2+3b^2n\sigma_1-13bn\sigma_1+4b(k-1)n^2\sigma_1+4b(k-1)n^2\sigma_2\\
&\quad +11b(k-1)n\sigma_1+12b(k-1)n\sigma_2-(12(k-1))\sigma_2-(13(k-1))n-(10(k-1))\sigma_1\\
&\quad -(9(k-1))n^2+4\sigma_2-(2(k-1))n^3+n^4+6n^3+13n^2+2b^2-b^2(k-1)n\sigma_1\\
&\quad -2b(k-1)^2n\sigma_1+4\sigma_1+2(k-1)^2-3b^2(k-1)+b^2(k-1)^2-b^2(k-1)\sigma_1\\
&\quad -2b^2(k-1)n+12b(k-1)n+4(k-1)^2n^2\sigma_2+2(k-1)^2n^2\sigma_1+7b(k-1)\sigma_1\\
&\quad -2b(k-1)^2\sigma_1+4b(k-1)n^2+12(k-1)^2n\sigma_2+8b(k-1)\sigma_2+6(k-1)^2n\sigma_1\\
&\quad -2b(k-1)^2n)k(b-n-1+2k)(b-n+2k-2)(2a+k)
\end{aligned}
$$

ab. Diese müssen aktualisiert werden, da die 1 in der Dispersionsmenge enthalten ist. Nach der bekannten Ersetzungsregel erhalten wir nun die Funktionen

$$\begin{aligned}
p_k =& -b^2kn\sigma_1 + b^2n^2\sigma_1 + b^2n^2\sigma_2 - 2bk^2n\sigma_1 + 4bkn^2\sigma_1 + 4bkn^2\sigma_2 - 2bn^3\sigma_1 - 2bn^3\sigma_2 \\
&+ 2k^2n^2\sigma_1 + 4k^2n^2\sigma_2 - 3kn^3\sigma_1 - 4kn^3\sigma_2 + n^4\sigma_1 + n^4\sigma_2 + b^2k^2 - 2b^2kn - b^2k\sigma_1 \\
&+ b^2n^2 + 3b^2n\sigma_1 + 3b^2n\sigma_2 - 2bk^2n - 2bk^2\sigma_1 + 4bkn^2 + 11bkn\sigma_1 + 12bkn\sigma_2 - 2bn^3 \\
&- 9bn^2\sigma_1 - 9bn^2\sigma_2 + k^2n^2 + 6k^2n\sigma_1 + 12k^2n\sigma_2 - 2kn^3 - 14kn^2\sigma_1 - 18kn^2\sigma_2 + n^4 \\
&+ 6n^3\sigma_1 + 6n^3\sigma_2 - 3b^2k + 3b^2n + 2b^2\sigma_1 + 2b^2\sigma_2 - 3bk^2 + 12bkn + 7bk\sigma_1 + 8bk\sigma_2 \\
&- 9bn^2 - 13bn\sigma_1 - 13bn\sigma_2 + 3k^2n + 4k^2\sigma_1 + 8k^2\sigma_2 - 9kn^2 - 21kn\sigma_1 - 26kn\sigma_2 \\
&+ 6n^3 + 13n^2\sigma_1 + 13n^2\sigma_2 + 2b^2 + 9bk - 13bn - 6b\sigma_1 - 6b\sigma_2 + 2k^2 - 13kn \\
&- 10k\sigma_1 - 12k\sigma_2 + 13n^2 + 12n\sigma_1 + 12n\sigma_2 - 6b - 6k + 12n + 4\sigma_1 + 4\sigma_2 + 4, \\
q_k =& (2(-n-3+k))(a+k-1)(2a-1+2k)(b+k-1), \\
r_k =& k(b-n-1+2k)(b-n+2k-2)(2a+k).
\end{aligned}$$

Nun erfüllen die Funktionen p_k, q_k und r_k die Bedingung (3.14). Lemma 3.8 liefert für die gesuchte Funktion f_k die obere Gradschranke 0. Wir setzen den Ansatz $f_k = b_0$ in die Rekursionsgleichung (3.11) ein und erhalten die folgende Gleichung

$$\begin{aligned}
0 =& -4 - 12n + 6b - n^4\sigma_1 - n^4\sigma_2 - b^2n^2 + 2bn^3 - 6n^3\sigma_1 - 6n^3\sigma_2 - 3b^2n - 2b^2\sigma_1 \\
&- 2b^2\sigma_2 + 9bn^2 - 13n^2\sigma_1 - 13n^2\sigma_2 + 13bn + 6b\sigma_1 + 6b\sigma_2 - 12n\sigma_1 - 12n\sigma_2 - b^2n^2\sigma_2 \\
&+ 9bn^2\sigma_1 + 2bn^3\sigma_1 + 13bn\sigma_2 - b^2n^2\sigma_1 + 9bn^2\sigma_2 + 2bn^3\sigma_2 - 3b^2n\sigma_2 - 3b^2n\sigma_1 \\
&+ 13bn\sigma_1 - 4\sigma_2 - n^4 - 6n^3 - 13n^2 - 2b^2 - 4\sigma_1 + (2bn\sigma_1 - b^2 - n^2 + 3b - 3n - 4\sigma_1 \\
&- 8\sigma_2 + 2bn - 4n^2\sigma_2 + 2b\sigma_1 - 6n\sigma_1 - 2n^2\sigma_1 - 12n\sigma_2 + (2((2(-n-2))a \\
&+ (-n-2+a)(2a+1) + (-2n-3+4a)b))b_0 - ((b-n-1)(b-n-2) \\
&+ (2(4b-4n-6))a)b_0 - 2)k^2 + ((2(-2n-3+4a+2b))b_0 - (4b-4n-6+8a)b_0)k^3 \\
&+ (2(-n-2))a(2a+1)bb_0 + (6 + 13n - 9b + 3n^3\sigma_1 + 4n^3\sigma_2 + 2b^2n + b^2\sigma_1 \\
&- 4bn^2 + 14n^2\sigma_1 + 18n^2\sigma_2 - 12bn - 7b\sigma_1 - 8b\sigma_2 + 21n\sigma_1 + 26n\sigma_2 - 4bn^2\sigma_1 - 12bn\sigma_2 \\
&- 4bn^2\sigma_2 + b^2n\sigma_1 - 11bn\sigma_1 - (2(b-n-1))(b-n-2)ab_0 + 12\sigma_2 + 2n^3 + 9n^2 + 3b^2 \\
&+ 10\sigma_1 + (2((-n-2)a(2a+1) + ((2(-n-2))a + (-n-2+a)(2a+1))b))b_0)k.
\end{aligned}$$

Nachdem wir einen Koeffizientenvergleich bezüglich k durchgeführt haben und das resultierende lineare Gleichungssystem nach den unbekannten Variablen b_0, σ_1 und σ_2 auflösen,

erhalten wir

$$\begin{aligned} b_0 &= -\frac{b^2-2bn+n^2-3b+3n+2}{4a^2-4ab+4an+b^2-2bn+n^2+6a-3b+3n+2} \\ \sigma_1(n) &= \frac{2ab-2an+2bn-2n^2-2a+3b-5n-3}{(n+1)(2a-b+n+1)} \\ \sigma_2(n) &= \frac{\begin{pmatrix}2ab^2-4abn+2an^2+b^2n-2bn^2+n^3-6ab \\ +6an+2b^2-7bn+5n^2+4a-6b+8n+4\end{pmatrix}}{(n+1)(2a-b+n+1)(n+2+2a-b)}. \end{aligned}$$

Schließlich ergibt sich folgende Rekursionsgleichung zweiter Ordnung

$$\begin{aligned} 0 = &(b-n-1)(b-n-2)(n+2+2a)s(n+2) \\ &+(b-n-1)(2a+3+2n)(n+2+2a-b)s(n+1) \\ &+(n+1)(2a-b+n+1)(n+2+2a-b)s(n). \end{aligned}$$

Wir haben mit dem Zeilberger-Algorithmus zwar eine Rekursionsgleichung gefunden, können jedoch keine geschlossene Form angeben, da die erhaltene Rekursionsgleichung der Ordnung 2 ist. Mit dem Zeilberger-Algorithmus können lediglich Rekursionsgleichungen erster Ordnung in eine geschlossene Form überführt werden. Um die gewünschte geschlossene Form zu erhalten, müssen wir auch an dieser Stelle einen weiteren Algorithmus heranziehen. Mit dem Petkovsek-Algorithmus werden wir in der Lage sein die vom Zeilberger-Algorithmus erhaltene holonome Rekursionsgleichung zweiter Ordnung in eine geschlossene Form umzuwandeln. Wir wenden auf die holonome Rekursionsgleichung 2. Ordnung den Algorithmus 4.12 an, um zunächst alle zugehörigen hypergeometrischen Termlösungen zu finden. Wir werden diesen nicht Schritt für Schritt durchgehen, sondern den implementierten Maple-Befehl aus dem hsum-Paket zur Hilfe nehmen. Mit

```
> rec2hyper((b-n-1)(b-n-2)(n+2+2a)s(n+2)+(b-n-1)(2a+3+2n)·
  (n+2+2a-b)s(n+1)+(n+1)(2a-b+n+1)(n+2+2a-b)s(n)=0,s(n));
```

erhalten wir alle hypergeometrischen Lösungen

$$\left\{\frac{t_{n+1}^{(1)}}{t_n}=-\frac{2a-b+n+1}{b-n-1}, \frac{t_{n+1}^{(2)}}{t_n}=-\frac{(2a-b+n+1)(n+1)}{(2a+n+1)(b-n-1)}\right\}.$$

Nun wenden wir den Algorithmus 4.13 an, um aus der erhaltenen Menge aus hypergeometrischen Lösungen, die als Quotient dargestellt sind, die gewünschte geschlossene Form herzuleiten. Für die geschlossene Form müssen wir die Quotienten mit Hilfe von

Algorithmus 2.11 zunächst in hypergeometrische Terme umwandeln. Es ergeben sich

$$t_n^{(1)} = \frac{(2a-b+1)_n}{(1-b)_n} \quad \text{und}$$
$$t_n^{(2)} = \frac{(1)_n(2a-b+1)_n}{(2a+1)_n(1-b)_n}.$$

Im nächsten Schritt des Algorithmus 4.13 bestimmen wir zwei Anfangswerte für s_n. Wir erhalten

$$s_0 = 1 \quad \text{und}$$
$$s_1 = 1 - \frac{2a(a+\frac{1}{2})}{(2a+1)((\frac{1}{2})b-\frac{1}{2})}.$$

Im vierten Schritt des Algorithmus stellen wir das folgende lineare Gleichungssystem auf

$$\alpha_1 t_0^{(1)} + \alpha_2 t_0^{(2)} = s_0$$
$$\alpha_1 t_1^{(1)} + \alpha_2 t_1^{(2)} = s_1.$$

Die Lösungen des linearen Gleichungssystems lauten

$$\alpha_1 = 1, \quad \alpha_2 = 0.$$

Schließlich können wir mit dem letzten Schritt des Algorithmus die geschlossene Form in Form einer Linearkombination der hypergeometrischen Termlösungen und der errechneten α_j, $j = 1, 2$ angeben:

$$s_n = \alpha_1 t_n^{(1)} + \alpha_2 t_n^{(2)} = \frac{(2a-b+1)_n}{(1-b)_n}.$$

Somit haben wir folgende interessante Identität

$${}_4F_3\left(\left.\begin{matrix}-n, a, a+\frac{1}{2}, b \\ 2a+1, \frac{b-n}{2}, \frac{b-n+1}{2}\end{matrix}\right| 1\right) = \frac{(2a-b+1)_n}{(1-b)_n}$$

bewiesen. □

Als nächstes betrachten wir die hypergeometrische Reihe s_n im Satz:

Satz 5.3 *Es gelte*

$$s_n = \sum_{k=0}^{\infty} \frac{(-1)^k \binom{r-s-k}{k}\binom{r-2k}{n-k}}{r-n-k+1} = -\frac{(-1)^n \Gamma(-s+n)}{(-r+2n-1)\Gamma(n+1)\Gamma(-s)}$$

Beweis: Wir werden versuchen die hypergeometrische Reihe s_n in eine geschlossene Form zu überführen, wobei wir nicht wie in dem vorigen Beispiel auf alle einzelnen Schritte eingehen werden, sondern anhand von Maple-Befehlen aus dem hsum-Paket direkt die jeweilige gesuchte Lösung angeben werden. Zu Beginn werden wir den Gosper-Algorithmus anwenden. Ist dieser erfolgreich, können wir die gesuchte geschlossene Form angeben. Mit dem Summanden

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

erhalten wir mit dem Maple-Befehl
> gosper$(F(n,k),k)$;

dass zum Summanden keine hypergeometrische Antidifferenz existiert. Das heißt, dass wir mit dem Gosper-Algorithmus nicht zum Erfolg kommen. Daher wenden wir den Zeilberger-Algorithmus mit dem Maple-Befehl
> zeilberger$(F(n,k),k,s(n))$;

an. Es wird ausgegeben, dass der Zeilberger-Algorithmus keine Rekursionsgleichung erster Ordnung finden kann. Also sind wir mit dem Zeilberger-Algorithmus ebenfalls nicht erfolgreich. Daher müssen wir uns den Petkovsek-Algorithmus zur Hilfe nehmen.Wir bestimmen zunächst die Rekursionsgleichung mit dem Maple-Befehl
> $rec :=$ sumrecursion$(F(n,k),k,s(n))$;

$$\begin{aligned} rec :=& (n+2)(2n-r+3)(n-r+s+1)s(n+2) \\ &+ (2n+1-r)(2n^2-2rn+rs-s^2+2n-2r)s(n+1) \\ &+ (-r+2n-1)(-r+n-1)(-s+n)s(n) = 0. \end{aligned}$$

Die erhaltene Rekursionsgleichung, welche 2. Ordnung ist, werden wir nun mit dem Petkovsek-Algorithmus in eine geschlossene Form umwandeln können. Hierzu bestimmen wir erst die hypergeometrischen Termlösungen anhand folgenden Maple-Befehls
> rec2hyper$(rec,s(n))$;

$$\left\{ \frac{t_{n+1}^{(1)}}{t_n} = -\frac{(-s+n)(-r+2n-1)}{(n+1)(2n+1-r)}, \frac{t_{n+1}^{(2)}}{t_n} = -\frac{(-r+2n-1)(-r+n-1)}{(-r+s+n)(2n+1-r)} \right\}.$$

Als nächstes wandeln wir erneut die resultierenden Quotienten in hypergeometrische Terme um und erhalten:

$$t_n^{(1)} = (-1)^n \frac{(-\frac{1}{2}r-\frac{1}{2})_n(-s)_n}{(1)_n(\frac{1}{2}-\frac{1}{2}r)_n} \quad \text{und}$$
$$t_n^{(2)} = (-1)^n \frac{(-\frac{1}{2}r-\frac{1}{2})_n(-r-1)_n}{(-r+s)_n(\frac{1}{2}-\frac{1}{2}r)_n}.$$

Im nächsten Schritt bestimmen wir zwei Anfangswerte für s_n. Wir erhalten

$$s_0 = \frac{1}{r+1} \quad \text{und}$$
$$s_1 = 1 - \frac{r-s-1}{r-1}.$$

Anschließend stellen wir das folgende lineare Gleichungssystem wie folgt auf

$$\alpha_1 t_0^{(1)} + \alpha_2 t_0^{(2)} = s_0$$
$$\alpha_1 t_1^{(1)} + \alpha_2 t_1^{(2)} = s_1.$$

Die Lösungen des linearen Gleichungssystems sind

$$\alpha_1 = \frac{1}{r+1}, \quad \alpha_2 = -\frac{\frac{r}{r+1} + \frac{1}{r+1} - 1}{r+1}.$$

Schließlich können wir die geschlossene Form für die Summe s_n durch

$$s_n = \alpha_1 t_n^{(1)} + \alpha_2 t_n^{(2)} = -\frac{(-1)^n \Gamma(-s+n)}{(-r+2n-1)\Gamma(n+1)\Gamma(-s)}$$

ausgeben, wodurch folgende Identität

$$\sum_{k=0}^{\infty} \frac{(-1)^k \binom{r-s-k}{k} \binom{r-2k}{n-k}}{r-n-k+1} = -\frac{(-1)^n \Gamma(-s+n)}{(-r+2n-1)\Gamma(n+1)\Gamma(-s)}$$

hergeleitet wurde. □

Zum Abschluss des Kapitels leiten wir eine weitere Identität her. Hierzu betrachten wir die hypergeometrische Funktion ${}_9F_8$ im Satz:

Satz 5.4 *Es gelte die Identität*

$$\begin{aligned} s_n &= {}_9F_8\left(\left.\begin{matrix} -n, 1, \frac{3}{2}, \frac{3}{2}, \frac{3}{2}, a, 1-a, b, 1-b \\ \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, n+2, 1+a, 2-a, 1+b, 2-b \end{matrix}\right| 1\right) \\ &= \sum_{k=0}^{n} \frac{(-n)_k (1)_k (\frac{3}{2})_k (\frac{3}{2})_k (\frac{3}{2})_k (a)_k (1-a)_k (b)_k (1-b)_k}{(\frac{1}{2})_k (\frac{1}{2})_k (\frac{1}{2})_k (n+2)_k (1+a)_k (2-a)_k (1+b)_k (2-b)_k} \frac{1}{k!} \\ &= \frac{n!(n+1)!}{(a+b-1)(a-b)} \left[\frac{(2b-1)^2 a(a-1)}{(b+1)_n (2-b)_n} - \frac{(2a-1)^2 b(b-1)}{(a+1)_n (2-a)_n} \right]. \end{aligned}$$

Beweis: Wir werden auch bei diesem Beweis zunächst überprüfen, ob wir mit dem Gosper-Algorithmus erfolgreich sein werden. Mit dem Summanden

$$F(n,k) := \frac{(-n)_k(1)_k(\frac{3}{2})_k(\frac{3}{2})_k(\frac{3}{2})_k(a)_k(1-a)_k(b)_k(1-b)_k}{(\frac{1}{2})_k(\frac{1}{2})_k(\frac{1}{2})_k(n+2)_k(1+a)_k(2-a)_k(1+b)_k(2-b)_k}\frac{1}{k!}$$

erhalten wir mit dem Befehl
> gosper$(F(n,k),k)$;

dass zum Summanden keine hypergeometrische Antidifferenz existiert. Folglich kann mit dem Gosper-Algorithmus keine geschlossene Form gefunden werden. Daher wenden wir den Zeilberger-Algorithmus mit dem Maple-Befehl
> zeilberger$(F(n,k),k,s(n))$;

an. Auch hier wird ausgegeben, dass der Zeilberger-Algorithmus keine Rekursionsgleichung erster Ordnung finden kann. Also sind wir mit dem Zeilberger-Algorithmus ebenfalls nicht erfolgreich. Daher müssen wir erneut den Petovsek-Algorithmus benutzen.Wir bestimmen zunächst die Rekursionsgleichung mit dem Befehl
> rec := sumrecursion$(F(n,k),k,s(n))$;

$$\begin{aligned} rec :=&(n+2+b)(-n-3+b)(n+2+a)(-n-3+a)s(n+2)\\ &+(a^2+b^2-2n^2-a-b-8n-8)(n+2)(n+3)s(n+1)\\ &+(n+2)^2(n+1)(n+3)s(n)=0. \end{aligned}$$

Da die erhaltene Rekursionsgleichung zweiter Ordnung ist, benötigen wir den Petkovsek-Algorithmus um diesen in eine geschlossene Form umwandeln zu können. Hierzu bestimmen wir erst die hypergeometrischen Termlösungen anhand von
> rec2hyper$(rec,s(n))$;

$$\left\{\frac{t_{n+1}^{(1)}}{t_n} = -\frac{(n+2)(n+1)}{(n+1+a)(-2+a-n)}, \frac{t_{n+1}^{(2)}}{t_n} = -\frac{(n+2)(n+1)}{(n+1+b)(-2+b-n)}\right\}.$$

Als nächstes wandeln wir erneut die resultierenden Quotienten in hypergeometrische Terme um und bekommen:

$$t_n^{(1)} = \frac{(2)_n(1)_n}{(1+a)_n(2-a)_n} \quad \text{und}$$
$$t_n^{(2)} = \frac{(2)_n(1)_n}{(1+b)_n(2-b)_n}.$$

Im nächsten Schritt bestimmen wir zwei Anfangswerte für s_n. Wir erhalten

$$s_0 = 1 \quad \text{und}$$
$$s_1 = 1 - \frac{9a(1-a)b(1-b)}{(1+a)(2-a)(1+b)(2-b)}.$$

Anschließend stellen wir das folgende lineare Gleichungssystem wie folgt auf

$$\alpha_1 t_0^{(1)} + \alpha_2 t_0^{(2)} = s_0$$
$$\alpha_1 t_1^{(1)} + \alpha_2 t_1^{(2)} = s_1.$$

Die Lösungen des linearen Gleichungssystems sind

$$\alpha_1 = 1 - \frac{a(4ab^2 - 4ab - 4b^2 + a + 4b - 1)}{a^2 - b^2 - a + b}, \quad \alpha_2 = \frac{a(4ab^2 - 4ab - 4b^2 + a + 4b - 1)}{a^2 - b^2 - a + b}.$$

Schließlich können wir die geschlossene Form für die Summe s_n durch

$$\begin{aligned} s_n &= \alpha_1 t_n^{(1)} + \alpha_2 t_n^{(2)} \\ &= \frac{n!(n+1)!}{(a+b-1)(a-b)} \left[\frac{(2b-1)^2 a(a-1)}{(b+1)_n (2-b)_n} - \frac{(2a-1)^2 b(b-1)}{(a+1)_n (2-a)_n} \right] \end{aligned}$$

ausgeben, wodurch folgende Identität

$$\begin{aligned} &{}_9F_8\left(\begin{matrix} -n, 1, \frac{3}{2}, \frac{3}{2}, \frac{3}{2}, a, 1-a, b, 1-b \\ \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, n+2, 1+a, 2-a, 1+b, 2-b \end{matrix} \middle| \, 1 \right) \\ &= \frac{n!(n+1)!}{(a+b-1)(a-b)} \left[\frac{(2b-1)^2 a(a-1)}{(b+1)_n (2-b)_n} - \frac{(2a-1)^2 b(b-1)}{(a+1)_n (2-a)_n} \right] \end{aligned}$$

hergeleitet wurde. □

In diesem Kapitel haben wir gesehen, dass der Petkovsek-Algorithmus ein mächtiges Instrument zum Finden einer geschlossenen Form ist. Denn bei allen drei Beispielidentitäten war nur der Petkovsek-Algorithmus erfolgreich, sowohl der Gosper als auch der Zeilberger-Algorithmus waren nicht in der Lage eine geschlossene Form zu finden. Außerdem haben wir die Maplet-Anwendung kennengelernt, mit welcher das Beweisen von Identitäten deutlich einfacher und schneller geschieht. Im Anhang kann gesehen werden, dass die Maplet-Anwendung die Beispielidentitäten ebenfalls richtig beweisen konnte.

6. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurden mächtige Algorithmen zur algorithmischen Summation von hypergeometrischen Termen vorgestellt. Das Ziel war es die Algorithmen richtig und verständlich darzustellen und ein Tool zur Verfügung zu stellen, welches die Implementierung des Petkovsek-Algorithmus enthält und eine beliebige hypergeometrische Summe in eine geschlossene Form überführt, falls solch eine existiert.

Einer dieser mächtigen Algorithmen ist der Gosper-Algorithmus. Dieser entscheidet ob zu einem hypergeometrischen Term a_k eine hypergeometrische Stammfunktion s_k existiert. Falls dem so ist, kann die Summe mittels Betrachtung der Teleskopsumme in eine geschlossene Form überführt werden. Obwohl der Gosper-Algorithmus in vielen Fällen eine geschlossene Form für die unbestimmte Summation liefert, wird sie hauptsächlich als Einbettung für den Zeilberger-Algorithmus verwendet. Dieser gibt eine holonome Rekursionsgleichung für die Summe aus, welche von n aber nicht mehr von der Summenlaufvariablen k abhängt. In einigen wenigen Fällen liefert der Zeilberger-Algorithmus eine Rekursionsgleichung höherer Ordnung, obwohl eine Rekursion erster Ordnung existiert. Ist die erhaltene Rekursionsgleichung in der Ordnung $J > 1$ oder scheitert der Algorithmus aufgrund zu hoher Komplexität, so ist die Anwendung des Petkovsek-Algorithmus eine gute Möglichkeit, die gewünschte geschlossene Form anhand der vom Zeilberger-Algorithmus erhaltenen Rekursionsgleichung auszugeben. Der Petkovsek-Algorithmus findet zu einer holonomen Rekursionsgleichung alle existierenden hypergeometrischen Lösungen. Falls jedoch die erhaltene Rekursionsgleichung inhomogen ist, so kann diese in eine homogene Rekursionsgleichung transformiert werden, wobei der Grad der Ordnung sich dabei um 1 erhöht. Aus den erhaltenen hypergeometrischen Lösungen durch den Petkovsek-Algorithmus können wir schließlich eine geschlossene Form in Form einer Linearkombination hypergeometrischer Terme angeben.

Gibt der Petkovsek-Algorithmus keine hypergeometrischen Lösungen aus, so kann gefolgert werden, dass keine geschlossene Form existiert. Der Petkovsek-Algorithmus ist nicht die einzige Methode um eine Rekursionsgleichung höherer Ordnung zu bestimmen. Eine viel effizientere Methode bietet Van Hoeij's Algorithmus. Dieser sucht ebenfalls alle hypergeometrischen Lösungen einer holonomen Rekursionsgleichung, jedoch unter Berücksichtigung des lokalen Verhaltens. Genaueres kann in [KOE14] nachgelesen werden.

Zum Abschluss sei noch erwähnt, dass die von mir zur Verfügung gestellte Maplet-Anwendung eine beliebige hypergeometrische Summe in eine geschlossene Form überführt, sofern sie existiert. Als Testfälle habe ich die in Kapitel 5 aufgeführten Beispiele genommen. Im Grunde müsste die Maplet-Anwendung korrekt funktionieren, ist aber nicht

abschließend gezeigt. Die Anwendung ist der Arbeit beigefügt.

A. Anhang

A.1 Maplet-Ausgaben der Beispielidentitäten aus Kapitel 5

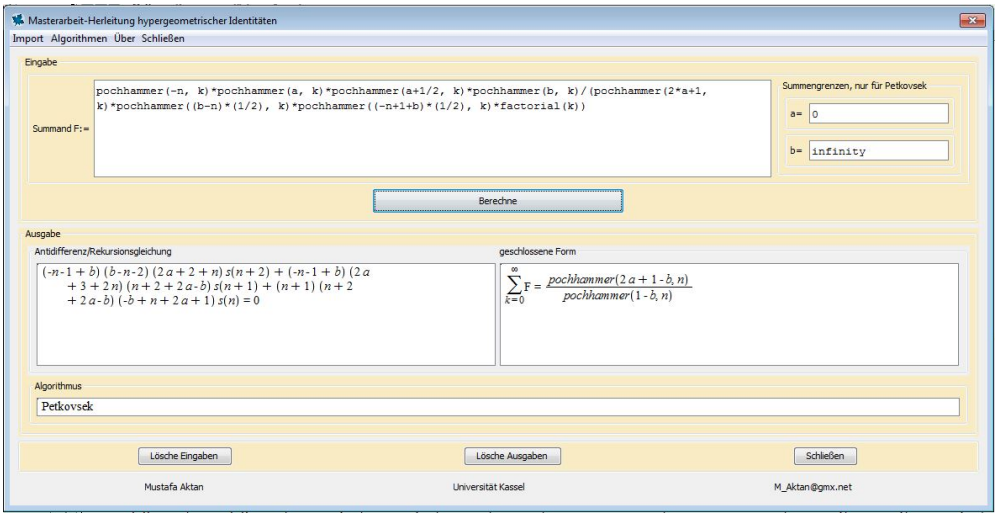

Abbildung A.1: Identität aus Satz 5.2

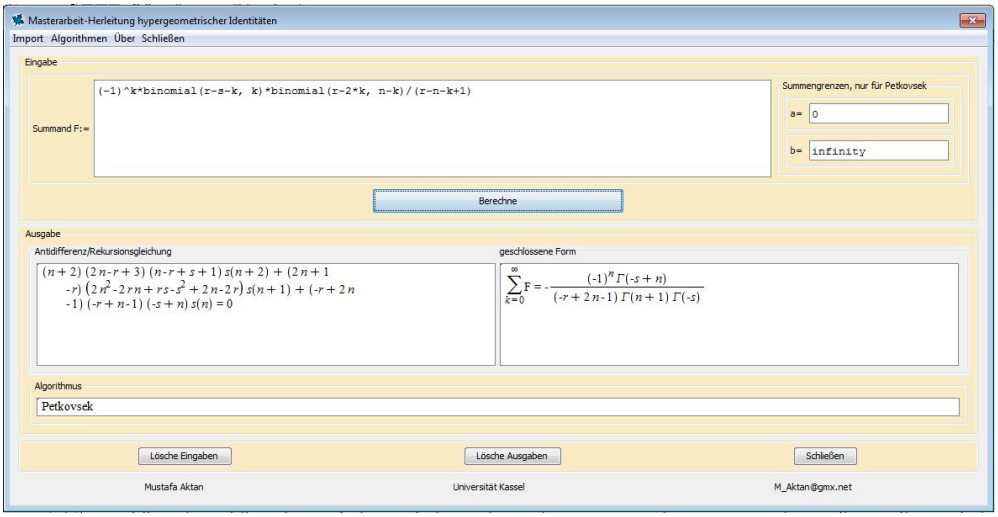

Abbildung A.2: Identität aus Satz 5.3

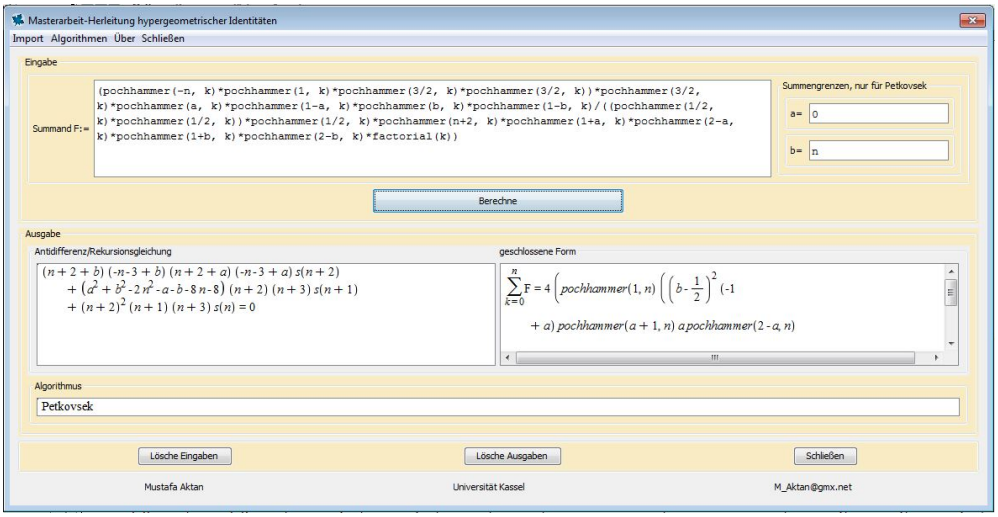

Abbildung A.3: Identität aus Satz 5.4

```
> maple_result := (4 pochhammer(1, n) pochhammer(2, n) ((a-1) a pochhammer(1+a, n) (-1/2+b)^2 pochhammer(2-a, n) - b pochhammer(2-b, n) pochhammer(1+b, n) (b-1) (-1/2+a)^2))/((a-b)(-1+b+a) pochhammer(2-a, n) pochhammer(1+a, n) pochhammer(1+b, n) pochhammer(2-b, n)):
> result := n!(n+1)!/((a+b-1)(a-b)) ((2b-1)^2 a(a-1)/(pochhammer(b+1, n) pochhammer(2-b, n)) - (2a-1)^2 b(b-1)/(pochhammer(a+1, n) pochhammer(2-a, n))):
> evalb(simplify(convert(maple_result, GAMMA)) = simplify(convert(result, GAMMA)))
                                  true
```

Literaturverzeichnis

[BCL82] Buchberger, B., Collins, G., Loos, R.: *Computer Algebra*: Symbolic and Algebraic Computation, Springer, 1982

[GKP94] Graham, R., Knuth, D., Patashnik, O.: *Concrete Mathematics*: A Foundation for Computer Science, 2. Auflage, 1994

[GOS78] Gosper, R. W. J.: *Decision procedure for indefinite hypergeometric summation*: Proc. Natl. Acad. Sci. USA, 1978

[KAP05] Kaplan, M.: *Computeralgebra*, Springer, 2005

[KOE06] Koepf, W.: *Computeralgebra*: Eine algorithmisch orientierte Einführung, Springer, 2006

[KOE14] Koepf, W.: *Hypergeometric Summation*: An Algorithmic Approach to Summation and Special Function Identities, Springer, 2. Auflage, 2014

[MPG15] Bernardin, Chin, DeMarco, Geddes, Hare, Heal, Labahn, May, McCarron, Monagan, Ohashi, Vorkoetter: *Maple Programming Guide*, 2015

[PBM90] Prudnikov, A., Brychkov, Y., Marichev, O.: *Integrals Series*: Volume 3: More Special Functions, 1990

[PFO93] Pforr, E.: *Mathematik für Ingenieure und Naturwissenschaftler*: Differential- und Integralrechnung für Funktionen mit einer Variablen, 9. Auflage, 1993

[PWZ96] Petkovšek, M., Wilf, H., Zeilberger, D.: *A=B*: With Foreword by DONALD E. KNUTH, 1996

[RZ16] Römisch, W., Zeugmann, T.: *Mathematical Analysis and the Mathematics of Computation*, Springer, 2016

[VAR13] Varnhorn, W.: *Maß- und Integrationstheorie*: Vorlesungsunterlagen, WS 2013/2014

[ZEI91] Zeilberger, D.: *The method of creative telescoping*:
Journal of Symbolic Computation **11**, 1991, 195-204

Eidesstattliche Erklärung

Hiermit versichere ich, die Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt zu haben.

Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Kassel, den 12.04.2017

_____

(Mustafa Aktan)