Integration in logarithmischen Erweiterungen

Bachelorarbeit von Ruben Debeerst

Betreuer: Prof. Dr. W. Koepf

Universität Kassel

Juli 2005

Inhaltsverzeichnis

0 Einleitung

Das Integrationsproblem gehört zu den Problemen, die am einfachsten beschrieben werden können: Wir haben eine Funktion f gegeben und suchen eine Funktion F, so dass die Ableitung von F wieder f ergibt:

$$F' = f \quad \text{oder} \quad F = \int f.$$

In der Schule haben wir dazu einige Verfahren kennengelernt, um dieses Problem zu lösen. Bei Polynomen gibt es hierfür die einfache Vorschrift

$$\int \lambda\, x^i dx = \frac{1}{i+1} \lambda\, x^{i+1}\,,$$

welche durch mehrmaliges Anwenden immer zum Ziel führt. Bei rationalen Funktionen wird das Problem mittels Substitution, partieller Integration oder Umformen des Integranden auf „einfach“ integrierbare Funktionen reduziert. Also auf Funktionen wie exp, sin oder log, von denen die Stammfunktion bekannt ist. Allerdings wissen wir nie, ob wir mit diesen Heuristiken auf dem richtigen Weg sind, und manchmal fehlt einfach der richtige „Trick“, um zum Ziel zu gelangen.

Der einzige Ansatz, der immer durchgeführt werden kann, ist die Partialbruchzerlegung. Dabei wird eine rationale Funktion f in mehrere Brüche $f_1 + \cdots + f_n$ zerlegt, wodurch sich das Problem aufgrund der Linearität in mehrere weniger komplexe Probleme aufteilt. Auf die Partialbrüche f_i können dann allerdings wieder nur Heuristiken angewendet werden.

Für Computeralgebrasysteme ist diese Vorgehensweise nicht besonders praktikabel, denn wir können von einem Computer nicht erwarten, dass er den richtigen Trick findet. Dies wird meistens mit großen Tabellen von bekannten Stammfunktionen gelöst.[1] Weiterhin kennen die Systeme Stammfunktionen von gewissen Kombinationen wie $x \exp(ax) \sin(bx)$ und können über Paternmatching – das Erkennen von gleichen Strukturen – feststellen, ob eine bestimmte Kombination vorliegt.[2] Wenn alle diese Versuche fehlschlagen, wird kein Ergebnis ausgegeben und es ist nicht klar, ob es überhaupt eine Stammfunktion gibt.

Die algorithmische Integration verfolgt bei dem Problem einen ganz anderen Weg. Dabei wird die Eingabe auf bestimmte Funktionenklassen eingeschränkt und es stellt sich die Frage, ob es eine Stammfunktion gibt, die sich auf die gleiche Weise ausdrücken lässt. Es werden zum Beispiel nur Funktionen, in denen Logarithmen, Exponentialfunktionen, trigonometrische Funktionen oder algebraische Funktionen auftauchen, betrachtet. Unter algebraischen Funktionen versteht man solche Funktionen, welche Nullstelle eines Polynoms sind, darunter auch die Wurzelfunktionen. Die Funktion $\sqrt{\log(x)}$ ist beispielsweise Nullstelle des Polynoms $p(y) = x - \exp(y^2)$.

Für das oben genannte Beispiel $x \exp(ax) \sin(bx)$ exisiert eine Stammfunktion, welche sich wieder durch diese Funktionen ausdrücken lässt. Bei der Funktion $\exp\left(-x^2\right)$

[1] Tabellen von unbestimmten Integralen befinden sich beispielweise in [BSMM97], Abschnitt 21.5.

[2] Das angegebene Beispiel ist aus [BSMM97], Abschnitt 21.5.4.2, Integral Nr. 463.

geht das aber nicht. Stattdessen wird hierfür eine „neue“ Funktion, die Error-Funktion, definiert:

$$\operatorname{erf}(z) := \frac{2}{\sqrt{\pi}} \int_0^z \exp\left(-x^2\right) dx.$$

Joseph Liouville hat im 19. Jahrhundert erste algebraische Ansätze zur algorithmischen Integration gemacht. Er hat die trigonometrischen Funktionen über die exponentielle Darstellung im Komplexen als spezielle exponentielle Funktionen betrachtet. Die Eingabe hat Liouville dadurch auf algebraische, logarithmische und exponentielle Funktionen eingeschränkt, welche auch als elementar bezeichnet werden. Ein großer Schritt wurde mit dem Satz von Liouville gemacht. Dieser Satz liefert eine Darstellung aller Funktionen, welche eine elementare Stammfunktion haben.

Ab Mitte des 20. Jahrhunderts wurde der Ansatz von Liouville wieder aufgegriffen, insbesondere durch Maxwell Rosenlicht und Robert Risch. Es wurden neue Erkenntnisse aus der Algebra genutzt und die Ableitung wurde durch den allgemeineren Abbildungsbegriff ersetzt. Funktionen werden dann als Elemente eines Differentialkörpers betrachtet und die Ableitung ist das Bild unter einer bestimmten Abbildung. Der daraus entstandene Algorithmus – auch Risch-Algorithmus genannt – bestimmt, ob es eine elementare Stammfunktion gibt und berechtet diese. In *Maple* wird dieser Algorithmus bei der Integration von Funktionen genutzt.

Ziel dieser Arbeit ist es, den logarithmischen Teil des Risch-Algorithmus vorzustellen. In Kapitel 1 wird nach der Einführung einiger Begriffe und Notationen die Differentialalgebra und ihre Anwendung auf die rationale Integration in Grundzügen erklärt. Im Anschluss beschäftigt sich Kapitel 2 mit Differentialkörpererweiterungen, welche uns die Möglichkeit geben, auch Logarithmen als Elemente eines Differentialkörpers darzustellen. Die verschiedenen Erweiterungen führen zu einem Turm von Differentialkörpern und der Integrationsalgorithmus wird mit teilweise sehr ähnlichen Verfahren wie im rationalen Fall erklärt. Kapitel 3 erläutert diesen Algorithmus für den logarithmischen Fall und in Kapitel 4 wird noch einmal kurz auf wichtige Details in der Implementierung eingegangen.

Von den Algorithmen selbst sind in dieser Arbeit nur die wichtigsten Teile in Pseudocode zu finden. Der gesamte Integrationsalgorithmus, der in *Mathematica* implementiert wurde, und alle Beispiele, die in dieser Arbeit vorkommen, befinden auf der beiliegenden CD.

1 Grundlagen

In diesem Kapitel sind einige Definitionen und Algorithmen zusammengestellt, die wir für die Integration in logarithmischen Erweiterungen benötigen.

1.1 Polynome

Unter einem Polynom p in x mit Koeffizienten in einem Körper k verstehen wir eine Summe $p = \sum_{i=0}^{n} a_i x^i$, mit $a_i \in k$ und $a_n \neq 0$. Die höchste Potenz n wird Grad des Polynoms p genannt und mit $\deg p$ bezeichnet. Die Menge dieser Polynome bildet den Polynomring $k[x]$.

Die Menge

$$k(x) := \left\{ \frac{p}{q} \,\middle|\, p, q \in k[x], q \neq 0, \gcd(p, q) = 1 \right\}$$

bezeichnen wir als Quotientenkörper. Über diesem Körper können wir wieder Polynome in einer zweiten Variable y betrachten: $p = \sum_{i=0}^{m} b_i y^i \in k(x)[y]$, mit $b_i \in k(x)$. Für solche Funktionen, die auch bezüglich beider Variablen polynomiell sein können, verwenden wir die Schreibweise $\deg_x p$ bzw. $\deg_y p$, um die Variable anzugeben, für die wir den Grad betrachten wollen. So gilt beispielweise für die Funktion $p = x^3 y^4$: $\deg_x p = 3$ und $\deg_y p = 4$.

1.1.1 Polynomdivision

Der Polynomring $k[x]$ ist ein euklidischer Ring, wir können also zwei Polynome $p_1(x), p_2(x) \in k[x]$ mit Rest teilen:

$$p_1(x) = q_1(x) p_2(x) + r(x), \tag{1.1}$$

wobei $q_1(x), r(x) \in k[x]$ und $\deg_x r(x) < \deg_x p_2(x)$ oder $r(x) = 0$.

Iteriert man diesen Prozess, indem man $p_3(x) := r(x)$ definiert und $p_2(x)$ durch $p_3(x)$ teilt, so erhält man eine Folge von Polynomen $p_i(x)$, bei denen der Grad immer weiter abnimmt. Das letzte Polynom der Folge, das nicht Null ist, ist der größte gemeinsame Teiler der Polynome $p_1(x)$ und $p_2(x)$. Diesen Algorithmus nennt man Euklidischen Algorithmus. Den größten gemeinsamen Teiler bezeichnen wir mit $\gcd(p_1(x), p_2(x))$.

Die Berechnungsvorschrift (1.1), welche angibt, wie man $r(x) = p_i(x)$ aus $p_{i-1}(x)$ und $p_{i-2}(x)$ erhält, wird im Erweiterten Euklidischen Algorithmus verwendet, um zusätzlich eine Darstellung

$$g(x) = \gcd\big(p_1(x), p_2(x)\big) = a(x) p_1(x) + b(x) p_2(x)$$

des größten gemeinsamen Teilers zu finden. Dazu starten wir mit den Gleichungen:

$$\begin{aligned} p_1(x) &= a_1(x) p_1(x) + b_1(x) p_2(x) \\ p_2(x) &= a_2(x) p_1(x) + b_2(x) p_2(x), \\ \text{mit} \quad & a_1(x) = 1, b_1(x) = 0, a_2(x) = 0 \text{ und } b_2(x) = 1. \end{aligned} \tag{1.2}$$

Die Berechnungsvorschrift der $p_i(x)$ wird nun auch auf $a_i(x)$ und $b_i(x)$ angewendet, so dass stets

$$p_i(x) = a_i(x)p_1(x) + b_i(x)p_2(x)$$

gilt. Da das vorletzte $p_i(x)$ der größte gemeinsame Teiler ist, haben wir mit $a_i(x)$ und $b_i(x)$ die gewünschte Darstellung gleichzeitig gefunden.

Beispiel 1.1
Wir berechnen den größten gemeinsamen Teiler von

$$p_1(x) = x^5 + 6x^4 + 8x^3 + x^2 + 3x + 5 \quad \text{und} \quad p_2(x) = x^4 + 2x^3 + 1$$

in $\mathbb{Z}[x]$:

Zunächst teilen wir $p_1(x)$ mit Rest durch $p_2(x)$ und erhalten damit:

$$p_1(x) = (x + 4)p_2(x) + (x^2 + 2x + 1).$$

Den Rest nennen wir $p_3(x)$ und wenn wir die Gleichung umstellen, ergibt sich:

$$p_3(x) = p_1(x) - (x + 4)p_2(x) = x^2 + 2x + 1.$$

Wir definieren $a_0(x), a_1(x), b_0(x)$ und $b_1(x)$ wie in (1.2) und können auch $a_3(x)$ und $b_3(x)$ berechnen:

$$a_3(x) = a_1(x) - (x + 4)a_2(x) = 1$$
$$\text{und} \quad b_3(x) = a_1(x) - (x + 4)b_2(x) = -x - 4.$$

Es gilt:

$$p_3(x) = p_1(x) - (x + 4)p_2(x) = a_3(x)p_1(x) + b_3(x)p_2(x).$$

Die weiteren Divisionen sind in folgender Tabelle zusammengefasst:

i	$p_i(x)$	$q_i(x)$	$a_i(x)$	$b_i(x)$
1	$x^5 + 6x^4 + 8x^3 + x^2 + 3x + 5$	$x + 4$	1	0
2	$x^4 + 2x^3 + 1$	$x^2 - 1$	0	1
3	$x^2 + 2x + 1$	$x/2 + 1/2$	1	$-x - 4$
4	$2x + 2$	0	$1 - x^2$	$x^3 + 4x^2 - x - 3$
5	0			

Der größte gemeinsame Teiler ist also $2x + 2$ und es gilt:

$$2x + 2 = (1 - x^2)p_1(x) + (x^3 + 4x^2 - x - 3)p_2(x).$$

Die Größe des Teilers bezieht sich bei Polynomen auf ihren Grad und dieser wird durch Multiplikation mit einem Element des Koeffizientenkörpers nicht beeinflusst. Der größte gemeinsame Teiler ist also bis auf eine Einheit eindeutig bestimmt und kann sich um ein Element aus k unterscheiden.

Im Gegensatz zu unserer Rechnung wird die *Mathematica*-Funktion **PolynomialGCD** den größten gemeinsamen Teiler immer normiert ausgeben:

```
In[1]:= PolynomialGCD[p1, p2]

Out[1]= x + 1
```

Leider kann **PolynomialGCD** nicht mit mehreren Variablen in einem Polynomring wie $k(x)[y]$ umgehen. Dies ist später für uns besonders wichtig, da Logarithmen als transzendente Erweiterungen und daher als zusätzliche Variable aufgefasst werden. Deswegen nutzen wir für den Erweiterten Euklidischen Algorithmus die Funktion **EEA**.

In **EEA** ist auch noch eine weitere Berechnung implementiert. Bei der Angabe eines zusätzlichen Polynoms $p_3(x)$, welches im Erzeugnis von $p_1(x)$ und $p_2(x)$ liegt (also ein Vielfaches λ vom größten gemeinsamen Teilers $g(x)$ ist), kann man auch Polynome $a(x)$ und $b(x)$ in $k[x]$ finden, so dass

$$p_3(x) = c(x) \cdot g(x) = a(x)p_1(x) + b(x)p_2(x)$$

gilt. Man muss hier nur $a(x)$ und $b(x)$ aus dem Erweitertem Euklidischen Algorithmus mit $c(x)$ multiplizieren. Wenn $a(x) \neq 0$ kann man weiterhin $a(x)$ durch $p_2(x)$ teilen:

$$\begin{aligned} & a(x) = q(x)p_2(x) + r(x) \\ \Longrightarrow \quad & p_3(x) = r(x) \cdot p_1(x) + \big(b(x) + q(x)p_1(x)\big) \cdot p_2(x). \end{aligned}$$

Mit $a(x) := r(x)$ und $b(x) := b(x) + q(x)p_1(x)$ können wir nun sogar sicherstellen, dass $a(x) = 0$ oder $\deg_x a(x) < \deg_x p_2(x)$ gilt. Diese Bedingung brauchen wir später bei der Partialbruchzerlegung.

Details zur Funktion **EEA** und allen anderen zusätzlich in *Mathematica* implementierten Funktionen – in Zukunft Hilfsfunktionen genannt – sind in Kapitel 4 ab Seite 50 zu finden.

1.1.2 Resultanten

Seien $f = a_n x^n + \cdots + a_1 x + a_0$ und $g = b_m x^m + \cdots + b_1 x + b_0$ zwei Polynome mit Koeffizienten im Körper k, so ist die Resultante $\mathrm{res}_x(f, g)$ die Determinante der so genannten Sylvester-Matrix:

$$S(f,g) = \begin{pmatrix} a_n & \cdots & \cdots & \cdots & a_1 & a_0 & & \\ & & & \ddots & & & & \\ & & a_n & \cdots & \cdots & \cdots & a_1 & a_0 \\ b_m & \cdots & b_1 & b_0 & & & & \\ & & \ddots & & & & & \\ & & & \ddots & & & & \\ & & & & \ddots & & & \\ & & & & b_m & \cdots & b_1 & b_0 \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} m\text{-Zeilen} \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} n\text{-Zeilen} \end{matrix}$$

Wie beim Grad $\deg_x$ gibt auch in $\mathrm{res}_x(f, g)$ der Index von res die Polynomvariable an.

Zwei Polynome $f, g \in k[x]$ der Form

$$f = \alpha \prod_{i=1}^{n} \big(x - \alpha_i\big) \quad \text{und} \quad g = \beta \prod_{j=1}^{m} \big(x - \beta_j\big)$$

mit Nullstellen im algebraischen Abschluss von k haben die Resultante:

$$\mathrm{res}_x(f, g) = \alpha^m \beta^n \prod_{i=1}^{n} \prod_{j=1}^{m} \big(\alpha_i - \beta_j\big). \tag{1.3}$$

Der Beweis wird über die Eigenschaften $\text{res}_x(f, g_1 g_2) = \text{res}_x(f, g_1)\,\text{res}_x(f, g_2)$ und $\text{res}_x\left(x - \alpha_i, g\right) = g(\alpha_i)$ geführt. Details hierzu sind in [Bos04] Kapitel 4.4, Korollar 9 zu finden.

In der Darstellung () der Resultante können wir sehen, dass die Resultante genau dann Null ist, wenn f und g eine gemeinsame Nullstelle haben. Diese besondere Eigenschaft werden wir beim Integrationsalgorithmus benötigen.

1.1.3 Quadratfreie Faktorisierung

Ein Polynom $p(x)$ heißt quadratfrei, wenn jeder Faktor der Primfaktorzerlegung genau einmal auftritt, d. h. es existiert kein $p_i(x)$ mit $\deg_x p_i(x) > 0$, so dass $p_i(x)^2 | p(x)$ gilt. Mit der Polynomdivision und dem größten gemeinsamen Teiler können wir die quadratfreie Faktorisierung eines Polynoms $p(x)$ berechnen:

$$\begin{aligned} p(x) &= \prod_{i=1}^{n} p_i(x)^i, \qquad (1.4)\\ \text{mit} \quad & p_i(x) \text{ quadratfrei} \\ \text{und} \quad & \gcd\left(p_i(x), p_j(x)\right) = 1 \ \forall i \neq j. \end{aligned}$$

Dabei kommen in $p_i(x)$ immer die Faktoren von $p(x)$ vor, die genau i-mal in $p(x)$ auftauchen. Im Gegensatz zur vollständigen Primfaktorzerlegung kann die quadratfreie Faktorisierung in polynomieller Zeit berechnet werden, da lediglich Algorithmen für Ableitung und Division sowie der Euklidische Algorithmus notwendig sind.

Es sei ein normiertes Polynom $p(x) \in k[x]$ gegeben. Wir nehmen an $p(x)$ hat die Darstellung (1.4) und wir wollen die Polynome $p_i(x)$ finden. Dazu berechnen wir den größten gemeinsamen Teiler mit der Ableitung:

$$a(x) = \gcd(p(x), p'(x)) = \prod_{i=2}^{n} p_i(x)^{i-1}.$$

Darin tauchen nur die Faktoren auf, die auch in der Ableitung auftauchen. Da in der Ableitung die Vielfachheit der Faktoren um eins abnimmt, haben wir in $a(x)$ alle Faktoren, die mindestens zwemal auftreten, mit verringerter Vielfachheit. Natürlich können in der Ableitung auch noch andere Faktoren auftauchen. Diese Faktoren sind aber keine Teiler von $p(x)$ und kommen deswegen auch nicht im größten gemeinsamen Teiler vor.

Als nächstes bilden wir den Quotienten

$$b(x) = p(x)/a(x) = \prod_{i=1}^{n} p_i(x),$$

in dem jeder Faktor von $p(x)$ genau einmal auftaucht. Dieser Teil wird auch quadratfreier Teil von $p(x)$ genannt. Der Teiler

$$c(x) = \gcd(b(x), a(x)) = \prod_{i=2}^{n} p_i(x)$$

ist das Produkt alle Faktoren, die mindestens zweimal auftauchen, mit einfacher Vielfachheit. Diesen Teil dividieren wir nun noch aus $b(x)$ heraus und erhalten den ersten Teil der quadratfreien Faktorisierung:

$$p_1(x) = b(x)/c(x).$$

Diesen Prozess setzen wir nun mit $a(x)$ fort und finden so die anderen Faktoren. Abgebrochen wird dann, wenn $\deg_x a(x) = 0$ ist.

Eine quadratfreie Faktorisierung ist in *Mathematica* mit **`FactorSquareFree`** möglich. Eine Matrix mit den Faktoren und deren Häufigkeiten erhalten wir durch die Funktion **`FactorSquareFreeList`**.

Beispiel 1.2
Wir betrachten das Polynom

$$p(x) = (x-1)(x+1)(x-2)^2 = x^4 - 4x^3 + 3x^2 + 4x - 4 \in \mathbb{Q}[x]$$

und führen die oben genannten Schritte durch:

$$\begin{array}{rcll} p'(x) & = & 4x^3 - 12x^2 + 6x + 4 & = & 2(x-2)(2x^2 - 2x - 1) \\ a(x) & = & \gcd(p(x), p'(x)) = x - 2 & & \\ b(x) & = & p(x)/a(x) = x^3 - 2x^2 - x + 2 & = & (x-1)(x+1)(x-2) \\ c(x) & = & \gcd(b(x), a(x)) = x - 2 & & \\ p_1(x) & = & b(x)/c(x) = x^2 - 1 & = & (x-1)(x+1) \end{array}$$

Die faktorisierten Darstellungen in der rechten Spalte dienen nur der Übersicht. Ein Computer würde diese Formen während der Berechnung nicht kennen!

Im nächsten Schritt muss nun $a(x)$ quadratfrei faktorisiert werden. Dies ist hier allerdings trivialerweise $p_2(x) = x - 2$ und wir haben die quadrafreie Faktorisierung von $p(x)$ gefunden:

$$p(x) = \prod_{i=1}^{n} p_i(x)^i = (x^2 - 1)(x - 2)^2.$$

Dieses Ergebnis stimmt auch mit dem überein, was *Mathematica* liefert:

```
In[2]:= FactorSquareFree[p]

Out[2]= (x - 2)^2 (x^2 - 1)
```

Bei Polynomen in mehreren Variablen nutzt *Mathematica* die Eigenschaft aus, dass bezüglich jeder einzelnen Variablen quadratfrei faktorisiert werden kann, und findet dadurch mehr Faktoren als der hier beschriebene Algorithmus. Weil wir in unserem Integrationspaket eine genaue quadratfreie Faktorisierung benötigen, erlaubt unsere Hilfsfunktion **`SquareFreeList`** die Angabe einer Polynomvariablen.

1.2 Partialbruchzerlegung

Seien $a(x), b(x) \in k[x]$ Polynome in x und $b(x) = b_1(x)\cdots b_n(x)$ eine Faktorisierung mit $\gcd\big(b_i(x), b_j(x)\big) = 1$ für $i \neq j$, so liefert die Partialbruchzerlegung durch mehrfaches Anwenden des Erweiterten Euklidischen Algorithmus Polynome $a_0(x), \ldots, a_n(x)$, so dass gilt:

$$\begin{aligned} \frac{a(x)}{b(x)} &= a_0(x) + \sum_{i=1}^{n} \frac{a_i(x)}{b_i(x)}\,, \\ \text{mit} \quad & \gcd\big(a_i(x), b_i(x)\big) = 1 \\ \text{und} \quad & 0 \leq \deg_x a_i(x) < \deg_x b_i(x). \end{aligned} \tag{1.5}$$

Um die Darstellung zu finden, gehen wir wie folgt vor: Das Polynom $a_0(x)$ erhält man durch Polynomdivision. Dann bestimmen wir mit dem Erweiterten Euklidischen Algorithmus Polynome $r(x)$ und $s(x)$, so dass gilt:

$$r(x)b_1(x) + s(x)\left(\prod_{i=2}^{n} b_i(x)\right) = a(x).$$

Dies ist möglich, weil $\gcd\big(b_i(x), b_j(x)\big) = 1$ ist und daher auch $\gcd(b_i(x), \prod_{i\neq j} b_j(x)) = 1$ für $i \neq j$ gilt. Teilen wir nun durch $b(x)$, erhalten wir:

$$\frac{a(x)}{b(x)} = \frac{s(x)}{b_1(x)} + \frac{r(x)}{\prod_{i=2}^{n} b_i(x)}\,.$$

Wir setzen also $a_1(x) := s(x)$, wenden die genannten Schritte nochmals auf den Bruch $r(x)/\prod_{i=2}^{n} b_i(x)$ an und erhalten die gewünschte Darstellung.

Wenn eine detailliertere Faktorisierung $b(x) = b_1(x)^{e_1}\cdots b_n(x)^{e_n}$ mit bekannten Exponenten vorliegt, können wir auch eine vollständige Partialbruchzerlegung

$$\begin{aligned} \frac{a(x)}{b(x)} &= a_0(x) + \sum_{i=1}^{n}\sum_{j=1}^{e_i} \frac{a_{ij}(x)}{b_i(x)^j}\,, \\ \text{mit} \quad & \gcd\big(a_{ij}(x), b_i(x)\big) = 1 \\ \text{und} \quad & 0 \leq \deg_x a_{ij}(x) < \deg_x b_i(x) \end{aligned} \tag{1.6}$$

bestimmen. Eine Faktorisierung mit bekannten Exponenten erhalten wir beispielsweise aus der quadratfreien Faktorisierung von $b(x)$.

Durch die einfache Partialbruchzerlegung erhalten wir (1.5) und betrachten nun jeweils einen Summanden $a_i(x)/b_i(x)^j$. Nun teilen wir sukzessive

$$a_i(x) = b_i(x)q(x) + a_{ij}(x)$$

für $j = e_i, \ldots, 1$, wobei im nächsten Schritt $a_i(x) := q(x)$ gilt. Falls am Schluss $a_i(x) \neq 0$ ist, so wird dieser Teil zu $a_0(x)$ addiert. Nachdem dies für alle Summanden $a_i(x)/q_i(x)^j$ durchgeführt wurde, haben wir die vollständige Partialbruchzerlegung gefunden.

Die einfache und vollständige Partialbruchzerlegung wurden in *Mathematica* als Hilfsfunktion **`PartialFraction`** implementiert (siehe Kapitel 4), welche die Polynome $a_i(x)$ bzw. $a_{ij}(x)$ direkt berechnet.

Beispiel 1.3
Wir betrachten

$$f(x) = \frac{A(x)}{B(x)} = \frac{2x^3 - 15x^2 + 38x - 41}{(x-2)(x-3)^2}\,.$$

Offensichtlich haben wir eine Faktorisierung des Nenners $b(x) = b_1(x)b_2(x)^2$ mit Faktoren $b_1(x) = (x-2)$ und $b_2(x)^2 = (x-3)^2$, die keinen gemeinsamen Teiler haben. Die Voraussetzungen an unseren Algorithmus sind also erfüllt.

Zunächst führen wir eine Polynomdivision durch, um den polynomiellen Teil abzuspalten:

```
In[3]:= PolynomialQuotient[A, B, x]

Out[3]= 2

In[4]:= PolynomialRemainder[A, B, x]

Out[4]= x^2 - 4 x - 5
```

$$\Longrightarrow \quad f(x) = a_0 + \frac{a(x)}{b(x)} = 2 + \frac{x^2 - 4x - 5}{(x-2)(x-3)^2}\,.$$

Mit dem Erweitertem Euklidischen Algorithmus suchen wir nun Polynome $r(x)$ und $s(x)$, so dass

$$r(x)b_1(x) + s(x)b_2(x) = a(x)$$

gilt, wobei $\deg_x s(x) < \deg_x b_1(x)$:

```
In[5]:= {s, r} = EEA[b2^2, b1, a, x]//Expand

Out[5]= {-9, 10 x - 38}
```

Mit $a_1 := s(x)$ und $a_2(x) := r(x)$ liefert uns die einfache Partialbruchzerlegung somit:

$$f(x) = 2 + \frac{-9}{x-2} + \frac{10x - 38}{(x-3)^2}\,.$$

Für eine vollständige Zerlegung müssen wir nun noch die Nenner der Partialbrüche durch den quadratfreien Anteil ihres Zählers teilen. Da im ersten Bruch bereits $\deg_x -9 < \deg_x(x-2)$ gilt, ist dieser schon vollständig zerlegt.

Beim zweiten Bruch berechnen wir für $j = 2$:

```
In[6]:= a22 = PolynomialRemainder[a2, b2, x]

Out[6]= -8

In[7]:= q = PolynomialQuotient[a2, b2, x]

Out[7]= 10
```

Für $j = 1$ nehmen mir nun den Quotienten $q(x)$ statt $a_2(x)$ und führen die gleichen Schritte aus:

```
In[8]:= a21 = PolynomialRemainder[q, b2, x]

Out[8]= 10

In[9]:= rest = PolynomialQuotient[q, b2, x]

Out[9]= 0
```

Der Rest, den wir zu a_0 addieren müssten, ist hier 0. In der vollständigen Partialbruchzerlegung wird also lediglich der letzte Bruch unterteilt:

$$f(x) = 2 + \frac{-9}{x-2} + \frac{10}{x-3} + \frac{-8}{(x-3)^2}\,.$$

1.3 Rationale Integration

Bevor wir den Integrationsalgorithmus für rationale Funktionen erklären, führen wir einige Grundbegriffe der Differentialalgebra ein.

Sei k ein Körper und $D : k \to k$ eine Abbildung mit den Eigenschaften

(i) $D(f + g) = D(f) + D(g)$,

(ii) $D(f \cdot g) = D(f) \cdot g + f \cdot D(g)$

für alle $f, g \in k$. Dann heißt D Differentialoperator oder Ableitung und (k, D) Differentialkörper. Wenn klar ist, welcher Differentialoperator gemeint ist, so sprechen wir auch vom Differentialkörper k und schreiben f' statt $D(f)$. Die Menge $k_0 = \{c \in k \mid D(c) = 0\}$ heißt Konstantenkörper des Differentialkörpers k.

Aus den Eigenschaften (i) und (ii) folgen direkt die aus der Differential- und Integralrechnung bekannten Regeln: Linearität, Produktregel, Quotientenregel, Potenzregel und partielle Integration. Die geforderten Eigenschaften an D reichen also, um die Ableitung zu beschreiben, wie wir sie kennen.

Wir betrachten ab jetzt in diesem Abschnitt den Quotientenkörper $k(x) = \mathbb{Q}(x)$ mit der formalen Ableitung $'$ als Differentialoperator und wir werden uns damit beschäftigen, wie eine rationale Funktion $p(x)/q(x) \in k(x)$ integrieren können.

1.3.1 Hermite Reduktion

Zunächst führen wir mit dem Nenner $q(x) \in k[x]$ eine quadratfreie Faktorisierung durch

$$q(x) = \prod_{i=1}^{n} q_i(x)^i$$

und wenden dann die vollständige Partialbruchzerlegung an. Damit erhalten wir:

$$\frac{p(x)}{q(x)} = p_0(x) + \sum_{i=1}^{n} \sum_{j=1}^{i} \frac{p_{ij}(x)}{q_i(x)^j}, \tag{1.7}$$

$$\begin{aligned} \text{mit} \quad & p_0(x), p_{ij}(x), q_i(x) \in k[x], \\ & \gcd\left(p_{ij}(x), q_i(x)\right) = 1 \\ \text{und} \quad & 0 \leq \deg_x p_{ij}(x) < \deg_x q_i(x). \end{aligned} \tag{1.8}$$

Aufgrund der Linearität der Ableitung können wir unser Problem nun wie folgt vereinfachen:

$$\int \frac{p(x)}{q(x)} dx = \int p_0(x) dx + \sum_{i=1}^{n} \sum_{j=1}^{i} \int \frac{p_{ij}(x)}{q_i(x)^j} dx.$$

Die Integration des Polynoms $p_0(x)$ ist mit $\int \lambda x^i dx = (i+1)^{-1} \lambda x^{i+1}$ für uns kein Problem mehr. Wenden wir uns also einem Partialbruch

$$\frac{p_{ij}(x)}{q_i(x)^j}$$

zu. Alle $q_i(x)$ sind quadratfrei, d. h. $\gcd(q_i(x), q_i'(x)) = 1$ und wir können mit dem Erweiterten Euklidischen Algorithmus zwei Polynome $a(x)$ und $b(x)$ finden, so dass

$$a(x) q_i(x) + b(x) q_i'(x) = p_{ij}(x) \tag{1.9}$$

gilt. Teilen wir diese Gleichung durch $q_i(x)^j$ und integrieren danach, so erhalten wir:

$$\int \frac{p_{ij}(x)}{q_i(x)^j} dx = \int \frac{a(x)}{q_i(x)^{j-1}} dx + \int \frac{b(x) q_i'(x)}{q_i(x)^j} dx.$$

Nun wird das letzte Integral partiell integriert, indem wir $b(x)$ ableiten und den Rest $q_i'(x)/q_i(x)^j$ integrieren:

$$\int \frac{q_i'(x)}{q_i(x)^j} dx = \frac{-1}{(j-1) q_i(x)^{j-1}}$$

$$\Longrightarrow \quad \int \frac{b(x) q_i'(x)}{q_i(x)^j} = \frac{-b(x)}{(j-1) q_i(x)^{j-1}} - \int \frac{-b'(x)}{(j-1) q_i(x)^{j-1}}.$$

Diese Umformungen liefern uns die Hermite Reduktion

$$\int \frac{p_{ij}(x)}{q_i(x)^j} = \frac{-b(x)/(j-1)}{q_i(x)^{j-1}} + \int \frac{a(x) + b(x)'/(j-1)}{q_i(x)^{j-1}} \tag{1.10}$$

eines Partialbruchs.

Durch die Bedingung (1.8) und die Anwendung des Erweiterten Euklidischen Algorithmus in (1.9) gilt:

$$\deg_x a(x) < \deg_x q_i'(x) = \deg_x q_i(x) - 1 \quad \text{und} \quad \deg_x b(x) < \deg_x q_i(x).$$

Damit gilt auch:

$$\deg_x \left(a(x) + b'(x)/(j-1)\right) \leq \max\left(\deg_x a(x), \deg_x b'(x)\right) < \deg_x q_i(x).$$

Der verbleibende Integrand der Hermite Reduktion (1.10) erfüllt also noch stets die Grad-Bedingungen (1.8) eines Partialbruchs und wir können den Reduktionsprozess so lange fortsetzen, bis der Exponent $j = 1$ ist und nur noch quadratfreie Faktoren mit einfacher Vielfachheit im Nenner stehen.

Beispiel 1.4
Wir betrachten $f(x) \in \mathbb{Q}(x)$ mit Zähler $p(x) = 4x^3 - 6x + 4$ und Nenner $q(x)^3 = (x^2+2)^3$. Es gilt $q'(x) = 2x$ und somit $\gcd(q(x), q'(x)) = 1$. Wir können also $a(x)$ und $b(x)$ finden, so dass gilt:

$$a(x)q(x) + b(x)q'(x) = p(x).$$

```
In[10]:= {a, b} = EEA[q, D[q, x], p, x]

Out[10]= {2, 2 x^2 - x - 3}
```

Wir berechnen noch die Ableitung $b'(x) = 4x - 1$ und haben den ersten Reduktionsschritt:

$$\int f(x)dx = \frac{-2x^2 + x + 3}{2(x^2+2)^2} + \int \frac{2 + (4x-1)/2}{(x^2+2)^2}dx.$$

Im rechten Term haben wir den Zähler $p(x) = 2x + 3/2$ und wir reduzieren erneut:

```
In[11]:= {a, b} = EEA[q, D[q, x], p, x]//Expand

Out[11]= {3/4, 1 - 3 x/8}
```

Es ist $b'(x) = -3/8$ und $a(x) + b'(x)/1 = 3/4 + (-3/8) = 3/8$

$$\Longrightarrow \quad \int f(x)dx = \frac{-2x^2 + x + 3}{2(x^2+2)^2} + \frac{-1 + 3x/8}{x^2+2} + \int \frac{3/8}{x^2+2}dx.$$

Die Integration des Restes $3/\left(8\,(x^2+2)\right)$ kann nicht mehr durch die Hermite Reduktion erfolgen.

1.3.2 Horowitz Reduktion

Da wir den polynomiellen Teil $p_0(x)$ sehr einfach integrieren können, betrachten wir nun nur eine rationale Funktion $p(x)/q(x)$ mit $\gcd(p(x), q(x)) = 1$, $q(x)$ normiert und $\deg_x p(x) < \deg_x q(x)$ und schauen uns eine Alternative zur Hermite Methode an.

Wenn man alle reduzierten Terme der Hermite Reduktion zu

$$\int \frac{p(x)}{q(x)} = \frac{p_1(x)}{q_1(x)} + \int \frac{p_2(x)}{q_2(x)} \tag{1.11}$$

zusammenfasst, stellt sich heraus, dass die beiden Nenner durch

$$q_1(x) = \gcd(q(x), q'(x)) \quad \text{und} \quad q_2(x) = q(x)/q_1(x)$$

festgelegt werden. Es gilt zusätzlich:

$$\deg_x p_1(x) < \deg_x q_1(x) \quad \text{und} \quad \deg_x p_2(x) < \deg_x q_2(x).$$

Für $p_1(x) = 1$ ist $q_1 = \gcd(q(x), q'(x)) = 1$ und damit fällt der linke Teil der Reduktion (1.10) weg. Andernfalls haben sowohl $q_1(x)$ als auch $q_2(x)$ einen echt positiven Grad und wir können für $p_1(x)$ und $p_2(x)$ wie folgt ansetzen:

$$\begin{aligned} p_1(x) &= \lambda_n x^n + \cdots + \lambda_1 x + \lambda_0 \\ \text{und} \quad p_2(x) &= \mu_m x^m + \cdots + \mu_1 x + \mu_0. \end{aligned}$$

Dies setzen wir in (1.11) ein und leiten ab:

$$\frac{p(x)}{q(x)} = \frac{p_1'(x)q_1(x) - p_1(x)q_1'(x)}{q_1(x)^2} + \frac{p_2(x)}{q_2(x)}\,.$$

Jetzt multiplizieren wir mit $q(x) = q_1(x)q_2(x)$:

$$\Longrightarrow \quad p(x) = q_2(x)p_1'(x) - p_1(x)\left(\frac{q_2(x)q_1'(x)}{q_1(x)}\right) + q_2(x)p_2(x).$$

Man kann zeigen, dass der einzige hier auftauchende Bruch in Wirklichkeit auch ein Polynom ist. Damit liegt eine Polynomgleichung vor und wir können durch Koeffizientenvergleich die Gleichung sehr leicht lösen. Details zur Herleitung sowie der zugehörige Beweis sind in [GCL92], Kapitel 11.4 zu finden.

1.3.3 Rothstein/Trager Resultante

Es bleibt nun also eine Funktion $p(x)/q(x)$ mit normiertem, quadratfreiem Nenner $q(x)$ und $\deg_x p(x) < \deg_x q(x)$ zu integrieren. Dies geschieht mit der so genannten Rothstein/Trager Resultante. Es gilt:

$$\int \frac{p(x)}{q(x)} = \sum_{i=1}^{n} c_i \log(v_i),$$

wobei c_i die verschiedenen Nullstellen von

$$R(z) = \operatorname{res}_x \big(p(x) - zq'(x), q(x)\big) \in k[z]$$

sind und v_i durch

$$v_i = \gcd\big(p(x) - c_i q'(x), q(x)\big) \in k[x]$$

definiert wird. Für c_i und v_i muss man gegebenenfalls den Konstantenkörper k_0 geeignet erweitern.

Für Funktionen in logarithmischen Erweiterungen werden wir in Kapitel 3 eine ähnliche Aussage beweisen und verzichten deshalb hier auf die Herleitung. Der Beweis für den rationalen Fall ist in [GCL92], Kapitel 11.5 zu finden.

Die nächsten Beispiele werden die Funktionsweise der Rothstein/Trager Methode zeigen.

Beispiel 1.5
Sei $f(x) = 3/(x+2) \in \mathbb{Q}(x)$. Der Bruch ist vollständig gekürzt und hat einen quadratfreien Nenner, welcher einen größeren Grad hat als der Zähler. Wir berechnen also die Resultante $R(z)$ von

$$\begin{aligned} a(x) &= p(x) - zq'(x) = 3 - z \\ \text{und} \quad b(x) &= q(x) = x + 2 \end{aligned}$$

mit der Formel (1.1.2) aus Abschnitt 1.1.2:

$$R(z) = \operatorname{res}_x(a(x), b(x)) = \alpha^n \beta^m \prod_{i=1}^{m} \prod_{j=1}^{n} \left(\alpha_i - \beta_j\right).$$

Da kein x die Gleichung $a(x) = 3 - z = 0$ löst, gibt es keine Nullstellen α_i. Also ist das Produkt leer und die Resultante ist das Produkt der höchsten Koeffizienten:

$$R(z) = 3 - z.$$

Die einzige Nullstelle c_1 von $R(z)$ ist $c_1 = 3$ und wir berechnen:

$$v_1 = \gcd\left(p(x) - c_1 q'(x), q(x)\right) = \gcd\left(0, q(x)\right) = q(x) = x + 2.$$

Die Stammfunktion ist somit:

$$\int f(x)dx = c_1 \log v_1 = 3\log(x+2).$$

Wenn wir die Rothstein/Trager Methode auf den Restintegrand in Beispiel 1.4 anwenden, sehen wir, dass durch die Berechnung aller Nullstellen ein geeigneter Erweiterungskörper gefunden wird.

Beispiel 1.6
Es sei

$$f(x) = \frac{3/8}{x^2 + 2}.$$

Wir berechnen also die Resultante $R(z)$ von

$$\begin{aligned} a(x) &= p(x) - zq'(x) = 3/8 - z \cdot 2x \\ \text{und} \quad b(x) &= q(x) = x^2 + 2 : \end{aligned}$$

```
In[12]:= r = Resultant[a, b, x]
```

$$\text{Out[12]= } \frac{1}{64}\left(512\, z^2 + 9\right)$$

Die Nullstellen von $R(z)$ sind

```
In[13]:= c = z/.Solve[r == 0, z]
```

$$\text{Out[13]= } \left\{-\frac{3\,i}{16\sqrt{2}}, \frac{3\,i}{16\sqrt{2}}\right\}$$

und die zugehörigen Logarithmen

```
In[14]:= v = PolynomialGCD[p - c D[q, x], q, Extension → Automatic]
Out[14]= {i x + √2, √2 - i x}
```

Wir setzen die Ergebnissen zusammen und erhalten:

$$\int f(x)dx = \sum_{i=1}^{n} c_i \log(v_i) = -\frac{3i \log\left(ix + \sqrt{2}\right)}{16\sqrt{2}} + \frac{3i \log\left(\sqrt{2} - ix\right)}{16\sqrt{2}}.$$

Die vorgestellten Algorithmen zur rationalen Integration wurden in den Funktionen **`IntPoly`**, **`IntRat`** und **`IntRatRothsteinTrager`** implementiert. Dadurch konnte komplett auf die interne Integrationsroutine von *Mathematica* verzichtet werden.

2 Differentialkörpererweiterungen

Unser Ziel ist es, auch Funktionen, in denen Logarithmen vorkommen, algorithmisch zu integrieren. Bei der Integration von rationalen Funktionen haben wir dafür Differentialkörper verwendet. Um nun auch Logarithmen mit einem Element eines Körpers zu identifizieren, benötigen wir Differentialkörpererweiterungen. In den nächsten Definitionen und Sätzen werden wir diesen Begriff definieren und einige Eigenschaften betrachten. Dabei bahandeln wir auch algebraische und exponentielle Erweiterungen und konzentrieren uns danach auf den Integrationsalgorithmus in logarithmischen Erweiterungen.

Definition 1 (Differentialkörpererweiterungen) *Ist (k, D_1) ein Differentialkörper, so nennen wir (K, D_2) eine Differentialkörpererweiterung, falls K ein Erweiterungskörper von k mit $K_0 = k_0$ ist, welcher D_1 nach K fortsetzt, d. h. $D_2(x) = D_1(x)$ für alle $x \in k$. Statt $D_1(x)$ und $D_2(x)$ können wir auch wieder x' schreiben.*

(i) Ein Element $t \in K$ heißt algebraisch über k, falls es ein Polynom $p \in k[x]$ gibt, so dass $p(t) = 0$. Ein Element heißt transzendent über k, falls es nicht algebraisch über k ist.

(ii) Ein Element $t \in K$ heißt logarithmisch über k, falls es transzendent über k ist und es ein $u \in k$ gibt, so dass $t' = u'/u$ (z.B. $t = \log u$).

(iii) Ein Element $t \in K$ heißt exponentiell über k, falls es transzendent über k ist und es ein $u \in k$ gibt, so dass $t'/t = u'$ (z.B. $t = \exp u$).

Eine Erweiterung $K = k(t_1, \ldots, t_N)$ heißt elementar über k, wenn für alle $i = 1, \ldots, N$ gilt: t_i ist algebraisch, logarithmisch oder exponentiell über $K_{i-1} = k(t_1, \ldots, t_{i-1})$. Jedes $t \in K$ heißt dann elementar über k.

Wenn t logarithmisch oder exponentiell über k ist, dann bezeichnen wir ein Polynom in t mit Koeffizienten aus k als logarithmisches bzw. exponentielles Polynom.

Es muss nicht zwingend vorausgesetzt werden, dass eine elementare Erweiterung den gleichen Konstantenkörper hat. Wir haben es hier allerdings so definiert, weil wir uns in dieser Arbeit nur mit solchen Erweiterungen beschäftigen werden. Weiterhin werden wir nur Körper der Charakteristik Null betrachten und dies immer stillschweigend voraussetzen.

Für einen Quotienten u'/u gibt es folgende Regeln, die auch als logarithmische Regeln bezeichnet werden:

$$\frac{(a\,b)'}{a\,b} = \frac{a'}{a} + \frac{b'}{b} \quad \text{und} \quad \frac{(a/b)'}{a/b} = \frac{a'}{a} - \frac{b'}{b}\,.$$

Mit Differentialkörpererweiterungen ist es möglich, für jede Funktion in logarithmischen und exponentiellen Ausdrücken einen Körper anzugeben, in dem diese Funktion liegt. Durch die Darstellungen von Sinus und Kosinus als exponentielle Funktionen im

Komplexen werden auch diese Funktionen durch unsere Definition erfasst und liegen in einer elementaren Erweiterung.

Ganz allgemein betrachten wir Elemente t über k, deren Ableitung ein Polynom in $k[t]$ ist. Diese Polynome wären bei logarithmischen und exponentiellen Erweiterungen u'/u bzw. $u't$ für ein beliebiges $u \in k$. Auf ähnliche Weise kann man eine Erweiterung definieren, in der die Tangens- oder Arcustanges-Funktion liegt. Die zugehörigen Polynome wären $t^2 + 1$ bzw. $1/(x^2 + 1)$.

Ein sehr wichtiges Instrument für den Aufbau unseres Integrationsalgorithmus wird der Satz von Liouville sein. Dieser liefert uns für Funktionen $f \in k$, die eine Stammfunktion F in einer beliebigen Differentialkörpererweiterung K haben, die Darstellung:

$$F' = f = v_0' + \sum_{i=1}^{m} c_i \frac{v_i'}{v_i},$$

mit $v_0, v_1, \ldots, v_m \in k$ und Konstanten $c_1, \ldots, c_m \in k_0$. Mit dieser Darstellung kann die Stammfunktion

$$F = \int f = v_0 + \sum_{i=1}^{m} c_i \log(v_i)$$

direkt angegeben werden, aber leider ist der Beweis nicht konstruktiv. Die Darstellung von Liouville ist jedoch für die Beweise zum Integrationsalgorithmus von zentraler Bedeutung.

Der Beweis zum Satz von Liouville wird in einem Zwischenschritt Polynome $v_i(t) \in k[t]$ und auch deren Ableitungen $v_i'(t)$ liefern. Um Aussagen über die Ableitungen machen zu können, untersuchen wir zunächst einige allgemeine Eigenschaften von Ableitungen in Differentialkörpererweiterungen.

2.1 Ableitung elementarer Funktionen

Satz 2 (Ableitung logarithmischer Polynome) *Sei k ein Differentialkörper und $K = k(t)$ eine Differentialkörpererweiterung. Außerdem sei t transzendent und logarithmisch über k mit $t' = u'/u$ für ein geeignetes $u \in k$.*
Dann gilt für alle $p(t) \in k[t]$ mit $\deg_t p(t) > 0$:

(i) $p'(t) \in k[t]$.

(ii) Falls der führende Koeffizient des Polynoms $p(t)$ konstant ist, so ist $\deg_t p'(t) = \deg_t p(t) - 1$.

(iii) Falls der führende Koeffizient des Polynoms $p(t)$ nicht konstant ist, dann gilt $\deg_t p'(t) = \deg_t p(t)$.

Beweis: Angenommen $p(t)$ hat die Darstellung:

$$p(t) = \sum_{i=0}^{n} p_i t^i,$$

mit $p_n \neq 0, n > 0$. Dann gilt für die Ableitung

$$p'(t) = \sum_{i=0}^{n-1} \left(p_i' + (i+1)\, p_{i+1}\, t'\right)\, t^i + p_n'\, t^n.$$

Da $t' = u'/u \in k$ gilt, folgt (i).
Falls $p_n' \neq 0$, d. h. p_n ist nicht konstant, dann gilt $\deg_t p'(t) = \deg_t p(t) = n \Rightarrow (iii)$.
Falls $p_n' = 0$, d. h. p_n ist konstant, dann gilt $\deg_t p'(t) < n$. Wir beweisen nun (ii) durch Widerspruch.
Angenommen $p_{n-1}' + n\, p_n\, t' = 0$. Betrachte $n\, p_n\, t + p_{n-1}$:
$\Longrightarrow (n\, p_n\, t + p_{n-1})' = n\, p_n'\, t + n\, p_n\, t' + p_{n-1}' = 0$
$\Longrightarrow n\, p_n\, t + p_{n-1} \in k(t)$ und wäre konstant.
Dies widerspricht jedoch der Voraussetzung, dass t transzendent über k ist.
Also muss gelten $p_{n-1}' + n\, p_n\, t' \neq 0$ und damit folgt (ii). □

Der Satz beweist, dass die Ableitung eines logarithmischen Polynoms $p(t)$ wieder ein logartithmisches Polynom ist. Dabei gilt immer $\deg_t p'(t) \leq \deg_t p(t)$. Zusätzlich wissen wir, dass wir eine Gleichheit haben, wenn der höchste Koeffizient nicht Null ist, und ansonsten eine echte Ungleichung.

Schauen wir uns nun einige Eigenschaften der exponentiellen Polynome an.

Satz 3 (Ableitung exponentieller Polynome) *Sei k ein Differentialkörper und $K = k(t)$ eine Differentialkörpererweiterung. Außerdem sei t transzendent und exponentiell über k mit $t'/t = u'$ für ein geeignetes $u \in k$.*
Dann gilt für alle $p(t) \in k[t]$ mit $\deg_t p(t) > 0$:

- *(i) Falls $\lambda \in k, \lambda \neq 0, n \in \mathbb{Z}, n \neq 0$, dann ist $(\lambda t^n)' = \overline{\lambda} t^n$, mit $\overline{\lambda} = \lambda' + n\lambda u' \in k$ und $\overline{\lambda} \neq 0$.*
- *(ii) $p'(t) \in k[t]$ und $\deg_t p'(t) = \deg_t p(t)$.*
- *(iii) Die Teilbarkeitsbedingung $p'(t) = c\, p(t)$ gilt genau dann für ein $c \in k[t]$, wenn $p(t)$ ein Monom bzgl. t ist.*

Beweis: (i) Es gilt: $(\lambda\, t^n)' = \lambda'\, t^n + n\, \lambda\, t^{n-1} t' = (\lambda' + n\, \lambda\, u')t^n$.
Noch z. z.: $\overline{\lambda} = \lambda' + n\, \lambda\, u' \neq 0$, falls $\lambda \neq 0$ und $n \neq 0$.
Angenommen $\overline{\lambda} = 0$. Dann folgt: $(\lambda t^n)' = 0 \Rightarrow \lambda t^n \in k_0$. Das Monom $\lambda t^n \in k_0$ ist konstant, aber dies widerspricht der Voraussetzung, dass t transzendent ist. Also gilt $\overline{\lambda} \neq 0$.

(ii) $p(t)$ habe die Darstellung:

$$p(t) = \sum_{i=0}^{n} p_i\, t^i, \quad p_n \neq 0, n \neq 0.$$

Mit (i) folgt:

$$p'(t) = \sum_{i=1}^{n} \left(p_i t^i\right)' = \sum_{i=0}^{n} \overline{p_i}\, t^i, \quad \overline{p_i} \in k \backslash \{0\}.$$

$\Longrightarrow \deg_t p(t) = \deg_t p'(t) \Longrightarrow$ (ii).

(iii) „$\Leftarrow$“ Angenommen $p(t)$ ist ein Monom: $p(t) = p_n t^n, p_n \in k, n > 0$. Dann folgt mit (i): $p'(t) = \bar{p}_n t^n \Longrightarrow p'(t) = c\,p(t)$ für $c = \bar{p}_n / p_n \in k$.
„$\Rightarrow$“ Angenommen $p'(t) = c\,p(t)$ für ein $c \in k[t]$, aber $p(t)$ sei kein Monom, d. h. $p(t) = p_n\,t^n + p_m\,t^m + b(t)$ mit $p_n \neq 0, p_m \neq 0, n > m \geq 0$ sowie $b(t) = 0$ oder $\deg_t b(t) < m$. Dies wollen wir jetzt zum Widerspruch führen.
Wegen (ii) gilt $c \in k$. Betrachten wir zunächst den Fall $m = 0$ und $p'_m = p'_0 = 0$:
Die Ableitung der Funktion $p(t) = p_n t^n + p_0$ ist $p'(t) = \bar{p}_n t^n = (p'_n + n\,p_n u')t^n$. Damit hat die Teilbarkeitsbedingung die Form $(p'_n + n\,p_n u')t^n = c\,p_n t^n + c\,p_0$. Ein Koeffizientenvergleich bei t^0 würde $c = 0$ liefern, aber da $n\,p_n u' \neq 0$ gilt, löst dieses c nicht die Gleichung.
Also können m und p'_m nicht beide 0 sein $\Longrightarrow \bar{p}_m = p'_m + n\,p_m u' \neq 0$. Für unser spezielles $p(t) = p_n\,t^n + p_m\,t^m + b(t)$ haben wir also die Ableitung:

$$p'(t) = \bar{p}_n t^n + \bar{p}_m t^m + b'(t), \tag{2.1}$$

mit $p_n \neq 0, p_m \neq 0$ und $b'(t) = 0$ oder $\deg_t b'(t) < m$. Koeffizientenvergleich bei t^n und t^m in der Teilbarkeitsbedingung $p'(t) = c\,p(t)$ liefert mit (2.1) das Gleichungssystem:

$$\begin{aligned} p'_n + n\,p_n\,u' &= c\,p_n \\ p'_m + m\,p_m\,u' &= c\,p_m. \end{aligned}$$

Wir teilen durch p_n bzw. p_m, setzen gleich und wenden die logarithmischen Regeln an:

$$\begin{aligned} & \frac{p'_n}{p_n} + n\,u' = \frac{p'_m}{p_m} + m\,u' \\ \Longleftrightarrow \quad & \frac{(p_n/p_m)'}{p_n/p_m} + (n-m)\,u' = 0. \end{aligned}$$

Betrachten wir nun die Ableitung von $p(t) = (p_n/p_m)t^{n-m} \in k[t]$ mit (i):

$$\begin{aligned} \left(\frac{p_n}{p_m}t^{n-m}\right)' &= \left(\left(\frac{p_n}{p_m}\right)' + (n-m)\frac{p_n}{p_m}u'\right)t^{n-m} \\ &= \frac{p_n}{p_m}t^{n-m}\underbrace{\left(\frac{(p_n/p_m)'}{p_n/p_m} + (n-m)u'\right)}_{=0} = 0. \end{aligned}$$

$(p_n/p_m)t^{n-m}$ wäre also konstant, was aber der Transzendenz von t widerspricht. Damit war unsere Annahme falsch und (iii) ist bewiesen. □

2.2 Satz von Liouville

Nachdem wir nun die benötigten Eigenschaften kennen, können wir den zuvor erwähnten Satz von Liouville beweisen.

Satz 4 (Satz von Liouville) *Sei k ein Differentialkörper mit Differentialkörpererweiterung K. Wenn es zu $f \in k$ ein über k elementares $F \in K$ mit $F' = f$ gibt, so existieren Elemente $v_0, v_1, \ldots, v_n \in k$ und Konstanten $c_1, \ldots, c_n \in k_0$, mit*

$$F' = f = v_0' + \sum_{i=1}^{n} c_i \frac{v_i'}{v_i} \tag{2.2}$$

$$\textit{bzw.} \quad F = \int f = v_0 + \sum_{i=1}^{n} c_i \log(v_i). \tag{2.3}$$

Beweis: $F = \int f$ ist elementar über k, d. h. wir können $t_1, \ldots, t_N$ so finden, dass $K = k\left(t_1, \ldots, t_N\right)$, wobei jedes t_i $(1 \leq i \leq N)$ logarithmisch, exponentiell oder algebraisch über $k\left(t_1, \ldots, t_{i-1}\right)$ ist. Der Beweis wird nun mit Induktion nach N geführt:
$N = 0$: $F \in k$ und mit $m = 0$ und $v_0 = F$ ist die gesuchte Darstellung gefunden.
$N - 1 \to N$:
Da $k(t_1, \ldots, t_N) = k(t_1)(t_2, \ldots, t_N)$, haben wir nach Induktionsvoraussetzung bereits die Darstellung:

$$f = v_0'(t_1) + \sum_{i=1}^{\tilde{n}} c_i \frac{v_i'(t_1)}{v_i(t_1)}, \tag{2.4}$$

mit $v_i(t_1) \in k(t_1), c_i \in k_0 (1 \leq i \leq \tilde{n})$. Wir schreiben nun $t := t_1$ und unterscheiden die drei Fälle t algebraisch, t logarithmisch oder t exponentiell.

t algebraisch:
Für den Beweis des algebraischen Falls muss man sich noch genauer mit Differentialkörpern beschäftigen. Seien $\mathrm{Tr} : k(t_1) \to k$ die Spur- und $\mathrm{N} : k(t_1) \to k$ die Normabbildung von $k(t_1)$ nach k. Man kann dann folgende Aussagen beweisen:

$$\mathrm{Tr}(v') = (\mathrm{Tr}(v))' \quad \text{und} \quad \mathrm{Tr}\left(\frac{v'}{v}\right) = \frac{(\mathrm{N}(v))'}{\mathrm{N}(v)}.$$

Details hierzu sind in [Bro97], Kapitel 3.2 zu finden. Die beiden benötigten Aussagen werden dort in Theorem 3.2.4 bewiesen.

Sei nun $m = [k(t_1) : k]$ der Körpergrad von $k(t_1)$ über k. Wir wenden auf die Gleichung (2.4) die Spur-Abbildung an und erhalten wegen $\mathrm{Tr}(f) = m f$:

$$\begin{aligned}
& \mathrm{Tr}(f) = \mathrm{Tr}\left(v_0'(t_1)\right) + \sum_{i=1}^{\tilde{n}} c_i \,\mathrm{Tr}\left(\frac{v_i'(t_1)}{v_i(t_1)}\right) \\
\iff \quad & m f = \mathrm{Tr}\left(v_0(t_1)\right)' + \sum_{i=1}^{\tilde{n}} c_i \frac{N(v_i(t_1))'}{N(v_i(t_1))} \\
\iff \quad & f = w_0' + \sum_{i=1}^{\tilde{n}} \frac{c_i}{m} \frac{w_i'}{w_i},
\end{aligned}$$

mit $w_0 = \mathrm{Tr}(v_0(t_1))/d$ und $w_i = N(v_i(t_1))$. Die letzte Gleichung ist bereits von der gewünschten Form (2.2).

t transzendent und logarithmisch:
Sei $t' = u'/u$ für ein geeignetes $u \in k$. Aus den logarithmischen Regeln folgt für

$1 \le i \le m$: $v_i(t)$ liegt in k oder ist normiert und irreduzibel und liegt in $k[t]$ mit $\deg_t v_i(t) > 0, c_i \neq 0$ und alle $v_i(t)$ sind paarweise verschieden.

Wir schreiben $v_0(t) \in k(t)$ in der Form $v_0(t) = a(t)/b(t)$ mit $a(t), b(t) \in k[t]$ und $\gcd(a(t), b(t)) = 1$. Weiterhin faktorisieren wir $b(t) = \prod_{i=1}^{\mu} b_i(t)^{e_i}$ wobei $e_i \in \mathbb{Z}_{>0}$ und $b_i(t)$ normiert, irreduzibel und paarweise verschieden sind. Dadurch bekommt die Partialbruchzerlegung von $v_0(t)$ die Form:

$$v_0(t) = a_0(t) + \sum_{i=1}^{\mu} \sum_{j=1}^{e_i} \frac{a_{ij}(t)}{b_i(t)^j},$$

wobei $a_0(t), a_{ij}(t), b_i(t) \in k[t]$ und $\deg_t a_{ij}(t) < \deg_t b_i(t)^j$. Diese Form liefert durch Einsetzen in (2.4):

$$f = a_0'(t) + \sum_{i=1}^{\mu} \sum_{j=1}^{e_i} \left(\frac{a_{ij}'(t)}{b_i(t)^j} - \frac{j\, a_{ij}(t)\, b_i'(t)}{b_i(t)^{j+1}} \right) + \sum_{i=1}^{\tilde{n}} c_i \frac{v_i'(t)}{v_i(t)}. \tag{2.5}$$

Sei nun $p(t) \in k[t]$ ein normiertes, irreduzibles Polynom mit $\deg_t p(t) > 0$. Dann folgt mit Satz 2 über die Ableitung logarithmischer Polynome: $p'(t) \in k[t]$ und $\deg_t p'(t) < \deg_t p(t) \Longrightarrow p'(t)/p(t) \notin k[t]$.

Falls $b_i(t) = p(t)$ gilt, dann gibt es in der Doppelsumme genau einen Term mit dem Nenner $b_i(t)^{e_i+1}$. Dieser Term kann sich nicht mit einem der andern Terme in der Summe aufheben, weil die Faktoren $b_i(t)$ irreduzibel und paarweise verschieden sind. Ebenso kann der Term sich auch nicht mit einem der Summanden in der rechten Summe aufheben, da der Exponent $e_i + 1$ echt größer als 1 ist. Also taucht $b_i(t)$ auch im Nenner von f auf. Da jedoch f unabhängig von t ist, ist dies ein Widerspruch und unsere Annahme, dass das Polynom $b_i(t)$ einen positiven Grad hat, ist widerlegt.

Die gleiche Argumentation können wir auf die Nenner $v_i(t)$ der rechten Summe anwenden: Falls $v_i(t) = p(t)$, dann gibt es genau einen Term mit dem Nenner $p(t)$, der auch im Nenner von f auftaucht.

Also sind alle Polynome $b_i(t)$ und $v_i(t)$ vom Grad 0 und (2.5) hat die Form:

$$f = a_0'(t) + \sum_{i=1}^{\tilde{n}} c_i \frac{v_i'}{v_i}, \tag{2.6}$$

mit $a_0(t) \in k[t], v_i \in k, c_i \in k_0$ $(1 \le i \le \tilde{n})$. Da f unabhängig von t ist, ist $a_0'(t)$ das auch. Das geht nach Satz 2 aber nur, wenn $a_0(t) = c\,t + d \in k[t]$, wobei $c \in k_0, d \in k$. Es gilt:

$$\begin{aligned} f &= d' + c\frac{u'}{u} + \sum_{i=1}^{\tilde{n}} c_i \frac{v_i'}{v_i} \\ &= d' + \sum_{i=1}^{n} c_i \frac{v_i'}{v_i}, \end{aligned}$$

mit $d, u, v_i \in k, c, c_i \in k_0$. Also hat f die geforderte Darstellung.

t transzendent und exponentiell:
Sei $t'/t = u'$ für ein geeignetes $u \in k$. Am Anfang der Argumentation oben haben wir die Eigenschaft, dass t logarithmisch ist, nicht verwendet. Wie eben erhalten wir also in diesem Fall die Form (2.5).

Sei $p(t) \in k[t]$ wieder ein normiertes, irreduzibles Polynom mit $\deg_t p(t) > 0$. Falls $p(t)$ kein Monom ist, folgt mit dem Satz 3 über die Ableitung exponentieller Polynome: $p'(t)/p(t) \notin k[t]$. Mit der gleichen Argumentation wie im logarithmischen Fall können wir darauf schließen, dass ein solcher Faktor nicht auftreten kann. Das einzige normierte Monom, das als Faktor im Nenner einer der $v_i(t)$ auftreten kann, ist $p(t) = t$.

Falls $v_i(t) = t$ gilt, dann ist $v_i'(t) = t\,u'$ und es gilt:

$$c_i \frac{v_i'(t)}{v_i(t)} = c_i\,u' \in k.$$

Dieser Term kann zu $v_0(t)$ addiert werden und somit können wir für $1 \le i \le n$ o. B. d. A. $v_i(t) \in k$ annehmen.

Für $v_0(t)$ können wir $v_0(t) = \sum_{j=-k}^{m} \lambda_j t^j$ ansetzen. Die Darstellung (2.4) aus der Induktion hat dadurch die Form:

$$\begin{aligned} f &= v_0'(t) + \sum_{i=1}^{n} c_i \frac{v_i'(t)}{v_i(t)} \\ &= \left(\sum_{j=-k}^{m} \lambda_j t^j \right)' + \sum_{i=1}^{n} c_i \frac{v_i'}{v_i} \\ &= \sum_{j=-k}^{m} \left(\lambda_j' + j\lambda_j u' \right) t^j + \sum_{i=1}^{n} c_i \frac{v_i'}{v_i}\,. \end{aligned}$$

Die Funktion f ist wiederum unabhängig von t, also gilt $k = l = 0$. Der Ansatz für $v_0(t)$ ergibt $v_0(t) = \lambda_0 \in k$ und wir haben auch für den exponentiellen Fall die Darstellung (2.2) gefunden. □

3 Risch-Algorithmus für logarithmische Erweiterungen

In diesem Kapitel werden wir den Risch-Algorithmus für Funktionen in logarithmischen Erweiterungen erklären. Dazu verwenden wir folgende Notationen:

Sei $(k,')$ ein Differentialkörper mit $x' = 1$. k lasse sich mit logarithmischen Erweiterungen aus seinem Konstantenkörper k_0 erzeugen, d. h. $k = k_0\left(x, t_1, \ldots, t_N\right)$ und jedes t_i ist logarithmisch über $k_0\left(x, t_1, \ldots, t_{i-1}\right)$.

Weiterhin sei $t = t_{N+1}$ logarithmisch über k und wir suchen eine Stammfunktion F der Funktion $f = f(t) \in k(t)$.

Die Funktion $f(t)$ habe die gekürzte Darstellung $f(t) = \tilde{p}(t)/q(t)$ mit $\tilde{p}(t), q(t) \in k[t]$ und wir erhalten durch Polynomdivision $p(t), r(t) \in k[t]$, so dass

$$\begin{aligned} \tilde{p}(t) &= q(t)p(t) + r(t) \\ \text{bzw.} \quad \int f(t)dx &= \int p(t)dx + \int \frac{r(t)}{q(t)}dx \end{aligned}$$

gilt. Die Integration der beiden Summanden im rechten Term wird getrennt betrachtet. Zunächst behandeln wir den polynomiellen Teil $p(t)$ und widmen uns danach dem rationalen Teil $r(t)/q(t)$.

3.1 Integration des polynomiellen Teils

Wir betrachten $p(t) = \lambda_n t^n + \cdots + \lambda_1 t + \lambda_0$. Angenommen es gibt eine elementare Stammfunktion, so gilt nach dem Satz von Liouville:

$$p(t) = v_0'(t) + \sum_{i=1}^{m} c_i \frac{v_i'(t)}{v_i(t)},$$

mit $v_i(t) \in k(t)$ und $c_i \in k_0$. Wenn wir wie im logarithmischen Teil des Beweises zum Satz von Liouville argumentieren, gibt es keine echten Faktoren im Nenner. Dadurch können wir annehmen, dass $v_1(t), \ldots, v_m(t)$ Elemente aus k sind und dass $v_0(t)$ ein Polynom in t sein muss. Wir erhalten wie in (2.6):

$$p(t) = P'(t) + \sum_{i=1}^{m} c_i \frac{v_i'}{v_i}, \tag{3.1}$$

mit $v_i \in k, c_i \in k_0$ und $P(t) \in k[t]$. Diesmal ist $p(t)$ zwar nicht unabhängig von t, aber nach Satz 2 gibt es eine Gradschranke der Form $\deg_t P(t) \leq \deg_t p(t) + 1$. Wir können also mit

$$P(t) = \mu_{n+1} t^{n+1} + \cdots + \mu_1 t + \mu_0 \tag{3.2}$$

ansetzen und erhalten die Gleichung:

$$\lambda_n t^n + \cdots + \lambda_1 t + \lambda_0 = \left(\mu_{n+1} t^{n+1} + \cdots + \mu_1 t + \mu_0\right)' + \sum_{i=1}^{m} c_i \frac{v_i'}{v_i}, \tag{3.3}$$

wobei $\lambda_i, \mu_i \in k$ für $i = 1, \ldots, n$, $\mu_{n+1} \in k_0$ und $v_i \in k$. Wir leiten ab und fassen die konstanten Terme auf der rechten Seite in $\tilde{\mu}_0'$ zusammen, wobei

$$\tilde{\mu}_0 = \mu_0 + \sum_{i=1}^{m} c_i \log v_i$$

gilt. Dadurch erhalten wir das Gleichungssystem:

$$\begin{aligned}
0 &= \mu_{n+1}' && (3.4)\\
\lambda_n &= (n+1)\mu_{n+1}t' + \mu_n' && (3.5)\\
\lambda_{n-1} &= n\mu_n t' + \mu_{n-1}' && (3.6)\\
&\vdots\\
\lambda_1 &= 2\mu_2 t' + \mu_1' && (3.7)\\
\lambda_0 &= \mu_1 t' + \tilde{\mu}_0'. && (3.8)
\end{aligned}$$

Dabei sind $\lambda_0, \ldots, \lambda_n$ gegeben und wir suchen $\mu_1, \ldots, \mu_n \in k, \mu_{n+1} \in k_0$ und $\tilde{\mu}_0 \in K := k(\log v_1, \ldots, \log v_m)$. Die Bedingung für $\tilde{\mu}_0'$ bedeutet lediglich, dass neue logarithmische Erweiterungen in der Stammfunktion $\tilde{\mu}_0$ erlaubt sind. Denn $\tilde{\mu}_0$ enthält auch die Logarithmen, die durch die Integration der Summe in (3.3) entstehen.

Aus der Gleichung (3.4) erhalten wir sofort, dass μ_{n+1} eine Konstante $C_{n+1} \in k_0$ ist. Nun setzen wir μ_{n+1} in Gleichung (3.5) ein und integrieren:

$$\begin{aligned}
\int \lambda_n dx &= \int (n+1)C_{n+1}t'dx + \int \mu_n' dx\\
&= (n+1)C_{n+1}t + \mu_n. && (3.9)
\end{aligned}$$

Der Integrationsalgorithmus wird nun rekursiv auf $\lambda_n \in k$ angewendet, d. h. λ_n wird als Quotient von Polynomen aus $k_0\left(x, t_1, \ldots, t_{N-1}\right)[t_N]$ dargestellt und es wird wieder zwischen Integration des polynomiellen und rationalen Teils unterschieden.

Dabei können wir wegen (3.9) folgende Bedingungen an $\int \lambda_n dx$ stellen:

(P1) Die Stammfunktion von λ_n muss elementar über k sein.

(P2) Die Stammfunktion muss in $k(t)$ liegen, d. h. es ist nur die logarithmische Erweiterung $t = \log u$ erlaubt.

Wenn eine der Bedingungen nicht erfüllt wird, können wir sofort folgern, dass $C_{n+1} \in k_0$ und $\mu_n \in k$ nicht gefunden werden können und dass die Stammfunktion von $p(t)$ nicht elementar über $k(t)$ ist. Ansonsten haben wir wegen (3.9):

$$\int \lambda_n dx = b_n t + a_n, \qquad (3.10)$$

mit $b_n \in k_0$ und $a_n \in k$. Es folgt mit (3.9):

$$C_{n+1} = \frac{b_n}{n+1}, \quad \text{und} \quad \mu_n = a_n + C_n,$$

wobei $C_n \in k_0$ eine Integrationskonstante ist.

Bei den nun folgenden Substitutionen gehen wir ähnlich vor. Wir setzen μ_n in Gleichung (3.6) ein und nutzen bei der folgenden Umformung $t' = u'/u$:

$$\begin{aligned} & \lambda_{n-1} = n(a_n + C_n)t' + \mu'_{n-1} \\ \Leftrightarrow \quad & \lambda_{n-1} - na_n\frac{u'}{u} = nC_nt' + \mu'_{n-1}. \end{aligned}$$

Das Element $a_n \in k$ haben wir aus der letzten Gleichung (3.10) bestimmt. Die linke Seite ist also bekannt und wir können wieder durch Rekursion integrieren. Wenn die Stammfunktion wieder die Bedingungen (P1) und (P2) erfüllt, erhalten wir:

$$\int \lambda_{n-1} - na_n\frac{u'}{u}dx = b_{n-1}t + a_{n-1},$$

mit $b_{n-1} \in k_0, a_{n-1} \in k$.

$$\Longrightarrow \quad C_n = \frac{b_{n-1}}{n}, \quad \text{und} \quad \mu_{n-1} = a_{n-1} + C_{n-1}$$

mit Integrationskonstante $C_{n-1} \in k_0$.

Dieses Verfahren wenden wir nacheinander bis zur vorletzten Gleichung, welche uns mittels (3.7) die Elemente C_2 und μ_1 liefert, an. Für die letzte Gleichung erhalten wir durch Einsetzen von μ_1 in (3.8):

$$\lambda_0 = \left(a_1 + C_1\right)t' + \tilde{\mu}'_0 = \left(a_1 + C_1\right)\frac{u'}{u} + \tilde{\mu}'_0$$

$$\Longrightarrow \quad \int \lambda_0 - a_1\frac{u'}{u}dx = C_1t + \tilde{\mu}_0 =: a_0.$$

Die einzige Bedingung an die Stammfunktion a_0 ist nun nur, dass sie elementar über k sein muss, denn in $\tilde{\mu}_0$ dürfen andere logarithmische Erweiterungen auftauchen. Die Unbekannte b_0 ist dann der Koeffizient von $t = \log(u)$ in a_0. Für μ_0 erhalten wir $\mu_0 = a_0 - b_0t$.

Es sind nun alle $b_0, \ldots, b_n \in k_0$ bekannt und wir können mit $C_i = b_{i-1}/i$ alle Integrationskonstanten $C_1, \ldots, C_{n+1}$ und dadurch auch alle Koeffizienten $\mu_i = a_i + C_i$ für $1 \leq i \leq n + 1$ bestimmen. Wir setzen nun noch die Koeffizienten in unseren Ansatz 3.2 für $P(t)$ ein und erhalten unsere Stammfunktion.

Wenn wir $a_{n+1} := 0$ setzen, können wir auch im ersten Schritt

$$\lambda_n = \left(\lambda_i - (i + 1)a_{i+1}u'/u\right)$$

integrieren und müssen nur noch für den letzten Schritt, in dem (P2) nicht geprüft wird, eine Fallunterscheidung machen.

Wir können somit unseren Algorithmus in folgenden Schritten durchführen:

Algorithmus: IntRischLogPoly

Eingabe: $p(t) = \lambda_n t^n + \cdots + \lambda_1 t + \lambda_0 \in k[t]$
Ausgabe: $P(t) \in K[t]$, so dass $P'(t) = p(t)$ und K elementar über k

```
1  a_{n+1} := 0
2  Für i = n, ..., 0
3      I = IntRisch[λ_i − (i + 1)a_{i+1}u′/u]
4      Wenn i > 0
5          Abbruch, wenn (P1) und (P2) nicht beide erfüllt
6          b_i := Coefficient[I, t]
7          a_i := Coefficient[I, 1]
8      sonst
9          Abbruch, wenn (P1) nicht erfüllt
10         b_0 := Coefficient[I, t]
11         μ_0 := I − b_0 t
12     Für i = 1, ..., n + 1
13         C_i := b_{i−1}/i
14         μ_i := a_i + C_i
```

Rückgabe: $\mu_{n+1}t^{n+1} + \cdots + \mu_1 t + \mu_0$

Mit **IntRisch** wird das gesamte Integrations-Paket, das hier beschrieben wird, rekursiv auf eine Funktion in k angewendet. Die *Mathematica*-Funktion **Coefficient** berechnet für das Polynom $I \in k[t]$ den Koeffizienten λ_1 von t^1 bzw. den konstanten Anteil λ_0.

Beispiel 3.1
Wir wenden nun diesen Algorithmus auf die Funktion $f(x) = \log(x)$ an.

Eingabe: Es gilt: $t = \log(x), t' = u'/u$ mit $u = x$ und
$p(t) = t \in \mathbb{Q}(x,t) \Longrightarrow n = 1, \lambda_1 = 1, \lambda_0 = 0.$

$i = 1$:

Integration des ersten Koeffizienten liefert $\int \lambda_1 dx = \int 1dx = x$

und wir setzen $b_1 = 0$ und $a_1 = x$.

$i = 0$:

Wir integrieren

$$\int \left(\lambda_0 - a_1 \frac{u'}{u}\right) dx = \int \left(0 - x\frac{1}{x}\right) dx = \int -1dx = -x$$

und erhalten $b_0 = 0$ und $\mu_0 = -x$.

Rücksubstitution:

$i = 1$:

$$C_1 = \frac{b_0}{1} = 0, \quad \mu_1 = a_1 + C_1 = x.$$

$i = 2$:

$$C_2 = \frac{b_1}{2} = 0, \quad \mu_2 = a_2 + C_2 = 0.$$

Ausgabe: $\mu_2 t^2 + \mu_1 t + \mu_0 = x\log(x) - x$.

Damit ist die Stammfunktion von $f(x)$:

$$\int f(x)dx = x\log(x) - x.$$

Wir überprüfen unser Ergebnis durch Ableiten:

```
In[15]:= D[x Log[x] - x, x]

Out[15]= Log(x)
```

Die in *Mathematica* programmierte Funktion liefert das gleiche Ergebnis:

```
In[16]:= IntRischLogPoly[Log[x], {x, Log[x]}]

Out[16]= x Log(x) - x
```

Als nächstes Beispiel betrachten wir eine Funktion, bei der zusätzliche Erweiterungen in der Stammfunktion auftreten.

Beispiel 3.2
Wir betrachten die Funktion

$$f(x) = \log(x) + \frac{1}{x+1} \in \mathbb{Q}(x, \log(x)).$$

Die Funktion $f(x)$ ist wieder ein Polynom in $\log(x)$ mit den Koeffizienten $\lambda_1 = 1$ und $\lambda_0 = 1/(x+1)$. Der erste Koeffizient ist wie im letzten Beispiel $\lambda_1 = 1$, d. h. der Algorithmus läuft zu Anfang genau wie oben. In Schritt 2 muss dann bei $i = 0$ nicht -1, sondern $\lambda_0 - 1 = 1/(x+1) - 1$ integriert werden. Die rationale Integration liefert:

$$\int \frac{1}{x+1} - 1dx = \log(x+1) - x$$

$$\Longrightarrow \quad b_0 = 0 \quad \text{und} \quad \mu_0 = \log(x+1) - x.$$

Da $b_0 = 0$ gilt, läuft auch die Rücksubstitution wie oben und liefert $\mu_1 = x$ und $\mu_2 = 0$. Wir haben damit das Ergebnis:

$$\int f(x)dx = x\log(x) - x + \log(x+1).$$

Der Integrationsalgorithmus liefert in diesem Beispiel eine neue logarithmische Erweiterung $\log(x+1)$, ohne die Bedingungen an die rekursive Integration zu verletzen. Dass der rekursive Aufruf aber auch zum Abbruch des Algorithmus führen kann, zeigt das folgende Beispiel.

Beispiel 3.3
Wir betrachten die Funktion $f(x) = \log(x)\log(x+1) \in \mathbb{Q}(x, log(x), log(x+1))$.

Eingabe: Es gilt: $t_1 = \log(x), t_1' = u_1'/u_1$ mit $u_1 = x$,
$t_2 = \log(x+1), t_2' = u_2'/u_2$ mit $u_2 = x+1$ und
$p(t) = t_1 t_2 \in \mathbb{Q}(x, t_1, t_2) \Rightarrow n = 1, \lambda_1 = t_1, \lambda_0 = 0.$

$i = 1$:
Integration des ersten Koeffizienten:

$$\int \lambda_1 dx = \int \log(x) dx = x \log(x) - x.$$

Hier wurde der Algorithmus rekursiv auf $\log(x) \in \mathbb{Q}(x, t_1)$ angewendet (siehe Beispiel 3.1). Wir erhalten $b_1 = 0$ und $a_1 = x \log(x) - x$.

$i = 0$:
Die gesuchte Stammfunktion in diesem Schritt ist:

$$\int \left(\lambda_0 - a_1 \frac{u_2'}{u_2} \right) dx = \int \left(0 - (x \log(x) - x) \frac{1}{x+1} \right) dx.$$

Nun muss also

$$f_2(x) = \frac{x}{x+1} - \frac{x}{x+1} \log(x) \in \mathbb{Q}(x, t_1)$$

integriert werden. Der Integrationsalgorithmus wird nun wieder rekursiv aufgerufen und der erste Koeffizient $-x/(x+1)$ wird mittels rationaler Integration integriert. Mit $-x/(x+1) = -1 + 1/(x+1)$ ist sehr leicht einzusehen, dass

$$\int \frac{-x}{x+1} dx = \log(x+1) - x$$

gilt. Dies verletzt jedoch die Bedingung (P2), denn die Stammfunktion muss im Körper $\mathbb{Q}(x, \log(x))$ liegen. Also kann es keine elementare Stammfunktion $f_2(x)$ geben und dadurch auch nicht für $f(x)$.

Der interne Integrationsbefehl **`Integrate`** von *Mathematica* zeigt uns, dass in der Stammfunktion eine nicht elementare Funktion $\mathrm{Li}_2(-x)$ auftaucht:

```
In[17]:= Integrate[Log[x + 1] Log[x], x]

Out[17]= -(Log(x) - 1) x + (Log(x) - 1) Log(x + 1) x + x + Log(x) Log(x + 1)
         - Log(x + 1) + Li2(-x)
```

Die Funktion $\mathrm{Li}_2(-x)$ ist in *Mathematica* als Stammfunktion von $-\log(x+1)/x$ definiert:

```
In[18]:= D[%[[-1]], x]

Out[18]= -Log(x + 1)
         ---------
             x
```

Die Stammfunktion von $f(x)$ ist also nicht elementar.

3.2 Integration des rationalen Teils

Wir behandeln nun die Integration einer in vollständig gekürzter Form vorliegenden Funktion $f(t) = r(t)/q(t)$, mit $r(t), q(t) \in k[t]$ und $\deg_t r(t) < \deg_t q(t)$.

Der Algorithmus läuft in etwa so wie bei der Integration rationaler Funktionen ab (siehe Abschnitt 1.3). Wir führen zuerst eine vollständige Partialbruchzerlegung durch und

wenden auf die einzelnen Brüche die Hermite Reduktion an. Danach bleiben wieder nur Terme übrig, die durch Bildung von Resultanten aufgelöst werden können.

Für die Partialbruchzerlegung benötigen wir zunächst die quadratfreie Faktorisierung des Nenners:

$$q(t) = \prod_{i=1}^{n} q_i(t)^i,$$

wobei $q_i(t)$ quadratfrei und $\gcd\left(q_i(t), q_j(t)\right) = 1$ für $i \neq j$. Die quadratfreie Faktorisierung ist auf einem Polynomring definiert. Der Polynomring, der hier in Frage kommt, ist $k[t]$ und damit bedeutet die Bedingung „$q_i(t)$ quadratfrei“ genauer:

$$\gcd\left(q_i(t), \frac{d}{dt} q_i(t)\right) = 1,$$

wobei mit d/dt die formale Ableitung nach t gemeint ist. Weil unsere Integrationsvariable x ist, leiten wir aber nach x ab. Damit wieder die Hermite Reduktion auf die einzelnen Partialbrüche anwendet werden kann, muss der Erweiterte Euklidische Algorithmus in

$$a(t)q_i(t) + b(t)q_i'(t) = r(t)$$

für jedes $r(t) \in k[t]$ eine Lösung liefern. Das geht aber nur, wenn $\gcd\left(q_i(t), q_i'(t)\right) = 1$ gilt. Wir benötigen also die stärkere Bedingung

$$\gcd\left(q_i(t), \frac{d}{dx} q_i(t)\right) = \gcd\left(q_i(t), q_i'(t)\right) = 1.$$

Dies liefert uns der folgende Satz.

Satz 5 *Gilt für ein normiertes Polynom $a(t) \in k[t]$, mit $\deg_t a(t) = n > 0$*

$$\gcd\left(a(t), \frac{d}{dt} a(t)\right) = 1, \tag{3.11}$$

dann gilt auch

$$\gcd\left(a(t), a'(t)\right) = 1. \tag{3.12}$$

Es ist zu beachten, dass der Euklidische Algorithmus in beiden Fällen auf $k[t]$ operiert. Das bedeutet insbesondere, dass gcd = 1 *und* gcd $\in k$ *äquivalent sind.*

Beweis: Es sei $a(t) \in k[t]$ und es gelte (3.11). Die faktorisierte Form von $a(t)$ sei:

$$a(t) = \prod_{i=1}^{n} \left(t - a_i\right),$$

mit $a_i \in k$. Dann hat die Ableitung die Form

$$a'(t) = \sum_{i=1}^{n} \left(\frac{u'}{u} - a_i'\right) \prod_{j \neq i} \left(t - a_j\right)$$

und nach Satz 2 gilt $a'(t) \in k[t]$. Falls einer der Faktoren $\left(u'/u - a_i'\right)$ Null ist, so gilt $\left(u'/u - a_i'\right) = \left(t - a_i\right)' = 0$. Dann wäre jedoch $t - a_i$ konstant in $k[t]$, also $t - a_i \in k_0$. Da dies aber der Transzendenz von t widerspricht, gilt $u'/u - a_i' \neq 0$ für $1 \leq i \leq n$.

Jeder Faktor $\left(t - a_i\right)$ von $a(t)$ taucht in genau einem Summanden von $a'(t)$ nicht auf. Weil dieser Summand wegen $u'/u - a_i' \neq 0$ nicht verschwinden kann, haben $a(t)$ und $a'(t)$ keine gemeinsamen Faktoren. Es gilt:

$$\gcd\left(a(t), \frac{d}{dx}a(t)\right) = 1$$

und die Behauptung ist bewiesen. □

Damit liefert die quadratfreie Faktorisierung über $k[t]$ auch eine Faktorisierung, die bezüglich der Ableitung nach x quadratfrei ist.

Mit der quadratfreien Faktorisierung und dem Erweitertem Euklidischen Algorithmus können wir eine Partialbruchzerlegung der Form

$$\begin{aligned} \frac{r(t)}{q(t)} &= \sum_{i=1}^{n}\sum_{j=1}^{k}\frac{r_{ij}(t)}{q_i(t)^j}\,, \\ \text{mit} \quad & \gcd\left(q_i(t), \frac{d}{dx}q_i(t)\right) = 1 \\ \text{und} \quad & 0 \leq \deg_t r_{ij}(t) < \deg_t q_i(t) \end{aligned} \tag{3.13}$$

bestimmen und die weitere Integration auf die einzelnen Summanden anwenden. Der polynomielle Teil der Partialbruchzerlegung taucht hier nicht auf, da dieser schon zu Beginn durch Polynomdivison abgespaltet wurde. Die Bedingung an die Grade von $r_{ij}(t)$ und $q_i(t)$ bezeichnen wir als Partialbruchbedingungen.

3.2.1 Hermite Reduktion

Wir betrachten einen der Partialbrüche $r_{ij}(t)/q_i(t)^j$. Da $\gcd\left(q_i(t), q_i'(t)\right) = 1$ ist, können wir wie bei der rationalen Integration vorgehen und mit dem Erweiterten Euklidischen Algorithmus Polynome $a(t), b(t) \in k[t]$ berechnen, so dass gilt:

$$a(t)q_i(t) + b(t)q_i'(t) = r_{ij}(t), \tag{3.14}$$

wobei $\deg_t a(t) < \deg_t q_i'(t)$. Wenn wir die Gleichung durch $q_i(t)^j$ teilen, erhalten wir

$$\int \frac{r_{ij}(t)}{q_i(t)^j}dx = \int \frac{a(t)}{q_i(t)^{j-1}}dx + \int \frac{b(t)q_i'(t)}{q_i(t)^j}dx$$

und partielle Integration des letzten Integrals liefert uns wie bei der rationalen Integration:

$$\int \frac{r_{ij}(t)}{q_i(t)^j}dx = \frac{-b(t)/(j-1)}{q_i(t)^{j-1}} + \int \frac{a(t) + b'(t)/(j-1)}{q_i(t)^{j-1}}dx. \tag{3.15}$$

Auch hier können wir aus (3.14) und $\deg_t r_{ij}(t) < \deg_t q_i(t)$ folgern, dass $\deg_t b(t) < \deg_t q_i(t)$ gilt. Das letzte Integral in (3.15) erfüllt also noch stets die Partialbruchbedingung

$$\deg_t\left(a(t) + b'(t)/(j-1)\right) \leq \max\left(\deg_t a(t), \deg_t b'(t)\right) < \deg_t q_i(t)$$

und wir können den Reduktionsprozess so lange wiederholen, bis nur noch Integranden mit quadratfreiem Nenner $q_i(t)$ übrig bleiben.

Algorithmus: IntRischLogHermiteReduce

Eingabe: $r(t), q(t)^j \in k[t]$, mit $\gcd(r(t), q(t)) = \gcd(q(t), q'(t)) = 1$ und $0 \le \deg_t r(t) < \deg_t q(t)$

Ausgabe: $(F, g) \in k(t)^2$, so dass $r(t)/q(t)^j = F' + g$ und g hat quadratfreien Nenner

1 Für $i = j, ..., 2$

2 $(a(t), b(t)) :=$ **EEA**$[q(t), q'(t), r(t)]$

3 $F := F + \dfrac{-b(t)/(i-1)}{q(t)^{i-1}}$

4 $r(t) := a(t) + b'(t)/(i-1)$

Rückgabe: $\left(F, r(t)/q(t)\right)$

Wir betrachten die Hermite Reduktion an einem Beispiel.

Beispiel 3.4
Sei $f(x) \in \mathbb{Q}(x, t), t = \log(x)$, mit

$$f(x) = \frac{1/x}{(t-1)^3}.$$

Der quadratfreie Faktor $q(t) = t - 1$ hat mit seiner Ableitung $q'(t) = 1/x$ keinen gemeinsamen Teiler, da $1/x$ eine Einheit in $\mathbb{Q}(x)$ ist. Die Partialbruchbedingungen sind ebenfalls erfüllt und wir können eine Hermite Reduktion durchführen.

Wir wenden den erweiterten euklidischen Algorithmus an, um Polynome $a(t), b(t) \in \mathbb{Q}(x)[t]$ zu finden, so dass gilt:

$$a(t)\,(t-1) + b(t)\,\frac{1}{x} = \frac{1}{x}.$$

Für dieses Beispiel ist dies sehr leicht: $a(t) = 0$ und $b(t) = 1$. Weil auch die Ableitung $b'(t) = 0$ ist, fällt der rechte Term in der Reduktion (3.15) weg und wir haben sofort das Ergebnis gefunden:

$$\int \frac{1/x}{(t-1)^3} = \frac{-b(t)/2}{(t-1)^2} + \int \frac{a(t) + b'(t)/2}{(t-1)^2} = \frac{-1}{2(t-1)^2}\,.$$

Wir bestätigen die Rechnung wieder durch Ableiten:

```
In[19]:= D[-1/(2 (Log[x] - 1)^2), x]

Out[19]=          1
          x (Log(x) - 1)^3
```

Die Hermite Reduktion ist in der Funktion **IntRischLogHermiteReduce** implementiert und liefert den intergierten Teil und den Restintegrand in einer Liste:

```
In[20]:= IntRischLogHermiteReduce[1/x, Log[x] - 1, 3, {x, Log[x]}]

Out[20]=  {-          1          , 0}
          2 (Log(x) - 1)^2
```

Wir konnten hier also schon die gesamte Stammfunktion

$$\int f(x)dx = \frac{-1}{2(\log x - 1)^2}$$

nur durch Anwendung des Erweiterten Euklidischen Algorithmus in $\mathbb{Q}(x)[t]$ finden!

Im folgenden Beispiel wird eine Funktion integriert, bei der mehrere Reduktionen durchgeführt werden können.

Beispiel 3.5
Wir betrachten die Funktion

$$f(x) = \frac{1/x + 1}{(t-1)^4} \in \mathbb{Q}(x,t),$$

mit $t = \log x$. Die 1 im Zähler und die höhere Potenz im Nenner beeinträchtigen die Bedingungen zur Durchführung der Hermite Reduktion nicht. Wir können also unseren Algorithmus starten. Über einen optionalen Parameter wird eingestellt, wieviele Nachrichten die Funktion **IntRischLogHermiteReduce** produzieren soll. Die Nummerierung der Nachrichten gibt Aufschluss über die unterschiedlichen Funktionsaufrufe. Im nächsten Kapitel wird darauf noch genauer eingegangen.

```
In[21]:= IntRischLogHermiteReduce[1/x + 1, t - 1, 4, {x, Log[x]},
          {Messages → Detail}]
```

1.IntRischLogHermiteReduce: IntRischLogHermiteReduce mit Parametern $\left\{1 + \frac{1}{x}, \text{Log(x)} - 1, 4, \{x, \text{Log(x)}\}\right\}$ aufgerufen.

1.IntRischLogHermiteReduce: Hermite Gleichung : $\frac{u}{x} + s\,(\text{Log(x)} - 1) == 1 + \frac{1}{x}$

1.IntRischLogHermiteReduce: Reduktion liefert die Teile $\frac{-x-1}{3\,(\text{Log(x)}-1)^3} + \text{Int}\,\frac{1}{3\,(\text{Log(x)}-1)^3}$

2.IntRischLogHermiteReduce: IntRischLogHermiteReduce mit Parametern $\left\{\frac{1}{3}, \text{Log(x)} - 1, 3, \{x, \text{Log(x)}\}\right\}$ aufgerufen.

2.IntRischLogHermiteReduce: Hermite Gleichung : $\frac{u}{x} + s\,(\text{Log(x)} - 1) == \frac{1}{3}$

2.IntRischLogHermiteReduce: Reduktion liefert die Teile $-\frac{x}{6\,(\text{Log(x)}-1)^2} + \text{Int}\,\frac{1}{6\,(\text{Log(x)}-1)^2}$

3.IntRischLogHermiteReduce: IntRischLogHermiteReduce mit Parametern $\left\{\frac{1}{6}, \text{Log(x)} - 1, 2, \{x, \text{Log(x)}\}\right\}$ aufgerufen.

3.IntRischLogHermiteReduce: Hermite Gleichung : $\frac{u}{x} + s\,(\text{Log(x)} - 1) == \frac{1}{6}$

3.IntRischLogHermiteReduce: Reduktion liefert die Teile $-\frac{x}{6\,(\text{Log(x)}-1)} + \text{Int}\,\frac{1}{6\,(\text{Log(x)}-1)}$

```
Out[21]= {(-x - 1)/(3 (Log(x) - 1)^3) - x/(6 (Log(x) - 1)) - x/(6 (Log(x) - 1)^2), 1/(6 (Log(x) - 1))}
```

Hier werden insgesamt drei Reduktionsschritte benötigt. In jedem Schritt nimmt der Exponent des Nenners um eins ab. Der Restintegrand im letzten Schritt hat quadratfreien Nenner.

Wir fassen den integrierten Teil zusammen:

```
In[22]:= Together[%[[1]]]
```

$$\text{Out[22]}= \frac{-x\,\text{Log}^2(x) + x\,\text{Log}(x) - 2x - 2}{6\,(\text{Log}(x) - 1)^3}$$

Die Stammfunktion hat also die Form:

$$\int f(x)dx = \frac{-x\,(\log x)^2 + x\,\log x - 2\,x - 2}{6\,(\log x - 1)^3} + \int \frac{1}{6\,(\log x - 1)}dx.$$

Die Integration des Restterms werden wir im nächsten Abschnitt behandeln.

Bei allen Umformungen des Integranden in unserem Algorithmus müssen wir stets darauf achten, dass der Integrand nicht so weit zerlegt wird, dass die einzelnen Terme keine elementare Stammfunktion mehr haben.

Das folgende Beispiel zeigt, dass bei den Resttermen der Hermite Reduktion Probleme auftauchen können.

Beispiel 3.6
Wir betrachten folgende Funktion:

```
In[23]:= f = (2 Log[x + 1/2] - 4x) / ((4x^2 + 4x + 1) Log[x + 1/2]^2);
```

Zunächst normieren wir den Nenner d bezüglich $\log(x+1/2)$ und fassen den Restterm in n zusammen:

```
In[24]:= t = Log[x + 1/2];
         {n, d} = Fraction[f, t]
```

$$\text{Out[24]}= \left\{-\frac{2\left(2x - \text{Log}\left(x + \frac{1}{2}\right)\right)}{(2x+1)^2}, \text{Log}^2\left(x + \frac{1}{2}\right)\right\}$$

Wir führen nun eine vollständige Partialbruchzerlegung durch. Dafür müssen wir der Funktion **PartialFraction** den Zähler, eine Liste mit Faktoren des Nenners und deren Häufigkeiten übergeben:

```
In[25]:= coefflist = PartialFraction[n, {t}, {2}, t]
```

$$\text{Out[25]}= \left\{0, \left\{\frac{2}{(2x+1)^2}, -\frac{4x}{(2x+1)^2}\right\}\right\}$$

Die resultierende Liste enthält die Nenner für $\log(x+1/2)^1$ und $\log(x+1/2)^2$ und wir erhalten:

$$f = \frac{2}{(2\,x+1)^2 \log\left(x+\frac{1}{2}\right)} - \frac{4\,x}{(2\,x+1)^2 \left(\log\left(x+\frac{1}{2}\right)\right)^2}\,.$$

Für jeden einzelnen Partialbruch wenden wir die Hermite Reduktion an. Der erste Term ist allerdings bereits vollständig reduziert und wir können direkt die Rothstein/Trager Methode anwenden. Der interne Integrationsbefehl von *Mathematica* zeigt uns aber, dass es dafür keine elementare Stammfunktion gibt:

```
In[26]:= Integrate[coefflist[[2, 1]]/t, x]
Out[26]= 1/2 Ei(-Log(x + 1/2))
```

Dies liegt aber nicht an der Funktion f, denn für diese findet *Mathematica* eine elementare Stammfunktion:

```
In[27]:= Integrate[f, x]
Out[27]= 2 x / ((2 x + 1) Log(x + 1/2))
```

Der Hintergrund ist, dass der erste Partialbruch nach der Hermite Reduktion des zweiten Partialbruchs wegfällt.

```
In[28]:= IntRischLogHermiteReduce[coefflist[[2, 2]], t, 2, {x, t}]
         //Together
Out[28]= {2 x / ((2 x + 1) Log(x + 1/2)), -2 / ((2 x + 1)^2 Log(x + 1/2))}
```

Wie wir sehen, stimmt der Restterm bis auf das Vorzeichen mit dem ersten Partialbruch überein und die Hermite Reduktion hat uns bereits die Stammfunktion geliefert:

$$\int f(x)dx = \frac{2\,x}{(2\,x+1)\,\log\left(x+\frac{1}{2}\right)}\,.$$

Es macht also keinen Sinn, den Rothstein/Trager-Algorithmus auf die einzelnen Restintegranden getrennt anzuwenden. Stattdessen müssen wir vorher die Restterme der Hermite Reduktion zu einem Term $r(t)/q(t)$ mit $r(t), q(t) \in k[t]$ zusammenfassen und mit diesem weiterarbeiten. Dabei gilt $\deg_t r(t) < \deg_t q(t)$, weil auch die Partialbrüche und Restterme der Hermite Reduktion diese Grad-Ungleichung erfüllen.

3.2.2 Rothstein/Trager Resultante

Der Restterm, der nun noch zu integrieren ist, ist von der Form $r(t)/q(t)$ mit $\deg_t r(t) < \deg_t q(t)$ und $q(t)$ ist normiert und quadratfrei. Der folgende Satz liefert uns dafür einen Algorithmus.

Satz 6 *Seien $r(t), q(t) \in k[t]$ mit $\gcd(r(t), q(t)) = 1, \deg_t r(t) < \deg_t q(t)$ und $q(t)$ sei normiert und quadratfrei. Die Variable z sei eine neue Unbekannte und wir definieren*

$$R(z) := \operatorname{res}_t\left(r(t) - z\,q'(t), q(t)\right) \in k[z].$$

Dann gilt

(i) $\int r(t)/q(t)$ *ist genau dann elementar, wenn die Nullstellen des Polynoms $R(z)$ konstant sind.*

(ii) Wenn $\int r(t)/q(t)$ *elementar ist, dann gilt*

$$\frac{r(t)}{q(t)} = \sum_{i=1}^{m} c_i \frac{v_i'(t)}{v_i(t)}\,,$$

wobei c_i ($1 \le i \le m$) die Nullstellen von $R(z)$ sind und für $v_i(t)$ gilt:

$$v_i(t) = \gcd\left(a(t) - c_i b'(t), b(t)\right) \in K[t].$$

Da $R(z)$ über $k[z]$ nicht in Linearfaktoren zerfallen muss, müssen nicht alle Nullstellen in k liegen. Mit K bezeichnen wir den kleinsten Erweiterungskörper von k, in dem $R(z)$ zerfällt und dadurch $c_1, \ldots, c_m \in K$ gilt.

Beweis: Wir beweisen (ii) und die erste Richtung von (i) gemeinsam, da dort die gleichen Voraussetzungen gelten. Dabei gehen wir schrittweise vor, indem wir folgende Behauptungen beweisen:

1. $\dfrac{r(t)}{q(t)} = \displaystyle\sum_{i=1}^{m} c_i \frac{v_i'(t)}{v_i(t)}$,
2. $q(t) = \displaystyle\prod_{i=1}^{m} v_i(t)$,
3. für alle j gilt: $v_j(t) \,|\, r(t) - c_j q'(t)$,
4. für alle j gilt: $v_j(t) = \gcd\left(r(t) - c_j q'(t), q(t)\right)$,
5. alle Nullstellen z_i von $R(z)$ tauchen in der Darstellung für genau eines der c_i auf.

Ab Schritt 2 wird nur noch im Ring $k[t]$ gearbeitet. Die Teilbarkeit in Schritt 3 bedeutet also:

$$\text{Es existiert ein } s(t) \in k[t], \text{ so dass } v_j(t) = s(t)\left(r(t) - c_j q'(t)\right) \text{ gilt.}$$

Schritt 1:
Zu zeigen: $\dfrac{r(t)}{q(t)} = \displaystyle\sum_{i=1}^{m} c_i \frac{v_i'(t)}{v_i(t)}$.

Angenommen $r(t)/q(t)$ hat eine elementare Stammfunktion, dann gilt nach dem Satz von Liouville:

$$\frac{r(t)}{q(t)} = v_0'(t) + \sum_{i=1}^{m} c_i \frac{v_i'(t)}{v_i(t)}, \tag{3.16}$$

mit $v_0(t), v_i(t) \in k(t)$ und $c_i \in k_0$ für $1 \le i \le m$. Wir können durch Anwenden der logarithmischen Regeln annehmen, dass für $i \le 1$ alle $v_i(t)$ normiert und quadratfrei sind und in $k[t]$ liegen. Durch Anpassung der Konstanten c_i können wir außerdem dafür sorgen, dass $v_1(t), \ldots, v_m(t)$ paarweise verschieden sind. Weiterhin habe $v_0(t)$ o.B.d.A. die Darstellung $v_0(t) = a(t)/b(t)$ mit $a(t), b(t) \in k[t]$ und $\gcd(a(t), b(t)) = 1$. Wenn wir wie im Beweis des Satzes von Liouville argumentieren, würde aus $\deg_t b(t) > 0$ folgen, dass $v_0'(t)$ einen irreduziblen Faktor mit Vielfachheit größer eins im Nenner hat. Dies widerspricht aber der Voraussetzung, dass $q(t)$ quadratfrei ist. Also ist auch $v_0(t) \in k[t]$ ein Polynom.

Nehmen wir nun an, dass $\deg_t v_0(t) > 0$ ist, und bringen die rechte Seite von (3.16) auf einen Nenner:

$$\frac{r(t)}{q(t)} = v_0'(t) + \sum_{i=1}^{m} c_i \frac{v_i'(t)}{v_i(t)} = \frac{v_0'(t)\prod_{i=1}^{m} v_i(t) + \sum_{i=1}^{m} c_i v_i'(t)\prod_{i\neq j} v_j(t)}{\prod_{i=1}^{m} v_i(t)}\,.$$

Da jeweils $\deg_t v_i'(t) \leq \deg_t v_i(t)$ gilt, hat im Zähler das Produkt mit $v_0'(t)$ in jedem Fall den höchsten Grad. Aus der Annahme $\deg_t v_0(t) > 0$ folgt $\deg_t v_0'(t) \geq 0$ und dadurch ist der Grad des Zählers größer oder gleich dem Grad des Nenners: $\deg_t r(t) \geq \deg_t q(t)$. Dies widerspricht jedoch den Voraussetzungen des Satzes.

Das Polynom $v_0(t)$ hat also den Grad $\deg_t v_0(t) = 0$, d. h. $v_0'(t) = 0$ und wir haben die gewünschte Darstellung:

$$\frac{r(t)}{q(t)} = \sum_{i=1}^{m} c_i \frac{v_i'(t)}{v_i(t)}\,. \tag{3.17}$$

Schritt 2:

Zu zeigen: $q(t) = \prod_{i=1}^{m} v_i(t)$.

Wir multiplizieren die Darstellung (3.17) mit dem Hauptnenner $V_i(t) := \prod_{i\neq j} v_j(t)$:

$$r(t)\prod_{i=1}^{m} v_i(t) = q(t)\sum_{i=1}^{m} c_i\, v_i'(t) V_i(t)\,. \tag{3.18}$$

Da $r(t)$ und $q(t)$ teilerfremd sind, gilt:

$$q(t) \,\Bigg|\, \prod_{i=1}^{m} v_i(t) \tag{3.19}$$

$$\text{und} \quad v_i(t) \,\Bigg|\, \sum_{j=1}^{m} q(t) c_j\, v_j'(t)\, V_j(t) \;\text{ für alle } 1 \leq i \leq m.$$

Wenn die Polynome $v_i(t)$ diese Summe teilen, teilen sie insbesondere auch jeden Summanden:

$$v_i(t) \,\big|\, q(t)\, c_i\, v_i'(t) V_i(t). \tag{3.20}$$

Weil alle $v_i(t)$ aufgrund der Annahmen in Schritt 1 quadratfrei und paarweise verschieden sind, gilt

$$\gcd\big(v_i(t), v_i'(t)\big) = \gcd\big(v_i(t), V_i(t)\big) = 1 \tag{3.21}$$

und aus der Teilbarkeit (3.20) folgt:

$$\begin{aligned} & v_i(t) \,\big|\, q(t) \text{ für alle } i \\ \Longrightarrow \quad & \prod_{i=1}^{m} v_i(t) \,\Bigg|\, q(t). \end{aligned} \tag{3.22}$$

Mit (3.19) gilt also in beide Richtungen die Teilbarkeit und wenn wir beachten, dass sowohl $q(t)$ als auch alle $v_i(t)$ quadratfrei sind, folgt die Gleichheit:

$$q(t) = \prod_{i=1}^{m} v_i(t). \tag{3.23}$$

Schritt 3:
Zu zeigen: für alle j gilt: $v_j(t) \mid r(t) - c_j q'(t)$.

Aus der Gleichheit (3.23) folgt mit (3.18)

$$r(t) = \sum_{i=1}^{m} c_i \, v_i'(t) V_i(t) \tag{3.24}$$

und für die Ableitung $q'(t)$ gilt:

$$q'(t) = \sum_{i=1}^{m} v_i'(t) V_i(t).$$

Wir betrachten für ein festes $0 \leq j \leq m$:

$$\begin{aligned} r(t) - c_j q'(t) &= \sum_{i=1}^{m} c_i \, v_i'(t) V_i(t) - c_j \sum_{i=1}^{m} v_i'(t) V_i(t) \\ &= \sum_{i=1}^{m} \left(c_i - c_j\right) v_i'(t) V_i(t). \end{aligned} \tag{3.25}$$

In der Summe fällt der Term für $i = j$ weg. Für $i \neq j$ teilt $v_j(t) \mid V_i(t)$ und damit auch die gesamte Summe:

$$v_j(t) \mid r(t) - c_j q'(t). \tag{3.26}$$

Schritt 4:
Zu zeigen: für alle j gilt: $v_j(t) = \gcd\left(r(t) - c_j q'(t), q(t)\right)$.

Aus den Aussagen in Schritt 2 und 3 wissen wir, dass $v_j(t)$ ein gemeinsamer Teiler der beiden Terme

$$r(t) - c_i q'(t) \quad \text{und} \quad q(t)$$

ist. Wir müssen nun nur noch zeigen, dass es auch der größte gemeinsame Teiler ist. Die einzigen möglichen Teiler sind $v_i(t)$ für $i \neq j$, denn in $q(t)$ tauchen keine anderen Faktoren auf. Es reicht also zu zeigen, dass für $i \neq j$ gilt:

$$\gcd\left(r(t) - c_i q'(t), v_j(t)\right) = 1.$$

Dazu formen wir zunächst den Teiler um:

$$\begin{aligned} \gcd\left(r(t) - c_i q'(t), v_j(t)\right) &= \gcd\left(\sum_{k=1}^{m} (c_k - c_i) \, v_k'(t) \, V_k(t), v_j(t)\right) \\ &= \gcd\left((c_j - c_i) \, v_j'(t) \, V_j(t), v_j(t)\right). \end{aligned}$$

Wir haben bereits oben gesehen, dass $v_j(t) \mid V_k(t)$ für $j \neq k$ gilt. Deswegen können wir in der letzten Umformung alle Summanden für $j \neq k$ weglassen.

Das Polynom $v_j(t)$ hat wegen den Annahmen in Schritt 1 aber keinen gemeinsamen Teiler mit $v_j'(t)$ oder $V_j(t)$ und da die Differenz $(c_i - c_j) \in k_0$ konstant ist, gilt:

$$\gcd\left(r(t) - c_i q'(t), v_j(t)\right) = 1. \tag{3.27}$$

Wir haben damit gezeigt, dass

$$v_j(t) = \gcd\big(r(t) - c_j q'(t), q(t)\big)$$

gilt. Da $v_j(t)$ ein echter gemeinsamer Faktor mit positivem Grad ist, bedeutet das insbesondere, dass die Resultante $\operatorname{res}_t\big(r(t) - c_j q'(t), q(t)\big)$ Null ist. Also ist c_j eine Nullstelle von $R(z)$.

Schritt 5:
Es bleibt zu zeigen, dass in der Darstellung von $r(t)/q(t)$ alle Nullstellen von $R(z)$ als eines der c_i auftauchen. Weil diese c_i aufgrund der Darstellung von Liouville konstant sind, wäre damit auch Aussage (i) bewiesen.

Sei also c eine Nullstelle von $R(z) \in k[z]$. Es gilt $\operatorname{res}_t(r(t) - c\,q'(t), q(t)) = 0$. Da eine Resultante genau dann Null wird, wenn es einen echten gemeinsamen Faktor gibt, hat

$$G(t) = \gcd\big(r(t) - c\,q'(t), q(t)\big) \in k[t]$$

positiven Grad: $\deg_t G(t) > 0$. Sei nun $g(t)$ ein irreduzibler Faktor von $G(t)$ mit $\deg_t g(t) > 0$, dann folgt einerseits:

$$g(t) \,\big|\, q(t) \Longrightarrow \text{ es existiert ein } j, \text{ so dass } g(t) \,\big|\, v_j(t) \text{ gilt.} \tag{3.28}$$

Andererseits teilt $g(t)$ auch $r(t) - c\,q'(t)$ und wegen (3.25) gilt:

$$r(t) - c\,q'(t) = \sum_{i=1}^{m} \big(c_i - c\big)\, v_i'(t) V_i(t).$$

Das Polynom $g(t)$ teilt jeden einzelnen Summanden auf der rechten Seite und damit insbesondere auch den Summanden für das spezielle j aus (3.28):

$$g(t) \,\Big|\, \big(c_j - c\big)\, v_j'(t) V_j(t).$$

Weil das Polynom $g(t)$ ein Teiler von $v_j(t)$ ist, ist es genau wie $v_j(t)$ teilerfremd zu $v_j'(t)$ und $V_j(t)$. Also hat $g(t)$ echt positiven Grad und teilt trotzdem $c_j - c \in k$. Das kann aber nur sein, wenn $c_j - c = 0$ ist. Also ist c eine der Konstanten c_i aus (3.17) und es tauchen alle Nullstellen von $R(z)$ in der Darstellung auf.

Es bleibt noch die Rückrichtung von (i) zu zeigen. Dazu nehmen wir an, dass alle verschiedenen Nullstellen $c_1, \ldots, c_m$ von $R(z)$ konstant sind. Wir definieren

$$v_i(t) := \gcd\big(r(t) - c_i\, q'(t), q(t)\big) \in k[t] \tag{3.29}$$

und zeigen folgende Aussagen:

1. $\gcd\big(v_i(t), v_j(t)\big) = 1$ für $i \neq j$,

2. $q(t) = \prod_{i=1}^{m} v_i(t)$,

3. $r(t) = \sum_{i=q}^{m} c_i\, v_i'(t) \prod_{i \neq j} v_j(t)$.

Schritt 1:
Zu zeigen: $\gcd\big(v_i(t), v_j(t)\big) = 1$ *für* $i \neq j$.

Es seien $i \neq j$ fest und $g(t) := \gcd\big(v_i(t), v_j(t)\big)$. Dann gilt wegen der Definition von $v_i(t)$ und $v_j(t)$ in (3.29):

$$\begin{aligned} & g(t) \,|\, r(t) - c_i\, q'(t), \\ & g(t) \,|\, r(t) - c_j\, q'(t) \\ \text{und} \quad & g(t) \,|\, q(t). \end{aligned} \tag{3.30}$$

Aus den ersten beiden Teilbarkeiten folgt:

$$g(t) \,|\, \big(c_i - c_j\big)\, q'(t). \tag{3.31}$$

Da $\big(c_i - c_j\big) \in k_0$ konstant ist, liefern die Aussagen (3.30) und (3.31), dass $q(t)$ und $q'(t)$ den gemeinsamen Teiler $g(t)$ haben. Da $q(t)$ jedoch nach Voraussetzung quadratfrei ist, gilt $g(t) = \gcd(v_i(t), v_j(t)) = 1$.

Schritt 2:
Zu zeigen: $q(t) = \prod_{i=1}^{m} v_i(t)$.

Nach der Definition in (3.29) teilt jedes $v_i(t)$ den Nenner $q(t)$ und da $v_i(t)$ paarweise teilerfremd sind, gilt auch für das Produkt:

$$\prod_{i=1}^{m} v_i(t) \,\Big|\, q(t).$$

Es gibt also ein $s(t) \in k[t]$, mit

$$q(t) = s(t) \prod_{i=1}^{m} v_i(t). \tag{3.32}$$

Wir zeigen nun, dass $s(t) = 1$ ist. Dazu nehmen wir an, dass $\deg_t s(t) > 0$ ist und führen dies zum Widerspruch. Laut Voraussetzung gilt $0 \leq \deg_t r(t) < \deg_t q(t)$. Das Polynom $q(t)$ hat also positiven Grad und $q'(t) \neq 0$. Die Sylvester-Matrix der Resultante

$$\tilde{R}(z) = \operatorname{res}_t\big(r(t) - z\, q'(t), s(t)\big) \in k[z]$$

hat dann mindestens ein Zeile, in der z auftaucht. Die Resultante $\tilde{R}(z)$ hat somit positiven Grad $\deg_z \tilde{R}(z) > 0$ und es gibt eine Nullstelle z_0.

Die Polynome $r(t) - z_0 q'(t)$ und $s(t)$ haben also den gemeinsamen Faktor:

$$g_1(t) := \gcd\big(r(t) - z_0\, q'(t), s(t)\big) \in k[t], \text{ mit } \deg_t g_1(t) > 0.$$

Wegen $s(t) | q(t)$ ist $g_1(t)$ auch ein echter Teiler von

$$g_2(t) := \gcd\big(r(t) - z_0\, q'(t), q(t)\big) \in k[t]$$

und es gilt:

$$\operatorname{res}_t\big(r(t) - z_0\, q'(t), q(t)\big) = 0.$$

Damit ist z_0 auch eine Nullstelle von $R(z)$. Es sei o.E. $z_0 = c_i$ für ein festes i. Dann gilt $g_2(t) = v_i(t)$ und da $g_1(t)$ jetzt sowohl $s(t)$ als auch $v_i(t)$ teilt, haben $s(t)$ und $v_i(t)$ einen gemeinsamen Teiler, d. h. $\gcd(s(t), v_i(t)) \neq 1$.

Aus der Definition von $s(t)$ in (3.32) folgt nun, dass $q(t)$ einen quadratischen Faktor hat. Das widerspricht jedoch der Voraussetzung und wir können folgern, dass $\deg_t s(t) = 0$ ist.

Wenn wir in (3.29) den größten gemeinsamen Teiler immer normiert wählen, muss $s(t) = 1$ sein, weil $q(t)$ nach Voraussetzung auch normiert ist. Somit gilt:

$$q(t) = \prod_{i=1}^{m} v_i(t).$$

Schritt 3:

Zu zeigen: $r(t) = \sum_{i=q}^{m} c_i\, v_i'(t) \prod_{i\neq j} v_j(t)$.

Wir definieren $\tilde{r}(t)$ wie in der Folgerung (3.24) aus dem ersten Beweisteil:

$$\tilde{r}(t) := \sum_{i=q}^{m} c_i v_i'(t) V_i(t), \tag{3.33}$$

wobei wieder $V_i(t) := \prod_{i\neq j} v_j(t)$ gilt. Wir betrachten auch hier ein festes $1 \leq j \leq m$ und wie oben gilt:

$$\begin{aligned}\tilde{r}(t) - c_j\, q'(t) &= \sum_{i=q}^{m} c_i v_i'(t) V_i(t) - c_j \left(\sum_{i=1}^{m} v_i'(t) V_i(t)\right) \\ &= \sum_{i=1}^{m} \big(c_j - c_j\big)\, v_i'(t)\, V_i(t) \\ \\ \Longrightarrow \quad & v_j(t) \,\big|\, \tilde{r}(t) - c_j\, q'(t). \end{aligned} \tag{3.34}$$

Nach Definition von $v_j(t)$ in (3.29) gilt zusätzlich:

$$v_j(t) \,\big|\, r(t) - c_j q'(t).$$

Also teilt $v_j(t)$ auch die Differenz $\tilde{r}(t) - r(t)$. Dies gilt für alle teilerfremden $v_i(t)$ und somit auch für das Produkt $q(t)$:

$$q(t) \,\big|\, \tilde{r}(t) - r(t).$$

Nun schauen wir uns den Grad dieser beiden Terme an. Nach Satz 2 nimmt bei normierten logarithmischen Polynomen der Grad beim Ableiten ab, d. h. es gilt $\deg_t v'(t) < \deg_t v_i(t)$ für alle $0 \leq i \leq m$ und aufgrund der Definition (3.33) gilt $\deg_t \tilde{r}(t) < \deg_t q(t)$. Da außerdem nach Voraussetzung $\deg_t r(t) < \deg_t q(t)$ gilt, hat die rechte

Seite $\tilde{r}(t) - r(t)$ einen echt kleineren Grad als $q(t)$. Dies kann aber nur sein, wenn $\tilde{r}(t) - r(t) = 0$ ist. Also gilt $\tilde{r}(t) = r(t)$ und die Behauptung ist bewiesen.

Wenn wir die Ergebnisse aus Schritt 2 und 3 in $r(t)/q(t)$ einsetzen, erhalten wir:

$$\frac{r(t)}{q(t)} = \frac{\sum_{i=1}^{m} c_i\, v_i'(t)\, \prod_{i\neq j} v_j(t)}{\prod_{i=1}^{m} v_i(t)} = \sum_{i=1}^{m} c_i \frac{v_i'(t)}{v_i(t)}\,.$$

Mit dieser Darstellung können wir die Stammfunktion direkt angeben und (i) ist bewiesen. □

Der Algorithmus kann somit wie folgt beschrieben werden:

Algorithmus: IntRischLogRothsteinTrager

Eingabe: $r(t)/q(t) \in k(t)$ mit $\gcd_t(r(t), q(t)) = 1, \deg_t r(t) < \deg_t q(t)$ und $q(t)$ normiert und quadratfrei
Ausgabe: $F \in K(t)$, so dass $F' = r(t)/q(t)g$

1 $R(z) := \mathrm{res}_t\,(r(t) - zq'(t), q(t))$
2 Abbruch, wenn $R(z)$ unabhängig von z
3 $R(z) :=$ **`MonicPart`**$[R(z), t]$
4 $\left(z_1, \ldots, z_m\right) :=$ **`Solve`**$[R(z) == 0, z]$
5 Abbruch, wenn ein z_i nicht konstant
6 $v_i := \gcd\left(r(t) - z_i\, q'(t), q(t)\right)$
7 $v_i :=$ **`MonicPart`**$[v_i, t]$

Rückgabe: $\sum_{i=1}^{m} z_i \log(v_i)$

Die Normierung von $R(z)$ in Zeile 3 ändert das Ergebnis nicht, da die Nullstellen bei der Multiplikation mit einem (bzgl. z) konstanten Faktor gleich bleiben. Man kann sogar beweisen, dass danach die Koeffizienten alle in k_0 liegen. Das würde bedeuteten, dass die Nullstellensuche in Zeile 4 unabhängig von x und dadurch leichter wird.

Die Normierung der Logarithmen v_i in Zeile 7 ist ebenfalls erlaubt, da wir dadurch die Stammfunktion nur um eine additive Konstante verändern. Diese Normierung hat einen speziellen Grund, den wir im nächsten Kapitel noch einmal aufgreifen werden.

Beispiel 3.7
Wir wenden die Rothstein/Trager Methode auf

$$f(x) = \frac{r(t)}{q(t)} = \frac{5/x}{\log x - 1} \in \mathbb{Q}(x, t), \quad t = \log x$$

an. Dafür berechnen wir zunächst die Resultante:

$$\begin{aligned} R(z) &= \mathrm{res}_t\left(r(t) - zq'(t), q(t)\right) \\ &= \mathrm{res}_t\left(5/x - z \cdot 1/x, \log x - 1\right). \end{aligned}$$

Da $\deg_t r(t) - zq'(t) = 0$ und $\deg_t q(t) = 1$ gilt, liefert uns die Produktdarstellung (1.3) der Resultante:

$$R(z) = 5/x - z/x.$$

Wir normieren bezüglich z und erhalten: $R(z) = z - 5$. Die einzige Nullstelle ist $z_1 = 5$ und wir berechnen:

$$\begin{aligned} v_1 &= \gcd\left(r(t) - z_1\, q'(t), q(t)\right) \\ &= \gcd\left(0, \log x - 1\right) = \log x - 1. \end{aligned}$$

Damit ist die Stammfunktion von $f(x)$:

$$\int f(x)dx = z_1 \log v_1 = 5 \log\left(\log x - 1\right).$$

3.3 Zusammenfassung der Algorithmen

Die drei vorgestellten Algorithmen **`IntRischLogPoly`**, **`IntRischLogHermiteReduce`** und **`IntRischLogRothsteinTrager`** vollenden den logarithmischen Teil des Risch-Algorithmus. Der folgende Algorithmus umfasst die Umformungen, die für den Aufruf der drei Funktionen benötigt werden.

Algorithmus: IntRischLog

Eingabe: $f = p/q \in k(t)$, mit $p, q \in k[t]$, q normiert und t logarithmisch über k
Ausgabe: $F \in K$ mit $F' = f$ und K elementar über $k(t)$

1 $(\tilde{q}, \tilde{e}) = \left(\left(q_1, \ldots, q_n\right), \left(e_1, \ldots, e_n\right)\right) :=$ **`SquareFreeList`**$[q, t]$
2 $\left(p_0, \left(p_{11}, \ldots, p_{1e_1}\right), \ldots, \left(p_{n1}, \ldots, p_{ne_n}\right)\right) :=$ **`PartialFraction`**$[p, \tilde{q}, \tilde{e}]$
3 $P :=$ **`IntRischLogPoly`**$[p_0]$
4 $(F, g) := (0, 0)$
5 Für $i = 1, \ldots, n$
6 Für $j = 1, \ldots, e_i$
7 $(\tilde{F}, \tilde{g}) :=$ **`IntRischLogHermiteReduce`**$[p_{ij}, q_i, j]$
8 $(F, g) := (F + \tilde{F}, g + \tilde{g})$
9 $G :=$ **`IntRischLogRothsteinTrager`**$[g]$

Rückgabe: $P + F + G$

Der Befehl **`IntRisch`** fasst die rationale Integration und den logarithmischen Teil des Risch-Algorithmus in einem Paket zusammen.

Im Unterschied zu *Mathematica*, wo der **`Integrate`**-Befehl in einigen Fällen kein Ergebnis liefert, ist unser Algorithmus deterministisch. Das heißt, entweder wir kommen zu einem Ergebnis und erhalten eine Stammfunktion, oder wir wissen, dass es keine elementare Stammfunktion gibt. Die folgenden Beispiele zeigen die unterschiedlichen Ergebnisse.

Beispiel 3.8
Wir betrachten folgende Funktion:

```
In[29]:= f = (Log[x])/((Log[x] - 1)(Log[x + 1]))
Out[29]=          Log(x)
          ---------------------
          (Log(x) - 1) Log(x + 1)
```

Von *Mathematicas* Integrationsbefehl erfahren wir nichts:

```
In[30]:= Integrate[f, x]
Out[30]= ∫ Log(x) / ((Log(x) - 1) Log(x + 1)) dx
```

Unser Algorithmus liefert jedoch, dass es keine elementare Stammfunktion geben kann:

```
In[31]:= IntRisch[f, {x, Log[x], Log[x + 1]}]

         Int::notelem: Es gibt keine elementare Stammfunktion.

Out[31]= $Failed
```

Wir wenden nun den Risch-Algorithmus auf einige komplexere Beispiele an und vergleichen die Laufzeit mit dem Integrationsbefehl von *Mathematica*.

Beispiel 3.9
Sei $f(x) = (\log x)^{1000}$. Wir berechnen die Stammfunktion $F(x) = \int f(x)dx$:

```
In[32]:= F = IntRisch[f, {x, Log[x]}];
```

und vergleichen $F'(x)$ mit $f(x)$:

```
In[33]:= D[F, x]

Out[33]= Log^1000(x)
```

Die berechnete Stammfunktion ist also korrekt. Da $f(x)$ polynomiell in $\log x$ ist, wird der polynomielle Teil des Algorithmus gestartet. Die Iterationen und rekursiven Aufrufe, die dort gemacht werden müssen, sorgen für eine längere Laufzeit:

```
In[34]:= Timing[IntRisch[f, {x, Log[x]}];]

Out[34]= {4.99042 Second, Null}
```

Mathematica scheint hier die einfache Struktur des Integranden zu verwenden und benötigt weniger als die Hälfte der Zeit:

```
In[35]:= Timing[Integrate[f, x];]

Out[35]= {1.98413 Second, Null}
```

Es könnte beispielsweise sein, dass *Mathematica* die Formel[3]

$$\int (\log x)^n dx = x(\log x)^n - n \int (\log x)^{n-1} dx, \quad n \in \mathbb{Z} \setminus \{-1\}$$

nutzt. Wenn diese Formel interativ angewendet wird, kann die Stammfunktion sehr leicht gefunden werden. Die Iteration im Risch-Algorithmus läuft zwar ähnlich ab, ist aber nicht speziell auf dieses Problem zugeschnitten. Auch wenn die Integranden $\lambda_i - (i+1)a_{i+1}u'/u$ konstant sind, wird der gesamte Integrationsalgorithmus aufgerufen.

Dass *Mathematica* aber nicht in allen Fällen schneller ist, zeigt folgendes Beispiel.

[3] [BSMM97] Abschnitt 21.5.4.3, Integral Nr. 468.

Beispiel 3.10
Wir betrachten nochmals die Funktion $f(x)$ aus Beispiel 3.8

```
In[36]:= f = (Log[x])/((Log[x] - 1)(Log[x + 1]))
```

$$\text{Out[36]}= \frac{\text{Log}(x)}{(\text{Log}(x)-1)\ \text{Log}(x+1)}$$

und berechnen die Ableitung $g(x) = f'(x)$:

```
In[37]:= g = D[f, x]//Together
```

$$\text{Out[37]}= \frac{-x\,\text{Log}^2(x) + x\,\text{Log}(x) - x\,\text{Log}(x+1) - \text{Log}(x+1)}{x\,(x+1)\,(\text{Log}(x)-1)^2\,\text{Log}^2(x+1)}$$

Die Stammfunktion $\int g(x)dx$ lässt sich relativ schnell finden:

```
In[38]:= Timing[IntRisch[g, {x, Log[x], Log[x + 1]}]]
```

$$\text{Out[38]}= \left\{0.046395\,\text{Second}, \frac{\text{Log}(x)}{(\text{Log}(x)-1)\ \text{Log}(x+1)}\right\}$$

Diesmal der Risch-Algorithmus doppelt so schnell wie *Mathematica*:

```
In[39]:= Timing[Integrate[g, x]]
```

$$\text{Out[39]}= \left\{0.103177\,\text{Second}, \frac{\text{Log}(x)}{(\text{Log}(x)-1)\ \text{Log}(x+1)}\right\}$$

Abschließend betrachten wir noch ein Beispiel, mit besonders langer Laufzeit.

Beispiel 3.11
Wir definieren $f(x)$ wie folgt:

```
In[40]:= f = ((x + 1) Log[x] - (3x^2 - 2) Log[x]^2)/
           (Log[x]^3 - 2x Log[x] + x^4 Log[x]) * Log[x + 1]
```

$$\text{Out[40]}= \frac{((x+1)\,\text{Log}(x) - (3x^2-2)\,\text{Log}^2(x))\,\text{Log}(x+1)}{\text{Log}(x)\,x^4 - 2\,\text{Log}(x)\,x + \text{Log}^3(x)}$$

Wir berechnen die Ableitung $g(x) = f'(x)$

```
In[41]:= g = D[f, x]//Together
```

$$\begin{aligned}\text{Out[41]}= {} & (-3\,\text{Log}(x)\,x^7 + 6\,\text{Log}(x)\,\text{Log}(x+1)\,x^7 - 3\,\text{Log}(x+1)\,x^7 + \\ & 6\,\text{Log}(x)\,\text{Log}(x+1)\,x^6 - 6\,\text{Log}(x+1)\,x^6 + x^6 + 2\,\text{Log}(x)\,x^5 - \\ & 8\,\text{Log}(x)\,\text{Log}(x+1)\,x^5 - 5\,\text{Log}(x+1)\,x^5 + x^5 + 6\,\text{Log}(x)\,x^4 - \\ & 2\,\text{Log}(x)\,\text{Log}(x+1)\,x^4 + 4\,\text{Log}(x+1)\,x^4 - 3\,\text{Log}^3(x)\,x^3 - \\ & 6\,\text{Log}^3(x)\,\text{Log}(x+1)\,x^3 + 3\,\text{Log}^2(x)\,\text{Log}(x+1)\,x^3 + \\ & 6\,\text{Log}(x)\,\text{Log}(x+1)\,x^3 + 6\,\text{Log}(x+1)\,x^3 - 2\,x^3 + \text{Log}^2(x)\,x^2 - \\ & 4\,\text{Log}(x)\,x^2 - 6\,\text{Log}^3(x)\,\text{Log}(x+1)\,x^2 + 4\,\text{Log}^2(x)\,\text{Log}(x+1)\,x^2 + \\ & 2\,\text{Log}(x)\,\text{Log}(x+1)\,x^2 - 2\,\text{Log}(x+1)\,x^2 - 2\,x^2 + 2\,\text{Log}^3(x)\,x + \\ & \text{Log}^2(x)\,x - \text{Log}^2(x)\,\text{Log}(x+1)\,x - 2\,\text{Log}(x+1)\,x - \\ & 2\,\text{Log}^2(x)\,\text{Log}(x+1) - 2\,\text{Log}(x)\,\text{Log}(x+1))\Big/ \\ & \left(x\,(x+1)\,(x^4 - 2\,x + \text{Log}^2(x))^2\right)\end{aligned}$$

und führen bei der Integration wieder einen Zeitvergleich durch:

```
In[42]:= Timing[IntRisch[g, {x, Log[x], Log[x + 1]}]]
```

$$\text{Out[42]}= \left\{0.123041\,\text{Second}, \frac{(-3\,\text{Log}(x)\,x^2 + x + 2\,\text{Log}(x) + 1)\,\text{Log}(x+1)}{x^4 - 2\,x + \text{Log}^2(x)}\right\}$$

```
In[43]:= Timing[Integrate[g, x]]

Out[43]= {0.431713 Second, -((3 Log(x) x^2 - x - 2 Log(x) - 1) Log(x + 1))/(x^4 - 2 x + Log^2(x))}
```

Um die Komplexität des Integranden und damit die absolute Laufzeit zu erhöhen, leiten wir mehrmals ab und betrachten die siebte und zehnte Ableitung $f^{(7)}(x)$ und $f^{(10)}(x)$:

```
In[44]:= g = D[f, {x, 7}];

In[45]:= Timing[IntRisch[g, {x, Log[x], Log[x + 1]}];]

Out[45]= {24.9874 Second, Null}

In[46]:= Timing[Integrate[g, x];]

Out[46]= {66.6418 Second, Null}

In[47]:= g = D[f, {x, 10}];

In[48]:= Timing[IntRisch[g, {x, Log[x], Log[x + 1]}];]

Out[48]= {82.0325 Second, Null}

In[49]:= Timing[Integrate[g, x];]

Out[49]= {157.138 Second, Null}
```

Der Risch-Algorithmus war bei diesen Beispielen immer etwa doppelt so schnell wie *Mathematicas* Integrationsalgorithmus.

4 Details zur Implementierung

Das gesamte Paket wurde in *Mathematica* implementiert. Wie bereits in den Grundlagen erwähnt wurde, mussten neben den bereits erwähnten Algorithmen noch einige weitere Funktionalitäten implementiert werden. Diese Hilfsfunktionen und eine andere Besonderheiten des Algorithmus sollen in diesem Kapitel vorgestellt werden.

4.1 Hilfsfunktionen

4.1.1 EEA

Da der *Mathematica*-Befehl **`PolynomialExtendedGCD`** nur mit einer Variablen arbeiten kann, wurde der Erweiterte Euklidische Algorithmus in **`EEA`** neu implementiert. Dabei wird die angegebene Variable als polynomiell und alle anderen Variablen als rationale Erweiterungen interpretiert. **`EEA`** kann auf zwei verschiedene Arten verwendet werden:

Die Funktion **`EEA`**$[a, b, x]$ liefert für zwei Polynome a und b in x die Polynome r und s, so dass gilt $r\,a + s\,b = g = \gcd(a, b)$. Falls ein weiteres Polynom c im Erzeugnis von a und b liegt, können mit **`EEA`**$[a, b, c, x]$ Polynome r und s erzeugt werden, welche den Bedingungen $c = r\,a + s\,b$ und $\deg_x r < \deg_x b$ genügen.

4.1.2 SquareFreeList

Die Hilfsfunktion **`SquareFreeList`** führt nur leichte Veränderungen an der Ausgabe der *Mathematica*-Funktion **`FactorSquareFreeList`** durch. **`FactorSquareFreeList`** faktorisiert immer bezüglich aller Polynomvariablen. Bei **`SquareFreeList`** kann hingegen durch einen weiteren Parameter gesteuert werden, in welchem Polynomring die quadratfreie Faktorisierung durchgeführt wird.

Beispiel 4.1
Wir betrachten das Polynom

$$p = (x-1)(x+1)(x-2)^2(x-t)^2(t-2)(t+2) \in \mathbb{Q}[x,t]\,.$$

`FactorSquareFreeList` liefert alle quadratfreien Faktoren, die aus einer quadratfreien Faktorisierung in $\mathbb{Q}[x][t]$ oder $\mathbb{Q}[t][x]$ bestimmt werden können:

```
In[50]:= FactorSquareFreeList[p]
```

$$\mathit{Out[50]}= \begin{pmatrix} 1 & 1 \\ t^2-4 & 1 \\ t-x & 2 \\ x-2 & 2 \\ x^2-1 & 1 \end{pmatrix}$$

Im Gegensatz dazu erhalten wir von **`SquareFreeList`** je nach Angabe des zweiten Parameters eine die quadratfreien Faktoren in $\mathbb{Q}[t][x]$ oder $\mathbb{Q}[x][t]$:

```
In[51]:= SquareFreeList[p, t]

Out[51]= {{(x - 2)^2 (x^2 - 1)   1}
          {      t^2 - 4         1}
          {       t - x          2}}

In[52]:= SquareFreeList[p, x]

Out[52]= {{t^2 - 4   1}
          {t - x     2}
          {x - 2     2}
          {x^2 - 1   1}}
```

4.1.3 PartialFraction

PartialFraction hat zwei Implementierungen, mit denen man wahlweise eine einfache oder auch eine vollständige Partialbruchzerlegung durchführen kann.

Im ersten Fall ist die Angabe des Zählers, der paarweise teilerfremden Faktoren des Nenners und der Polynomvariablen notwendig. **PartialFraction** erzeugt dann eine Liste der neuen Zähler. Sei beispielsweise p der Zähler, $q = \prod_{i=1}^{n} q_i$ eine Faktorisierung des Nenners und x die Variable. **PartialFraction**$[p(x), (q_1(x), \ldots, q_n(x)), x]$ liefert eine Liste $\big(p_0(x), \ldots, p_n(x)\big)$, so dass gilt:

$$\frac{p(x)}{q(x)} = p_0(x) + \sum_{i=1}^{n} \frac{p_i(x)}{q_i(x)}\,.$$

Mit einer Faktorisierung $q(x) = \prod_{i=1}^{n} q_i(x)^{e_i}$ können wir die vollständige Partialbruchzerlegung bestimmen, indem wir die Exponenten getrennt angeben. Der Funktionsaufruf **PartialFraction**$[p(x), (q_1(x), \ldots, q_n(x)), (e_1, \ldots, e_n), x]$ erzeugt dann eine verschachtelte Liste der Form:

$$\Big(p_0, \big(p_{11}, \ldots, p_{1e_1}\big), \ldots, \big(p_{n1}, \ldots, p_{ne_n}\big)\Big)\,.$$

Das erste Element ist wieder der polynomielle Anteil und jedes Polynom p_{ij} bildet den Zähler von $q_i(x)^j$. Die Zerlegung hat dann die Form:

$$\frac{p(x)}{q(x)} = \frac{p(x)}{q_1(x)^{e_1} \cdots q_n(x)^{e_n}} = p_0 + \sum_{i=1}^{n} \sum_{j=1}^{e_i} \frac{p_{ij}}{q_i(x)^j}\,.$$

4.2 Integrationsalgorithmus

Bei allen Algorithmen des Risch-Algorithmus ist hinter den individuellen Parametern die Liste der Erweiterungen $(x, t_1, \ldots, t_n)$ über dem festen Grundkörper $\mathbb{C}$ anzugeben. Dabei ist das erste Element x die Integrationsvariable, nach der auch abgeleitet wird, und die anderen Erweiterungen t_i sind logarithmische Erweiterungen. Die Reihenfolge spielt für die Integration nur dann eine Rolle, wenn es verschachtelte Logarithmen gibt. Dann muss die Erweiterung t_i immer elementar über $\mathbb{C}(t_1, \ldots, t_{i-1})$ sein.

Zur Integration der Funktion $x^{-1}\log\log x$ geben wir $(x, \log x, \log(\log x))$ als Erweiterungsliste an. Eine weitere einfache Erweiterung $\log(x+1)$, die bei der Integration von $x^{-1}\log\log x + \log(x+1)$ benötigt wird, kann an eine beliebige Stelle der Liste hinter x stehen. Dies hat natürlich Einfluss auf den Ablauf des Algorithmus und die Ergebnisse können um eine Konstante oder auch in der Darstellung voneinander abweichen.

Hinter der Angabe der Erweiterungsliste können Optionen in der in *Mathematica* üblichen Form `Parameter→Wert` angegeben werden. Als einzige Option steht `Messages` mit folgenden Werten zur Verfügung:

- `None`: keine Meldungen.
- `Basic`: Funktionsaufrufe und Ereignisse, die zum Abbruch führen.
- `More`: Ergebnis von jeder aufgerufenen Funktion und Details der Hermite Reduktion.
- `Detail`: Alle Meldungen. Damit kann man den Algorithmus komplett nachvollziehen.

Mit Ereignissen, die zu einem Abbruch führen, ist beispielsweise das Auftreten von zusätzlichen logarithmischen Erweiterungen bei der Integration von Polynomen gemeint. Die Meldung „Es gibt keine elementare Stammfunktion“ und Meldungen bei falschen Parametern werden immer ausgegeben.

Um den Quellcode der Algorithmen nicht mit Debug-Meldungen zu überlasten, wird die Ausgabe der Meldungen von der separaten Funktion **`IntMessage`** gesteuert.

Mit **`IntRisch`** wird der gesamte Algorithmus gestartet. Als Parameter werden der Integrand und die Liste der Erweiterungen erwartet. Zunächst werden die Parameter überprüft:

- Die erste Erweiterung muss ein Symbol sein, z. B. x oder y,
- alle anderen Erweiterungen müssen logarithmisch sein,
- die Reihenfolge der Erweiterungen muss richtig sein und
- der Integrand muss im angegebene Erweiterungskörper liegen.

Für die letzten beiden Bedingungen wurde die Funktion **`IntRischExtensions`** implementiert. Diese wird auch im polynomiellen Integrationsteil zur Überprüfung der Bedingung (P2) verwendet.

Danach wird der Integrand an die Funktion **`IntRisch2`** weitergegeben. **`IntRisch2`** erwartet die gleichen Parameter wie **`IntRisch`** und wird auch für interne Rekursionen genutzt, da die Überprüfung der Parameter dann nicht mehr erforderlich ist. Je nachdem, ob in der Liste der Erweiterungen nur noch die Integrationsvariable x oder auch andere logarithmische Erweiterungen stehen, wird die rationale Integration mit **`IntRat`** oder der logarithmische Teil des Risch-Algorithmus **`IntRischLog`** aufgerufen.

Ein Problem, das unser Algorithmus nicht bewältigen kann, ist, dass unterschiedliche logarithmischen Erweiterungen den gleichen Differentialkörper liefern können. So sind beispielweise $\log(x+1/2)$ und $\log(2x+1)$ beides Stammfunktionen der Funktion $(x+1/2)^{-1}$, die sich nur um die Konstante $\log 2$ unterscheiden. Weiterhin sind die

Funktionen $\log(x^2 + x/2)$ und $\log x + \log(x + 1/2)$ wegen

$$\log\left(x^2 + x/2\right)' = \frac{2x + 1/2}{x^2 + x/2} = \frac{1}{x} + \frac{1}{x + 1/2} = \left(\log x + \log(x + 1/2)\right)'$$

zwei verschiedene Darstellungen der gleichen Stammfunktion. Für die korrekte Integration ist es also erforderlich, dass wir uns auf einen Standard einigen.

In unserem Algorithmus werden immer Logarithmen erzeugt, die normiert sind und möglichst kleinen Grad haben. Diese Vereinbarung muss auch in der Eingabe eingehalten werden, denn unser Algorithmus wird für $f(x) = (x+1/2)^{-1}\left(\log(x + 1/2) + \log 2\right)$ eine Stammfunktion finden, aber nicht für $f(x) = (x + 1/2)^{-1}\log(2x + 1)$. Dies liegt daran, dass bei der Integration der Koeffizienten von logarithmischen Polynomen keine zusätzlichen Logarithmen auftauchen dürfen.

Um die Vereinbarung einzuhalten, ist es zwingend erforderlich, dass im Rothstein/Trager-Algorithmus die neuen Logarithmen normiert werden. Desweiteren wird bei der rationalen Integration nach der Horowitz Reduktion eine Partialbruchzerlegung durchgeführt. Dadurch wird bei der Eingabe von $(2x + 1/2)/(x^2 + x/2)$ die Umformung

$$\frac{2x + 1/2}{x^2 + x/2} = \frac{1}{x} + \frac{1}{x + 1/2}$$

durchgeführt und die Stammfunktion

$$\int \frac{2x + 1/2}{x^2 + x/2} dx = \log x + \log(x + 1/2)$$

berechnet.

5 Ausblick

Wir haben in dieser Arbeit gesehen, wie wir die Methoden der rationalen Integration auf ähnliche Weise auf die Integration in logarithmischen Erweiterungen anwenden können. Dabei haben wir als Grundlage die Theorie der Differentialkörpererweiterungen genutzt, dessen wichtigstes Instrument der Satz von Liouville war. Wir hatten dadurch eine einfache Darstellung aller einfach integrierbarer Funktionen und haben in den darauf folgenden Beweisen diese Darstellung oft als Ansatz genutzt.

Da wir die Differentialkörpertheorie auch für exponentielle und algebraische Fälle betrachtet haben, könnte man sich überlegen, den Algorithmus auch dafür zu erweitern.

Der rationale Teil im exponentiellen Fall lässt sich wieder sehr leicht integrieren. Wie bei den Logarithmen können wir beweisen, dass eine quadratfreie Faktorisierung für ein exponentielles t auch quadratfrei für die Integrationsvariable x ist. So kann wieder die Hermite Reduktion und anschließend die Rothstein/Trager Resultante angewendet werden.

Der polynomielle Teil bereitet jedoch mehr Schwierigkeiten, denn beim Ableiten bleibt nach dem Satz über die Ableitung exponentieller Polynome (Kapitel 2, Satz 3, S. 22) der Grad des Polynoms in jedem Fall erhalten. Wir können somit nicht einfach iterativ vorgehen. Stattdessen führt das Problem auf Differentialgleichungen einer ganz bestimmten Form, die Risch-Differentialgleichungen $y' + fy = g$. Lösungsalgorithmen hierfür werden unter anderem in [Bro97], Kapitel 6 diskutiert.

Wie schon beim Beweis des Satzes von Liouville auffällt, spielt der algebraische Fall eine Sonderrolle im Integrationsalgorithmus. Die Methoden, die für die beiden transzendenten Fälle aus der rationalen Integration übertragen werden konnten, greifen dort leider nicht mehr.

Weiterhin kann die in der Einleitung angesprochen Form der Eingabe verändert werden. So kann beispielweise die Funktion

$$\mathrm{Li}(x) := \int \frac{1}{\log x} dx$$

im Differentialkörper aufgenommen werden und wir können uns fragen, ob es in einer elementaren Erweiterung dieses Differentialkörpers eine Stammfunktion gibt. Um die Darstellung der trigonometrischen Funktionen im Komplexen zu umgehen, können wir die Eingabe auch auf diese Funktionenklasse erweitern.

Wie bereits erwähnt wurde, ist der Risch-Algorithmus in *Maple* integriert. Überraschenderweise scheiterte *Maple* aber an Beispiel 12.11 aus [GCL92], obwohl *Mathematicas* Integrationsbefehl und auch der im Rahmen dieser Arbeit implementierte Algorithmus ein Ergebnis lieferten.[4] Es ließ sich allerdings nicht feststellen, wieso die Integration scheiterte. Der in Beispiel 3.6 angesproche Fehler scheint es jedenfalls nicht zu sein, denn die dort betrachtete Funktion wurde auch von *Maple* ordnungsgemäß integriert. Außerdem wäre ein Laufzeitvergleich mit dem in *Maple* integrierten Risch-Algorithmus interessant.

[4] Das Beispiel befindet sich auf der beiliegenden CD.

Literatur

[Beh04] Eberhard Behrends. *Analysis Band 2*. Vieweg, Wiesbaden, 2004.

[Bos04] Siegfried Bosch. *Algebra*. Springer, Berlin, 5th edition, 2004.

[Bro97] Manuel Bronstein. *Symbolic Integration I*. Springer, Berlin, 2nd edition, 1997.

[Bro98] Manuel Bronstein, editor. *Symbolic Integration Tutorial*, Rostock, August 1998. ISSAC.

[BSMM97] Ilja N. Bronstein, Konstantin A. Semendjajew, Gerhard Musiol, and Heiner Mühlig. *Taschenbuch der Mathematik*. Harri Deutsch, Frankfurt am Main, 3rd edition, 1997.

[GCL92] Keith O. Geddes, Stephen R. Czapor, and George Labahn. *Algorithmns for Computer Algebra*. Kluwer, Boston, 1992.

[Kas80] Toni Kasper. Integration in finite terms: The liouville theory. *Mathematics Magazine*, 53(4):195–201, September 1980.

[Ris69] Robert H. Risch. The problem of integration in finite terms. *Trans. of AMS*, 139:167–189, 1969.

[Ros72] Maxwell Rosenlicht. Integration in finite terms. *Am. Math. Monthly*, 9:963–972, 1972.

[vzGG03] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, 2nd edition, 2003.

[Wol] Wolfram Research. *Mathematica Dokumentation*.

Ich versichere hiermit, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

RUBEN DEBEERST