# Symbolisches Rechnen

## Wintersemester 2014/15

Prof. Dr. Hans-Gert Gräbe

http://bis.informatik.uni-leipzig.de/HansGertGraebe

zusammen mit Prof. Dr. Johannes Waldmann, HTWK Leipzig

# Was ist Symbolisches Rechnen?

Selbstverständnis der Hersteller der großen Systeme:

Mathematica – The world's definitive system for modern technical computing.

> *For more than 25 years, Mathematica has defined the state of the art in technical computing – and provided the principal computation environment for millions of innovators, educators, students, and others around the world. [. . . ] Mathematica provides a single integrated, continually expanding system that covers the breadth and depth of technical computing – and with Mathematica Online, it is now seamlessly available in the cloud through any web browser, as well as natively on all modern desktop systems.*

Maple: The Essential Math Software for Mathematics and Modeling

> *Maple is the essential technical computing software for today's engineers, mathematicians, scientists, and students.*

# CAS im Einsatz – erste Beobachtungen

Ein Beispiel: Was ist $\sqrt{2}$?

Die (im Kernbereich ausgeführten) Rechnungen eines CAS sind *grundsätzlich exakt*.

CAS ordnen symbolischen Ausdrücken *Eigenschaften* zu, die deren mathematischen Gehalt widerspiegeln.

CAS verfügen über die Fähigkeit zur *Manipulation symbolischer Ausdrücke*.

CAS bieten eine *ausgebaute Programmiersprache*, mit der man den eingebauten Interpreter gut steuern kann. Dabei werden alle gängigen Sprachkonstrukte einer imperativen Programmiersprache unterstützt und noch um einige Spezifika erweitert, die aus der Natur des symbolischen Rechnens folgen.

# CAS im Einsatz – erste Beobachtungen

CAS sind *Expertensysteme*, in denen algorithmische Fertigkeiten aus sehr verschiedenen Bereichen der Mathematik nahtlos miteinander zusammenspielen

## CAS als Expertensystem:

Seine Nutzung erfordert *Expertise des Nutzers* in

- der adäquaten Aufbereitung der Problemstellung
- der geschickten Nutzung des Werkzeugs zur Problemlösung
- der korrekten Interpretation der Antworten.

# Was ist Symbolisches Rechnen?

Synonymer Gebrauch der Begriffe *Symbolisches Rechnen* und *Computeralgebra*.

**Von der Natur der Inhalte und der Form der Manipulationen her:**

Symbolisch-algebraische Manipulationen mathematischer Inhalte.

Diese „Natur der Inhalte“ (Semantik) ist dem Computer nicht direkt zugänglich, sondern bedarf zur adäquaten Anwendung des an Menschen und menschliches Wissen gebundenen *Verständnisses* dieser Inhalte. Der Computer ist nur zu syntaktischen Manipulationen fähig.

# Was ist Symbolisches Rechnen?

## Von der syntaktischen Form her:

Rechnen mit Symbolen, die mathematische Objekte repräsentieren.

## Symbolisch bedeutet:

Ziel ist die Suche nach einer geschlossenen oder approximativen Formel im Sinne des deduktiven Mathematikverständnisses.

## Algebraisch bedeutet:

Es wird eine *exakte* mathematische Ableitung aus den Ausgangsgrößen durchgeführt.

# Was ist Symbolisches Rechnen?

Das Symbolische Rechnen hat damit mathematische Konstruktionen zum Gegenstand, die zwar syntaktisch endlich (und damit *exakt* im Computer darstellbar) sind, aber semantisch unendliche Strukturen repräsentieren können. Sie kommt mathematischen Arbeitstechniken näher als Anwendungen und implementierte Kalküle der numerischen oder diskreten Mathematik.

Die Bedeutung der Verfügbarkeit von CAS reicht weit über den Bereich der algorithmischen Mathematik hinaus: Moderne CAS mit ihrer *einheitlichen* Oberfläche sind **metamathematische Werkzeuge** für Anwender aus den verschiedensten Bereichen von Wissenschaft und Technik.

# Was ist Symbolisches Rechnen?

Die zusätzlichen Visualisierungs- und Präsentationsmöglichkeiten, die weltweite Vernetzung, der Zugang zu ständig aktualisierten Datenbeständen und vieles mehr machen die führenden CAS heute zu **integrierten Systemen für wissenschaftlich-technisches Rechnen**.

## Deshalb Erweiterung um den Aspekt

Entwicklung des zu Implementierung und Management solcher Systeme notwendigen informatik-theoretischen und -praktischen Instrumentariums.

Die Computeralgebra befindet sich damit an der Schnittstelle zentraler Entwicklungen verschiedener Gebiete sowohl der Mathematik als auch der Informatik.

# Was ist Symbolisches Rechnen?

## Computeralgebra-Handbuch 2003:

Die Computeralgebra ist ein Wissenschaftsgebiet, das sich mit Methoden zum Lösen mathematisch formulierter Probleme durch symbolische Algorithmen und deren Umsetzung in Soft- und Hardware beschäftigt. Sie beruht auf der exakten endlichen Darstellung endlicher oder unendlicher mathematischer Objekte und Strukturen und ermöglicht deren symbolische und formelmäßige Behandlung durch eine Maschine. Strukturelles mathematisches Wissen wird dabei sowohl beim Entwurf als auch bei der Verifikation und Aufwandsanalyse der betreffenden Algorithmen verwendet. Die Computeralgebra kann damit wirkungsvoll eingesetzt werden bei der Lösung von mathematisch modellierten Fragestellungen in zum Teil sehr verschiedenen Gebieten der Informatik und Mathematik sowie in den Natur- und Ingenieurwissenschaften.

# Zur Genese einer Computermathematik

Neben Pflege, Weiterentwicklung und Vermittlung entsprechender *Kalküle* als traditionellem Gegenstand *mathematischer* Bildung formiert sich eine weitere Querschnittswissenschaft, welche die Erstellung, Pflege, Nutzungsunterweisung und Einbettung für technikbasierte Hilfsmittel geistiger Arbeit, kurz, eine sich neu herausbildende *technologische Seite des Denkens*, zum Gegenstand hat.

Ein solches Verständnis von Informatik lässt Raum für eine weitergehende Symbiose von Kalkül und Technologie als Gegenstand eines Faches zwischen Mathematik und Informatik, dem [Grabmeier 1995] den provisorischen Namen **Computermathematik** gegeben hat.

Das symbolische Rechnen ist ein wesentlicher Teil dieses Gebiets.

# Das Simplifizieren von Ausdrücken

Manipulationen von Ausdrücken („Termersetzungen“) sind eine Kernfähigkeit von CAS, da derartige (syntaktische) Manipulationen in vielen Kalkülen bedeutsam sind – syntaktisch verschiedene, aber semantisch gleichwertige Ausdrücke.

Semantische Interpretation innerhalb einer Theorie, hier die Theorie trigonometrischer Ausdrücke als Teilgebiet der Theorie reellwertiger Funktionen.

Automatische Vereinfachungen

$$\begin{aligned} \sin(\arcsin(x)) &\to x \\ \sin(\arctan(x)) &\to \frac{x}{\sqrt{x^2+1}} \\ \mathrm{abs}(\mathrm{abs}(x)) &\to \mathrm{abs}(x) \end{aligned}$$

# Zum Begriff der Simplifikation

Vereinfachungen können an verschiedenen Stellen einer Rechnung mit unterschiedlichen Intentionen und sogar einander widersprechenden Zielvorgaben erforderlich sein.

## Simplifikation

*Zielgerichtete* Transformation eines Ausdrucks in eine semantisch gleichwertige, aber syntaktisch verschiedene Form *mit gewissen vorgegebenen Eigenschaften*.

Unterscheide

- *syntaktische Korrektheit* der Simplifikation und
- *semantische Korrektheit* der Simplifikation.

# Das funktionale Transformationskonzept

Das funktionale Transformationskonzept

## Konzept

Beim funktionalen Konzept werden Transformationen als Funktionsaufrufe realisiert, in welchen eine genaue syntaktische Analyse der (ausgewerteten) Aufrufparameter erfolgt und danach entsprechend verzweigt wird.

**Nachteile**:

- Die Funktionsdefinitionen sind sehr unübersichtlich und erfordern eine Vielzahl von Fallunterscheidungen.
- Die Typanalyse ist bei vielen Funktionen von ähnlicher Bauart, so dass unnötig Code dupliziert wird.
- Die Transformationsregeln sind relativ „starr“ und lassen sich nur mit großem Aufwand modifizieren.

# Das regelbasierte Transformationskonzept

## Konzept

Beim regelbasierten Zugang wird die im funktionalen Zugang notwendige Code-Redundanz vermieden, indem der Transformationsvorgang als `Apply(Expression, Rules)` aus einem allgemeinen Programmteil und einem speziellen Datenteil aufgebaut wird.

Der Datenteil `Rules` enthält die jeweils konkret anzuwendenden Ersetzungsregeln, also Informationen darüber, welche Kombinationen von Funktionssymbolen wie zu ersetzen sind.

Der Programmteil `Apply` stellt die erforderlichen Routinen zur Mustererkennung und Unifikation bereit.

Beispiel Regelsystem für sin in REDUCE.

`http://reduce-algebra.sourceforge.net/`

# Konzeptionelle Anforderungen

1. *Pattern Matching*: Das CAS muss einen **Unifikator** zur Lokalisierung entsprechender Anwendungsmöglichkeiten sowie der Zuordnung von Belegungen für die formalen Parameter bereitstellen.
2. Wie bei Funktionen ist zwischen **Regeldefinition** und **Regelanwendung** zu unterscheiden.
3. Wie bei Funktionen können in Regeldefinitionen **formale Parameter** auftreten.
   Bei Bezeichnern in einer Regeldefinition ist zu unterscheiden, ob der Bezeichner literal als Symbol für sich selbst steht oder als formaler Parameter eine Platzhalterfunktion hat.
   Wir fassen als Platzhalter verwendete Bezeichner in einer Liste $(u_1, \ldots, u_n)$ zusammen.

# Konzeptionelle Anforderungen

4. Im Gegensatz zu Funktionen kann die Anwendung einer passenden Regel **konditional** sein, d. h. vom Wert eines vorab zu berechnenden booleschen Ausdrucks abhängen.
   Eine Regeldefinition besteht damit aus vier Teilen:

$$\texttt{Rule}(\mathit{lhs}, \mathit{rhs}, \mathit{bool})(u_1, \ldots, u_n)$$

Syntax für Regeldefinitionen in verschiedenen CAS:

| | |
|---|---|
| MATHEMATICA | `lhs /; bool -> rhs` |
| MAXIMA | `tellsimpafter(lhs, rhs, bool)` |
| REDUCE | `lhs => rhs when bool` |

# Simplifizieren und mathematische Exaktheit

Beispiele:

- $\arctan(\tan(x)) = x$ ?
- $\log(\exp(x)) = x$ ?
- $\sqrt{u \cdot v} = \sqrt{u} \cdot \sqrt{v}$ ?

# Simplifizieren und mathematische Exaktheit

| Ausdruck | Axiom | Maxima | Maple | Mma | Reduce |
|---|---|---|---|---|---|
| | 2010 | 5.20 | 13 | 8.0 | 3.8 |
| $\lvert -\pi \cdot x \rvert$ | $\lvert \pi \cdot x \rvert$ | $\pi \cdot \lvert x \rvert$ | $\pi \cdot \lvert x \rvert$ | $\pi \cdot \lvert x \rvert$ | $\pi \cdot \lvert x \rvert$ |
| $\arctan(\tan(x))$ | $x$ | $*$ | $*$ | $*$ | $*$ |
| $\arctan(\tan(\frac{25}{7}\pi))$ | $\frac{25}{7}\pi$ | $-\frac{3}{7}\pi$ | $-\frac{3}{7}\pi$ | $-\frac{3}{7}\pi$ | (2) |
| $\sqrt{x^2}$ | $*$ | $\lvert x \rvert$ | $*$ | $*$ | $\lvert x \rvert$ |
| $\sqrt{x\,y} - \sqrt{x}\,\sqrt{y}$ | $*$ | $*$ | $*$ | $*$ | $*$ |
| $\sqrt{\frac{1}{z} - \frac{1}{\sqrt{z}}}$ | (1) | 0 | $*$ | $*$ | (1) |
| $\ln(\exp(x))$ | $x$ | $x$ | $*$ | $*$ | $x$ |
| $\ln(\exp(10\,\iota))$ | $10\,\iota$ | $10\,\iota$ | $*$ | $10\,\iota - 4\pi\,\iota$ | $10\,\iota$ |

$$(1) = \frac{\sqrt{z}\sqrt{\frac{1}{z}} - 1}{\sqrt{z}} \qquad (2) = \arctan\left(\tan\left(\frac{4}{7}\,\pi\right)\right)$$

# Annahmen-Systeme

| | |
|---|---|
| MAXIMA | `declare(x,real)` |
| MAPLE | `assume(x,real)` |
| MATHEMATICA | `SetOptions[Assumptions -> x ∈ Reals]` |
| MUPAD | `assume(x,Type::Real)` |

Schwierigkeiten des neuen Gegenstands:

- Das Problem eines konsistenten Typsystems ist eine Teilfrage eines konsistenten Annahmen-Systems.
- Annahmen sind oft nicht kanonisch einzelnen Bezeichnern zuzuordnen.
- Selbst für eine überschaubare Menge von erlaubten Annahmen führt das Inferenzproblem auf mathematisch und rechnerisch schwierige, oft in ausreichender Allgemeinheit algorithmisch unlösbare Probleme.

# Annahmen-Systeme

Praktisch erlaubte Annahmen beschränken sich deshalb meist auf wenige Eigenschaften wie etwa

- Annahmen über die Natur einer Variablen (`assume(x,integer)`, `assume(x,real)`),
- die Zugehörigkeit zu einem reellen Intervall (`assume(x>0)`) oder
- die spezielle Natur einer Matrix (`assume(m,quadratic)`)

Das Inferenzproblem wird stets nur im schwachen Sinne gelöst: es wird eine (ggf. keine), nicht unbedingt die strengste ableitbare Annahme gesetzt.

Die Möglichkeit, einzelnen Bezeichnern Eigenschaften aus einem Spektrum von Vorgaben zuzuordnen, macht die Sprache des CAS reichhaltiger, ändert jedoch nichts am prinzipiellen Transformationskonzept, sondern wertet nur den konditionalen Teil auf.

# Das allgemeine Simplifikationsproblem

Als *wohlgeformte Ausdrücke* (oder kurz: Ausdrücke) bezeichnen wir (LISP-Notationskonvention)

- alle Symbolvariablen (Bezeichner) und Konstanten des Systems (**atomare Ausdrücke**)
  sowie
- Listen $A = [f, a, b, c, \ldots]$, deren Elemente selbst wohlgeformte Ausdrücke sind (**zusammengesetzte Ausdrücke**)

$f, a, b, c, \ldots$ heißen *Teilausdrücke erster Ebene* von $A$,
*Teilausdrücke* werden daraus rekursiv definiert.

Der Bezeichner $u$ *kommt in* $A$ *vor*, wenn $u$ ein Teilausdruck von $A$ ist.

# Das allgemeine Simplifikationsproblem

$A$ Ausdruck, in dem Bezeichner $x = (x_1, \ldots, x_n)$ vorkommen, $U_1, \ldots, U_n$ $x$-freie Ausdrücke. $A(x \vdash U)$ ist der Ausdruck, welcher entsteht, wenn alle Vorkommen von $x_i$ in $A$ durch $U_i$ ersetzt werden.

$\mathcal{E}$ Menge der *wohlgeformten* Zeichenketten (Ausdrücke) mit einer Äquivalenzrelation $\sim$, die semantische Gleichwertigkeit von Ausdrücken modelliert.

$\sim$ wird als *kontextfrei* vorausgesetzt, d. h. es gilt

$$A(x \vdash U) \sim B(x \vdash V)$$

für alle $x$-freien Ausdrücke $U, V$ mit $U \sim V$ und alle Ausdrücke $A(x), B(x)$ mit $A(x) \sim B(x)$.

# Simplifikatoren

## Simplifikator:

Eine effektiv berechenbare Funktion $S : \mathcal{E} \longrightarrow \mathcal{E}$ mit

$$S(S(t)) = S(t) \qquad \text{(Idempotenz)} \qquad \text{(I)}$$

$$S(t) \sim t \qquad \text{(Äquivalenz)} \qquad \text{(E)}$$

für alle $t \in E$.

Wird in einem regelbasierten Transformationskonzept als fortgesetzte Anwendung eines Arsenals von Transformationsregeln ausgeführt.

# Simplifikation durch Regelsysteme

Beschreibung durch ein (endliches) Regelsystem $\mathcal{R}$ mit Regeln der Gestalt $\texttt{Rule}(L, R, B)(u)$.

Dabei sind

- $u = (u_1, \ldots, u_n)$ Liste von formalen Parametern,
- $L, R, B \in \mathcal{E}$ Ausdrücke, so dass für alle *zulässigen* Belegungen $u \vdash U$ der formalen Parameter mit $u$-freien Ausdrücken $U = (U_1, \ldots, U_n)$, d. h. solchen mit $\texttt{bool}(B(u \vdash U)) = \texttt{true}$, die entsprechenden Ausdrücke semantisch äquivalent sind, d.h. $L(u \vdash U) \sim R(u \vdash U)$ in $\mathcal{E}$ gilt.
- $\texttt{bool} : \mathcal{E} \to \{\texttt{true}, \texttt{false}, \texttt{fail}\}$ eine boolesche Auswertefunktion auf der Menge der Ausdrücke.

# Simplifikation durch Regelsysteme

Die Regel $r = \texttt{Rule}(L, B, R)(u)$ ist auf einen Ausdruck $A$ *anwendbar*,

(1) wenn es eine zu $u$ disjunkte Liste von Bezeichnern $x = (x_0, x_1, \ldots, x_n)$ gibt, so dass $A$ $x$-frei ist.
Wir arbeiten dann mit $L' = L(u \vdash x)$ und $R' = R(u \vdash x)$ **(gebundene Umbenennung)**

(2) wenn es weiter einen Ausdruck $A'$ und Teilausdrücke $U = (U_1, \ldots, U_n)$ von $A$ mit $A = A'(x \vdash U)$ gibt, so dass $L'$ ein Teilausdruck von $A'$ ist, d. h. $A' = A''(x_0 \vdash L')$ für einen weiteren Ausdruck $A''$ gilt **(Matching)**

(3) und $\texttt{bool}(B(u \vdash U)) = \texttt{true}$ gilt **(Konditionierung)**.

Im Ergebnis der Anwendung der Regel $r$ wird $A$ durch $A^{(1)} = A''(x_0 \vdash R(u \vdash U))$ ersetzt.

# Simplifikation durch Regelsysteme

Wir schreiben auch $A \to_r A^{(1)}$.

Der zugehörige Simplifikator $S : \mathcal{E} \longrightarrow \mathcal{E}$ ist der transitive Abschluss der Ersetzungsrelationen aus $\mathcal{R}$.

$S$ ist *effektiv*, wenn gilt:

*(Matching)* Es lässt sich effektiv entscheiden, ob es zu gegebenem Ausdruck $A$ und Regel $r \in \mathcal{R}$ ein Matching gibt.

*(Termination)* Nach endlich vielen Schritten

$$A \to_{r_1} A^{(1)} \to_{r_2} A^{(2)} \to_{r_3} \ldots \to_{r_N} A^{(N)}$$

mit $r_1, \ldots, r_N \in \mathcal{R}$ ist keine Ersetzung mehr möglich.

Vergleiche Vorlesungsteil zu Termersetzungssystemen.

# Kanonischer Simplifikator

Sinnvolle Aussagen sind nur innerhalb eingeschränkter Klassen $\mathcal{U} \subset \mathcal{E}$ von Ausdrücken möglich.

**Identifikationsproblem**:
Angabe eines effektiven Verfahrens, das $s \sim t$ beantwortet.

## Kanonischer Simplifikator:

Ein Simplifikator mit

$$s \sim t \Rightarrow S(s) = S(t) \qquad \text{für alle } s, t \in \mathcal{U} \tag{C}$$

heißt *kanonischer Simplifikator*.

Für einen kanonischen Simplifikator gilt $s \sim t$ genau dann, wenn die Normalformen (Zeichen für Zeichen) übereinstimmen.

# Starker Nulltester

Annahme: Existenz eines speziellen Symbol $0 \in \mathcal{U}$ sowie einer Funktion $M : \mathcal{U} \times \mathcal{U} \to \mathcal{U}$ mit

$$s \sim t \Leftrightarrow M(s,t) \sim 0$$

## Starker Nulltester

Als *starken Nulltester* bezeichnen wir einen Simplifikator $S$, für den zusätzlich

$$t \sim 0 \Rightarrow S(t) = 0 \qquad \text{für alle } t \in \mathcal{U} \tag{N}$$

gilt, d. h. jeder Nullausdruck $t \in \mathcal{U}$ wird durch $S$ auch als solcher erkannt.

$s \sim t$ gilt genau dann, wenn die Differenz zu Null vereinfacht werden kann:

$$s \sim t \Leftrightarrow S(M(s,t)) = 0.$$

Kern des Arguments ist die Existenz einer booleschen Funktion $\texttt{iszero} : \mathcal{U} \to \texttt{boolean}$ mit der Eigenschaft

$$t \sim 0 \iff \texttt{iszero}(t) = \texttt{true} \qquad \text{für alle } t \in \mathcal{U}$$

# Simplifikation polynomialer Ausdrücke

$S := R[x_1, \ldots, x_n]$

Eine Darstellung

$$f = \sum_a c_a X^a$$

eines Polynoms $f \in S$ mit paarweise verschiedenen Termen, die in fallender Reihenfolge bgzl. einer fixierten Ordnung auf den Termen angeordnet sind, heißt **distributive Darstellung**.

# Simplifikation polynomialer Ausdrücke

$T = T(X) = \{X^a : a \in \mathbb{N}^n\}$ heißt *Monoid der Terme* in $(x_1, \ldots, x_n)$.

$(T, <)$ heißt *Termordnung*, wenn $<$ eine lineare monotone Ordnung ist, d.h.

$$s < t \Rightarrow s \cdot u < t \cdot u \quad \text{für alle } s, t, u \in T$$

gilt.

**Satz:**

Existiert auf $R$ eine kanonische Form, dann ist die distributive Darstellung von Polynomen $f \in S$ mit Koeffizienten in kanonischer Form eine kanonische Form auf $S$.
Hat $R$ einen starken Nulltester, so auch $S$.

# Simplifikation polynomialer Ausdrücke

Als **rekursive Darstellung** bezeichnet man die Darstellung als Polynom in $x_n$ mit Koeffizienten aus $R[x_1, \ldots, x_{n-1}]$, wobei die Summanden nach fallenden Potenzen von $x_n$ angeordnet und die Koeffizienten rekursiv nach demselben Prinzip dargestellt sind.

## Folgerung:

Existiert auf $R$ eine kanonische Form, dann ist auch die rekursive Darstellung von Polynomen $f \in R[x]$ mit Koeffizienten in kanonischer Form eine kanonische Form auf $R[x]$.
Hat $R$ einen starken Nulltester, so auch $R[x]$.

# Simplifikation rationaler Funktionen

## Simplifikation rationaler Funktionen

Hat $R$ einen starken Nulltester, so auch der Ring $R(x_1, \ldots, x_n)$ der rationalen Funktionen über $R$.

Mit Blick auf die guten Simplifikationseigenschaften stellen CAS intern Funktionsausdrücke als rationale Funktionen mit **Kernen** als verallgemeinerten Variablen dar.

Ein Kern kann ein Symbol oder ein Ausdruck mit einem nicht-arithmetischen Funktionssymbol als Kopf sein.

Algebraische Abhängigkeitsrelationen zwischen diesen Kernen werden vom Simplifikator rationaler Funktionen nicht berücksichtigt.

| System | Rationale Normalform |
|---|---|
| Maxima | `rat(u)` |
| Maple | `normal(u)` |
| Mathematica | `Together[u]` |
| MuPAD | `normal(u)` |
| Reduce | standardmäßig |

## CAS und Polynomarithmetik

Eine eigenständige, im Systemkern verankerte Simplifikationsschicht für polynomiale und rationale Ausdrücke in solchen Kernen spielt eine wichtige Rolle im Simplifikationsdesign aller CAS.

# Bezeichnungen

$S := R[x_1, \ldots, x_n]$ sei ein Polynomring und $T(X, <) = \{X^a : a \in \mathbb{N}^n\}$ das Termmonoid mit einer fixierten Termordnung.

Für ein Polynom $f = \sum_a c_a X^a \neq 0$ und $X^{a_0} = \max(X^a : c_a \neq 0)$ bezeichnen wir

- $lt(f) = X^{a_0}$ als *Leitterm*,
- $lc(f) = c_{a_0} \neq 0$ als *Leitkoeffizienten*,
- $lm(f) = c_{a_0} X^{a_0}$ als *Leitmonom* und
- $red(f) = f - lm(f)$ als *Reduktum* von $f$.

Ein Polynom $f \neq 0$ mit $lc(f) = 1$ heißt *monisch*.

# Algebraische Zahlen

Nullstellen von Polynomen: $a = \sqrt{2},\ b = \sqrt{2+\sqrt{3}}$

$$c = \sqrt{2\sqrt{3}+4} = 1+\sqrt{3}$$

$$d = \sqrt{11+6\sqrt{2}} + \sqrt{11-6\sqrt{2}} = 6$$

## Satz:

Sei $R$ ein Körper mit kanonischer Form und $a$ eine Nullstelle des über $R$ irreduziblen Polynoms $p(x) = x^k - r(x) \in R[x]$ mit $\deg(r) < k$.

Stellt man polynomiale Ausdrücke in $R[a]$ in distributiver Form dar und wendet zusätzlich die algebraische Regel $\left\{a^k \to r(a)\right\}$ an, so erhält man eine kanonische Form in $R[a]$.

Genauer: Jedes Element aus $R[a]$ hat eine eindeutige Darstellung als $R$-lineare Kombination der Terme $T_{red} = \left\{a^0 = 1, a, \ldots, a^{k-1}\right\}$.

# Minimalpolynom

Eine solche Nullstelle $a$ eines Polynoms $p(x) \in R[x]$ bezeichnet man auch als *algebraische (über R) Zahl*.

Zum Beweis des Satzes wird das folgende Lemma benötigt.

**Lemma:**

Unter allen Polynomen

$$P := \{q(x) \in R[x] : q(a) = 0 \text{ und } lc(q) = 1\}$$

mit Leitkoeffizient 1 und Nullstelle $a$ gibt es genau Polynom $p(x)$ kleinsten Grades. Dieses ist irreduzibel und jedes andere Polynom $q(x) \in P$ ist ein Vielfaches von $p(x)$.

$p(x)$ bezeichnet man als *Minimalpolynom* von $a$.

# Der Ring der algebraischen Zahlen

**Satz:**

Die Summe und das Produkt zweier algebraischer Zahlen ist wieder eine algebraische Zahl.

Seien $\alpha_i, i = 1, \ldots, s$ algebraische Zahlen mit den Minimalpolynomen

$$p_i(x) = x^{d_i} - q_i(x), \ i = 1, \ldots, s$$

aus denen sich die algebraischen Ersetzungsformeln

$$\{\alpha_i^{d_i} \to q_i(\alpha_i), \ i = 1, \ldots, s\}$$

ergeben.

# Der Ring der algebraischen Zahlen

Menge der reduzierten Terme

$$T_{red} := \left\{ \alpha_1^{j_1} \cdots \alpha_s^{j_s} \ : \ 0 \leq j_i < d_i \right\}$$

**Folgerung:**

Sind $\alpha_1, \ldots, \alpha_s$ algebraische Zahlen über $k$ vom Grad $d_1, \ldots, d_s$, so bildet die Menge der $k$-linearen Kombinationen von Elementen aus $T_{red}$ einen Ring.

Zwischen Elementen der Menge $T_{red}$ kann es lineare Abhängigkeitsrelationen geben.

# Die Inverse einer algebraischen Zahl

Beispiele: $\frac{1+\iota}{1-\iota}$, $\frac{1}{\sqrt{2}+\sqrt{3}}$ und sogar

$$\frac{1}{\sqrt{2}+\sqrt{3}+\sqrt{5}} = \frac{1}{6}\sqrt{3}+\frac{1}{4}\sqrt{2}-\frac{1}{12}\sqrt{30}$$

*Aufgabe:* Finde für $a = \sqrt[5]{2}$ eine Darstellung von $b = \frac{a}{a+2}$ als $\mathbb{Q}$-lineare Kombination von Potenzen von $a$.

# Die Inverse einer algebraischen Zahl

## Satz:

Ist $Q(x) \in k[x]$ das Minimalpolynom der algebraischen Zahl $a \neq 0$, so gilt

$$a^{-1} = -\frac{1}{Q(0)} \cdot \left.\frac{Q(x) - Q(0)}{x}\right|_{x=a}$$

Hierbei ist $Q(0) \neq 0$ das Absolutglied des irreduziblen Polynoms $Q(x)$, so dass $Q(x) - Q(0)$ durch $x$ teilbar ist.

## Folgerung:

Sind $\alpha_1, \ldots, \alpha_s$ algebraische Zahlen über $k$ vom Grad $d_1, \ldots, d_s$, so lässt sich jeder $k$-rationale Ausdruck $\frac{P(\alpha)}{Q(\alpha)} \in k(\alpha_1, \ldots, \alpha_s)$, für dessen Nenner $Q(\alpha) \neq 0$ gilt, als $k$-lineare Kombination von Elementen aus $T_{red}$ darstellen.

## Hauptsatz über das Rechnen in einer algebraischen Erweiterung

Ist $\alpha$ eine algebraische Zahl über $k$ vom Grad $d$, so bildet die Menge $R := k[\alpha]$ der $k$-linearen Kombinationen von Termen aus

$$T_{red} := \{\alpha^i,\ i = 0, \ldots, d-1\}$$

einen Körper.

Kann man in $k$ effektiv rechnen, so auch in $R$: Jeder rationale Ausdruck $A = \frac{P(\alpha)}{Q(\alpha)} \in k(\alpha)$ (mit $Q(\alpha) \neq 0$) kann eindeutig als $k$-lineare Kombination von Termen aus $T_{red}$ dargestellt werden.

Ist das Minimalpolynom von $\alpha$ bekannt, so kann diese reduzierte Form effektiv berechnet werden, was auf eine kanonische Form in $R$, die *algebraische Normalform*, führt, wenn wir eine kanonische Form in $k$ voraussetzen.