Computeralgebra

Prof. Dr. K. Madlener

12. April 2010

Einführung „Computer Algebra“

- Berechnungen in algebraischen Strukturen: Monoide, Gruppen, Ringe, Polynomringe, Körper, Körpererweiterungen, Moduln, Vektorräume ...
- Lineare Algebra: Vektorräume, Matrizen, Determinanten, Gleichungssysteme.
- Analysis: Grenzwerte, Funktionen, Differentiation, Integration, Differentialgleichungen.
- Algebra: Gruppen, Ringe, Körper, Moduln, Konstruktionen: Produkte, Quotienten, Unterstrukturen.
- Zahlentheorie: Primzahlen, Faktorisierung, Kryptographie.
- Numerische Berechnungen vs. Symbolische Berechnungen.
- Computer Algebra Systeme, seit 60’er Jahren.
- Inhalt–Umriss

Moderne CA-Systeme

Derive, Macsyma, Maple, Mathematica, Reduce, Scratchpad, Mupad, Mumath, Axiom, Magma, Mathlab

Ziele:

- Breite Funktionalität
- Einfache Bedienung
- Effizienz
- Erweiterbarkeit

Probleme:

- Darstellung der Strukturen und ihrer Elemente
- Effiziente Lösungen: Darstellungsabhängig
- Effiziente Transformationen zwischen Darstellungen

Vorteile CA-Systeme

Verarbeitung großer algebraischer Berechnungen
$\rightsquigarrow$ genaue Berechnungen „fehlerfrei“

Grundoperationen: Multiplikation, Division, Addition, Substraktion, Exponentation
$\rightsquigarrow$ Arithmetik, Langzahlarithmetik

- GGT, KGV:
  Euklidischer Algorithmus (Ringe euklidisch z. B. $\mathbb{Z}, \mathbb{Q}[x]$)
- Faktorisierung:
  UFD (ZPE)-Ringe, Prim-Elemente (z. B. $\mathbb{Z}, \mathbb{Z}[x, y, z], .$)
- Klassische Algorithmen sind nicht immer effizient.
  Problem: Zwischengrößenwachstum
- Kosten arithmetischer Operatoren hängt von der Länge der Operanden ab.

Probleme bei der Implementierung von CA-Systemen

- Allgemeine Systeme: Sprachumgebung, Notationen, Ein/Ausgaben,...
- Erfordern oft spezielle Programmiersprachen und Umgebungen
- Spezielle Systeme, z. B. Gruppen oder Gröbnerbasen, können oft nicht in andere Systeme verwendet werden.
- Vielzahl algorithmischer Lösungen, Vergleich schwer.
- Analyse der Algorithmen erfordert oft tiefe mathematische Ergebnisse.
- Wahl der Implementierungs- und Programmiersprachen

Symbolische Numerische Berechnungen

1.1 Beispiel Chebyshev-Polynome. Rekursive Definition.

$$T_0(x) = 1;\ T_1(x) = x;\ T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \text{ für } k \geq 2.$$

Liste der Polynome: $1, x, 2x^2 - 1, 4x^3 - 3x, 8x^4 - 8x^2 + 1, \ldots$

Werte die Polynome an bestimmten Stellen aus.
Etwa für $x = 0.3$: $1, 0.3, -0.82, -0.792, 0.3448, \ldots$

Programm: Berechnung der 5 ersten Werte an einer Stelle x.

Für 0.3 sollte das Programm die Ausgabe:
$T_0[0.3] = 1.0;\ T_1[0.3] = 0.3;\ T_2[0.3] = -0.82;\ T_3[0.3] = -0.792;$
$T_4[0.3] = 0.3448$ liefern.

Standard Algorithmus in Programmiersprachen

procedure Chebyshev (input, output);
begin
var x: real; T: array[0..4] of real; n: integer;
writeln(x eingeben);
read(x);
$T[0] := 1; T[1] := x;$
for $n := 2$ **to** 4 **do**
    $T[n] := 2 \cdot x \cdot T[n-1] - T[n-2];$
for $n := 0$ **to** 4 **do**
    writeln('T'$_n[x] = T[n]$)
end.

Maple Version für Chebyshev Polynome

$T[0] := 1;$

$T[1] := x;$

for $n = 2$ **to** 4 **do**
$T[n] := \text{expand}(2 \cdot x \cdot T[n-1] - T[n-2]);$

Interne Darstellung

$$T[0] := 1$$

$$T[1] := x$$

$$T[2] := 2x^2 - 1$$
$$T[3] := 4x^3 - 3x$$
$$T[4] := 8x^4 - 8x^2 + 1$$

Externe Darstellung.

Historische Entwicklung der Case

Faktoren:

- Systeme (Programmiersprachen,HW.)
- Algorithmen (spezielle Lösungen)
- Anwendungen (Erweiterungen)

Höhere Programmiersprachen: Ende der 50er Anfang 60er.
Fortran (58), Algol (60), Lisp (61).

Systeme **1961-1966**:

- I. Slagle (MIT): Lisp-Programm SAINT (Symbolic Automatic Integration): Lösen von unbestimmten Integralen unter Ausnutzung von Heuristiken.
- J. Sammet, R. Tobey (IBM): FORMAC (Fortran-Preprozessor): Symbolisches Rechnen mit elementaren Funktionen: Polynome, rationale Funktionen,u.a.
- W.S. Brown (Bell Labs): ALPAK (in Assembler geschriebene Subroutinen für Fortran): Symbolisches Rechnen mit Polynomen und rationalen Funktionen.
- G. Collins (IBM, University of Wisconsin at Madison): PM: symbolisches Rechnen mit Polynomen.
- C. Engelman (MIT): MATHLAB (LISP-basiert): symbolisches Rechnen mit Polynomen und rationale Funktionen, erstes interaktives System.

Systeme **1966-1971**:

- J. Moses (MIT): LISP-Programm SIN (Symbolic INtegrator).
- T. Hearn (Stanford University): REDUCE (LISP-basiert, interaktiv): für physikalische Berechnungen, hohe Portabilität.
- C. Engelman (MIT):MATHLAB-68 (graphische Ausgaben).
- A.D. Hall: ALTRAN (ALgebraic TRANslator): Sprache und System für das symbolische Rechnen mit Polynomen und rationale Funktionen.
- G. Collins: SAC-1 (Symbolic and Algebraic Calculations).
- D. Barton, S. Bourne, J. Fitch (University of Cambridge): CAMAL (CAMbridge ALgebra system: für astronomische Berechnungen und für Berechnungen der allgemeinen Relativitätstheorie.
- T. Hearn: REDUCE-2: allgemeines System mit Schwerpunkt für Berechnungen in der Hochenergie-Physik, geschrieben in RLISP (ALGOL-ähnlich).

Systeme **1971-1981**:

Alle bisherigen Systeme rein experimenteller Natur, wurden auch außerhalb der Gruppe der Entwickler verwendet. Insbesondere REDUCE weite Verbreitung aufgrund der leichten Portierbarkeit.

- J. Griesmer, R. Jenks (IBM Research): SCRATCHPAD: LISP-basiert, interaktiv, beinhaltet MATHLAB-68, REDUCE-2 und SIN.
- J. Moses, W. Martin (MIT): MACSYMA: algebraische Berechnungen, Grenzwert-Berechnungen, symbolisch Integrieren, Lösen von Gleichungen.
- G. Collins, R. Loos: SAC/ALDES: Bibliothek von Modulen, die in ALDES (ALgebraic DEScription language) geschrieben sind, zusammen mit einem Übersetzer nach ANSI FORTRAN. Alle verwendeten Algorithmen waren vollständig und ausführlich dokumentiert.
- D. Stautemeyer, A. Rich (University of Hawai): muMATH: eigene Programmiersprache, lief auf PC.

Spezielle Systeme:

- I. Frick (University of Stockholm): SHEEP: Berechnungen von Tensorprodukten.
- W. Jeffreys (University of Texas at Austin): TRIGMAN: in FORTRAN geschrieben, zur Berechnung von Poisson-Reihen.
- H. Veltman (NL): SCHOONSHIP: für Berechnungen in der Hochenergie-Physik.
- V.M. Glushkov (Hiev): ANALYTIK: Implementierung in Hardware.

CASe, die portabel sind meistens C-basiert. Wegen der stark angestiegenen Rechenleistung der Computer finden CASe mehr und mehr Anwendungen und Benutzer. Insbesondere entstehen nun auch kommerzielle CASe.

Systeme **1981-1991**:

- G. Gonnet, K. Geddes (University of Waterloo): MAPLE: modulare Struktur, bestehend aus einem kleinen kompilierten Kern in C, und einer großen Library von mathematischen Subroutinen, die alle in der eigenen MAPLE Sprache geschrieben sind. Interpreter für die Kommandos, Integer und rationale Arithmetik, Polynom-Routinen und ein effizientes Speicherverwaltungssystem.
- S. Wolfram (Caltech): SMP (Symbolic Manipulation Program): in C geschrieben, Regel-basiert.
- S. Wolfram: MATHEMATICA: symbolische und numerische Berechnungen, graphische Wiedergabe (2-D und 3-D, inkl. Animation), C-basiert mit eigener Programmiersprache.
- D. Stoutemeyer, A. Rich: DERIVE: interaktiv, nicht als Programmierumgebung.
- weitere allgemeine Systeme: REDUCE 3, DOE-MACSYMA, MuPAD, AXIOM (SCRATCHPAD II).

Spezielle Systeme:

- J. Cannon (University of Sydney): CAYLEY: Gruppentheoretische Berechnungen. Mittlerweile MAGMA.
- J. Neubüser (RWTH Aachen): GAP (Group Algorithms and Programming). Mittlerweile in St. Andrews neu implementiert.
- J. Vermaseren: FORM: Berechnungen in der Hochenergie-Physik.
- A.M. Cohen: Lie: Berechnungen in Lie Algebren.
- M. Stillman: MACAULAY: Algebraische Geometrie und komm. Algebra.
- H. Cohen: PARI: Zahlentheorie.
- Greuel, Pfister (KL): SINGULAR: Gröbner Basen, Algebraische Geometrie, Singularitäten.
- COCOA (Genova) Kommutative Algebra. MAGNUS Gruppen.

Literatur

- von zur Gathen/Gerhard: Modern Computer Algebra, 1999, Cambridge University Press, ISBN 0-521-64176-4, INF 235/167 und L inf 92
- Geddes/Czapor/Labahn: Algorithms for Computer Algebra, INF 235/132, L inf 694.
- Davenport/Siret/Tournier: Computer Algebra, INF 235/116.
- Buchberger et al. (Eds.): Computer Algebra, INF 235/095.
- Mignotte: Mathematics for Computer Algebra, INF 235/126.
- Mignotte/Stefanescu: Polynomials: An Algorithmic Approach, INF 246/057.
- Winkler: Polynomial Algorithms in Computer Algebra, INF 235/132.
- Zippel: Effective Polynomial Computation, INF 246/054.
- Kreuzer, Robbiano Computational Commutative Algebra (0,1,2)

Inhalt Teil I

Grundlegende Konzepte der Algebra

Axiome:

A_1	Assoziativität:	$a \circ (b \circ c) = (a \circ b) \circ c$
A_2	Neutrales Element:	$e \circ a = a \circ e = a$ (alle a)
A_3	Inverse:	$a \circ a^{-1} = a^{-1} \circ a = e$ (alle a)
A_4	Kommutativität:	$a \circ b = b \circ a$ (alle a, b)
A_5	Distributivität:	$a \circ (b + c) = (a \circ b) + (a \circ c)$
		$(a + b) \circ c = (a \circ c) + (b \circ c)$
A_6	Nullteilerfreiheit:	$a \circ b = 0 \Rightarrow a = 0$ oder $b = 0$.

Additiv neutrales Element: 0 null. Multiplikativ neutrales Element: 1 eins.

Struktur	Notation	Axiome
Gruppe	$(G; \circ)$	$A_1; A_2; A_3$
Abelsche Gruppe	$(G; \circ)$	$A_1; A_2; A_3; A_4$
Ring	$(R; +, \cdot)$	$A_1; A_2; A_3; A_4$ bzgl. $+$, $A_1; A_2$ bzgl. $\cdot$, A_5
Kommutative Ringe	$(R; +, \cdot)$	$+A_4$ bzgl. $\cdot$
Integritätsbereich	$(D; +, \cdot)$	$+A_6$
Körper	$(F; +, \cdot)$	$+A_3$ für $F \backslash \{0\}$ bzgl. $\cdot$ (A_6 folgt daraus)

Grundlegende Konzepte der Algebra

2.1 Beispiel Strukturen

$\mathbb{Z}$ Integritätsbereich, $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ Körper,
$\mathbb{Z}_n$ Ring, $n \in \mathbb{N}$, Primzahl p $\mathbb{Z}_p$ Körper, sonst kein Integritätsbereich.
Beachte: jeder endliche Integritätsbereich ist Körper.
Teilbarkeit und Faktorisierung in Integritätsbereichen

$$a, b \in D \quad a \mid b \ (a \text{ teilt } b) \ \text{gdw} \ b = ax \text{ für ein } x \in D.$$

a Teiler von b, b Vielfaches von a.
GGT (GCD) größter gemeinsamer Teiler von $a, b \in D$ ist $c \in D$ mit $c \mid a$ und $c \mid b$, wobei c Vielfaches von jedem d mit $d \mid a$ und $d \mid b$ ist.
KGV (LCM) kleinstes gemeinsames Vielfaches von $a, b \in D$ ist $c \in D$ mit $a \mid c$ und $b \mid c$, wobei c Teiler von jedem d mit $a \mid d$ und $b \mid d$ ist.
Beachte: nicht eindeutig.
Assoziierte Elemente: $a, b \in D$ sind assoziiert, falls $a \mid b \wedge b \mid a$
Einheiten: $a \in D$ ist Einheit, falls a eine multiplikative Inverse besitzt.

Grundlegende Konzepte der Algebra

2.2 Beispiel Einheiten, assoziierte Elemente:
$\mathbb{Z}$ :: Einheiten $1, -1$ $\quad$ $6, -6$ sind GGT von 18 und 30,
$6, -6$ sind assoziiert.
Assoziert sein ist Äquivalenzrelation, Klassen assoziierter Elemente.
$\mathbb{Z}$ :: $\{0\}, \{1, -1\}, \{2, -2\}, \ldots$
Auswahl von Repräsentanten aus assoziierten Klassen:$\rightsquigarrow$
Einheits-normale Elemente:: $n : D \to D$ $\quad$ $n(a)$ EN.

$\mathbb{Z}$ :: $\mathbb{N}$ als Einheits-normale Elemente
F Körper:: $0, 1$ Einheits-normal.
Einheits-normaler GGT und KGV sind dann eindeutig.

In der Regel: 0 ist EN, 1 ist EN, a, b EN, so auch ab.
Für $a \in D$, so $a = u(a)n(a)$ eindeutig, wobei $u(a)$ Einheit und $n(a)$ EN.
Diese Begriffe, geeignet angepasst, sind auch für Ringe und Halbgruppen sinnvoll.
Nullteiler: $0 \neq a \in R$ ist Nullteiler, falls $ab = 0$ für ein $0 \neq b \in D$.

ZPE (UFD)-Bereiche

Prim-Elemente (irreduzible Elemente):: $p \in D - \{0\}$ ist Prim, falls

a) p ist keine Einheit, b) $p = ab$, so a oder b Einheit.

$a, b \in D$ heißen relativ Prim (teilerfremd), falls $\mathrm{GGT}(a, b) = 1$.

Ein Integritätsbereich heißt ZPE-Ring (UFD), falls für $a \in D - \{0\}$ gilt, a ist Einheit oder a kann als (endliches Produkt) von Primelementen dargestellt werden und diese Darstellung ist eindeutig bis auf Assoziativität und Umordnung:
$a = p_1 \cdots p_n$ Primelemente p_i $1 \leq i \leq n$ und ist $a = q_1 \cdots q_m$ mit Primelemente q_i, so $m = n$ und bei geeigneter Umordnung der q_j gilt p_i ist assoziert mit q_i.
Wähle **Einheitsnormale Primelemente** : Eindeutige Primfaktorisierung:
$a = u(a) p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$ p_i Einheitsnormale Primelemente, $p_i \neq p_j$ für $i \neq j$.
2.3 Satz
Ist D ein ZPE-Ring $\{a, b\} \neq \{0\}$, so existiert $\mathrm{GGT}(a, b)$ und ist eindeutig.

Polynomringe

Polynomring über R: $R[x]$

$a(x) = \sum_{k=0}^{m} a_k x^k$, $a_k \in R$, $m \geq 0$, Koeffizienten a_k

Grad von $a(x)$: größtes n mit $a_n \neq 0$:: $\operatorname{grad}(a(x))$

Standardform $\sum_{k=0}^{n} a_k x^k$, $a_n \neq 0$: a_n Leitkoeffizient

0 Polynom: $a_k = 0$ für alle k:: $\operatorname{grad}(0) := -\infty$

Monisch, falls $a_n = 1$

Addition, Multiplikation definiert wie üblich.

Eigenschaften von $\boldsymbol{R[x]}$

i) R ist kommutativ, so auch $R[x]$, 0 add. Einheit, 1 mult. Einheit.

ii) D Integritätsbereich, so auch $D[x]$.
Einheiten sind die konstanten Polynome a_0 mit a_0 ist Einheit in D.

iii) Ist D ein ZPE-Ring, so auch $D[x]$.

Polynome mit einheitsnormalen Leitkoeffizienten heißen einheitsnormal.

Polynomringe: Beispiele

2.4 Beispiel

$\mathbb{Z}[x]$: Einheiten $1, -1$, einheitsnormale Polynome sind solche mit positiven Leitkoeffizienten.

$\mathbb{Q}[x]$: Einheiten sind konstante Polynome $(\neq 0)$. Einheitsnormale Polynome: Leitkoeffizienten 1 bzw. 0.

$\mathbb{Z}_p[x]$ p prim: wie in $\mathbb{Q}[x]$.

Beachte

$\text{grad}(a(x) + b(x)) \leq \max(\text{grad}(a(x), \text{grad}(b(x)))$
$(=$ falls $\text{grad}(a(x)) \neq \text{grad}(b(x))$

$\text{grad}(a(x) \cdot b(x)) = \text{grad}(a(x)) + \text{grad}(b(x))$

Euklidische Bereiche

2.5 Definition

Euklidischer Ring ist Integritätsbereich D mit einer Bewertung $v : D - \{0\} \to \mathbb{N}$, die folgende Eigenschaft hat:

1) für alle $a, b \in D - \{0\} : v(ab) \geq v(a)$
2) für alle $a, b \in D$ mit $b \neq 0$ gibt es $q, r \in D$: $a = bq + r$, wobei $r = 0$ oder $v(r) < v(b)$.

2.6 Beispiel

- $\mathbb{Z}$ mit $v(a) = |a|$ erfüllt 1), 2).
- F Körper $v(a) = 1$, $a \neq 0$ (r immer 0).
- $\mathbb{Q}[x]$ (allgemeiner $F[x]$ F Körper) $v(a(x)) = \text{grad}(a(x))$ erfüllt 1) + 2).

Beachte:

q (Quotient), r (Rest) in 2) müssen nicht eindeutig bestimmt sein (falls $r \neq 0$).

Euklidische Bereiche

2.7 Beispiel

$\mathbb{Z}: a = -8 \quad b = 3$, so
$-8 = 3 \cdot (-2) - 2 = 3 \cdot (-3) + 1$,
d. h. $q = -2$, $r = -2$ und $q = -3$, $r = 1$ erfüllen 2).

Vereinbarungen um Eindeutigkeit zu erreichen:

- In $\mathbb{Z}$
  a) Wähle q, r mit $r = 0$ oder $\text{sign}(r) = \text{sign}(a)$
  b) Wähle q, r mit $r = 0$ oder $\text{sign}(r) = \text{sign}(b)$
- In $F[x]$ sind q, r eindeutig. (warum?)

Euklidische Ringe sind ZPE-Ringe.
$g = \text{GGT}(a, b)$, so gibt es $s, t \in D$ mit $g = sa + tb$ (nicht eindeutig!)
s, t heißen Bezout-Koeffizienten.
Annahme: In "effektiven" Euklidischen Ringen sein zu a, b stets eindeutige q, r berechenbar.

Euklidischer Algorithmus

2.8 Beispiel

In $\mathbb{Z}$:: GGT-Berechnung von 126 35

$$\begin{aligned} 126 &= 3 \cdot 35 + 21 \\ 35 &= 1 \cdot 21 + 14 \\ 21 &= 1 \cdot 14 + 7 \\ 14 &= 2 \cdot 7 + 0 \end{aligned}$$

7 ist $\mathrm{GGT}(126, 35)$

Anwendung: Simplifikation rationaler Ausdrücke: $35/126 \rightsquigarrow 5/18$

Nutzen: Zahlen „klein“ halten.

Sei D euklidischer Bereich $a, b \in D$, $b \neq 0$. Seien q, r Quotient und Rest mit $a = bq + r$, wobei $r = 0$ oder $v(r) < v(b)$ setze

$$\begin{aligned} \mathrm{quo}(a, b) &= q \quad (\text{ auch } a \text{ quo } b) \text{ und} \\ \mathrm{rem}(a, b) &= r \quad (\text{ auch } a \text{ rem } b \text{ oder } a \bmod b) \end{aligned}$$

Es gilt dann $\mathrm{GGT}(a, b) = \mathrm{GGT}(b, r)$

Grundlage für Euklidischen Algorithmus

2.9 Lemma $\mathrm{GGT}(a,b) = \mathrm{GGT}(b,r)$

Beweis: Sei $a = bq + r$, dann gilt
$\mathrm{GGT}(b,r)|a$ und $|b \rightsquigarrow \mathrm{GGT}(b,r)|\ \mathrm{GGT}(a,b)$, wegen $r = a - bq$ folgt
$\mathrm{GGT}(a,b)|r$ und $|b \rightsquigarrow \mathrm{GGT}(a,b)|GGT$(b,r), d. h.
$\mathrm{GGT}(a,b)$ und $\mathrm{GGT}(b,r)$ sind assoziiert, da EN sind sie gleich.

Seien $a, b \in D$, $b \neq 0$, $v(a) \geq v(b)$.
Eine Restefolge für a, b ist definiert durch die Folge $\{r_i\}$ mit
$r_0 := a$, $r_1 := b$ und $r_i = \mathrm{rem}(r_{i-2}, r_{i-1})$, $i = 2, 3, 4 \ldots$

Es gilt $v(r_0) \geq v(r_1) > v(r_2) > v(r_3) \ldots$
Es gibt ein k mit $r_{k+1} = 0 \quad (k \leq v(b))$ und $\mathrm{GGT}(a,b) = n(r_k)$.

Procedure Euclid

procedure Euclid (a,b)
{Berechne $g = \mathrm{GGT}(a, b)$ $a, b \in D$ euklid. Bereich}
begin
$c := n(a); d := n(b);$
while $d \neq 0$ **do**
begin
$r := \mathrm{rem}(c, d);$
$c := d;$
$d := r;$
end
$g := n(c);$ **return** g
end.

Korrektheit und Terminierung folgen aus Lemma und Restefolgeneigenschaften. Komplexitätsanalyse folgt.

Erweiterter euklidischer Algorithmus (EEA)

procedure $EEA(a, b; s, t)$
$\{$Berechne $g = \mathrm{GGT}(a, b)$ und $s, t \in D$ mit $g = sa + tb\}$
begin
$c := n(a); d := n(b); c_1 := 1; d_1 := 0; c_2 := 0; d_2 := 1;$
while $d \neq 0$ **do**
begin
$q := \mathrm{quo}(c, d); r := c - q \cdot d;$
$r_1 := c_1 - q \cdot d_1; r_2 := c_2 - q \cdot d_2;$ $\{Invariante::\}$
$c := d; c_1 := d_1; c_2 := d_2;$ $\{c = c_1 n(a) + c_2 n(b) \wedge\}$
$d := r; d_1 := r_1; d_2 := r_2;$ $\{d = d_1 n(a) + d_2 n(b)\}$
end
$g := n(c);$
$s := c_1/(u(a) \cdot u(c)); t := c_2/(u(b) \cdot u(c));$ **return** $(g, s, t);$
end.

Beachte:
$n(c) = c_1 \cdot \frac{n(a)}{u(c)} + c_2 \cdot \frac{n(b)}{u(c)}$: d.h. s, t sind die Bezout-Koeffizienten.

Erweiterter euklidischer Algorithmus: Beispiel

2.10 Beispiel In $\mathbb{Z}$:: $a = 18$ $b = 30$

Wertefolge::

Iteration	q	c	c_1	c_2	d	d_1	d_2
0	–	18	1	0	30	0	1
1	0	30	0	1	18	1	0
2	1	18	1	0	12	−1	1
3	1	12	−1	1	6	3	−1
4	2	6	2	−1	0	−5	3

$g = 6,\ s = 2,\ t = -1,\ \mathrm{GGT}(18, 30) = 2 \cdot 18 - 1 \cdot 30 = 6$

Erweiterter euklidischer Algorithmus: Beispiel

In $\mathbb{Q}[x]$:: $a = 12x^3 - 28x^2 + 20x - 4$, $b = -12x^2 + 10x - 2$

$u(a) = 12$ $\qquad$ $u(b) = -12$

Iter.	q	c	c_1	c_2	d
–	–	$x^3 - \frac{7}{3}x^2 + \frac{5}{3}x - \frac{1}{3}$	1	0	$x^2 - \frac{5}{6}x + \frac{1}{6}$
1	$x - \frac{3}{2}$	$x^2 - \frac{5}{6}x + \frac{1}{6}$	0	1	$\frac{1}{4}x - \frac{1}{12}$
2	$4x - 2$	$\frac{1}{4}x - \frac{1}{12}$	1	$-x + \frac{3}{2}$	0

$g = n(c) = x - \frac{1}{3}$, $s = \frac{c_1}{u(a)u(c)} = \frac{1}{12\frac{1}{4}} = \frac{1}{3}$

$t = \frac{-x+\frac{3}{2}}{(-12)\frac{1}{4}} = \frac{x-\frac{3}{2}}{3} = \frac{x}{3} - \frac{1}{2}$

$x - \frac{1}{3} = \frac{1}{3}a + \left(\frac{x}{3} - \frac{1}{2}\right) b$

Kostenanalyse von EAA für $\mathbb{Z}$ und $F[x]$

Seien $a, b \in R$ mit $n = v(a) \geq v(b) = m \geq 0$.
Die Anzahl l der Durchläufe der While-Schleife wird durch $l \leq v(b) + 1$ beschränkt. Die wesentliche Operation ist die Division mit Rest.
Diese ist l-mal durchzuführen: $l \leq v(b) + 1 = m + 1$.

Sei $R = F[x]$, F Körper, dann $v(a) = \text{grad}(a)$.
Zähle Grundoperationen (go) in F:
Kosten der Division mit Rest: Seien $\text{grad}(a) = n$, $\text{grad}(b) = m$.
Ein Durchgang der Division kostet: Eine Division, m Multiplikationen, m Additionen in F, $n - m + 1$ Durchläufe, d. h.

$$(2m + 1)(n - m + 1) = (2\ \text{grad}(b) + 1)(\text{grad}(q) + 1) \in O(n^2)$$

Operationen in F. Ist b monisch, so spart man die Division.
Sei $n_i = \text{grad}(c)$ in Durchlauf i $(0 \leq i \leq l + 1)$, wobei d in Durchlauf l Null wird. Dann gilt $n_0 = n \geq n_1 = m > n_2 > \cdots > n_l$ und $\text{grad}(q_i) = n_{i-1} - n_i$ für $1 \leq i \leq l$ (q_i Wert von q in Durchlauf i). Kosten der Division mit Rest: $(2n_i + 1)(n_{i-1} - n_i + 1)$ arithm. Operationen in F.

Kostenanalyse von EAA für $F[x]$: Kosten für s und t

Die Kosten für die r_i und q_i sind $\sum_{1\le i\le l}(2n_i+1)(n_{i-1}-n_i+1)$ Operationen in F. Normaler Fall: $n_i = n_{i-1}-1 = \cdots = m-i+1$
$2 \le i \le l = m+1 \le 2mn+2m$.

2.11 Lemma Sei s_i Wert von c_1 in Durchgang i und t_i Wert von c_2 in Durchgang i. Dann gilt

1. $\text{grad } s_i = \sum_{2\le j<i} \text{grad } q_j = n_1 - n_{i-1} \quad 2 \le i \le l+1$
2. $\text{grad } t_i = \sum_{1\le j<i} \text{grad } q_j = n_0 - n_{i-1} \quad 1 \le i \le l+1$

Beweis: Wir zeigen nur 1) und $\text{grad } s_{i-1} < \text{grad } s_i \quad (2 \le i \le l)$ durch Induktion nach i.
$i = 2$:: $s_2 = (s_0 - q_1 s_1) = 1 - q_1 \cdot 0$, $\text{grad } s_1 = -\infty < 0 = \text{grad } s_2$.
Sei $i \ge 2$ Behauptung richtig für $2 \le j \le i$, dann

$$\text{grad } s_{i-1} < \text{grad } s_i < n_{i-1} - n_i + \text{grad } s_i = \text{grad } (q_i s_i)$$

Kostenanalyse von EAA für $F[x]$: Kosten für s und t

Also grad $s_{i+1} = \text{grad}\ (s_{i-1} - q_i s_i) = \text{grad}\ q_i + \text{grad}\ s_i > \text{grad}\ s_i$
und
$\text{grad}\ s_{i+1} = \text{grad}\ q_i + \text{grad}\ s_i = \sum_{2 \le j < i} \text{grad}\ q_j + \text{grad}\ q_i = \sum_{2 \le j < i+1} \text{grad}\ q_j$

Die Berechnung $t_{i+1} = (t_{i-1} - q_i t_i)$ bzw. $s_{i+1} = (s_{i-1} - q_i s_i)$.
Multiplikation von Pol grad $n, m :\le 2(n+1)(m+1)$ Operationen.
$2(\text{grad}\ t_i + 1)(\text{grad}\ q_i + 1) + (\text{grad}\ t_{i+1} + 1)$, d. h.

$$\sum_{2 \le i \le l} 2(n_0 - n_{i-1} + 1)(n_{i-1} - n_i + 1) + (n_0 - n_i + 1)$$

Normalfall

$$\sum_{2 \le i \le m+1} 2(n - m + i - 1)2 + n - (m - i + 1) + 1 =$$

$$\sum_{2 \le i \le m+1} 5n - 5m + 5i - 4 = 5nm - 5mm + \frac{5}{2} m(m+1) + O(m)$$

Kostenanalyse für $\mathbb{Z}$: Langzahlarithmetik

Darstellung von Zahlen: Wort 64 Bits. 2^{64}-Standard Darstellung: Zahl als Feld von Wörtern. Erstes Wort für Vorzeichen und Länge des Feldes, d. h. $a \in \mathbb{Z}$

$$a = (-1)^s \sum_{0 \le i \le n} a_i 2^{64i}$$

$s \in \{0,1\}, 0 \le n+1 < 2^{63}, a_i \in \{0, \ldots, 2^{64}-1\}$.

Als Feld: $s2^{63}+n+1, a_0, \ldots, a_n$ von 64 Bit-Wörtern, z. B. $-1 : 2^{63}+1, 1$ und $1 : 1, 1$.

Bereich: $-2^{64 \cdot 2^{63}} + 1$ bis $2^{64 \cdot 2^{63}} - 1$.

Länge von a: $\lambda(a) = \lfloor \log_{2^{64}} |a| \rfloor + 1 = \left\lfloor \frac{\log_2 |a|}{64} \right\rfloor + 1$.

Allgemein: Darstellung zur Basis b mit $2 \le b < \frac{|w|}{2}$, wobei $|w|$ Wortlänge ist (Multiplikation der Koeffizienten in Wort).

$a = (u_1 \ldots u_n)_b \quad 0 \le u_i < b$, d. h. $a = \sum_{i=1}^{n} u_i b^{n-i}$

$= u_n + u_{n-1} b + \cdots + u_1 b^{n-1}$ a ist n-stellig zur Basis b.

$a < b^n \rightsquigarrow a$ hat Länge $\le n$.

Langzahlarithmetik: Klassische Algorithmen

Klassische Algorithmen für: $+, -, \cdot,$ quo, Exponentation

Maß in **Grundoperationen** (go):

- Addition, Substraktion von 1-stelligen Zahlen
- Multiplikation von 1-stelligen Zahlen
- Division von 1-stelligen Zahlen

Algorithmen: Addition

A: Addition nicht negativer ganzer Zahlen zur Basis b.

Eingabe: $(u_1 \cdots u_n)_b \quad (v_1 \cdots v_n)_b$
Ausgabe: $(w_0 \cdots w_n)_b$ w_0 Übertrag mit

$$(u_0 \cdots u_n)_b + (v_1 \cdots v_n)_b = (w_0 \cdots w_n)_b$$

begin
$j := n; k := 0$ $\{k = \textit{Übertrag}\}$
while $j > 0$ **do**
begin
$w_j := (u_j + v_j + k) \bmod b;$ $\{k \in \{0, 1\}\}$
$k := \lfloor (u_j + v_j + k)/b \rfloor;$
$j := j - 1;$
end
$w_0 := k;$
end.

Korrektheit! Aufwand $\approx 2n$ go.

Algorithmen: Substraktion

S: Substraktion nicht negativer ganzer Zahlen.

Eingabe: $(u_1 \cdots u_n)_b \geq (v_1 \cdots v_n)_b$
Ausgabe: Nichtnegative Differenz: $u - v = (w_1 \cdots w_n)_b$

begin
$j := n; k := 0$
while $j > 0$ **do**
begin
$w_j := (u_j - v_j + k) \bmod b;$
$k := \lfloor (u_j - v_j + k)/b \rfloor$ $\{k \in \{0, -1\}\}$
$j := j - 1;$
end
end.

Korrektheit! Aufwand $\approx 2n$ go.

Algorithmen: Multiplikation

M: Multiplikation nicht negativer ganzer Zahlen Basis b.
Eingabe: $(u_1 \cdots u_n)_b \geq (v_1 \cdots v_m)_b$, d. h. $n \geq m$
Ausgabe: Produkt $u \cdot v = (w_1 \cdots w_{m+n})_b$

for i from 1 **to** n **do**
    $w_{m+i} := 0$; {*Initialisierung* $m+i-$ *te Stelle*}
$j := m$;
while $j > 0$ **do**
    begin
    if $v_j = 0$ **then**
        $w_j := 0$
    else
        begin
        $i := n$; $k := 0$;
        while $i > 0$ **do**
            $t := u_i v_j + w_{i+j} + k$; $w_{i+j} := t \bmod b$; $k := \lfloor t/b \rfloor$; $i := i - 1$;
        $w_j := k$;
        end
    $j := j - 1$;
    end {Korrektheit! Aufwand $\approx 3nm$ go}

Algorithmen: Motivation für Multiplikationsalg.

$$(u_1 \cdots u_n)(v_1 \cdots v_m)$$

$$\left.\begin{array}{rr} (u_1 v_m) \cdots & (u_{n-1} v_m)(u_n v_m) \\ (u_1 v_{m-1}) \cdots & (u_n v_{m-1}) \end{array}\right\} m$$

$$(u_1 v_1) \cdots (u_n v_1)$$

$$w_1 \cdots w_m w_{m+1} \qquad \cdots w_{n+m}$$

Algorithmen: Division

D: Division mit Rest nicht negativer ganzer Zahlen Basis b.
Eingabe: $(m+n)$ stellige Zahl, n stellige Zahl.
Ausgabe: $(m+1)$ stelliger Quotient, n stelliger Rest.
Reduktion auf: Division mit Rest einer $(n+1)$ stelligen Zahl u durch n-stellige Zahl v, mit $0 \leq \left\lfloor \frac{u}{v} \right\rfloor < b$.

Rest r ist jeweils kleiner als v, d. h. $rb +$ (nächste Stelle des Dividenden) als „neues“ u,
z. B.

$$\begin{array}{l} \underline{3142} : \underline{47} = \underline{66} \text{ Rest } 40 \\ \underline{282} \\ \quad 322 \\ \quad \underline{282} \\ \quad\quad 40 \end{array}$$

Algorithmen: Division

Problem
Eingabe: $u = (u_0 u_1 \cdots u_n)_b \quad v = (v_1 \cdots v_n)_b$ mit $\left\lfloor \frac{u}{v} \right\rfloor < b$ (einstellig).
Bestimme: $q = \left\lfloor \frac{u}{v} \right\rfloor$ mit $u = qv + r$, wobei $0 \leq r < v$.

Schätzung für q: $\hat{q} = \min\left(\left\lfloor \frac{u_0 b + u_1}{v_1} \right\rfloor, b - 1 \right)$ erste Stelle für q.

2.12 Lemma (Übung): Es gilt
1) $\hat{q} \geq q$
2) Für $v_1 \geq \left\lfloor \frac{b}{2} \right\rfloor$ gilt $\hat{q} - 2 \leq q \leq \hat{q}$

D: Division mit Rest nicht negativer ganzer Zahlen Basis t.

Eingabe: $u = (u_1 \cdots u_{m+n})_b \quad v = (v_1 \cdots v_n)_b$, $v_1 \neq 0, n > 1$
Ausgabe: Quotient $\left\lfloor \frac{u}{v} \right\rfloor = (q_0 \cdots q_m)_b$, Rest $u \bmod v = (r_1 \cdots r_n)_b$

Algorithmen: Division

begin

$d := \left\lfloor \frac{b}{(v_1+1)} \right\rfloor$; $\{d \in \{\lfloor b/2 \rfloor, \ldots 1\}\}$

$(u_0 \cdots u_{m+n})_b := (u_1 \cdots u_{m+n}) \cdot d; (v_1 \cdots v_n)_b := (v_1 \cdots v_n) \cdot d; \{\textit{Normierung}\}$

for j from 0 **to** m **do**

begin

if $u_j = v_1$ **then**

$\hat{q} := b - 1$

else

$\hat{q} := \left\lfloor \frac{u_j b + u_j + 1}{v_1} \right\rfloor$

while $v_2 \hat{q} > (u_j b + u_{j+1} - \hat{q} v_1) b + u_{j+2}$ **do**

$\hat{q} := \hat{q} - 1;$

if $(u_j \cdots u_{j+n})_b < \hat{q} \cdot (v_1 \cdots v_n)_b$ **then**

$\hat{q} := \hat{q} - 1;$

$(u_j \cdots u_{j+n})_b := (u_j \cdots u_{j+m})_b - \hat{q} \cdot (v_1 \cdots v_n)_b; q_j := \hat{q};$

end

$(r_1 \cdots r_n)_b := (u_{m+1} \cdots u_{m+m})_b / d;$

end.

Korrektheit! Aufwand $O(m \cdot n)$ go.

Algorithmen: Exponentation

E: Exponentation:: **Eingabe**: x Basis b, $n \in \mathbb{N}$. **Ausgabe**: x^n
Naive Lösung: n-Multiplikationen.
Durch Quadrieren: $\log n$ Multiplikationen, d. h. $x^2, x^4, x^8, \ldots$
Länge der Zahlen: $\lambda(x) = h \rightsquigarrow \lambda(x^n) = n \cdot h$

begin
$y := x; z := 1;$ $\{$*Ergebnis in* $z, y \rightsquigarrow x, x^2, x^4, \ldots$$\}$
while $n > 1$ **do**
begin
$m := \left\lfloor \frac{n}{2} \right\rfloor;$
if $n > 2m$ **then**
$z := zy;$
$y := yy; n := m;$
end
$z := zy;$
end.

Algorithmen: Exponentation Beispiel

x^{13}

n	13	13	6	3
m		6	3	1
y	x	x^2	x^4	x^8
z	1	x	x	x^5	x^{13}

Grundlage: Ist $n = \sum_{i=0}^{k} e_i 2^i \quad e_i \in \{0,1\}$, so

$$x^n = x^{\sum_{i=0}^{k} e_i 2^i} = \prod_{i=0}^{k} x^{e_i \cdot 2^i} = \prod_{i:e_i \neq 0} x^{2^i}$$

Anzahl der Multiplikationen:
$N = k + e_0 + e_1 + \cdots + e_k - 1 \leq 2k = 2 \log n$

Problem:
Naiver Algorithmus $x^n \quad \lambda(x)$ fest $x^i \cdot x$ kostet $c \cdot i \cdot \lambda(x)^2$
Hingegen $y \cdot y$ kostet $c \cdot \lambda(y) \cdot \lambda(y)$. D.h. es kommen größere Zahlen vor!

Algorithmen: Exponentation Analyse

$$c_{\exp}(n) \approx c \cdot \lambda(x)^2 \sum_{i=0}^{k-1} 2^{2^{i+1}} + c \cdot \lambda(x)^2 \sum_{i=1}^{k} e_i \left(\sum_{j=0}^{i-1} e_j 2^j \right) 2^i$$

$$c_{\text{naiv}}(n) \approx \tfrac{1}{2} c \cdot n^2 \cdot \lambda(x)^2 = c \cdot \lambda(x) \cdot \sum_{i=1}^{n-2} i \cdot \lambda(x)$$

d. h. $n = 2^k$

$$c_{\exp}(n) \cong \frac{4}{3} c \cdot n^2 \lambda(x)^2 \cong \frac{8}{3} c_{\text{naiv}}(n)$$

Für $n = 2^k + 2^{k-1}$
$c_{\exp}(n) \cong \frac{4}{3} c \cdot 2^{2k} \lambda(x)^2 + c \cdot 2^{2k-1} \lambda(x)^2 \simeq \frac{11}{6} c \cdot 2^k \lambda(x)^2$
$c_{\text{naiv}}(n) = \frac{9}{4} c \cdot 2^{2k} \lambda(x)^2 \simeq \frac{27}{12} c_{\exp}(n)$
Falls $x \in R$, R endlich, so können die Kosten der Multiplikation als konstant gesehen werden und exp ist erheblich schneller als naiv.
Anwendungen: Cryptographie: Kodierung und Decodierung
RSA-Methode: $y = x^n \bmod a$, $n > 10^{50}$,
Rekurrenzgleichungen, Potenzreihenentwicklungen.

GGT Kosten für $\mathbb{Z}$: $v(a) = |a|$

$a = r_0 \geq b = r_1 > r_2 > \cdots > r_l \geq 0 \qquad q_i \geq 0$ alle i
Darstellung der Zahlen z. B. 2^{64}-Standard Darstellung
Länge $\lambda(a) = \left\lfloor \frac{\log_2 |a|}{64} \right\rfloor + 1$
Verwendet man $l \leq v(b) + 1 = b + 1 \leq 2^{64\lambda(b)} \rightsquigarrow$ exp in $\lambda(b)$.
Polynomiale Schranke für l : $1 \leq i \leq l$
$r_{i-1} = q_i r_i + r_{i+1} \geq r_i + r_{i+1} > 2r_{i+1}$, d. h.
$\prod_{2 \leq i < l} r_{i-1} > 2^{l-2} \prod_{2 \leq i < l} r_{i+1}$ für $l \geq 2 \quad r_{l-1} \geq 2$ folgt
$2^{l-2} < \frac{r_1 \cdot r_2}{r_{l-1} r_l} < \frac{r_1^2}{2}$ oder $l \leq \lfloor 2 \log r_1 \rfloor + 1 \approx 128\lambda(b)$

2.13 Satz Lamé 1845
Sei $n \in \mathbb{N}^+$ und u kleinste positive Zahl, für die der EA für Eingabe u, v' n Iterationen benötigt für mindestens eine Zahl v' mit $v' \leq u$. Dann gilt $u = F_{n+1}$ und $v' = F_n$, wobei F_k k-te Fibonacci Zahl.

GGT Kosten für $\mathbb{Z}$: $v(a) = |a|$

Alle Quotienten gleich 1, z. B. $(a, b) = (13, 8)$ EA

$$
\begin{aligned}
13 &= 1 \cdot 8 + 5 \\
8 &= 1 \cdot 5 + 3 \\
5 &= 1 \cdot 3 + 2 \\
3 &= 1 \cdot 2 + 1 \\
2 &= 2 \cdot 1
\end{aligned}
$$

l für $(a, b) = (F_{n+1}, F_n)$
$\rightsquigarrow l = n - 1 \approx 1.44 \log F_n + O(1)$
für b fest und a Var gilt
im Mittel $l \approx 0.584 \log b$

Beachte: Dirichlet / Lejeune 1849 Cesaro 1881

Für zufällig gewählte Zahlen a, b gilt
$PR(\mathrm{GGT}(a, b) = 1) = \frac{6}{\pi^2} \approx 0.6079$
Verwende: $PR(p \nmid n \wedge p \nmid m) = 1 - \frac{1}{p^2}$
$\prod_p (1 - \frac{1}{p^2}) \approx \frac{6}{\pi^2}$

Aufwand für EEA über $\mathbb{Z}$

Sei $n = \lambda(a), m = \lambda(b) \rightsquigarrow O(nm)$ für EA
(Kosten der Div mit Rest $a = qb + r \quad O((\lambda(a) - \lambda(b)) \cdot \lambda(b))$ go)
Für die Bezout Koeffizienten gilt analog
$|s_i| \leq \frac{b}{r_{i-1}}$ und $|t_i| \leq \frac{a}{r_{i-1}} \qquad 1 \leq i \leq l+1$

2.14 Satz Der EEA für Zahlen $a, b \in \mathbb{N}$ $\lambda(a) = n \geq \lambda(b) = m$, kann mit $O(nm)$ go durchgeführt werden.

Weitere Ergebnisse und Bemerkungen siehe von zu Gathen, Gerhard bzw. Mignotte. Siehe auch Knuth Kap. 4.5.3, Bach/Shallit 4.2, 4.3.
Viele Varianten zur Berechnung vom GGT (z. B. ohne Division).
KGV Kleinste gemeinsamer Vielfache (LCM)

$$\mathrm{KGV}(a, b) = \frac{|ab|}{\mathrm{GGT}(a, b)}$$

Reduktion auf GGT-Berechnung.

Ringkonstruktionen: $R[x]$ Polynomring

R ZPE, so $R[x]$ ZPE-Ring. R euklidisch $\not\Rightarrow$ $R[x]$ euklidisch
z. B. $\mathbb{Z}[x]$ nicht euklidisch, da kein Hauptidealring (z. B. $\langle 2, x\rangle$ wird nicht von $a(x) \in \mathbb{Z}[x]$ erzeugt oder $\mathbb{Q}[x, y]$ nicht euklidisch, da kein Hauptidealring (z. B. $\langle x, y\rangle$).
Vorteile E-Ringe: Euklidischer Algorithmus für GGT Berechnung.
Anwendungen: Lösung diophantischer Gleichungen in $F[x]$: $a(x), b(x), c(y)$ gesucht $\sigma(x)$ und $\tau(x)$ mit

$$\sigma(x)a(x) + \tau(x)b(x) = c(x)$$

Lösbar für $g(x) = \mathrm{GGT}(a(x), b(x))|c(x)$. Eindeutigkeit und Schranken für die Grade von $\sigma(x), \tau(x)$ (Übung).
Zerlegung rationaler Funktionen:

$$\frac{c(x)}{a(x)b(x)} = \frac{\tau(x)}{a(x)} + \frac{\sigma(x)}{b(x)} \qquad \rightsquigarrow \text{Integration}$$

Problem: wie berechnet man GGT in $\mathbb{Z}[x]$ oder $\mathbb{Q}[x, y]$.
$\rightsquigarrow$ Pseudodivision primitiver EA.

Quotienten-Körper von Integritätsbereichen

Übergang von $\mathbb{Z} \rightsquigarrow \mathbb{Q}$. D: Integritätsbereich $\rightsquigarrow$ Körper.
Setze:: $S = \{a/b : a \in D, b \in D - \{0\}\}$ formale Quotienten.
$\sim$ auf S: $a/b \sim c/d$ gdw $ad = bc$ ist Äquivalenzrelation auf S $[a/b]$
$S/\sim = \{[a/b] : a \in D, b \in D - \{0\}\}$, $a/b \in [a/b]$ Repräsentant.
Addition + Multiplikation auf $S/\sim$:
$(a/b) + (c/d) = (ad + bc)/bd$
$(a/b) \cdot (c/d) = ac/bd$ wohldefiniert auf Äquivalenzklassen.
$S/\sim$ ist Körper:: $Q(D)$ (F_D):: Quotientenkörper von D.
Kleinster Körper, der D enthält, $D \cong \{[a/1] : a \in D\}$
$0/1$ $1/1$ $a/1$ mit a identifiziert.
Praxis: eindeutige Repräsentanten für $[a/b]$, Entscheidung für $\sim$.
Falls GGT in D existiert:
$a/b \in [a/b] \in S$ ist Repräsentant, falls $\mathrm{GGT}(a, b) = 1$, b ist einheitsnormal, a, b in „Normalform“.
z. B. $\mathbb{Z}$ Quotientenkörper $Q(\mathbb{Z}) = \mathbb{Q}$ a/b „kanonisch“, $(b > 0)$.
$-2/4, 2/-4, 100/-200, -600/1200$ Kan. repräsentant: $-1/2$.

Quotienten-Körper rationaler Funktionen

$D[x]$ mit D ZPE-Ring, $Q(D[x])$ Körper der rationalen Funktionen (Ausdrücke) in x :: Schreibe $D(x)$.

Beachte: Operationen $+, \cdot$ sind „teuer“.
Addition: 3-Multiplikationen + Addition + GGT Berechnung
Multiplikation: 2 Multiplikationen und GGT Berechnung.

Wähle geeignete Darstellungen
Fall $\mathbb{Z}[x]$ $\mathbb{Q}[x]$ bzw. $\mathbb{Z}(x)$ $\mathbb{Q}(x)$

in $\mathbb{Q}(x)$: $a(x)/b(x) = \left(\frac{17}{100}x^2 - \frac{3}{113}x + \frac{1}{2}\right) / \left(\frac{5}{9}x^2 + \frac{4}{5}\right)$

Die Äquivalenzlasse enthält Repräsentanten mit ganzzahligen Koeffizienten: z. B.
$a(x)/b(x) = (4284x^2 - 675x + 12600)/(14000x^2 + 20160) \in \mathbb{Z}(x)$.

D mit Quotienten-Körper F_D dann $D(x) \cong F_D(x)$.

Beachte: unterschiedliche kanonische Repräsentanten möglich.
Siehe Beispiel oben.

Potenzreihen - erweiterte Potenzreihen

$R[[x]]$ Potenzreihen mit Koeffizienten in R: Ausdrücke

$$a(x) = \sum_{k=0}^{\infty} a_k x^k \qquad a_k \in R$$

$\mathrm{ord}(a(x)) = \min\{k : a_k \neq 0\}$.
0 alle $a_k = 0$, $a_k = 0$ für $k \geq 1$ Konstante PR.
Addition + Multiplikation wie üblich!
$d(x) = a(x) \cdot b(x) = \sum_{k=0}^{\infty} d_k x^k$ mit $d_k = a_0 b_k + \cdots + a_k b_0 \quad k \geq 0$

Eigenschaften:

1. $R[x] \hookrightarrow R[[x]]$
2. R kommutativ, so auch $R[[x]]$ $0, 1$
3. R Intbereich, so auch $D[[x]]$. Einheiten sind PR mit a_0 Einheit in R.
4. F Körper, so ist $F[[x]]$ euklidischer Ring mit Bewertung $v(a(x)) = \mathrm{ord}(a(x))$.

Potenzreihen - Einheiten

$a(x) = \sum a_k x^k \qquad b(x) = \sum b_k x^k \quad a(x) \cdot b(x) = 1$ so

$$\begin{aligned} 1 &= a_0 b_0 \\ 0 &= a_0 b_1 + a_1 b_0 \\ &\vdots \\ 0 &= a_0 b_n + a_1 b_{n-1} + \cdots + a_n b_0 \end{aligned}$$

$\rightsquigarrow a_0$ ist Einheit

Ist a_0 Einheit in R, so wird b bestimmt durch

$b_0 = a_0^{-1},\ b_1 = -a_0^{-1}(a_1 b_0), \cdots,\ b_n = -a_0^{-1}(a_1 b_{n-1} + \cdots + a_n b_0)$

In $\mathbb{Z}[[x]]$ gilt $\qquad (1-x)^{-1} = 1 + x + x^2 + x^3 + \cdots$

Beachte

$\operatorname{ord}(a(x) + b(x)) \geq \min\{\operatorname{ord}(a(x)), \operatorname{ord}(b(x))\}$

$\operatorname{ord}(a(x) \cdot b(x)) = \operatorname{ord}(a(x)) + \operatorname{ord}(b(x))$.

Für $a(x), b(x) \in F[[x]], a(x) \neq 0 \neq b(x)$, so $a(x) \mid b(x)$ oder $b(x) \mid a(x)$.

Sei $\operatorname{ord}(a(x)) = l \qquad \operatorname{ord}(b(x)) = m$, d. h.

$a(x) = x^l \bar{a}(x) \quad b(x) = x^m \bar{b}(x) \quad \bar{a}(x), \bar{b}(x)$ Einheiten.

$l \geq m$, so $a(x)/b(x) = x^{l-m} \bar{a}(x) \cdot \bar{b}(x)^{-1} \in F[[x]]$.

Potenzreihen - Einheiten, GCD in $F[[x]]$

Für $a(x), b(x) \in F[[x]]$, $b(x) \neq 0$ gibt es $q(x), r(x)$ mit
$a(x) = b(x) \cdot q(x) + r(x)$ mit
$r(x) = 0$ falls $\mathrm{ord}(a(x)) \geq \mathrm{ord}(b(x))$, $r(x) = a(x)$ falls
$\mathrm{ord}(a(x)) < \mathrm{ord}(b(x))$.

Quotientenkörper: $Q(D[[x]])$ Schreibe $D((x))$.

Achtung: D ZPE Ring $\not\hookrightarrow$ $D[[x]]$ ZPE Ring, d. h. Normalformen schwer, assoziierte Elemente!

$a(x) = 2 + 2x + 2x^2 + 3x^3 + 4x^4 + \cdots$
$b(x) = 2 + 4x + 6x^2 + 9x^3 + 13x^4 + \cdots$
$c(x) = 2 + x^3 + x^4 + x^5 + x^6 + \cdots$

sind assoziiert

$b(x) = a(x)(1 + x + x^2 + x^3 + x^4 + \cdots$
$c(x) = a(x)(1 - x)$

Welche PR soll als einheitsnormal gewählt werden! In $F[[x]]$ geht dies:
$a(x) = x^l \cdot b(x)$, $l = \mathrm{ord}(a(x))$ $b(x) = a_l + a_{l+1}x + \cdots$ $a_l \neq 0$ also $b(x)$ Einheit. Die Monome $x^l (l \geq 0)$ und 0 sind einheitsnormal.

$$\mathrm{GCD}(a(x), b(x)) = x^{\min\{\mathrm{ord}(a(x)), \mathrm{ord}(b(x))\}}$$

Erweiterte Potenzreihen

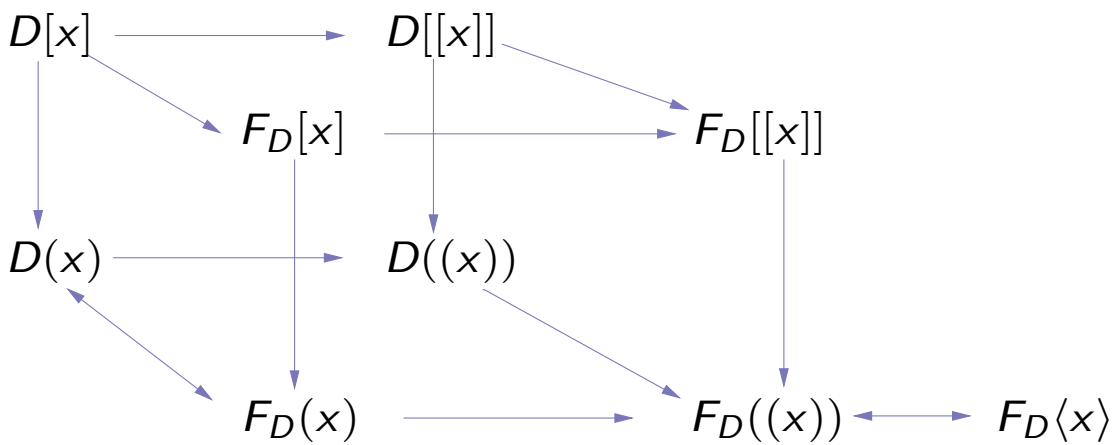
In $F((x))$ $\quad \left(\sum_{k=0}^{\infty} a_k x^k\right) / x^n \qquad n \geq 0$

$F\langle x\rangle$: $a(x) = \sum_{k=m}^{\infty} a_k x^k \quad a_k \in F, k \geq m, m \in \mathbb{Z}$

$\mathrm{ord}(a(x)) = \min\{k : a_k \neq 0\}$ $(< 0!)$

$F\langle x\rangle$ ist Körper.

Zusammenhang:



Standard Ringkonstruktionen

- $i \leq R, i$ ideal, so R/i Ring: Quotientenring
  Idealkongruenz: $x \equiv_i y \quad (x \equiv y \mod i)$ gdw. $x - y \in i$.
- R_1, R_2 Ringe, $R_1 + R_2 = \{(r_1, r_2) : r_1 \in R_1, r_2 \in R_2\}$ mit
  $(r_1, r_2) + (r_1', r_2') = (r_1 + r_1', r_2 + r_2')$
  $(r_1, r_2) \cdot (r_1', r_2') = (r_1 r_1', r_2 r_2')$.

  $(0_{R_1}, 0_{R_2})$ Nullelement, $(1_{R_1}, 1_{R_2})$ Einselement.

  Produkt
- Ist R bzw. sind R_1, R_2 effektiv, so stellt sich die Frage ob der Quotientenring bzw. das Produkt effektiv sind.

Inhalt Kapitel 3

Normalformen - Algebraische Darstellungen

Algebraische Strukturen $\leftrightarrow$ Datenstrukturen (Typen) $\simeq$ „Klassen“
Menge und Operationen:: $(\mathbb{Z}; 0, 1, +, -, \cdot, /, \mathrm{mod}, \mathrm{ggt}, \mathrm{kgv}, \exp, \dots)$

- Darstellung der Objekte:
  oft Konstruktionsvorschriften „Konstruktoren“ für Definitionsbereich
- Termalgebra: Terme in Konstanten und Operatoren.
  Grundterme stellen Elemente des Definitionsbereich dar. Terme sind gleich, wenn sie das gleiche Element des Definitionsbereichs darstellen.
  z. B. $2^3 = (2 \cdot 2) + 4 = \mathrm{ggt}(24, 16)$.

Probleme:

- Welche Darstellungen sind erlaubt (Operatoren z. B. für Ringe: $0, 1, +, \cdot$)
- Transformation von Darstellungen
- Eindeutige oder mehrdeutige Darstellungen
- Gleichheit von Darstellungen

Verschiedene Darstellungsebenen

Elemente der algebraischen Struktur, Darstellungen, Rechnerdarstellung.

Objektebene, Formebene, Darstellungsebene

3.1 Beispiel Funktionenringe, Differentiation als Operator
$\frac{\partial}{\partial x}(ax + xe^{x^2})$ Regeln (Axiome-Gleichungen)

$$\frac{\partial c}{\partial x} \to 0 \qquad \frac{\partial x}{\partial x} \to 1 \qquad \frac{\partial(u+v)}{\partial x} \to \frac{\partial u}{\partial x} + \frac{\partial v}{\partial x}$$

$$\frac{\partial(uv)}{\partial x} \to u\frac{\partial v}{\partial x} + \frac{\partial u}{\partial x}v$$

$$\frac{\partial(u^v)}{\partial x} \to vu^v\frac{\partial u}{\partial x} + (\log_e u)u^v\frac{\partial v}{\partial x}$$

$\rightsquigarrow$ Simplifikation symbolischer Ausdrücke $\rightsquigarrow$ Reduktionsmethoden.

Einfache Simplifikationsregeln in CA-Systemen

- Unterdrücken von Klammern: Präfix-Postfix Notationen:: Formebene
- Identitäten Vereinfachung: z. B. $0 \cdot u \to 0$, $1 \cdot u \to u$, $u/1 \to u$, $v^0 \to 1$ $(v \neq 0)$, $0^w \to 0$ $(w > 0)$
- Vorzeichenregeln: z. B. $(-u)(-v^3) = uv^3$, $-(u+v) \to -u - v$?
- Numerische Vereinfachungen: $\frac{5}{8} - \frac{1}{8} \to \frac{1}{2}$, $9! \to 362880$

  Vorsicht! oft nicht einfach: $(33282)\frac{1}{2} \sin\left(\frac{13\pi}{6}\right) \to \frac{122}{\sqrt{2}}$, $e^e, e, \pi, \ldots$
- Assoziativ-kommutative Gesetze

  $(uv)w + (p+q) \to uvw + p + q \qquad q + p \to p + q$?
- Anordnung: z. B. Polynomdarstellung

Einfache Simplifikationsregeln in CA-Systemen

- Zusammenfassung gemeinsamer Faktoren
  $u + \left(\frac{2}{3}\right) u \longrightarrow \frac{5}{3}u$, $2^{x+2} \longrightarrow 4 \cdot 2^x$, $e^{5+\log u} \longrightarrow e^5 e^{\log u}$
- Operationen mit Exponenten: $(u^w)^v \longrightarrow u^{wv}$, $(uv)^w \longrightarrow u^w v^w$
- Distributiv Gesetze: $(u+v)w \longrightarrow uw + vw$
- Potenzen erweitern: $(a+b)^2 \longrightarrow a^2 + 2ab + b^2$, $(1+x)^{100} \longrightarrow ?$
- GGT-Vereinfachungen: $\frac{4u^2+12u^3+12u^2+4u}{2u^4-2u^3-2u^2+2u} \longrightarrow \frac{2u+2}{u-1}$

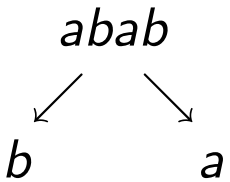
Wortproblem - Simplifikation

$(M, \sim)$ WP:: Gegeben $u, v \in M$, Frage: $u \sim v$?

Wie ist M gegeben: oft endlich erzeugt, z. B. Termalgebra.
Wie ist $\sim$ gegeben: oft Axiome (Gleichungen)

3.2 Beispiel Monoide, Gruppen: Erzeugende, Definierende Relationen

$M = (\{a, b\}; aba = \lambda, bab = \lambda)$
$G = (a, b, \bar{a}, \bar{b}; a\bar{a} = \bar{a}a = b\bar{b} = \bar{b}b = \lambda)$ freie Gruppe.
Frage: gilt $a =_M b$ $\quad a\bar{b}b\bar{a} =_G \lambda$?



Wortersetzungssysteme:: $M \cong (a; a^3 = \lambda)$,
allg: Termersetzungssysteme

Simplifikation: Terme in „einfachste“ Form zu bringen.
Methode: Maß: wohlfundierte (Partial)-Ordnung $\succ$ auf M.
$\mathrm{rep}(u) = \min_{v \sim u} v$ sollte eindeutig sein.
Frage: Ist rep effektiv berechenbar? i. Allg. nicht, da WP damit lösbar.

Wortproblem - Simplifikation

Termersetzungssysteme: Methoden zur Behandlung von WP:
Regeln, Konfluenz, Terminierung, Vervollständigung (KB).

Oft genügt es ein spezielles Wortproblem zu betrachten:

Rolle der Konstanten z. B. $0, 1$.

Gruppen: $u = v \text{ gdw } uv^{-1} = 1$

Ringe: $u = v \text{ gdw } u - v = 0$

$\rightsquigarrow$ Eigenschaften einer speziellen Äquivalenzklasse.

Wortproblem - 0-Äquivalenz

Richardson: Some unsolvable problems involving elementary funtions of a real var. J. Symb. Logic 33 (1968). How to recognize zero. J. Symb. Comp. 24 (1997). Funktionsklassen: $\{f : f : \mathbb{R} \to \mathbb{R}\}$, $+, \cdot, 0, 1$.

3.3 Satz Sei R die Klasse aller Terme, die man aus

1. $\mathbb{Q}$ (rationale Zahlen), π Konstanten.
2. Einer Variablen x und Funktionen $\sin(x)$, $|x|$.
3. Operatoren: add$(+)$, mult$(*)$, komp$(\circ)$.

Wie üblich definiert: $t_1(x) \circ t_2(x) = t_1(t_2(x))$

Interpretiert man die Konstanten als konstante Funktionen auf $\mathbb{R}$, x als Identitätsfunktion und $\sin(x)$, $|x|$ durch die Standardfunktionen, so stellt jeder Term eine eindeutige Funktion aus $(\mathbb{R} \to \mathbb{R})$ dar.
Folgendes Problem ist unentscheidbar:
Eingabe: $E \in R$ **Frage**: Gilt $E = 0$?

Wortproblem - 0-Äquivalenz

3.4 Beispiel

$\frac{1}{2} \qquad \sin\left(\frac{1}{2}\right) = \sin(x) \circ \frac{1}{2} \qquad \sin(|x|)$

$|\sin(x)|, \quad \sin(\sin(\sin(x))), \quad \sin(x)^2, \quad \sin(x) + |x|$

Beweisidee: Reduziere Hilbert's 10. Problem auf „$E = 0$".
Matiasevic 1970: Unentscheidbarkeit.

Hilbert's 10. Problem

Es gibt eine Menge $\mathcal{P} = \{p(x_1, \ldots, x_n) \mid p \text{ Polynome über } \mathbb{Z}\}$ mit Prädikat: $p \in \mathcal{P}$ hat p Nullstelle in $\mathbb{N}^n$ ist nicht rekursiv entscheidbar.

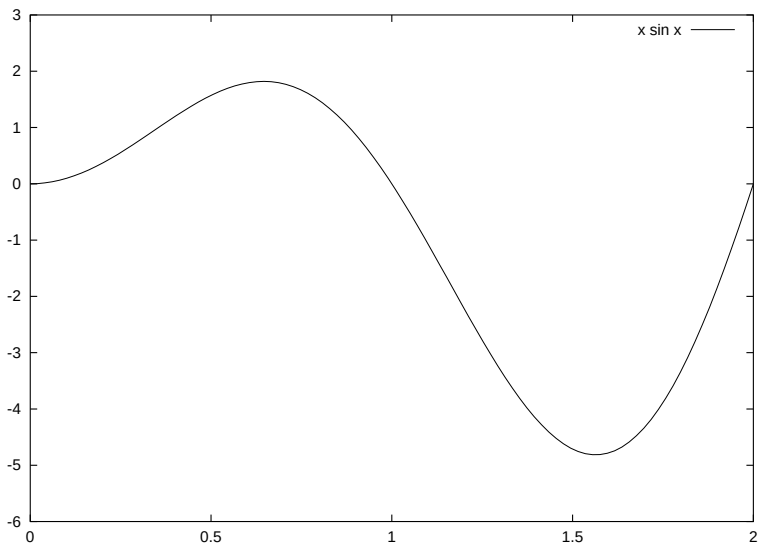
Idee der Reduktion: Sei $F : \mathbb{R} \to \mathbb{R}$, so

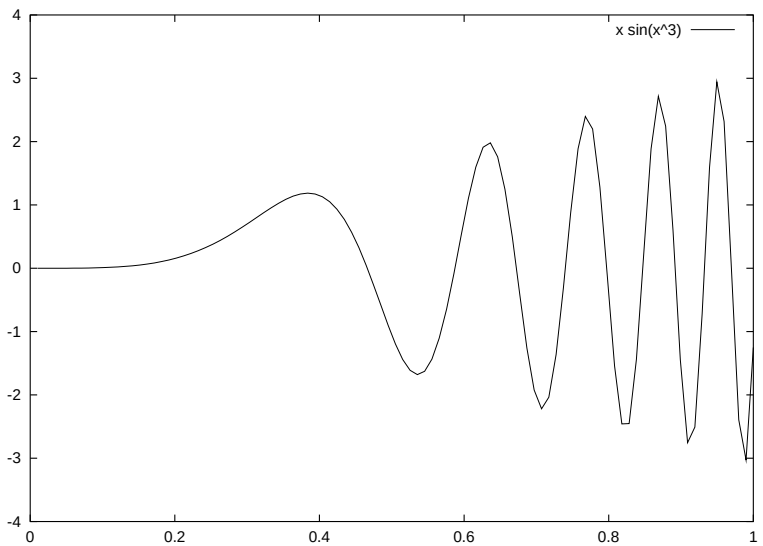
$f^0(x) = x \qquad f^{(n+1)}(x) = f(f^n(x)) \quad n > 0$

Wortproblem - 0-Äquivalenz

3.5 Lemma Jedes Tupel reeller Zahlen kann durch eine reelle Zahl dargestellt werden (bis auf ε). Seien $h(x) = x \sin x, g(x) = x \sin(x^3)$. Dann gibt es für alle $a_1, a_2, \ldots, a_n \in \mathbb{R}, 0 < \varepsilon < 1$ ein $b \in \mathbb{R}$ mit $|h(g^{(k+1)}(b)) - a_k| < \varepsilon \quad 1 \leq k \leq n$.

$h(x) = x \sin x \qquad g(x) \qquad = \qquad x \sin(x^3)$





Wortproblem - 0-Äquivalenz

3.6 Definition Eine Funktion (Term) $F(x_1, \ldots, x_n) \in R$ wird von $G(x_1, \ldots, x_n) \in R$ dominiert, wenn für alle $x_i \in \mathbb{R}$

i) $G(x_1, \ldots, x_n) > 1$

ii) $\forall \delta_1, \ldots, \delta_n \in \mathbb{R}, |\delta_i| < 1 : F(x_1 + \delta_1, \ldots, x_n + \delta_n) < G(x_1, \ldots, x_n)$

3.7 Lemma Zu jeder Funktion $F \in R$ gibt es eine Funktion $G \in \mathcal{P}$, die F dominiert.

3.8 Lemma Für alle $P \in \mathcal{P}$ existiert $F \in R$ mit: Es gibt $(a_1, \ldots, a_n) \in \mathbb{N}^n$ mit $P(a_1, \ldots, a_n) = 0$ gdw es gibt $(b_1, \ldots, b_n) \in \mathbb{R}^n_{\geq 0}$ mit $F(b_1, \ldots, b_n) < 0$.

Wortproblem - 0-Äquivalenz: Positive Ergebnisse

3.9 Satz Richardson
Betrachte Funktionenklasse, die durch Termmenge Λ definiert wird mit

1. $\mathbb{Q}, \pi \in \Lambda$
2. Var x Identität
3. $F, G \in \Lambda$, so $(F + G), (F * G), (F/G)$
4. $F \in \Lambda$, so $\log(|F|)$ und $\exp(F)$ in Λ

Interpretiere $F \in \Lambda$ als $F : \mathbb{R} \to \mathbb{R}$.

Das Prädikat „$F(x) \equiv 0$“ auf Λ ist entscheidbar.

Wortproblem - 0-Äquivalenz: Positive Ergebnisse

Beweisidee: Komplexität von Ausdrücken + Induktion. Sei etwa y Unterausdruck mit größter Komplexität etwa $y = \log u$

$F = a_n y^n + a_{n-1} y^{n-1} + \cdots + a_1 y + a_0$

wende Verfahren an:

$a_n = 0 \rightsquigarrow F_1 = a_{n-1} y^{n-1} + \cdots + a_0$, $F \equiv 0$ gdw $F_1 \equiv 0$.

$a_n \neq 0 \rightsquigarrow F_2 := \frac{F}{a_n} = y^n + \frac{a_{n-1}}{a_n} y + \ldots$

$F_3 = F_2' = n y^{n-1} y' + \cdots + \frac{a_n a_0' - a_0 a_n'}{a_n^2}$

$F_2 \equiv 0 \rightsquigarrow F_3 \equiv 0$ $\quad$ $F_3 \equiv 0 \rightsquigarrow F_2$ konstant.

Klasse ist abgeschlossen gegen Ableitungen und die Ableitungen sind weniger komplex.

$y = \log u \rightsquigarrow y' = \frac{u'}{u}$ $\quad$ u', u weniger komplex.

$y = e^u$ ist dies nicht der Fall, unterscheide hier

$$F \equiv a_n y^n + \cdots + a_0 \begin{cases} a_0 \equiv 0 \rightsquigarrow F_1 = a_n y^n + \ldots + a_1 y = Qy \\ a_0 \not\equiv 0 \rightsquigarrow F_2 = F / a_0 \rightsquigarrow F_2' / y \equiv 0 \rightsquigarrow F \equiv c \end{cases}$$

Formalisierung des Simplifikationsbegriffs

Zwei Ziele:

1. „Einfachere“ äquivalente Objekte zu definieren und sie bei Operationen zu verwenden.
2. Wenn möglich kanonische (d. h. eindeutige) Darstellung in (einigen/allen) Äquivalenzklassen festzulegen und wenn möglich effektiv zu bestimmen.

3.10 Definition Sei E Menge syntaktischer Objekte (z. B. Terme über Signatur, Formeln, Wörter, Programme) und sei $\sim$ eine Äquivalenzrelation auf E. Sei weiterhin $\preceq$ eine Partialordnung auf E. Eine Simplifikationsfunktion für $[E;\sim]$ bzgl. $\preceq$ ist eine rekursive Funktion $f : E \to E$ mit

$$\text{i) } f(t) \sim t \qquad \text{ii) } f(t) \preceq t$$

i. Allg. $\preceq$ wohlfundierte Partialordnung auf E, d. h. es gibt keine ∞-Ketten $e_1 \succ e_2 \succ e_3 \succ \cdots$,
z. B. $|e|$ Länge des Ausdrucks $e_1 \succ e_2$ gdw $|e_1| \succ |e_2|$.

Formalisierung des Simplifikationsbegriffs

Eine Normalisierungsfunktion bzgl. $\preceq$ ist eine Simplifikationsfunktion f bzgl. $\preceq$ mit $f(f(t)) = f(t)$ für alle t.
D. h. $f(t)$ ist simplifiziert oder in Normalform.
Oft wird verlangt, dass für bestimmte Äquivalenzklassen z. B.
$[0], [1] : t \sim 0$ so $f(t) = f(0) = 0$.
Eindeutige Normalformen für spezielle Äquivalenzklassen in der Regel solche, die ausgewählte Konstanten der Signatur enthalten.

Eine kanonische Funktion ist eine Simplifikationsfunktion f mit

$$s \sim t \text{ so } f(s) = f(t) \text{ für alle } s, t \in E$$

Sie berechnet eindeutige (kanonische) Repräsentanten für jede Äquivalenzklasse.

Formalisierung des Simplifikationsbegriffs

Beachte: Ist f kanonisch, so ist f auch Normalisierungsfunktion und

$$s \sim t \text{ gdw } f(s) = f(t)$$

d. h. das Wortproblem für $\sim$ ist entscheidbar.
f ist idempotent (d. h. $f \circ f = f$) und in jeder Äquivalenzklasse gibt es genau ein Element in kanonischer Form.

3.11 Satz Sei E eine entscheidbare Menge syntaktischer Objekte und $\sim$ eine Äquivalenzrelation auf E. Dann gilt
$\sim$ entscheidbar (WP-entscheidbar) gdw es gibt eine kanonische Funktion f für $[E; \sim]$.

Berechenbare Quotientenstrukturen

3.12 Satz Sei E entscheidbar, R berechenbare Operation auf E, d. h. $R : E^n \to E$ und $\sim$ eine Kongruenz bzgl. R.
Hat E eine kanonische Funktion f bzgl. $\sim$ und ist $\mathrm{rep}(E) = \{t \in E : f(t) = t\}$ die Menge der kanonischen Repräsentanten, so lässt sich die Quotientenstruktur wie folgt darstellen:

$$R'(s_1, \ldots, s_n) := f(R(s_1, \ldots, s_n)) \text{ für } s_1, \ldots, s_n \in \mathrm{rep}(E)$$

und

$$(\mathrm{rep}(E), R') \cong (E/\sim, R/\sim)$$

$\mathrm{rep}(E)$ ist entscheidbar, R' ist berechenbar.

Beispiele: Monoide und Gruppen

3.13 Beispiel

- $(a, b : ba = \lambda)$ Normalformen $a^n b^m \quad n, m \geq 0$

  Wortersetzungssystem: $ba \longrightarrow \lambda$ terminierend (Längenkürz.)

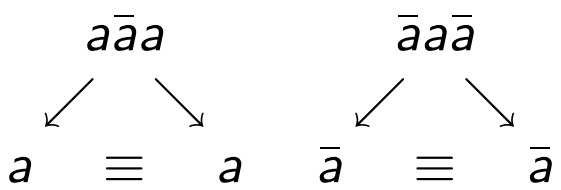
  konfluent (d. h. eindeutige NF)

$$(a^n b^m) \circ (a^{n'} b^{m'}) = \begin{cases} a^{n+(n'-m)} b^{m'} & n' \geq m \\ a^n b^{(m-n')+m'} & m > n' \end{cases}$$

- $(a, b, \bar{a}, \bar{b} : a\bar{a} = \bar{a}a = b\bar{b} = \bar{b}b = \lambda)$

  WES: $a\bar{a} \longrightarrow \lambda \qquad \bar{a}a \longrightarrow \lambda \qquad b\bar{b} \longrightarrow \lambda \qquad \bar{b}b \longrightarrow \lambda$ terminierend.

  Konfluent: Kritische Paare:



Beispiele: Monoide und Gruppen

- Normalformen: Wörter, die keine linke Seite als Teilwort enthalten $\rightsquigarrow$ reguläre Sprache
- $(a, b : aba = bab = \lambda)$

  WES: $aba \to \lambda \quad bab \to \lambda$ terminierend.

  Nicht konfluent $abab$ mit $b \not\equiv a$

  Hinzunahme von Regeln $b \to a \rightsquigarrow$ Knuth Bendix Vervollständigung.
  Länge-Lexikographische Ordnung $b \succ a$

  $(a, b; b \to a, a^3 \to \lambda)$, Repr: λ, a, a^2

  Multiplikation:

	λ	a	a^2
λ	λ	a	a^2
a	a	a^2	λ
a^2	a^2	λ	a

Abstraktionsebenen für algebraische Strukturen

- $\mathbb{Z}_m$ $\quad f(n) = n \bmod m$ positive Reste

  Repr. $0, 1, \ldots, m-1$, Definition von $+, \cdot$ auf $\mathbb{Z}_m$.

I) **Objektebene**: Menge $\quad$ Operationen = Elemente der Mengen

II) **Form-Ebene**

Objekte werden explizit dargestellt „Bezeichner“

mehrere Gleichheiten $\quad \equiv$ syntaktische $\quad =$ semantische

gleiche Bezeichner $\quad$ gleiche Objekte

Typische Bezeichner: Terme $12x^2y - 4xy + 9x - 3 \quad (3x-1)(4xy+3)$
$(12y)x^2 + (-4y+9)x - 3$

Syntaktisch verschieden, aber semantisch gleich.

Abstraktionsebenen für algebraische Strukturen

III) **Datenstrukturebene**

Darstellung der Objekte aus Ebenen I), II) im Rechner:
Speicherorganisation
Listen, Felder, Verbunde usw.
Simplifikation definiert auf Ebene II).
Realisiert in Ebene III).

Wichtige Entscheidungen: Welche Darstellungen erlaubt man in Ebene II), wie werden diese in III) dargestellt.

Oft Unterscheidung nötig: Eingabe, Intern, Ausgabe.

Beispiele

a) $E = \mathbb{Z}[x]$

Formebene

- Jedes Polynom $\sum_{i=0}^{n} a_i x^i \in \mathcal{F}$
- $p_1, p_2 \in \mathcal{F}$, so auch $(p_1 * p_2) \in \mathcal{F}$
- $p_1, p_2 \in \mathcal{F}$, so auch $(p_1 + p_2) \in \mathcal{F}$

Normalisierungsfunktionen:

f_2 {
  f_1 {
    Multipliziere Produkte aus (Distributivgesetz) Σ Monom.
    Fasse Monome mit gleichem Grad zusammen.
  }
  Ordne Monome nach aufsteigendem Grad
  (absteigendem)
}

Beispiele (2)

f_1 ist Normalisierungsfunktion, f_2 ist kanonische Funktion.

Normalform bzgl. f_1:

$a_1 x^{e_1} + a_2 x^{e_2} + \cdots + a_m x^{e_m} \quad e_i \neq e_j$ für $i \neq j$

Kanonische Form bzgl. f_2:

$a_1 x^{e_1} + a_2 x^{e_2} + \cdots + a_m x^{e_m} \quad e_i < e_j$ für $i < j$

Oft gilt $s \sim t$ gdw $M(s,t) \sim 0$, $\exists$ Normalisierungsfunktion $\rightsquigarrow$ kanonische Funktion.

Beispiele (3)

b) Abelsche Halbgruppen — Varietät
Erzeugende — Relationen
$\Sigma :: a, b, c, f, s$ — $E :: as = c^2 s, bs = cs, s = f$
$+$ Kommutativität

Faktorhalbgruppe des freien komm. Monoids in a, b, c, f, s

Formebene: $\{a^{n_1} b^{n_2} c^{n_3} f^{n_4} s^{n_5} \mid n_i \in \mathbb{N}\}$

$\circ : M \times M \to M$ Addition der Exponenten.

Kongruenz, die von E erzeugt wird: Ersetzungsregeln

$\underline{s} \to f \quad \underline{c}f \to bf \quad \underline{b^2}f \to af$ „Modulo Kommutativität“

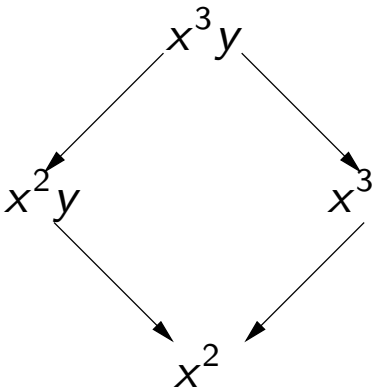
Definiere kanonische Funktion $\xrightarrow{*}$ mit kanonischen Formen
$\subseteq a^{n_1} b^{n_2} c^{n_3} f^{n_4}$

Beispiele (4)

c) $E = \mathbb{Q}[x, y] : x^3 - x^2, x^2y - x^2$

$i = \langle x^3 - x^2, x^2y - x^2 \rangle \qquad E/i$

Regeln: $x^3 \to x^2 \qquad x^2y \to x^2$ Reduktionsfunktion



$x^3 - x^2y \xrightarrow{*} 0$

$\to$ definiert Simplifikationsfunktion $p \xrightarrow{*} NF(p)$, sie ist kanonisch ($\rightsquigarrow$ Gröbner Basen).

Normalformen für Polynomringe und Quotientenkörper, d. h. Normalformen für Polynome und rationale Ausdrücke.

Beispiele (5)

Ringe: Axiome kommutative Ringe mit 1.
Signatur: $0, 1, -, +, *$
Axiome: $+$ Komm., Ass., 0 neutr. El., Gruppe inv. -
$*$ Komm., Ass., Einh. $+$ Distributivgesetz
Gleichheitsaxiome $\rightsquigarrow$ Varietät.
Univariate Polynome: Formebene.
$R[x] : a_n x^n + \cdots + a_1 x + a_0,\ n \geq 0, a_i \in R, a_n \neq 0 \cup \{0\}$
System kanonischer Formen für $R[x]$ (**dicht**) oder **dünn** alle Koeffizienten $\neq 0$.
Multivariate Polynome: Formebene.
Rekursive Darstellung: $R[x_1 \ldots x_n] = R[x_1 \ldots x_{n-1}][x_n]$

$$a(x_1, \ldots, x_n) = \sum_{i=0}^{\text{grad}(a(\bar{x}))} a_i(x_1 \ldots x_{n-1}) x_n^i$$

dicht/dünn

Beispiele (6)

3.14 Beispiel

$a(x,y,z) = (3y^2+(-2z^3)y+5z^2)x^2+4x+((-6z+1)y^3+3y^2+(z^4+1))$

Distributive Darstellung $a(\bar{x}) \in D[x]$

$a(\bar{x}) = \sum_{e\in\mathbb{N}^n} a_e x^e$ mit $a_e \in D$ dicht /dünn $a_e \neq 0$

$x^e \quad e \in \mathbb{N}^n$ werden oft Terme genannt.

$a(x,y,z) = 3x^2y^2 - 2x^2yz^3 + 5x^2z^2 + 4x - 6y^3z + y^3 + 3y^2 + z^4 + 1$

Reihenfolge der Terme? Ordnungen auf Termmengen, die kompatibel mit Termmultiplikation sind, z. B.

Lex $x > y > z$ $x^2y^2 > x^2yz^3 > x^2z^2 > x > y^3z \cdots$

oder

Grad-Lex $x^2yz^2 > x^2y^2 > x^2z^2 > y^3z > z^4 > y^3 > y^2 > x$

Beispiele (7)

3.15 Beispiel

$a(x,y) =$
$((x^2-xy+x)+(x^2+3)(x-y+1))\cdot((y^3-3y^2-9y-5)+x^4(y^2+2y+1))$

Distributive Darstellung:

$$f_1(a(x,y)) = 5x^2y^3+3x^2y^2-13x^2y-10x^2+3x^6y+2x^6-xy^4+7xy^3\cdots$$

Kanonische distributive Darstellung:

$$f_2(a(x,y)) = x^7y^2+2x^7y+x^7-x^6y^3+3x^6y+2x^6-x^5y^3+2x^5y^2+\cdots$$

Faktorisierte Normalform:

$f_3(a(x,y)) =$
$(x^3-x^2y+2x^2-xy+4x-3y+3)(x^4y^2+2x^4y+x^4+y^3-3y^2-9y-5)$

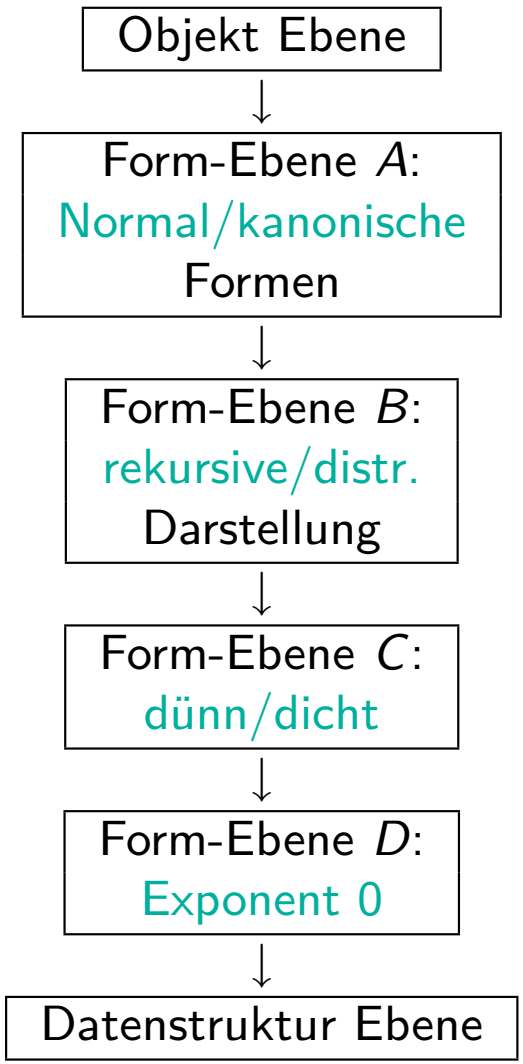
Faktorisierte kanonische Form:

$$f_4(a(x,y)) = (x-y+1)(x^2+x+3)(x^4+y-5)(y+1)^2$$

Beispiele (8)

- Rekursive Darstellung
- Distributive Darstellung
  f_1
- Kanonische distributive Darstellung
  (Ordnung auf Termen) f_2
- Faktorisierte Normalform
  $\prod_{i=1}^{k} p_i \longrightarrow \prod_{i=1}^{k} f_2(p_i)$
  f_3
- Faktorisierte kanonische Form (D ZPE)
  $\prod_{i=1}^{k} p_i \longrightarrow \prod_{i=1}^{k} f_2(p_i)$

Faktorisiere die $f_2(p_i)$, fasse gleiche Faktoren zusammen.
Einheitsnormale Faktorisierung + Ordnung der Faktoren



Beispiele (9)

- f_1, f_2, f_3 sind „einfach“ zu berechnen.
- f_4 kostspielig!
- f_2, f_3 werden am häufigsten verwendet.
- $(x + y)^{1000} - y^{1000}$ von f_2 und f_3 expandiert!
- Weitere Transformationen erwünscht!

Normalformen für rationale Ausdrücke

D Integritätbereich, Quotientenkörper (D) F_D

Annahme: D ZPE-Ring, d. h. GGT existiert.

$D[x_1, \ldots, x_n]$ $D(x_1 \ldots x_n)$

Formebene: $\frac{a}{b}$ :: GGT $(a, b) = 1$, b EN, a, b kanon.

$(\exp * \exp)$ $(\exp + \exp)$ $\frac{p}{q}$

Normalisierungsfunktion für rationale Ausdrücke

f_5 ::

1. Bringe in Gestalt $\frac{a}{b}$ mit $a, b \in D[x_1, \ldots, x_n]$

   (Gemeinsamer Nenner, Ausmultiplizieren) $\frac{a}{b} + \frac{a'}{b'} \longrightarrow$

2. GGT $(a, b) = 1$ $\qquad \frac{a}{b} \longrightarrow \frac{a'}{b'} \qquad a = a' \cdot g,\ b = b' \cdot g$

3. b Einheitsnormal: $\frac{a'}{b'} \longrightarrow \frac{a''}{b''},\ a'' = a' \cdot (u(b'))^{-1},\ b'' = b' \cdot (u(b'))^{-1}$

4. $\frac{a''}{b''} \longrightarrow \frac{f_2(a'')}{f_2(b'')}$

Andere Formen: a/b
Fakt/Fakt Fakt / erweitert erweitert / Faktor
mit GGT$(a, b) = 1$, b einheitsnormal.

Normalformen Potenzreihen

Potenzreihen Truncated Power Series: Abbruchgrad t

$$a(x) = \sum_{k=0}^{t} a_k x^k + 0(x^{t+1})$$

$\rightsquigarrow$ Problem Normalformen

Explizite Darstellung unendlicher Reihen:

$$e^x = \sum_{k=0}^{\infty} \frac{1}{k!} x^k, \text{ d. h.}$$

$$a(x) = \sum_{k=0}^{\infty} f_a(k) x^k$$

Koeffizientenfunktion $f_a : \mathbb{N} \longrightarrow \mathbb{Q}$ rekursiv.

Datenstrukturebene

Darstellung der Objekte der Formebene im Rechner.
Entscheidung:
Alle nur Normalformen nur kanonische Formen
Ziel: Effiziente Unterstützung (Realisierung) der grundlegenden Operationen.
1. $\mathbb{Z}, \mathbb{Q}$
Single-Precision $\rightsquigarrow$ Wortlänge z. B. 64 Bits
Multi-Precision $\rightsquigarrow$ Langzahlen
SP-Zahl mit Vorzeichen: $-2^{63}+1 \leq$ SP-Zahl $\leq 2^{63}-1$
Langzahlen als Listen von SP-Zahlen.

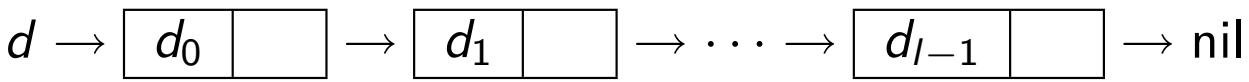
$$(d_0, \ldots, d_{l-1}) \longleftrightarrow \sum_{i=0}^{l-1} d_i \beta^i$$

Wahl von β

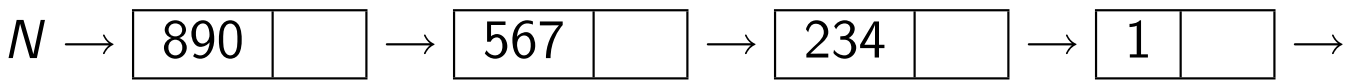
$1 \leq \beta - 1 \leq$ SP-Zahl oder als Feld var. Länge (wie gehabt!)

Datenstrukturebene (Forts.)

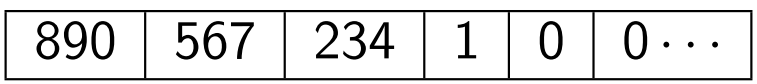
Wahl von β:
i) $\beta - 1$ größte SPZ
ii) $\beta = 10^p$ p so groß wie möglich.
Länge l der Liste: Dynamisch oder statisch.
Implementierung: Zeiger oder Felder.
Referenzierte / sequentielle Zuweisung (-Vorzeichen, -Länge)



$\beta = 10^3$ $N = 1/234/567/890$



Feld



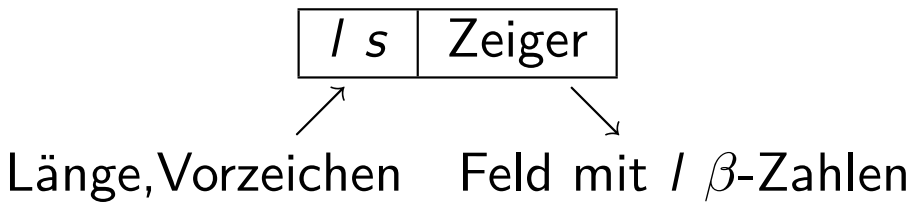
Probleme

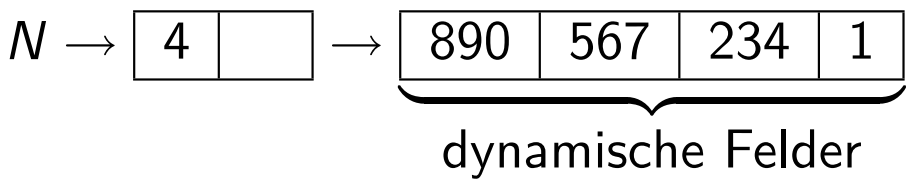
- Feste Länge (Überlauf), auffüllen mit 0 (Platz Verschwendung).
- Listen, Pointer Kosten Platz, Kosten für nächste Stelle.

Descriptor Allocation

Mischung

Beschreibungsblock

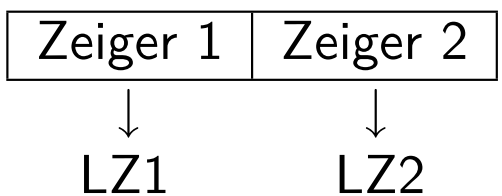




Problem: Speicherverwaltung kostspielig

Garbagecollection

Rationale Zahlen $Q \rightarrow$



Datenstrukturen für Polynome

Datenstrukturen für Polynome / rationale Funktionen.
Hängen von Entscheidungen auf Formebenen $B \mid C \mid D$ ab.

B : Rekursive, dünn $\longrightarrow$ Listen
C : Distributive, dünn $\longrightarrow$ Felder.

Koeff.-Zeiger	Exponent	Next
$\hookrightarrow a_i$	$\hookrightarrow i$	(SP-Zahl)

$a_i x_1^i$
$a_i \in D[x_2, \ldots, x_n]$

Als Headerknoten

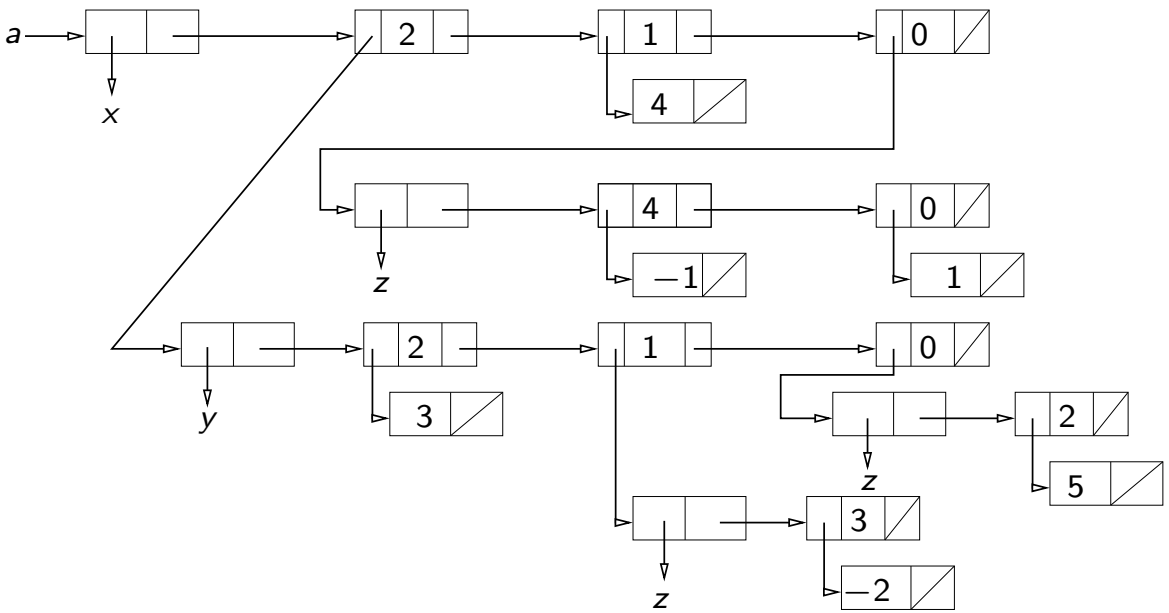
Indet_Zeiger	First_Zeiger
$\hookrightarrow$ Name_Var	$\hookrightarrow$ erstes Monom in Stufe

Beispiel

3.16 Beispiel $a(x,y,z) \in \mathbb{Z}[x,y,z]$ mit

$$\begin{aligned} a(x,y,z) &= 3x^2y^2 - 2x^2yz^3 + 5x^2z^2 + 4x - z^4 + 1 \\ &= (3y^2 + (-2z^3)y + 5z^2)x^2 + 4x + (-z^4 + 1) \end{aligned}$$

DS-Darstellung für rekursive dünn



Beachte hier Koeffizienten aus $\mathbb{Z}$ als SPZ (können auch als LZ oder rationale Zahlen dargestellt sein!) Erlaubt Unterscheidung (Polynom, Zahl in $\mathbb{Z}, \mathbb{Q}$).

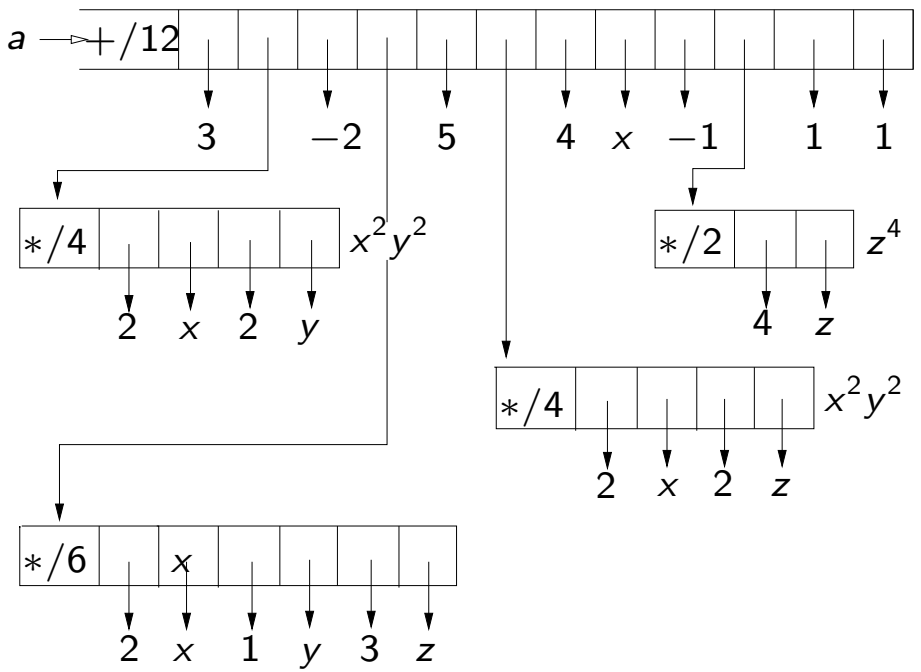
Distributive Form Darstellung
Dynamische Felder als DS für Pol. (Maple)

Type/Length	Coeff	Term	$\cdots$	Coeff	Term

Sum Product in Maple

Type/Length	Exponent	Fact	$\cdots$	Exponent	Fact

3.17 Beispiel



Beispiel (Forts.)