Notizen zu meinem Teil der Vorlesung „Symbolisches Rechnen“

Gemeinsame Vorlesung
mit J. Waldmann (HTWK Leipzig)

Wintersemester 2014/15

H.-G. Gräbe, Institut für Informatik
http://bis.informatik.uni-leipzig.de/HansGertGraebe

20. Januar 2015

Inhaltsverzeichnis

Kapitel 1

Das Simplifizieren von Ausdrücken

Eine wichtige Eigenschaft von CAS ist die Möglichkeit, *zielgerichtet* Ausdrücke in eine semantisch gleichwertige, aber syntaktisch verschiedene Form zu transformieren. Wir hatten im letzten Kapitel gesehen, dass solche *Transformationen* eine zentrale Rolle im symbolischen Rechnen spielen und dass dazu – wieder einmal ähnlich einem Compiler zur Compilezeit – die syntaktische Struktur von Ausdrücken zu analysieren ist.

Zum besseren Verständnis der dabei ablaufenden Prozesse ist zunächst zu berücksichtigen, dass einige zentrale Funktionen wie etwa die Polynomaddition aus Effizienzgründen als Funktionsaufrufe, zudem auf teilweise speziellen Datenstrukturen, implementiert sind und deshalb Vereinfachungen wie $(x+2)+(2\,x+3) \to 3\,x+5$ unabhängig von jeglichen Transformationsmechanismen ausgeführt werden.

Weiterhin gibt es eine Reihe von Vereinfachungen, die automatisch ausgeführt werden.
Jedoch ist nicht immer klar, in welcher Richtung eine mögliche Umformung auszuführen ist.

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

1.1 Simplifikationen als zielgerichtete Transformationen

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

Zur Berechnung von $\int \sin(2\,x)\,\cos(3\,x)\,d\,x$ ist es etwa angezeigt, den Ausdruck der Form $\sin(2\,x)\,\cos(3\,x)$ nach dem Additionstheorem

$$\sin(a)\,\cos(b) = \frac{1}{2}\,(\sin(a+b)+\sin(a-b))$$

in die Differenz $\frac{1}{2}\,(\sin(5\,x)-\sin(x))$ zu zerlegen, um dann diese Differenz termweise integrieren zu können.

```
int(sin(2*x)*cos(3*x),x);
```

$$\frac{\cos(x)}{2} - \frac{\cos(5\,x)}{10}$$

Um die Lösung der Gleichung $\sin(2\,x) = \cos(3\,x)$ zu bestimmen, ist es dagegen sinnvoll, nach der Umformung des Ausdrucks in $\sin(2\,x)+\sin(3\,x-\frac{\pi}{2}) = 0$ darauf das umgekehrte Additionstheorem

$$\sin(a)+\sin(b) = 2\,\sin\left(\frac{a+b}{2}\right)\sin\left(\frac{a-b}{2}\right)$$

anzuwenden, um die Differenz in das Produkt

$$2\,\sin\left(\frac{10\,x-\pi}{4}\right)\cos\left(\frac{2\,x-\pi}{4}\right)=0$$

zu verwandeln. Hieraus lässt sich die Lösungsmenge unmittelbar ablesen als

$$\begin{aligned}L&=\left\{\frac{\pi}{10}+\frac{2}{5}\,k\,\pi\,|\,k\in\mathbb{Z}\right\}\cup\left\{-\frac{\pi}{2}+2\,k\,\pi\,|\,k\in\mathbb{Z}\right\}\\&=\left\{\frac{\pi}{10}+2\,k\,\pi\,|\,k\in\mathbb{Z}\right\}\cup\left\{\frac{5\,\pi}{10}+2\,k\,\pi\,|\,k\in\mathbb{Z}\right\}\cup\left\{\frac{9\,\pi}{10}+2\,k\,\pi\,|\,k\in\mathbb{Z}\right\}\\&\quad\cup\left\{\frac{13\,\pi}{10}+2\,k\,\pi\,|\,k\in\mathbb{Z}\right\}\cup\left\{\frac{17\,\pi}{10}+2\,k\,\pi\,|\,k\in\mathbb{Z}\right\}\cup\left\{-\frac{\pi}{2}+2\,k\,\pi\,|\,k\in\mathbb{Z}\right\}\end{aligned}$$

in guter Übereinstimmung mit dem Ergebnis, welches MuPAD berechnet

```
solve(sin(2*x)=cos(3*x),x);
```

$$\left\{\frac{\pi}{2}+k\,\pi\,|\,k\in\mathbb{Z}\right\}\cup\left\{\frac{\pi}{10}+2\,k\,\pi\,|\,k\in\mathbb{Z}\right\}\cup\left\{\frac{9\,\pi}{10}+2\,k\,\pi\,|\,k\in\mathbb{Z}\right\}\cup\left\{\frac{13\,\pi}{10}+2\,k\,\pi\,|\,k\in\mathbb{Z}\right\}\cup\left\{\frac{17\,\pi}{10}+2\,k\,\pi\,|\,k\in\mathbb{Z}\right\}$$

Ein nicht ganz so überzeugendes Ergebnis liefert Wolfram-Alpha auf die Eingabe

```
solve sin(2*x)=cos(3*x)
```

$$\begin{aligned}&x=2\left(\pi\,n+\arctan\left(1-\sqrt{5}-\sqrt{5-2\sqrt{5}}\right)\right)\ \text{and}\ n\in\mathbb{Z}\\&x=2\left(\pi\,n+\arctan\left(1-\sqrt{5}+\sqrt{5-2\sqrt{5}}\right)\right)\ \text{and}\ n\in\mathbb{Z}\\&x=2\left(\pi\,n+\arctan\left(1+\sqrt{5}-\sqrt{5+2\sqrt{5}}\right)\right)\ \text{and}\ n\in\mathbb{Z}\\&x=2\left(\pi\,n+\arctan\left(1+\sqrt{5}+\sqrt{5+2\sqrt{5}}\right)\right)\ \text{and}\ n\in\mathbb{Z}\\&x=\frac{1}{2}\,\pi\,(4\,n-1)\ \text{and}\ n\in\mathbb{Z}\\&x=\frac{1}{2}\,(4\,\pi\,n+\pi)\ \text{and}\ n\in\mathbb{Z}\,,\end{aligned}$$

da die genauen Werte als rationale Vielfache von π nicht gefunden werden. Immerhin bietet Wolfram-Alpha auf die Frage

```
simplify arctan(1-sqrt(5)-sqrt(5-2 sqrt(5)))
```

als eine mögliche Antwort $-\frac{7}{20}\,\pi$.
Intern wird dabei MATHEMATICA aufgerufen und das folgende Kommando ausgewertet:

```
u = Reduce[Sin[2*x]==Cos[3*x],x]
```

$$\begin{aligned}c_1\in\mathbb{Z}\wedge\Big(&x=2\,\pi\,c_1-\frac{\pi}{2}\vee x=2\,\pi\,c_1+\frac{\pi}{2}\\&\vee x=2\,\arctan\left(\text{Root}\left[\texttt{\#1}^4-4\,\texttt{\#1}^3-14\,\texttt{\#1}^2-4\,\texttt{\#1}+1\&,1\right]\right)+2\,\pi\,c_1\\&\vee x=2\,\arctan\left(\text{Root}\left[\texttt{\#1}^4-4\,\texttt{\#1}^3-14\,\texttt{\#1}^2-4\,\texttt{\#1}+1\&,2\right]\right)+2\,\pi\,c_1\\&\vee x=2\,\arctan\left(\text{Root}\left[\texttt{\#1}^4-4\,\texttt{\#1}^3-14\,\texttt{\#1}^2-4\,\texttt{\#1}+1\&,3\right]\right)+2\,\pi\,c_1\\&\vee x=2\,\arctan\left(\text{Root}\left[\texttt{\#1}^4-4\,\texttt{\#1}^3-14\,\texttt{\#1}^2-4\,\texttt{\#1}+1\&,4\right]\right)+2\,\pi\,c_1\Big)\end{aligned}$$

`u//Simplify` löst die `Root`-Ausdrücke wie bei Wolfram-Alpha zu Wurzelausdrücken auf, während `u//FullSimplify` explizite Lösungen als rationale Vielfache von π liefert.

```
u // FullSimplify
```

$$c_1 \in \mathbb{Z} \wedge \left(2\,x + \pi = 4\,\pi\,c_1 \vee 4\,\pi\,c_1 + \pi = 2\,x \vee x = \pi\,\left(2\,c_1 - \frac{7}{10}\right) \vee x = \pi\,\left(2\,c_1 - \frac{3}{10}\right)\right.$$
$$\left. \vee\, x = \pi\,\left(2\,c_1 + \frac{1}{10}\right) \vee x = \pi\,\left(2\,c_1 + \frac{9}{10}\right)\right)$$

In dem gerade betrachteten Beispiel wurde dasselbe Additionstheorem in jeweils unterschiedlicher Richtung angewendet. Ähnlich kann man polynomiale Ausdrücke expandieren oder aber in faktorisierter Form darstellen, Basen in Potenzfunktionen zusammenfassen oder aber trennen, Additionstheoreme anwenden, um trigonometrische Ausdrücke eher als Summen oder eher als Produkte darzustellen, die Gleichung $\sin(x)^2 + \cos(x)^2 = 1$ verwenden, um eher sin durch cos oder eher cos durch sin zu ersetzen usw.

Eine solche *zielgerichtete Transformation* von Ausdrücken in semantisch gleichwertige *mit gewissen vorgegebenen Eigenschaften* wollen wir als **Simplifikation** bezeichnen.

In den meisten CAS gibt es für solche Simplifikationen eine Reihe spezieller Transformationsfunktionen wie `expand`, `collect`, `factor` oder `normal`, welche verschiedene, häufig erforderliche, aber fest vorgegebene Simplifikationsstrategien (Ausmultiplizieren, Zusammenfassen von Termen nach gewissen Prinzipien, Anwendung von Additionstheoremen für Winkelfunktionen, Anwendung von Potenz- und Logarithmengesetzen usw.) *lokal* auf einen Ausdruck anwenden.

Daneben existiert meist eine (oder mehrere) komplexere Funktion `simplify`, welche das Ergebnis verschiedener Transformationsstrategien miteinander vergleicht und an Hand des Ergebnisses entscheidet, welches denn nun das „einfachste“ ist.

Dies kann zu durchaus überraschenden Ergebnissen führen, wie das folgende MATHEMATICA-Beispiel zeigt:

`u =` $\dfrac{a^n}{(a-b)(a-c)} + \dfrac{b^n}{(b-a)(b-c)} + \dfrac{c^n}{(c-a)(c-b)}$

```
v = Table[u /. n -> i // Simplify, {i, 2, 7}]
```

$$1,\ a+b+c,\ a^2+(b+c)\,a+b^2+c^2+b\,c,$$
$$a^3+(b+c)\,a^2+\left(b^2+b\,c+c^2\right)\,a+b^3+c^3+b\,c^2+b^2c,$$
$$\frac{a^6}{(a-b)(a-c)} + \frac{\frac{b^6}{b-a}+\frac{c^6}{a-c}}{b-c},\ \frac{a^7}{(a-b)(a-c)} + \frac{\frac{b^7}{b-a}+\frac{c^7}{a-c}}{b-c}$$

Im MATHEMATICA-Hilfesystem heißt es dazu:

> There are many situations where you want to write a particular algebraic expression in the simplest possible form. Although it is difficult to know exactly what one means in all cases by the 'simplest form', a worthwhile practical procedure is to look at many different forms of an expression, and pick out the one that involves the smallest number of parts.

Diese *Anzahl von Teilen* lässt sich mit der Funktion `LeafCount` bestimmen. Für die ausgeführte Simplifikation erhält man maximal 50 Terme, während die Expansion als ganzrationale Ausdrücke für $n \geq 6$ längere Terme liefert. Das Beispiel zeigt also, dass in der Tat der „einfachste“ Ausdruck in einer wohlbestimmten Semantik gefunden wurde, auch wenn das Ergebnis möglicherweise nicht mit Ihren Erwartungen übereinstimmt.

```
LeafCount /@ v
```

$$\{1, 4, 18, 39, 50, 50\}$$

```
w = Table[u /. n -> i // Together, {i,2,7}]
LeafCount /@ w
```

$$\{1, 4, 19, 44, 79, 124\}$$

Um Vereinfachungen in verschiedenen wohldefinierten Richtungen zu erreichen, halten die CAS spezielle Transformationsfunktionen für unterschiedliche Simplifikationsaufgaben bereit. Damit kann die Simplifikationsrichtung im Laufe des interaktiven Dialogs leicht geändert werden. Nur ein kleiner Satz „allgemeingültiger“ Vereinfachungen wird automatisch durch das System ausgeführt. Der Nachteil dieses Herangehens besteht in der relativen Starrheit des Simplifikationssystems, womit ein Abweichen von den fest vorgegebenen Simplifikationsstrategien nur unter erheblichem Aufwand möglich ist.

In diesem Kapitel werden wir deshalb untersuchen, welche Möglichkeiten CAS zur Verfügung stellen, um eigene Simplifikationsstrategien zu definieren und anzuwenden. Dabei sind zwei grundlegend verschiedene Herangehensweisen im Einsatz, ein *funktionales* (Maple, MuPAD) und ein *regelbasiertes Transformationskonzept* (Reduce, Maxima, Mathematica, Axiom).

1.2 Das funktionale Transformationskonzept

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

Da in die Abarbeitung eines solchen Funktionsaufrufs die *Struktur* der Argumente mit eingeht, werden dazu Funktionen benötigt, welche diese Struktur wenigstens teilweise analysieren. Maple verfügt für diesen Zweck über die Funktion `type(A,T)`, die prüft, ob ein Ausdruck A den „Typ“ T hat.

Wie bereits an anderer Stelle erwähnt handelt es sich dabei allerdings **nicht um ein strenges Typkonzept für Variablen**, sondern um eine **syntaktische Typanalyse für Ausdrücke**, die in der Regel nur die Syntax der obersten Ebene des Ausdrucks analysiert (z. B. entsprechende Schlüsselworte an der Stelle 0 der zugehörigen Liste auswertet) oder ein mit dem Bezeichner verbundenes Typschlüsselwort abgreift. Dies erkennen wir etwa an nebenstehendem Beispiel.

```
exp(ln(x));
```

$$x$$

```
h:=exp(ln(x)+ln(y));
```

$$e^{\ln(x)+\ln(y)}$$

```
simplify(h);
```

$$x\,y$$

Die Struktur des Arguments verbirgt im zweiten Beispiel die Anwendbarkeit der Transformationsregel. Erst nach eingehender Analyse, die mit `simplify` angestoßen wird, gelingt die Umformung.

Betrachten wir als Beispiel die Definition der Exponentialfunktion in Maple, was mit `print(exp)` angezeigt werden kann, nachdem `interface(verboseproc=2)` gesetzt wurde:

```
proc(x::algebraic)
local res, i, t, q, n, f, r;
```

```
option builtin = HFloat_exp,
‘Copyright (c) 1992 by the University of Waterloo. All rights reserved.‘;
    if nargs <> 1 then error "expecting 1 argument, got %1", nargs
    elif type(x, ’complex(float)’) then return evalf(’exp’(x))
    elif type(x, ’rational’) then res := ’exp’(x)
    elif type(x, ’function’) and op(0, x) = ’ln’ then res := op(1, x)
    elif ...
    elif type(x, ’‘*‘’) and type(-I*x/Pi, ’rational’) then
        i := -I*x/Pi;
        if 1 < i or i <= -1 then
            t := trunc(i); t := t + irem(t, 2); res := exp((i - t)*Pi*I)
        elif type(6*i, ’integer’) or type(4*i, ’integer’) then
            res := cos(-I*x) + sin(-I*x)*I
        else res := ’exp’(x)
        end if
    elif  ...
    elif type(x, ’function’) and nops(x) = 1 then
        n := op(0, x);
        t := op(1, x);
        if n = ’arcsinh’ then res := t + sqrt(1 + t^2)
        elif n = ’arccosh’ then res := t + sqrt(t + 1)*sqrt(t - 1)
        elif n = ’arctanh’ then res := (t + 1)/sqrt(1 - t^2)
        elif n = ’arccsch’ then res := 1/t + sqrt(1 + 1/t^2)
        elif n = ’arcsech’ then res := 1/t + sqrt(1/t - 1)*sqrt(1/t + 1)
        elif n = ’arccoth’ then res := 1/sqrt((t - 1)/(t + 1))
        else res := ’exp’(x)
        end if
    elif ...
    else res := ’exp’(x)
    end if;
    exp(args) := res
end proc
```

Zunächst

```
if nargs <> 1 then error "expecting 1 argument, got %1", nargs
```

wird die Korrektheit der Anzahl der Aufrufargumente geprüft. Die meisten Zeilen des folgenden Codes beginnen mit `type(x, Art)` und analysieren den Kopfterm des aufgerufenen Arguments. Sehen wir uns die einzelnen Zeilen nacheinander an:

```
elif type(x, ’complex(float)’) then return evalf(’exp’(x))
```

untersucht, ob x eine (reelle oder komplexe) float-Zahl ist und ruft in diesem Fall `evalf(exp(x))` auf, was nach einer durch `evalf` ausgelösten Funktionstransformation zu `‘evalf/exp‘(x)` transformiert wird. `‘evalf/fff‘(x)` definiert ein einheitliches Konzept des Aufrufs von Funktionsdefinitionen zur numerischen Auswertung von Funktionen `fff`, das zur Laufzeit erweitert werden kann.

Ehe wir uns genauer anschauen, was im Fall eines Produkts ausgeführt wird, analysieren wir die an mehreren Stellen auftretende Rückgabe `res:=‘exp‘(x)` genauer.

In diesem Fall wird ein Funktionsausdruck mit dem Kopf `exp` und dem *ausgewerteten* Argument x gebildet.
Am letzten Beispiel sehen wir noch einmal, dass das Argument vor dem Aufruf von `exp` wirklich ausgewertet wurde.

```
exp(2/3);
```

$$\exp\left(\frac{2}{3}\right)$$

```
exp(27/3);
```

$$\exp(9)$$

Die Zeile

```
elif type(x, 'function') and op(0, x) = 'ln' then res := op(1, x)
```

untersucht, ob das Argument x ein Funktionsausdruck mit dem Kopf `ln` ist, und gibt in dem Fall das erste Element von $x = \ln(z)$, also z, zurück: $\exp(\ln(z)) = z$.
Weiter

```
elif type(x, '`*`') and type(-I*x/Pi, 'rational') then
```

wird geprüft, ob x ein Produkt ist, also mit dem Kopf `*` beginnt, und zusätzlich die Gestalt $x = y{\cdot}\pi\,\mathrm{i}$ mit $y \in \mathbb{Q}$ hat, da dann $\mathrm{e}^{y\,\pi\,\mathrm{i}}$ zu $\cos(y\,\pi) + \mathrm{i}\,\sin(y\,\pi)$ vereinfacht werden kann. Mit den Zeilen

```
y := -I*x/PI;
if 1 < y or y <= -1 then
      t := trunc(i); t := t + irem(t, 2); res := exp((y - t)*Pi*I)
```

wird untersucht, ob $-1 \le y < 1$ gilt, und anderenfalls in einem neuen Aufruf das Argument y durch $y - t$ ersetzt, wobei t eine ganze gerade Zahl mit $-1 \le y - t < 1$ ist. Dies entspricht der Anwendung der Identität $\mathrm{e}^{2\pi\,\mathrm{i}} = 1$.

```
exp(24/7*Pi*I);
```

$$\mathrm{e}^{-4/7\ \mathrm{i}\ \pi}$$

```
elif type(6*i, 'integer') or type(4*i, 'integer') then
      res := cos(-I*x) + sin(-I*x)*I
else res := 'exp'(x)
```

Die Verwandlung in $\cos(y\,\pi) + \mathrm{i}\,\sin(y\,\pi)$ wird nur dann vorgenommen, wenn $4\,y$ oder $6\,y$ eine ganze Zahl ist. Zusammen mit den Maple bekannten expliziten Werten der Winkelfunktionen an diesen Stellen ergibt sich das hier gezeigte Verhalten. Anderenfalls wird ein Funktionsausdruck mit dem Kopf `exp` zurückgegeben.

```
exp(23/6*Pi*I);
```

$$\frac{1}{2}\sqrt{3} - \frac{1}{2}\,\mathrm{i}$$

Die restlichen Zeilen realisieren die Funktionstransformationen $\exp(\operatorname{arcsinh}(t)) = t + \sqrt{1+t^2}$ usw., die sich aus den entsprechenden Funktionsdefinitionen ergibt.

```
exp(arcsinh(t));
```

$$t + \sqrt{1+t^2}$$

Das Beispiel zeigt, dass `exp` eine *Transformationsfunktion* ist und die symbolischen Fähigkeiten eines CAS eingesetzt werden können, um Polymorphie zu simulieren. Neue Funktionsbezeichner können während des Aufrufs durch Stringoperationen erzeugt werden und auf vorhandene Funktionsdefinition verweisen, wie wir beim Zusammensetzen von `evalf(exp(x))` zu `` `evalf/exp`(x) `` gesehen hatten.

Dieses Prinzip findet bei inerten MAPLE-Funktionen Anwendung. Schauen wir uns dazu noch einmal das Beispiel der Funktion `Factor` an, welche beim modularen Faktorisieren von Polynomen zu verwenden ist.

```
Factor(f) mod 2;
```

$$(x+1)^3$$

`Factor(f)` wird unverändert zurückgegeben und `mod` erkennt an der Struktur `mod(Factor(f),p)`, dass modular zu faktorisieren ist. Schauen wir uns mit `print(‘mod/Factor‘)` den Quellcode der speziellen modularen Faktorisierungsroutine `‘mod/Factor‘(f,p)` an, die hier aufgerufen wird.

```
proc(a)
local K, f, p;
option
‘Copyright (c) 1990 by the University of Waterloo. All rights reserved.‘;
    if nargs = 3 then K := args[2]; p := args[3]
    else K := NULL; p := args[2]
    end if;
    f := Factors(a, K) mod p;
    f[1]*convert(map(proc(x) x[1]^x[2] end proc, f[2]), ‘*‘) mod p
end proc
```

Der Funktionsrumpf enthält die Kombination `Factors(a, K) mod p`, die nach denselben Regeln in `‘mod/Factors‘(f,p)` umgesetzt wird und im Wesentlichen eine Liste von Paaren aud Primfaktor und Exponent zurückgibt, die in der letzten Zeile mit `convert` in ein Produkt verwandelt wird.

Die Nachteile des funktionalen Transformationskonzepts fallen sofort ins Auge:

1. Man hat mit jedem Funktionsaufruf eine Menge verschiedener Typinformationen zu ermitteln, womit jeder Funktionsaufruf relativ aufwändige Operationen anstößt. Deshalb ist es oft sinnvoll, bereits berechnete Funktionswerte zu speichern, wenn man weiß, dass sie sich nicht verändern.
2. Die Typanalyse ist bei vielen Funktionen von ähnlicher Bauart, so dass unnötig Code dupliziert wird.
3. Die so definierten Transformationsregeln sind relativ „starr“. Möchte man z. B. das Verhalten der `exp`-Funktion dahingehend ändern, dass sie bei rein imaginärem Argument *immer* die trigonometrische Darstellung verwendet, so müsste man den gesamten oben gegebenen Code kopieren, an den entsprechenden Stellen ändern und dann die (natürlich außerdem vor Überschreiben geschützte) `exp`-Funktion entsprechend redefinieren.

1.3 Das regelbasierte Transformationskonzept

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`, der *Simplifikator*, stellt die erforderlichen Routinen zur Mustererkennung und Unifikation bereit.

Der Simplifikator `Apply` ist also eine zweistellige Funktion, welche einen symbolischen Ausdruck A und einen Satz von *Transformationsregeln* übergeben bekommt und diese Regeln so lange auf A und die entstehenden Folgeausdrücke anwendet, bis keine Ersetzungen mehr möglich sind. Im

Gegensatz zum funktionalen Zugang sind hier der Simplifikator und die jeweils anzuwendenden Regelsätze voneinander getrennt, was es auf einfache Weise ermöglicht, Regelsätze zu ergänzen und für spezielle Zwecke zu modifizieren und anzupassen.

Damit enthält die Programmiersprache eines CAS neben funktionalen und imperativen auch Elemente einer logischen Programmiersprache. Wir werden uns in einem späteren Abschnitt genauer mit der Funktionsweise eines solchen Regelsystems vertraut machen. An dieser Stelle wollen wir uns anschauen, welche Regeln REDUCE zum Simplifizieren verschiedener Funktionen kennt. Die jeweiligen Regeln sind unter dem Funktionssymbol als Liste gespeichert und können mit der Funktion `showrules` ausgegeben werden:

```
showrules log;

{log(1) => 0,
 log(e) => 1,
 log(e^~x) => x,
 df(log(~x),~x) => 1/x,
 df(log(~x/~y),~z) => df(log(x),z) - df(log(y),z)}
```

Die ersten beiden Regeln ersetzen spezielle Kombinationen von fest vorgegebenen Symbolen durch andere. Solche Regeln werden auch als *spezielle Regeln* bezeichnet, denn in ihnen sind alle Bezeichner nur in ihrer literalen Bedeutung präsent.

Anders in den beiden letzten Regeln, in denen Bezeichner auch als *formale Parameter* vorkommen, die als Platzhalter für beliebige Teilausdrücke auftreten. So vereinfacht REDUCE etwa `log(exp(A))` zu A, egal wie der Teilausdruck A beschaffen ist. Der Bezeichner `e` steht dagegen für das Symbol e in seiner literalen Bedeutung. Wir haben also auch hier zwischen Bezeichnern in ihrer literalen Bedeutung (als Symbolvariable) (neben `e` sind das in obigen Regeln die Funktionssymbole `df` und `log`) und Bezeichnern als Wertcontainer (hier: als formale Parameter) zu unterscheiden. Regeln mit formalen Parametern werden auch als *allgemeine Regeln* bezeichnet.

Ein solches Regelsystem kann durchaus einen größeren Umfang erreichen:

```
showrules sin;

{sin(pi) => 0,
 sin(pi/2) => 1,
 sin(pi/3) => sqrt(3)/2,
 sin(pi/4) => sqrt(2)/2,
 sin(pi/6) => 1/2,
 sin((5*pi)/12) => sqrt(2)/4*(sqrt(3) + 1),
 sin(pi/12) => sqrt(2)/4*(sqrt(3) - 1),
 sin((~(~ x)*i)/~(~ y)) => i*sinh(x/y) when impart(y)=0,
 sin(atan(~u)) => u/sqrt(1 + u**2),
 sin(2*atan(~u)) => 2*u/(1 + u**2),
 sin(~n*atan(~u)) => sin((n - 2)*atan(u))*(1 - u**2)/(1 + u**2)
        + cos((n - 2)* atan(u))*2*u/(1 + u**2)  when fixp(n) and n>2,
 sin(acos(~u)) => sqrt(1 - u**2),
 sin(2*acos(~u)) => 2*u*sqrt(1 - u**2),
 sin(2*asin(~u)) => 2*u*sqrt(1 - u**2),
 sin(~n*acos(~u)) => sin((n - 2)*acos(u))*(2*u**2 - 1)
        + cos((n - 2)*acos(u))*2* u*sqrt(1 - u**2) when fixp(n) and n>2,
 sin(~n*asin(~u)) => sin((n - 2)*asin(u))*(1 - 2*u**2)
        + cos((n - 2)*asin(u))*2* u*sqrt(1 - u**2) when fixp(n) and n>2,
 sin((~x + ~(~ k)*pi)/~d) => sign(k/d)*cos(x/d)
                when x freeof pi and abs(k/d)=1/2,
 sin((~(~ w) + ~(~ k)*pi)/~(~ d)) =>
```

```
       (if evenp(fix(k/d)) then 1 else  - 1)*sin(( w + remainder(k,d)*pi)/d)
              when w freeof pi and ratnump(k/d) and abs(k/d)>=1,
sin((~(~ k)*pi)/~(~ d)) => sin((1 - k/d)*pi) when ratnump(k/d) and k/d>1/2,
sin(asin(~x)) => x,
df(sin(~x),~x) => cos(x)}
```

Manche der angegebenen Regeln sind noch konditional untersetzt, d. h. werden nur dann angewendet, wenn die Belegung der formalen mit aktuellen Parametern noch Zusatzvoraussetzungen erfüllt. Diese Effekte sind von regelorientierten Programmiersprachen wie etwa Prolog aber gut bekannt.

Konzeptionelle Anforderungen

1. Transformationen von Ausdrücken können über Regelanwendungen realisiert werden.

   Dazu muss das CAS eine *Mustererkennung* (pattern matcher) 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.

   In den Systemen werden Bezeichner, die als Platzhalter verwendet werden, besonders gekennzeichnet. Dies kann am einfachsten geschehen, indem diese Bezeichner in einer separaten Liste $(u_1, \ldots, u_n)$ zusammengefasst werden.

4. Im Gegensatz zu Funktionen kann die Anwendung einer passenden Regel konditional sein, d. h. vom Wert einer vorab zu berechnenden booleschen Wächterbedingung (guard clause) abhängen.

   Eine Regeldefinition besteht damit aus vier Teilen: `Rule(lhs, rhs, bool)`$(u_1, \ldots, u_n)$

MATHEMATICA	`lhs /; bool` $\rightarrow$ `rhs`
MAXIMA	`tellsimpafter(lhs, rhs, bool)`
MUPAD	`Rule(lhs, rhs, bool)`
REDUCE	`lhs => rhs when bool`

5. Regelanwendungen haben viel Ähnlichkeit mit der Auswertung von Ausdrücken. Insbesondere ist zwischen einfachen Regelanwendungen und iterierten Regelanwendungen zu unterscheiden.

6. Das Ergebnis hängt sowohl von der Reihenfolge der Regelanwendungen als auch von der Strategie der Mustererkennung ab.

Zusammenhang mit anderen CAS-Konzepten

Regelanwendungen haben viel Ähnlichkeit mit der Auswertung von Ausdrücken:

- Nach einmaliger Regelanwendungen kann es sein, dass dieselbe oder weitere Regeln anwendbar sind bzw. werden. Es ist also sinnvoll, Regeln iteriert anzuwenden.
- Iterierte Regelanwendungen bergen die Gefahr von Endlosschleifen in sich.
- Auswertungen können als Spezialfall von Regelanwendungen betrachtet werden, da die Einträge in der Symboltabelle als spezielles Regelwerk aufgefasst werden können.

Regelanwendungen können wie Wertzuweisungen lokal oder global vereinbart werden.

- Globale Regeldefinitionen ergänzen und modifizieren das automatische Transformationsverhalten des Systems und haben damit ähnliche Auswirkungen wie globale Wertzuweisungen.
- Lokale Regelanwendungen haben viel Ähnlichkeit mit der Substitutionsfunktion, indem sie das regelbasierte Transformationsverhalten auf einen einzelnen Ausdruck beschränken.
- Substitutionen und Wertzuweisungen können als spezielle Regelanwendungen formuliert werden. Einige CAS realisieren deshalb einen Teil dieser Funktionalität über Regeln.
- Das gleiche gilt für Funktionsdefinitionen. Diese können als spezielle Regeldefinitionen realisiert werden.

1.4 Simplifikation und mathematische Exaktheit

Wir hatten bereits gesehen, dass es in beiden Zugängen zur Simplifikationsproblematik einen

Kern allgemeingültiger Simplifikationen

gibt, die allen Simplifikationsstrategien gemeinsam sind und deshalb stets automatisch ausgeführt werden.

Dazu gehört zunächst einmal die Strategie, spezielle Werte von Funktionsausdrücken, sofern diese durch „einfachere“ Symbole exakt ausgedrückt werden können, durch diese zu ersetzen wie in diesen Maxima-Beispielen.

`sqrt(36)`	$\Rightarrow$	6
`sin(%pi/4)`	$\Rightarrow$	$\frac{1}{\sqrt{2}}$
`tan(%pi/6)`	$\Rightarrow$	$\frac{1}{\sqrt{3}}$
`asin(1)`	$\Rightarrow$	$\frac{\pi}{2}$

Dies trifft auch für kompliziertere Funktionsausdrücke zu, die auf „elementarere“ Funktionen zurückgeführt werden, in denen mehr oder weniger gut studierte spezielle mathematische Funktionen auftreten wie in diesen Maxima-Beispielen.

`gamma(1/2)`	$\Rightarrow$	$\sqrt{\pi}$
`integrate(exp(-x^2),x,0,inf)`	$\Rightarrow$	$\frac{\sqrt{\pi}}{2}$
`assume(y>0)$ integrate(exp(-x^2),x,0,y)`	$\Rightarrow$	$\frac{\sqrt{\pi}\,\mathrm{erf}(y)}{2}$
`sum(1/i^2,i,1,inf), simpsum`	$\Rightarrow$	$\pi^2/6$
`sum(1/i^7,i,1,inf), simpsum`	$\Rightarrow$	$\mathrm{zeta}(7)$

In den Beispielen treten als Transformationsergebnis die Gamma-Funktion $\Gamma(x)$, die Gaußsche Fehlerfunktion $\mathrm{erf}(x)$ sowie die Riemannsche Zeta-Funktion $\zeta(n)$ auf.

Weiterhin wird auch eine Reihe komplizierterer Umformungen von einigen der Systeme automatisch[1] ausgeführt wie z. B.:

$$\sqrt{24} = 2\sqrt{6}, \qquad \sqrt{2\sqrt{3}+4} = \sqrt{3}+1, \qquad \sqrt{11+6\sqrt{2}} + \sqrt{11-6\sqrt{2}} = 6\,.$$

Auch werden eindeutige Simplifikationen von Funktionsausdrücken ausgeführt wie etwa die folgenden von Maxima

[1] Maple automatisch, MuPAD erst mit `radsimp`, Mathematica erst mit `FullSimplify`, Maxima gar nicht.

`abs(abs(x))`	$\Rightarrow$	$\lvert x\rvert$
`tan(atan(x))`	$\Rightarrow$	x
`tan(asin(x))`	$\Rightarrow$	$\frac{x}{\sqrt{1-x^2}}$
`abs(-%pi*x)`	$\Rightarrow$	$\pi\,\lvert x\rvert$
`cos(-x)`	$\Rightarrow$	$\cos(x)$
`exp(3*log(x))`	$\Rightarrow$	x^3

Auf den ersten Blick mag es deshalb verwundern, dass folgende Ausdrücke von den meisten CAS[2] nicht vereinfacht werden:

`sqrt(x^2)`	$\Rightarrow$	$\sqrt{x^2}$
`log(exp(x))`	$\Rightarrow$	$\log(\exp(x))$
`arctan(tan(x))`	$\Rightarrow$	$\arctan(\tan(x))$

In jedem der drei Fälle würde der durchschnittliche Nutzer als Ergebnis wohl x erwarten. Für die letzte Beziehung ist das allerdings vollkommen falsch, wie ein Plot der Funktion mit Maxima unmittelbar zeigt:

```
plot2d(atan(tan(x)),[x,-5,5]);
```

Wir sehen, dass arctan nur im Intervall $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ die Umkehrfunktion von tan ist. Die korrekte Antwort lautet für $x \in \mathbb{R}$ also

$$\arctan(\tan(x)) = x - \left\lfloor \frac{x}{\pi} + \frac{1}{2} \right\rfloor \cdot \pi\,,$$

wobei $\lfloor a \rfloor$ für den ganzen Teil der Zahl $a \in \mathbb{R}$ steht.

Dass auch $\sqrt{x^2} = x$ mathematisch nicht exakt ist, dürfte bei einigem Nachdenken ebenfalls einsichtig sein und als Ergebnis der Simplifikation $|x|$ erwartet werden. Diese Antwort wird auch von Reduce und Maxima gegeben. Maple allerdings gibt nach expliziter Aufforderung

`simplify(sqrt(x^2))`	$\Rightarrow$	$\operatorname{csgn}(x)\,x$

zurück, obwohl Maple auch die Betragsfunktion kennt. Der Grund liegt darin, dass das nahe liegende Ergebnis $|x|$ nur für *reelle* Argumente korrekt ist, nicht dagegen für komplexe. Für komplexe Argumente ist die Wurzelfunktion mehrwertig, so dass $\sqrt{x^2} = \pm x$ eine korrekte Antwort wäre. Da man in diesem Fall oft vereinbart, dass der Wert der Wurzel der Hauptwert ist, also derjenige, dessen Realteil positiv ist, wird hier die *komplexe Vorzeichenfunktion* `csgn` verwendet. In diesem Kontext ist auch die Vereinfachung des Ergebnisses zu $|x|$, dem Betrag der komplexen Zahl x, fehlerhaft.

Für noch allgemeinere mathematische Strukturen, in denen Multiplikationen und deren Umkehrung definiert werden können, wie etwa Gruppen (quadratische Matrizen oder ähnliches), ist allerdings selbst diese Simplifikation nicht korrekt. MuPAD und Mathematica vereinfachen deshalb den Ausdruck auch unter `simplify` nicht.

> Die Exaktheit von Umformungen hängt auch von der mathematischen Theorie ab, innerhalb derer die entsprechenden Ausdrücke interpretiert werden.

So ist die dritte Beziehung $\log(\exp(x)) = x$ wegen der Monotonie der beteiligten Funktionen in der Theorie der reellwertigen Funktionen $f : \mathbb{R} \to \mathbb{R}$ richtig. Für komplexe Argumente kommt aber, ähnlich wie für die Wurzelfunktion, die Mehrdeutigkeit der Logarithmusfunktion ins Spiel.

[2] Maxima vereinfacht die ersten beiden Ausdrücke unzulässigerweise.

Die in der gymnasialen Oberstufe und im Grundkurs Analysis diskutierte Theorie der stetigen reellwertigen Funktionen wird bei unvorsichtigem Gebrauch von Symbolen schnell verlassen. Betrachten wir dazu die Ausdrücke

$$\sqrt{\frac{1}{x}} - \frac{1}{\sqrt{x}} \qquad \text{und} \qquad \sqrt{x\cdot y} - \sqrt{x}\cdot\sqrt{y}\,,$$

die für solche Argumente, für welche sie „sinnvoll“ definiert sind (also hier etwa für positive reelle x), zu null vereinfacht werden können. Für negative reelle Argumente verlassen wir allerdings die Theorie der stetigen reellwertigen Funktionen und haben nicht nur die Mehrdeutigkeit der Wurzelfunktion im Bereich der komplexen Zahlen zu berücksichtigen, sondern kollidieren mit anderen, wesentlich zentraleren Annahmen, wie etwa der automatischen Ersetzung von $\sqrt{-1}$ durch die imaginäre Einheit i. Setzen wir in obigen Ausdrücken $x = y = -1$ und führen diese Ersetzung aus, so erhalten wir im ersten Fall $\mathrm{i} - 1/\mathrm{i} = 2\,\mathrm{i}$ und im zweiten Fall $\sqrt{1} - \mathrm{i}^{\,2} = 2$. Solche Inkonsistenzen tief in komplexen Berechnungen versteckt können zu vollkommen falschen Resultaten führen, ohne dass der Grund dafür offensichtlich wird.

Aus ähnlichen Gründen sind übrigens auch die Transformationen der Logarithmusfunktion nach den bekannten Logarithmengesetzen mathematisch nicht allgemeingültig:

`log((-1)*(-1))=log(-1)+log(-1)` $\quad\Rightarrow\quad 0 = 2\,\mathrm{i}\,\pi$

> Mit Blick auf die Bedeutung polynomialer Strukturen im Design der CAS und dem Umstand, dass Nullstellen polynomialer Gleichungssysteme grundsätzlich komplexe Zahlen sind, interpretieren moderne CAS deshalb ihre Terme, soweit dies sinnvoll ist, in der Theorie der meromorphen Funktionen über den komplexen Zahlen.

Natürlich sind Simplifikationssysteme mit zu rigiden Annahmen für die meisten Anwendungszwecke untauglich, wenn sie derart simple Umformungen „aus haarspalterischen Gründen“ nicht oder nur nach gutem Zureden ausführen. Jedes der CAS muss deshalb für sich entscheiden, auf welchen Grundannahmen seine Simplifikationen sinnvollerweise basieren, um ein ausgewogenes Verhältnis zwischen mathematischer Exaktheit einerseits und Praktikabilität andererseits herzustellen.

In der folgenden Tabelle sind die Simplifikationsergebnisse der verschiedenen CAS (in der Grundeinstellung) auf einer Reihe von Beispielen zusammengestellt ($*$ bedeutet unsimplifiziert):

Ausdruck	Axiom	Maxima	Maple	Mma	MuPAD	Reduce
	2010	5.24	16	9.0	5.7	3.8
$\lvert -\pi\cdot x\rvert$	$\lvert\pi\cdot x\rvert$	$\pi\,\lvert x\rvert$	$\pi\,\lvert x\rvert$	$\pi\,\lvert x\rvert$	$\pi\,\lvert x\rvert$	$\pi\,\lvert x\rvert$
$\arctan(\tan(x))$	x	$*$	$*$	$*$	$*$	$*$
$\arctan(\tan(\frac{25}{7}\pi))$	$\frac{25}{7}\pi$	$-\frac{3}{7}\pi$	$-\frac{3}{7}\pi$	$-\frac{3}{7}\pi$	$-\frac{3}{7}\pi$	(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)
$\log(\exp(x))$	x	x	$*$	$*$	$*$	x
$\log(\exp(10\,\mathrm{i}))$	$10\,\mathrm{i}$	$10\,\mathrm{i}$	$*$	$10\,\mathrm{i} - 4\pi\,\mathrm{i}$	$10\,\mathrm{i} - 4\pi\,\mathrm{i}$	$10\,\mathrm{i}$

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

Tabelle 1: Simplifikationsverhalten der verschiedenen Systeme an ausgewählten Beispielen

Assume-Mechanismus

Derartigen Fragen der mathematischen Exaktheit widmen die großen CAS seit Mitte der 90er Jahre verstärkte Aufmerksamkeit. Eine natürliche Lösung ist die Einführung von **Annahmen**

(assumptions) zu einzelnen Bezeichnern. Hierfür haben in den letzten Jahren die meisten der großen CAS `assume`-Mechanismen eingeführt, mit denen es möglich ist, im Rahmen der durch das CAS vorgegebenen Grenzen einzelnen Bezeichnern einen gültigen Definitionsbereich als Eigenschaft zuzuordnen.

Maxima	`declare(x,real)`
Maple	`assume(x,real)`
Mathematica	`SetOptions[Assumptions -> x ∈ Reals]`
MuPAD	`assume(x,Type::Real)`

Tabelle 2: Variable x als reell deklarieren

Die folgenden Bemerkungen mögen zunächst die Schwierigkeiten des neuen Gegenstands umreißen:

- Die Probleme, mit welchen eine solche zusätzliche logische Schicht über der Menge der Bezeichner konfrontiert ist, umfasst die Probleme eines konsistenten Typsystems als Teilfrage.
- Annahmen wie etwa $x < y$ betreffen nicht nur einzelne Bezeichner, sondern Gruppen von Bezeichnern und sind nicht kanonisch einzelnen Bezeichnern als Eigenschaft zuzuordnen.
- Selbst für eine überschaubare Menge von erlaubten Annahmen über Bezeichner (vorgegebene Zahlbereichen, Ungleichungen) führt das Inferenzproblem, d. h. die Bestimmung von erlaubten Bereichen von Ausdrücken, welche diese Bezeichner enthalten, auf mathematisch und rechnerisch schwierige Probleme wie das Lösen von Ungleichungssystemen. Unter einigermaßen allgemeinen Annahmen kann bewiesen werden, dass das Inferenzproblem algorithmisch nicht lösbar ist.

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.

Mit speziellen Funktionen (Maxima: `facts`, `properties`, Maple: `about`, MuPAD: `getprop`) können die gültigen Eigenschaften ausgelesen werden.

Wie bei Regeldefinitionen können Eigenschaften global oder lokal zur Anwendung kommen. Maxima, Maple und MuPAD erlauben im Rahmen eines speziellen Assume-Mechanismus die globale Definition von Annahmen. In Mathematica werden globale Annahmen als Optionen in einer Systemvariablen `$Assumptions` gespeichert und können wie andere Optionen auch global mit `SetOptions[Assumptions -> ... ]` gesetzt und modifiziert werden. Maple und Mathematica erlauben darüber hinaus die Vereinbarung lokaler Annahmen zur Auswertung oder Vereinfachung von Ausdrücken.

Maple führt unter zusätzlichen Annahmen die oben beschriebenen mathematischen Umformungen automatisch aus. Ähnlich ist das Vorgehen in MuPAD oder Maxima.

In diesem Beispiel ist $x < 0$ und $y \in \mathbb{R}$ angenommen und mit Maxima angeschrieben.
Die meisten CAS vereinfachen die ersten beiden Ausdrücke, nicht jedoch den dritten, da für $x < 0$ das einfache Expandieren der Wurzel nicht korrekt ist. Maxima liefert die ebenfalls korrekte Vereinfachung $\sqrt{-x}\,\sqrt{-y} - \sqrt{x}\,\sqrt{y}$.

```
declare(y,real); assume(x<0);
facts(x); facts(y);

[0 > x] [kind(y, real)]

abs(-%pi*x); sqrt(x^2);
sqrt(x*y) - sqrt(x)*sqrt(y);
```

`assume` überschreibt in MAPLE und MUPAD die bisherigen Eigenschaften, während diese Funktion in MAXIMA kumulativ wirkt. Die (zusätzliche) Annahme $x > 0$ führt in diesem Fall zu einem inkonsistenten System von Eigenschaften, weshalb zunächst `forget(x<0)` eingegeben werden muss.

Neben globalen Annahmen sind in einigen CAS auch lokale Annahmen möglich. So vereinfacht MAPLE unter der lokal mit `assuming` zugeordneten Annahme $x < 0$.

```
sqrt(x^2) assuming x<0;
```

$$-x$$

In MATHEMATICA können Annahmen über die Option `Assumptions` für die Befehle `Simplify`, `FullSimplify`, `Refine` oder `FunctionExpand` angeschrieben werden.
Diese Optionen können in der Systemvariablen `$Assumptions` global gespeichert und durch das `Assuming`-Konstrukt auch in allgemeineren Kontexten lokal erweitert werden.

```
Simplify[√(x^2), Assumptions -> {x<0}]
```

$$-x$$

```
Simplify[√x √y - √(x y),
      Assumptions -> {x∈ Reals, y>0}]
```

$$0$$

```
Assuming[x<0, Simplify[Sqrt[x^2]]]
```

$$-x$$

In MAXIMA und REDUCE lässt sich außerdem die Strenge der mathematischen Umformungen durch verschiedene Schalter verändern. So kann man in REDUCE über den (allerdings nicht dokumentierten) Schalter `reduced` die klassischen Vereinfachungen von Wurzelsymbolen, die für komplexe Argumente zu fehlerhaften Ergebnissen führen können, zulassen. In MAXIMA können die entsprechenden Schalter wie `logexpand`, `radexpand` oder `triginverses` sogar drei Werte annehmen: `false` (keine Simplifikation), `true` (bedachtsame Simplifikation) oder `all` (ständige Simplifikation).

Die enge Verzahnung des Assume-Mechanismus mit dem Transformationskonzept ist nicht zufällig. 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.

1.5 Das allgemeine Simplifikationsproblem

Wir hatten gesehen, dass sich das allgemeine Simplifikationsproblem grob als **das Erkennen der semantischen Gleichwertigkeit syntaktisch verschiedener Ausdrücke** charakterisieren lässt. Wir wollen dies nun begrifflich genauer fassen.

Die Formulierung des Simplifikationsproblems

Ausdrücke in einem CAS können nach der Auflösung von Operatornotationen und anderen Ambiguitäten als geschachtelte Funktionsausdrücke in linearer, eindimensionaler Notation angesehen werden. Als *wohlgeformte Ausdrücke* (oder kurz: Ausdrücke) bezeichnen wir

- alle Bezeichner und Konstanten des Systems (atomare Ausdrücke) sowie
- Listen $[f, a, b, c, \ldots]$, deren Elemente selbst wohlgeformte Ausdrücke sind (zusammengesetzte Ausdrücke), wobei der Ausdruck entsprechend der LISP-Notationskonvention für den Funktionsausdruck $f(a, b, c, \ldots)$ steht.

Ausdrücke sind also rekursiv aus Konstanten, Bezeichnern und anderen Ausdrücken zusammengesetzt.

Als *Teilausdruck erster Ebene* eines Ausdrucks $A = [f, a, b, c, \ldots]$ bezeichnen wir die Ausdrücke $f, a, b, c, \ldots$, als *Teilausdrücke* diese Ausdrücke sowie deren Teilausdrücke. Ein Bezeichner *kommt in einem Ausdruck vor*, wenn er ein Teilausdruck dieses Ausdrucks ist.

Sind $x_1, \ldots, x_n$ Bezeichner, $A, U_1, \ldots, U_n$ Ausdrücke und die Bezeichner $x_i, i = 1, \ldots, n$, kommen in keinem der $U_j, j = 1, \ldots, n$, vor, so schreiben wir $A(x \vdash U)$ für den Ausdruck, der entsteht, wenn alle Vorkommen von x_i in A durch U_i ersetzt werden.

$\mathcal{E}$ sei die Menge der wohlgeformten Ausdrücke, also der Zeichenfolgen, welche ein CAS „versteht“. Die semantische Gleichwertigkeit solcher Ausdrücke kann als eine (nicht notwendig effektiv berechenbare) Äquivalenzrelation $\sim$ auf $\mathcal{E}$ verstanden werden. Diese Relation wird stets durch die mathematische Theorie vorgegeben, in der die Ausdrücke interpretiert werden. Wir hatten bereits gesehen, dass die semantische Gleichwertigkeit von Ausdrücken von dieser mathematischen Theorie abhängen kann. So gelten eine Reihe von Beziehungen zwischen Funktionen in der Theorie der stetigen reellwertigen Funktionen, sind aber in der Theorie der meromorphen Funktionen nicht mehr gültig, etwa die Vereinfachung $\sqrt{x^2} = |x|$.

Wir wollen im Weiteren die *Kontextfreiheit* dieser semantischen Relation $\sim$ voraussetzen, dass also das Ersetzen eines Teilausdrucks durch einen semantisch äquivalenten Teilausdruck in einem Gesamtausdruck zu einem semantisch äquivalenten Gesamtausdruck führt.

> Genauer fordern wir für alle x-freien Ausdrücke U, V mit $U \sim V$ und alle Ausdrücke $A(x), B(x)$ mit $A(x) \sim B(x)$, dass $A(x \vdash U) \sim B(x \vdash V)$ gilt.

Als *Simplifikator* bezeichnen wir eine effektiv berechenbare Funktion $S : \mathcal{E} \longrightarrow \mathcal{E}$, welche die beiden Bedingungen

$$(I) \quad S(S(t)) = S(t) \quad \text{(Idempotenz)} \qquad \text{und} \qquad (E) \quad S(t) \sim t \quad \text{(Äquivalenz)}$$

für alle $t \in \mathcal{E}$ erfüllt.

Wir hatten gesehen, dass in einem regelbasierten Transformationskonzept ein solcher Simplifikator als fortgesetzte Anwendung eines Arsenals von Transformationsregeln ausgeführt ist, wobei die Regeln immer wieder auf den entstehenden Zwischenausdruck angewendet werden, bis keine Ersetzungen mehr möglich sind. Sei dazu $\mathcal{R}$ ein (endliches) Regelsystem, also eine Menge von Regeln der Gestalt $\texttt{Rule}(L, R, B)(u)$.

Dabei bezeichnet $u = (u_1, \ldots, u_n)$ eine Liste von formalen Parametern und $L, R, B \in \mathcal{E}$ 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.

$\texttt{bool} : \mathcal{E} \to \{\texttt{true}, \texttt{false}, \texttt{fail}\}$ ist dabei eine boolesche Auswertefunktion auf der Menge der Ausdrücke. Die letzte Bedingung ist wegen der Kontextfreiheit von $\sim$ insbesondere dann erfüllt, wenn bereits $L(u) \sim R(u)$ in $\mathcal{E}$ gilt. Eine konditionale Restriktion B hat in diesem Fall keine Auswirkungen auf die semantische Korrektheit. Allerdings kann eine konditionale Restriktion für die Termination des Regelsystems erforderlich sein.

Die Regel $r = \texttt{Rule}(L, R, B)(u) \in \mathcal{R}$ 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. Zur Vermeidung von Namenskollisionen arbeiten wir mit den Regeln $L' = L(u \vdash x)$ 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)**.

Dann wird also $A = A''(x_0 \vdash L'(x \vdash U)) = A''(x_0 \vdash L(u \vdash U))$ im Ergebnis der Anwendung der Regel r durch $A^{(1)} = A''(x_0 \vdash R(u \vdash U))$ ersetzt. Wir schreiben auch $A \to_r A^{(1)}$.
Weniger formal gesprochen bedeutet die Anwendung einer Regel also, in einem zu untersuchenden Ausdruck A

- einen Teilausdruck $L'' = L(u \vdash U)$ der in der Regel spezifizierten Form zu finden,
- die formalen Parameter in L'' zu „matchen“,
- aus den Teilen den Ausdruck $R'' = R(u \vdash U)$ zusammenzubauen und
- schließlich L'' durch R'' zu ersetzen.

Der zugehörige Simplifikator $S = S(\mathcal{R}) : \mathcal{E} \longrightarrow \mathcal{E}$ ist der transitive Abschluss der durch die Regelmenge $\mathcal{R}$ definierten Ersetzungsrelationen[3].

Termination

S ist somit *effektiv*, wenn die Implementierung von $\mathcal{R}$ die folgenden beiden Bedingungen erfüllt:

(Matching) Es lässt sich effektiv entscheiden, ob es zu einem gegebenem Ausdruck A und einer 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.

Die erste Bedingung lässt sich offensichtlich durch entsprechendes Absuchen des Ausdrucks A unabhängig vom gegebenen Regelsystem erfüllen. Die einzige Schwierigkeit besteht darin, verschiedene Vorkommen desselben formalen Parameters in der linken Seite von r korrekt zu matchen. Dazu muss festgestellt werden, ob zwei Teilausdrücke U' und U'' von A syntaktisch übereinstimmen. Dies kann jedoch leicht durch eine `equal`-Funktion realisiert werden, die rekursiv die Teilausdrücke erster Ebene von U' und U'' vergleicht und bei atomaren Ausdrücken von der eindeutigen Darstellung in der Symboltabelle Gebrauch macht. Letzterer Vergleich wird auch als `eq`-Vergleich bezeichnet, da hier nur zwei Referenzen verglichen werden müssen. Aus Effizienzgründen wird man solche `eq`-Vergleiche auch für entsprechende Teilausdrücke von U' und U''durchführen, denn wenn es Referenzen auf denselben Ausdruck sind, kann der weitere Vergleich gespart werden.

Die zweite Bedingung dagegen hängt wesentlich vom Regelsystem $\mathcal{R}$ ab. Am einfachsten lässt sich die Termination sichern, wenn es eine (teilweise) Ordnungsrelation $\leq$ auf $\mathcal{E}$ gibt, bzgl. derer die rechten Seiten der Regeln „kleiner“ als die linken Seiten sind. In diesem Sinne ist dann auch $S(t)$ „einfacher“ als t, d. h. es gilt

$$S(t) \leq t \qquad \text{für alle } t \in \mathcal{E}. \tag{S}$$

Die Termination ist gewährleistet, wenn zusätzlich gilt:

(1) $\leq$ ist wohlfundiert.

(2) **Simplifikation**: Für jede Regel $r = \texttt{Rule}(L, R, B)(u) \in \mathcal{R}$ und jede zulässige Belegung $u \vdash U$ gilt $L(u \vdash U) > R(u \vdash U)$.

(3) **Monotonie**: Sind U_1, U_2 zwei x-freie Ausdrücke mit $U_1 > U_2$ und A ein weiterer Ausdruck, in welchem x vorkommt, so gilt $A(x \vdash U_1) > A(x \vdash U_2)$.

[3]Das ist nicht ganz korrekt, da das Ergebnis der fortgesetzten Regelanwendung auch im (hier vorausgesetzten) Terminationsfall von der Wahl der möglichen Matchings und passenden Regeln abhängt. Wir setzen deshalb stillschweigend eine feste Auswahlstrategie als gegeben voraus.

Die zweite und dritte Eigenschaft sichern, dass in einem elementaren Simplifikationsschritt die Vereinfachung zu einem „kleineren“ Ausdruck führt, die erste, dass eine solche Simplifikationskette nur endlich viele Schritte haben kann. In der Tat, ist $A \to A^{(1)}$ durch

$$A''(x_0 \vdash L(u \vdash U)) \to A''(x_0 \vdash R(u \vdash U))$$

beschrieben, so sichert (2), dass $U_1 = L(u \vdash U) > U_2 = R(u \vdash U)$ gilt, und (3) schließlich $A = A''(x_0 \vdash U_1) > A''(x_0 \vdash U_2) = A^{(1)}$.

Es reicht aus, dass es sich bei $>$ um eine teilweise Ordnung mit den Eigenschaften (1) – (3) handelt. Eine solche partielle Ordnung wird z. B. durch

$$\forall\, e_1, e_2 \in \mathcal{E} \quad e_1 > e_2 \;:\Longleftrightarrow\; l(e_1) > l(e_2)$$

für ein geeignetes *Kompliziertheitsmaß* $l : \mathcal{E} \longrightarrow \mathbb{N}$ beschrieben. In diesem Fall sind nur die Eigenschaften (2) und (3) zu prüfen. l kann z. B. die verwendeten Zeichen, die Klammern, die Additionen, den `LeafCount` usw. zählen. Obwohl Kompliziertheitsmaße auf $\mathcal{E}$ nur für sehr einfache Regelsysteme explizit angegeben werden können, spielen sie eine zentrale Rolle beim Beweis der Termination von Regelsystemen.

Allgemein ist die Termination von Regelsystemen schwer zu verifizieren und führt schnell zu (beweisbar) algorithmisch nicht lösbaren Problemen.

Kanonoische Simplifikatoren und starke Nulltester

Ein Simplifikationsprozess kann wesentlich von den gewählten Simplifikationsschritten abhängen, wenn für einen (Zwischen-)Ausdruck mehrere Simplifikationsmöglichkeiten und damit Simplifikationspfade existieren. Dies kann nicht nur Einfluss auf die Rechenzeit, sondern auch auf das Ergebnis selbst haben. Sinnvolle Aussagen dazu sind nur innerhalb eingeschränkterer Klassen von Ausdrücken möglich. Sei dazu $\mathcal{U} \subset \mathcal{E}$ eine solche Klasse von Ausdrücken.

Als *kanonischen Simplifikator* innerhalb $\mathcal{U}$ bezeichnet man einen Simplifikator $S : \mathcal{U} \longrightarrow \mathcal{U}$, der zusätzlich der Bedingung

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

genügt. Für einen solchen Simplifikator führen wegen (E) alle möglichen Simplifikationspfade zum selben Ergebnis. $S(t)$ bezeichnet man deshalb auch als ***die*** *kanonische Form* des Ausdrucks t innerhalb der Klasse $\mathcal{U}$. Ein solcher Simplifikator hat eine in Bezug auf unsere Ausgangsfrage starke Eigenschaft:

> Ein kanonischer Simplifikator erlaubt es, semantisch äquivalente Ausdrücke innerhalb einer Klasse $\mathcal{U}$ an Hand des (syntaktischen) Aussehens der entsprechenden kanonischen Form eindeutig zu identifizieren.

Ein solcher kanonischer Simplifikator kann deshalb insbesondere dazu verwendet werden, die semantische Äquivalenz von zwei gegebenen Ausdrücken zu prüfen, d.h. das **Identifikationsproblem** zu lösen: Zwei Ausdrücke aus der Klasse $\mathcal{U}$ sind genau dann äquivalent, wenn ihre kanonischen Formen literal (Zeichen für Zeichen) übereinstimmen.

Solche Simplifikatoren existieren nur für spezielle Klassen von Ausdrücken $\mathcal{E}$. Deshalb ist auch die folgende Abschwächung dieses Begriffs von Interesse. Nehmen wir an, dass $\mathcal{U}/\sim$, wie in den meisten Anwendungen in der Computeralgebra, die Struktur einer additiven Gruppe trägt, d. h. ein spezielles Symbol $0 \in \mathcal{U}$ für das Nullelement sowie eine Funktion $M : \mathcal{U} \times \mathcal{U} \longrightarrow \mathcal{U}$ existiert, welche die Differenz zweier Ausdrücke (effektiv syntaktisch) aufzuschreiben vermag. Letzteres bedeutet insbesondere, dass

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

gilt. Als *starken Nulltester* in der Klasse $\mathcal{U}$ bezeichnen wir dann einen Simplifikator $S : \mathcal{U} \longrightarrow \mathcal{U}$, 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[4].

Diese Forderung ist schwächer als die an einen kanonischen Simplifikator: Es kann sein, dass für zwei Ausdrücke $s, t \in \mathcal{U}$, die keine Nullausdrücke sind, zwar $s \sim t$ gilt, diese jedoch zu verschiedenen Normalformen simplifizieren.

Gleichwohl können wir mit einem solchen Normalform-Operator S ebenfalls das Identifikationsproblem innerhalb der Klasse $\mathcal{U}$ lösen, denn zwei **vorgegebene** Ausdrücke s und t sind offensichtlich genau dann semantisch äquivalent, wenn ihre 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

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

Oft wird auch eine solche Funktion, die nicht unbedingt über einen Simplifikator definiert sein muss, als *starker Nulltester* bezeichnet.

1.6 Simplifikation polynomialer und rationaler Ausdrücke

Wir hatten bei der Betrachtung der Fähigkeiten eines CAS bereits gesehen, dass sie besonders gut in der Lage sind, polynomiale und rationale Ausdrücke zu vereinfachen. Der Grund dafür ist die Tatsache, dass in der Menge dieser Ausdrücke kanonische bzw. normale Formen existieren.

Polynome in distributiver Darstellung

Betrachten wir dazu den Polynomring $S := R[x_1, \ldots, x_n]$ in den Variablen $X = (x_1, \ldots, x_n)$ über dem Grundring R. Wir hatten gesehen, dass solche Polynome $f \in S$ in expandierter Form als Summe von Monomen $f = \sum_a c_a X^a$ dargestellt werden, wobei $a = (a_1, \ldots, a_n) \in \mathbb{N}^n$ ein Multiindex ist und X^a kurz für $x_1^{a_1} \cdots x_n^{a_n}$ steht. Eine solche Darstellung kann in den meisten CAS durch die Funktion `expand` erzeugt werden. REDUCE verwendet sie standardmäßig für die Darstellung von Polynomen.

Diese Darstellung ist eindeutig, d. h. eine kanonische Form für Polynome $f \in S$, wenn für die Koeffizienten, also die Elemente aus R, eine solche kanonische Form existiert und die Reihenfolge der Summanden festgelegt ist. Zur Festlegung der Reihenfolge definiert man gewöhnlich eine Ordnung auf $T(X) = \{X^a : a \in \mathbb{N}^n\}$, dem *Monoid der Terme* in $(x_1, \ldots, x_n)$.

Als *distributive Darstellung* eines Polynoms $f \in S$ bzgl. einer solchen Ordnung bezeichnet man eine Darstellung $f = \sum_a c_a X^a$, in welcher die Summanden paarweise verschiedene Terme enthalten, diese in fallender Reihenfolge angeordnet sind und die einzelnen Koeffizienten in ihre kanonische Form gebracht wurden. In dieser Darstellung ist die Addition von Polynomen besonders effizient ausführbar. Ist die gewählte Ordnung darüber hinaus *monoton*, d. h. gilt

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

so kann man auch die Multiplikation recht effektiv ausführen, da dann beim gliedweisen Multiplizieren einer geordneten Summe mit einem Monom die Summanden geordnet bleiben. Wohlfundierte Ordnungen mit dieser Zusatzeigenschaft bezeichnet man als *Termordnungen*.

[4] Für $t = 0$ ergibt sich insbesondere $S(0) = 0$.

Zusammenfassend können wir folgenden Satz formulieren:

Satz 1 *Existiert auf R eine kanonischer Simplifikator, dann ist die distributive Darstellung von Polynomen $f \in S$ mit Koeffizienten in kanonischer Form ein kanonischer Simplifikator auf S.*
Hat R einen starken Nulltester, so auch S.

Der Beweis ist offensichtlich. Damit kann in Polynomringen über gängigen Grundbereichen wie den ganzen oder rationalen Zahlen, für die kanonische Formen existieren, effektiv gerechnet werden.

Polynome in rekursiver Darstellung

Als *rekursive Darstellung* bezeichnet man die Darstellung von Polynomen aus S als Polynome 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. Da die rekursive Darstellung für $n = 1$ mit der distributiven Darstellung zusammenfällt, führt die rekursive Natur des bewiesenen Satzes unmittelbar zu folgendem

Folgerung 1 *Existiert auf R eine kanonischer Simplifikator, dann ist auch die rekursive Darstellung von Polynomen $f \in R[x]$ mit Koeffizienten in kanonischer Form ein kanonischer Simplifikator auf $R[x]$.*
Hat R einen starken Nulltester, so auch $R[x]$.

Die rekursive Darstellung des in distributiver kanonischer Form gegebenen Polynoms

$$f := 2\,x^3\,y^2 + 3\,x^2\,y^2 + 5\,x^2\,y - x\,y^2 - 3\,x\,y + x - y - 2$$

als Element von $R[y][x]$ ist

$$(2\,y^2)\,x^3 + (3\,y^2 + 5\,y)\,x^2 + (-y^2 - 3\,y + 1)\,x + (-y - 2)$$

Im Gegensatz dazu führt die **faktorisierte Darstellung von Polynomen** nicht zu einer kanonischen Form, da eine Faktorzerlegung in $R[x_1, \ldots, x_n]$ immer nur eindeutig bis auf Einheiten aus R bestimmt werden kann. Auch für die Polynomoperationen ist diese Darstellung eher ungeeignet.

Rationale Funktionen

Wir hatten bereits mehrfach gesehen, dass CAS hervorragend in der Lage sind, auch rationale Funktionen zu vereinfachen. Allerdings ist es in einigen Beispielen besser, die spezielle Simplifikationsstrategie `normal` zu verwenden als den allgemeinen Ansatz `simplify`, wie folgendes Beispiel (MuPAD) demonstriert:

```
u:= a^3/((a-b)*(a-c)) + b^3/((b-c)*(b-a)) + c^3/((c-a)*(c-b));
```

$$\frac{a^3}{(a-b)\,(a-c)} + \frac{b^3}{(b-a)\,(b-c)} + \frac{c^3}{(c-a)\,(c-b)}$$

```
simplify(u);
```

$$\frac{a^3}{-a\,b - a, c + b, c + a^2} - \frac{b^3}{a, b - a, c + b, c - b^2} + \frac{c^3}{a, b - a, c - b, c + c^2}$$

```
normal(u);
```

$$a + b + c$$

`normal` verfährt nach folgendem einfachen Schema: Es bestimmt einen gemeinsamen Hauptnenner und überführt dann das entstehende Zählerpolynom in seine distributive Form. Auf diese Weise entsteht zwar keine kanonische Form wie im Fall der Polynome, da ja Zähler und Nenner nicht unbedingt teilerfremd sein müssen, allerdings eine Normalform, denn auf diese Weise werden Null-Ausdrücke sicher erkannt. Es gilt damit folgender

Satz 2 *Existiert auf R ein starker Nulltester, dann liefert die beschriebene Normalisierungsstrategie einen starken Nulltester auf dem Ring $R(x_1, \ldots, x_n)$ der rationalen Funktionen über R.*

Eine solche Darstellung bezeichnet man deshalb auch als **rationale Normalform**. Von ihr gelangt man zu einer kanonischen Form, indem man den gcd von Zähler und Nenner ausdividiert und dann die beiden Teile des Bruches noch geeignet normiert. Da die Berechnung des gcd aber im Vergleich zu den anderen arithmetischen Operationen aufwändiger sein kann, verzichtet z. B. REDUCE auf diese Berechnung, wenn nicht der Schalter `gcd` gesetzt ist.

```
(x^5-1)/(x^3-1);
```

$$\frac{x^5 - 1}{x^3 - 1}$$

```
on gcd; ws;
```

$$\frac{x^4 + x^3 + x^2 + x + 1}{x^2 + x + 1}$$

Die meisten CAS wandeln Eingaben nicht automatisch in eine rationale Normalform um, sondern nur auf spezielle Anforderung hin (wobei dann auch der gcd von Zähler und Nenner ausgeteilt wird) oder im Zuge von Funktionsaufrufen wie `factor`. In REDUCE (immer) oder MAXIMA können Teile des Ergebnisses in dieser „compilierten“ Form vorliegen, was bei der Suche nach Mustern im Rahmen des regelbasierten Transformationszugangs zu berücksichtigen ist. MAXIMA zeigt durch eine Marke `/R/` an, wenn ein Ausdruck oder Teile davon in dieser rationalen Normalform vorliegen.

Rationale Normalformen sind auf der Basis der distributiven Darstellung von Polynomen (MAPLE, MUPAD, MATHEMATICA) oder aber der rekursiven Darstellung (REDUCE, MAXIMA) möglich.

MAXIMA	`ratsimp(f)`
MAPLE	`normal(f)`
MATHEMATICA	`Together[f]`
MUPAD	`normal(f)`
REDUCE	standardmäßig

Tabelle 3: Rationale Normalform bilden

Verallgemeinerte Kerne

Auf Grund der guten Eigenschaften, welche die beschriebenen Simplifikationsverfahren polynomialer und rationaler Ausdrücke besitzen, werden sie auch auf Ausdrücke angewendet, die nur teilweise eine solche Struktur besitzen.

Betrachten wir etwa die Vereinfachung, die REDUCE automatisch an einem trigonometrischen Ausdruck vornimmt, und vergleichen ihn mit einem analogen polynomialen Ausdruck.
Die innere Struktur beider Ausdrücke ist ähnlich, einzig statt der Variablen a und b stehen verallgemeinerte Variablen (sin x) und (cos x) (die Lisp-Notation von sin(x) und cos(x)).

```
u1:=(sin(x)+cos(x))^3;
```

$$\cos(x)^3 + 3\,\cos(x)^2\,\sin(x) + 3\,\cos(x)\,\sin(x)^2 + \sin(x)^3$$

```
u2:=(a+b)^3;
```

$$a^3 + 3\,a^2\,b + 3\,a\,b^2 + b^3$$

```
lisp prettyprint prop 'u2;
((avalue scalar
  (!*sq
    ((((a . 3) . 1)
      ((a . 2) ((b . 1) . 3))
      ((a . 1) ((b . 2) . 3))
      ((b . 3) . 1))
      . 1)
    t)))
```

```
lisp prettyprint prop 'u1;
((avalue scalar
  (!*sq
    (((((cos x) . 3) . 1)
      (((cos x) . 2) (((sin x) . 1) . 3))
      (((cos x) . 1) (((sin x) . 2) . 3))
      (((sin x) . 3) . 1))
      . 1)
    t)))
```

Ein ähnliches Ergebnis liefert die Funktion `expand` bei „konservativeren“ Systemen wie z. B. MuPAD. Die interne Struktur als rationaler Ausdruck kann hier mit `rationalize` bestimmt werden.

```
u:=expand((sin(x)+cos(x))^3);
rationalize(u);
```

$$D3^3 + D4^3 + 3\,D3\,D4^2 + 3\,D3^2\,D4,$$
$$\{D3 = \cos(x), D4 = \sin(x)\}$$

In beiden Fällen entstehen polynomiale Ausdrücke, aber nicht in (freien) Variablen, sondern in allgemeineren Ausdrücken, wie in diesem Fall `sin(x)` und `cos(x)`, welche für die vorzunehmende Normalform-Expansion als algebraisch unabhängig angenommen werden. Die zwischen ihnen bestehende algebraische Relation $\sin(x)^2 + \cos(x)^2 = 1$ muss ggf. durch andere Simplifikationsmechanismen zur Wirkung gebracht werden. Solche allgemeineren Ausdrücke werden als *Kerne* bezeichnet.

In Maxima kann man statt mit `ratsimp` Ausdrücke mit solchen Kernen auch mit der Funktion `rat` in deren rationale Normalform umwandeln. Dabei merkt sich das System, dass es sich um eine rationale Normalform handelt. Die Marke /R/ in der Ausgabe weist darauf hin.

```
u:(sin(x)+cos(x)^3)$
showratvars(u);
```

$$[\cos(x), \sin(x)]$$

```
rat(u);
```

$$/R/ \quad \cos(x)^3 + 3\,\cos(x)^2\,\sin(x) + 3\,\cos(x)\,\sin(x)^2 + \sin(x)^3$$

CAS stellen allgemeine Ausdrücke dar als rationale Ausdrücke in verallgemeinerten Variablen, die als *Kerne* bezeichnet werden. Ein solcher Kern kann dabei ein Symbol oder ein Ausdruck mit einem nicht-arithmetischen Funktionssymbol als Kopf sein.

Maxima	`showratvars(f)`
Maple	`indets(f)`
Mathematica	`Variables[f]`
MuPAD	`indets(f,RatExpr)`
Reduce	`prop 'f;`

Tabelle 4: Kerne eines Ausdrucks bestimmen

Hier noch ein etwas komplizierteres Beispiel, das von den verschiedenen CAS auf unterschiedliche Weise „compiliert“ wird.

```
u:=(exp(x)*cos(x)+cos(x)*sin(x)^4+2*cos(x)^3*sin(x)^2+cos(x)^5)/
(x^2-x^2*exp(-2*x))-(exp(-x)*cos(x)+cos(x)*sin(x)^2+cos(x)^3)/
(x^2*exp(x)-x^2*exp(-x));
```

$$\frac{\cos(x)\,\sin(x)^4 + 2\,\cos(x)^3\,\sin(x)^2 + \cos(x)^5 + \mathrm{e}^x\,\cos(x)}{x^2 - x^2\,\mathrm{e}^{-2\,x}} - \frac{\cos(x)\,\sin(x)^2 + \cos(x)^3 + \mathrm{e}^{-x}\,\cos(x)}{x^2\,\mathrm{e}^x - x^2\,\mathrm{e}^{-x}}$$

Maple und MuPAD interpretieren den Ausdruck ähnlich

```
indets(u); // Maple
```

$$\{x,\, \cos(x),\, \sin(x),\, \exp(x),\, \exp(-x),\, \exp(-2\,x)\}$$

```
rationalize(u); // MuPAD
```

$$\frac{D8\,D9 + D9^5 + D9\,D10^4 + 2\,D9^3\,D10^2}{x^2 - x^2\,D7} - \frac{D6\,D9 + D9^3 + D9\,D10^2}{x^2\,D8 - x^2\,D6}$$

$$\{D9 = \cos(x),\, D8 = \exp(x),\, D10 = \sin(x),\, D6 = \exp(-x),\, D7 = \exp(-2\,x)\}$$

Die Wirkung von `normal` folgt der beobachteten Zerlegung. Insbesondere werden die Terme $\exp(x)$, $\exp(-x)$ und $\exp(-2\,x)$ als eigenständige Kerne behandelt. MuPAD fasst allerdings ab Version 4 $\exp(m\,x)\cdot\exp(n\,x)$ für $m, n \in \mathbb{Z}$ zu $\exp((m+n)\,x)$ automatisch zusammen.

```
v:=normal(u);
```

$$\frac{\left(\begin{array}{c} \mathrm{e}^{3\,x}\cos(x) - \cos(x) - \mathrm{e}^{x}\cos(x)^3 + \mathrm{e}^{2\,x}\cos(x)^5 + \\ 2\,\mathrm{e}^{2\,x}\cos(x)^3\sin(x)^2 - \mathrm{e}^{x}\cos(x)\sin(x)^2 + \mathrm{e}^{2\,x}\cos(x)\sin(x)^4 \end{array}\right)}{x^2\,\mathrm{e}^{2\,x} - x^2}$$

Ersetzt man in zusätzlich $\sin(x)^2$ durch $1 - \cos(x)^2$, so ergibt sich eine besonders einfache Form des Ausdrucks.

```
simplify(v);
```

$$\frac{e^x\cos(x) + \cos(x)}{x^2}$$

Maple belässt bei der Berechnung von `normal(u)` den Nenner in faktorisierter Form

$$x^2\left(\mathrm{e}^{-2x} - 1\right)\left(\mathrm{e}^{x} - \mathrm{e}^{-x}\right),$$

was die Normalformeigenschaft nicht zerstört, aber die Ähnlichkeit der beiden Faktoren $\mathrm{e}^{-2x} - 1$ und $\mathrm{e}^{x} - \mathrm{e}^{-x}$ nicht erkennt und damit zu einem komplizierteren Ausdruck führt als MuPAD.

`v1:=normal(u,expanded)` expandiert zusätzlich den Nenner, fasst aber auch die verschiedenen exp-Kerne zusammen, was die Zahl der Kerne reduziert und auf dasselbe Ergebnis wie MuPAD führt. `simplify(u)` kann deshalb die Kerne nicht alle eliminieren, sondern erst `simplify(v1)`.

Maxima, Reduce und Mathematica wenden sofort die Regel $\exp(n\,x) = \exp(x)^n$ für $n \in \mathbb{Z}$ an und reduzieren auf diese Weise die Zahl der Kerne.

```
showratvars(u1);
```

$$[x,\ \mathrm{e}^{x},\ \cos(x),\ \sin(x)]$$

Hier das Ergebnis der Berechnung der rationalen Normalform mit Maxima. An der Darstellung ist zu erkennen, dass intern eine rekursive Polynomdarstellung verwendet wird.

```
u1a:ratsimp(u1);
```

$$\frac{\left(\begin{array}{c} e^{2x}\cos(x)\sin^4(x) + \left(2\,e^{2x}\cos^3(x) - e^{x}\cos(x)\right)\sin^2(x) + \\ e^{2x}\cos^5(x) - e^{x}\cos^3(x) + \left(e^{3x} - 1\right)\cos(x) \end{array}\right)}{x^2\,e^{2x} - x^2}$$

Die Zahl der Kerne kann weiter reduziert werden, wenn $\sin^4(x)$ und $\sin^2(x)$ durch entsprechende Ausdrücke in $\cos(x)$ ersetzt werden. Eine solche Verringerung der Zahl der Kerne „verstärkt“ den rationalen Charakter des Ausdrucks. Bringt man diesen Ausdruck in die rationale Normalform, so vereinfacht er sich wie oben.

```
u1b:subst(sin(x)=sqrt(1-cos(x)^2),u1a);
ratsimp(u1b);
```

$$\frac{(e^x + 1)\,\cos x}{x^2}$$

MATHEMATICA behauptet zwar, dass $\exp(x)$ nicht mit zu den Kernen dieses Ausdrucks gehört, aber die Wirkung von `Together` zeigt, dass das nicht stimmt.

```
Variables[u]
```

```
{x, Cos[x], Sin[x]}
```

```
Together[u]
```

$$\frac{\cos(x)\left(-1+\mathrm{e}^{3x}-\mathrm{e}^{x}\cos(x)^2+\mathrm{e}^{2x}\cos(x)^4-\mathrm{e}^{x}\sin(x)^2+2\,\mathrm{e}^{2x}\cos(x)^2\sin(x)^2+\mathrm{e}^{2x}\sin(x)^4\right)}{(-1+\mathrm{e}^{2x})\,x^2}$$

Eine eigenständige, im Systemkern verankerte Simplifikationsschicht für polynomiale und rationale Ausdrücke in solchen Kernen spielt eine wichtige Rolle im Simplifikationsdesign aller CAS. Sowohl die Herstellung rationaler Normalformen und anschließende Anwendung weitergehender Vereinfachungen wie in obigem Beispiel als auch die gezielte Umformung anderer funktionaler Abhängigkeiten in rationale wie etwa beim Expandieren trigonometrischer Ausdrücke werden angewendet.

Kapitel 2

Algebraische Zahlen

Wir hatten im letzten Kapitel gesehen, dass sich Polynome mit Koeffizienten aus einem Grundbereich R, auf dem eine kanonische Form existiert, selbst stets in eine kanonische Form bringen lassen und so das Rechnen in diesen Strukturen besonders einfach ist. Dies gilt insbesondere für Polynome über den ganzen oder rationalen Zahlen, da für beide Bereiche kanonische Formen (die Darstellung ganzer Zahlen mit Vorzeichen im Dezimalsystem oder die rationaler Zahlen als unkürzbare Brüche mit positivem Nenner) existieren.

Treten als Koeffizienten Wurzelausdrücke wie $\sqrt{2}$ oder $\sqrt{2+\sqrt{3}}$ auf, so kann man die Frage nach einer kanonischen oder wenigstens normalen Form schon nicht mehr so einfach beantworten. Eine solche Darstellung müsste ja wenigstens die bereits früher betrachteten Identitäten

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

oder

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

erkennen, wenn die entsprechenden Wurzelausdrücke als Koeffizienten auftreten. Während dies in Maple automatisch erfolgt, sind die anderen Systeme nur mit viel gutem Zureden bereit, diese Simplifikation vorzunehmen. Selbst das Normalformproblem, also jeweils die Vereinfachung der Differenz beider Seiten der Identitäten zu 0, wird weder in MuPAD noch in Mathematica oder Maxima allein durch den Aufruf von `simplify` gelöst. MuPAD (`radsimp`) und Mathematica (`RootReduce`) stellen spezielle Simplifikationsroutinen für geschachtelte Wurzelausdrücke zur Verfügung, was bei Mathematica im stärkeren `FullSimplify` integriert ist. Maxima bietet eine solche Simplifikationsroutine `sqrtdenest` im Paket `sqdnst`. Mit Reduce und Axiom habe ich keinen direkten Weg gefunden, diese Aufgabe zu lösen.

Solche Wurzelausdrücke sind stets Nullstellen von Polynomen. Für $a = \sqrt{2\sqrt{3}+4}$ etwa rechnet man $(a^2-4)^2 = 4\cdot 3 = 12$ leicht nach, also ist a Nullstelle des Polynoms

$$P(x) = \left(x^2-4\right)^2 - 12 = x^4 - 8\,x^2 + 4,$$

welches sich in die algebraische Ersetzungsregel $a^4 \to 8\,a^2 - 4$ transformieren lässt. Wegen

$$P(x) = \left(x^2-2\,x-2\right)\cdot\left(x^2+2\,x-2\right)$$

ist a aber auch Nullstelle eines der beiden Faktorpolynome und damit existiert sogar eine algebraische Ersetzungsregel für a^2.

2.1 Die RootOf-Notation

Lösungen polynomialer Gleichungssysteme werden (allerdings nicht von MAXIMA) oft mit ROOTOF-Symbolen angeschrieben. So liefert etwa die REDUCE-Eingabe

```
solve(x^3+x+1,x);
```

die Antwort

$$\{x = \text{ROOTOF}(X^3 + X + 1, X)\}$$

Es wird ein Funktionsausdruck mit dem Funktionssymbol ROOTOF und dem zugehörigen Polynom als Parameter erzeugt, das stellvertretend für die einzelnen Nullstellen dieses Polynoms steht und von dem nichts weiter bekannt ist als die Gültigkeit der Ersetzungsregel $X^3 \Rightarrow -(X + 1)$.

Neben einer höheren Konsistenz der Darstellung spricht für ein solches Vorgehen auch die Tatsache, dass die Wurzeln einer Gleichung dritten Grades schon ein recht kompliziertes Aussehen haben können, aus dem sich die genannte Ersetzungsinformation nur noch schwer rekonstruieren lässt. Dies gilt erst recht für Nullstellen einer Gleichung vierten Grades. Für Nullstellen allgemeiner Gleichungen höheren Grades gibt es gar keine Darstellung in Radikalen, so dass für solche Zahlen *nur* auf eine ROOTOF-Darstellung zurückgegriffen werden kann.

Neben diesen praktischen Erwägungen ist eine solche Darstellung auch aus theoretischen Gründen günstig, denn es gilt der folgende

Satz 3 *Sei R ein Körper mit kanonischer Form und a eine Nullstelle des über R irreduziblen Polynoms $p(x) = x^k - r(x) \in R[x]$, wobei $\deg(r) < k$ sei.*

Stellt man polynomiale Ausdrücke in $R[a]$ in distributiver Form dar und wendet zusätzlich die algebraische Regel `a^k => r(a)` *an, so erhält man eine kanonische Form in $R[a]$.*

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

Lemma 1 *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 ein Polynom $p(x)$ kleinsten Grades. Dieses ist irreduzibel und jedes andere Polynom $q(x) \in P$ ist ein Vielfaches von $p(x)$.

Beweis: (des Lemmas) Division mit Rest in $R[x]$ ergibt

$$q(x) = s(x) \cdot p(x) + r(x)$$

mit $r = 0$ oder $\deg(r) < \deg(p)$. Wäre $r \neq 0$, so ergäbe sich wegen $q(a) = p(a) = 0$ auch $r(a) = 0$ und $\frac{1}{lc(r)} r(x) \in P$ im Gegensatz zur Annahme, dass $p(x) \in P$ minimalen Grad hat.

Wäre $p(x)$ reduzibel, so wäre a auch Nullstelle eines der Faktoren. Dieser würde also zu P gehören im Widerspruch zur Auswahl von $p(x)$. □

Beweis: (des Satzes) Mit dem beschriebenen Verfahren kann man jeden Ausdruck $U \in R[a]$ in der Form $U = \sum_{i=0}^{k-1} r_i a^i$ darstellen. Es bleibt zu zeigen, dass diese Darstellung eindeutig ist.

Wie früher auch genügt es zu zeigen, dass aus $0 = \sum_{i=0}^{k-1} r_i a^i$ bereits $r_i = 0$ für alle $i < k$ folgt. Wäre das nicht so, so wäre a eine Nullstelle des (nicht trivialen) Polynoms $q(x) = \sum_{i=0}^{k-1} r_i x^i$ vom Grad $deg\ q < k$ im Widerspruch zur Aussage des Lemmas. □

Es stellt sich heraus, dass $R[a]$ nicht nur ein Ring, sondern seinerseits wieder ein Körper ist, wie wir weiter unten noch genauer untersuchen werden. Wendet man das beschriebene Verfahren rekursiv

an, so erhält man eine für Rechnungen sehr brauchbare Darstellung für solche algebraischen Zahlen. Zur Einführung einer neuen algebraischen Zahl a muss man dazu jeweils „nur“ ein über dem bisherigen Bereich irreduzibles Polynom finden, dessen Nullstelle a ist, d.h. im wesentlichen in solchen Bereichen faktorisieren können.

RootOf-Symbole werden deshalb inzwischen von fast allen CAS verwendet. Der `Solve`-Operator von Maple und MuPAD etwa unterscheidet zwischen Nullstellen einzelner Gleichungen und Nullstellen von Gleichungssystemen. Für erstere wird angenommen, dass der Nutzer nach einer möglichst expliziten Lösung sucht und die RootOf-Notation erst ab Grad 4 (Maple), Grad 5 (MuPAD 1.4) bzw. Grad 3 (MuPAD 2.0) eingesetzt. Im zweiteren Fall werden in Maple standardmäßig alle nichtrationalen Nullstellen durch RootOf-Ausdrücke dargestellt, in MuPAD dagegen werden auch Quadratwurzel-Ausdrücke erzeugt. Dieses Verhalten kann man durch Setzen eines entsprechenden Parameters (`_EnvExplicit` in Maple, `MaxDegree` in MuPAD) ändern. Ähnlich gehen auch Reduce (RootOf ab Grad 3) und Mathematica (RootOf ab Grad 5, was man aber durch die Optionen $Cubics \to False$ und $Quartics \to False$ – dummerweise nicht im `Solve`-Kommando – abstellen kann) vor.

Auch in Maxima lassen sich Variablen mit `tellrat` als algebraische Zahlen vereinbaren und entsprechende Transformationen mit `ratsimp` ausführen, wenn zusätzlich der Kontext-Schalter `algebraic` aktiviert wird.

```
tellrat(a^5=2);
b:ratsimp(1/(a+2)),algebraic;
```

$$\frac{a^4 - 2\,a^3 + 4\,a^2 - 8\,a + 16}{34}$$

Die Probe

```
expand(b*(a+2));
```

$$\frac{a^5}{34} + \frac{16}{17}$$

(oder gleich `ratsimp(b*(a+2)),algebraic`) zeigt wegen $a^5 = 2$, dass richtig gerechnet wurde.

Allerdings ist in Maxima diese Möglichkeit des Rechnens mit algebraischen Zahlen nicht in das `solve`-Kommando integriert.

2.2 Mit algebraischen Zahlen rechnen

Wir hatten gesehen, dass mit Nullstellen von Polynomen, also algebraischen Zahlen, am besten gerechnet werden kann, wenn deren Minimalpolynom bekannt ist.

Oft sind algebraische Zahlen aber in einer Form angegeben, aus der sich dieses Minimalpolynom nicht unmittelbar ablesen lässt. Am einfachsten geht das noch bei Wurzelausdrücken:

Beispiele (über $k = \mathbb{Q}$):

$$\begin{aligned} \alpha_1 &= \sqrt{2} & p_1(x) &= x^2 - 2 \\ \alpha_2 &= \sqrt[3]{5} & p_2(x) &= x^3 - 5 \\ \alpha_3 &= \sqrt{1 - \sqrt{2}} & p_3(x) &= x^4 - 2\,x^2 - 1 \end{aligned}$$

Die Irreduzibilität von p_3 ist nicht ganz offensichtlich, kann aber leicht mit Maxima getestet werden:

```
factor(x^4-2*x^2-1);
```

$$x^4 - 2\,x^2 - 1$$

Analog erhalten wir für $\alpha_4 = \sqrt{9+4\sqrt{5}}$ ein Polynom $p_4(x) = x^4 - 18\,x^2 + 1$, dessen Nullstelle α_4 ist. Allerdings ist p_4 nicht irreduzibel

```
factor(x^4-18*x^2+1);
```

$$\left(x^2 - 4\,x - 1\right)\,\left(x^2 + 4\,x - 1\right)$$

und α_4 als Nullstelle des ersten Faktors in Wirklichkeit eine algebraische Zahl vom Grad 2. Das weiß auch MAXIMA nach Laden des Pakets `sqdnst`:

```
load(sqdnst);
sqrtdenest(sqrt(9+4*sqrt(5)));
```

$$\sqrt{5} + 2$$

Andere Beispiele algebraischer Zahlen hängen mit Winkelfunktionen spezieller Argumente zusammen. Aus der Schule bekannt sind die Werte von $\sin(x)$ und $\cos(x)$ für $x = \frac{\pi}{n}$ mit $n = 3, 4, 6$. MUPAD und MATHEMATICA kennen auch für $n = 5$ interessante Ausdrücke:

```
cos(PI/5), cos(2*PI/5);
```

$$\frac{1}{4}\,\sqrt{5} + \frac{1}{4}, \frac{1}{4}\,\sqrt{5} - \frac{1}{4}$$

In MAPLE können solche Darstellungen seit Version 7 mit `convert(cos(Pi/5),radical)` erzeugt werden. Damit lassen sich Radikaldarstellungen von deutlich mehr algebraischen Zahlen trigonometrischer Natur finden, etwa die von 3^o:

```
convert(cos(Pi/60),radical);
```

$$\left(-\frac{1}{16}\,\sqrt{2} + \frac{1}{8}\,\sqrt{5+\sqrt{5}} + \frac{1}{16}\,\sqrt{2}\sqrt{5}\right)\,\sqrt{3} + \frac{1}{8}\,\sqrt{5+\sqrt{5}} + \frac{1}{16}\,\sqrt{2} - \frac{1}{16}\,\sqrt{2}\sqrt{5}$$

Zur Bestimmung des charakteristischen Polynoms von $\cos\left(\frac{\pi}{5}\right)$ benutzen wir

$$\cos\left(5 \cdot \frac{\pi}{5}\right) = \cos(\pi) = -1.$$

Wenden wir die Mehrfachwinkelformeln für $\sin(n \cdot x)$ und $\cos(n \cdot x)$ auf $\cos(5\,x) + 1$ an, so erhalten wir ein Polynom in $\cos(x)$, das für $x = \frac{\pi}{5}$ verschwindet. Die entsprechende MAXIMA-Rechnung lautet

```
u:trigexpand(cos(5*x)+1);
p:expand(subst(sin(x)=sqrt(1-cos(x)^2),u));
```

$$16\,\cos(x)^5 - 20\,\cos(x)^3 + 5\,\cos(x) + 1\,.$$

Um diese Umformungen nicht jedes Mal nacheinander aufrufen zu müssen, definieren wir uns zwei Funktionen

```
sinexpand(u):= expand(subst(cos(x)=sqrt(1-sin(x)^2),trigexpand(u)));
cosexpand(u):= expand(subst(sin(x)=sqrt(1-cos(x)^2),trigexpand(u)));
```

und bekommen nun obige Darstellung durch einen einzigen Aufruf `cosexpand(cos(5*x)+1)`. Ein Regelsystem wäre an dieser Stelle natürlich besser, denn diese Funktionen nehmen die Vereinfachungen nur für Vielfache von x als Argument der trigonometrischen Funktionen vor und nicht für allgemeinere Kerne.

Weiter mit unserem Beispiel:

```
p:subst(cos(x)=z,p);
```

$$16\,z^5 - 20\,z^3 + 5\,z + 1$$

Diese Polynom ist allerdings noch nicht das Minimalpolynom.

```
factor(p);
```

$$(z+1)\,\left(4\,z^2 - 2\,z - 1\right)^2$$

Dasselbe Programm kann man für $\sin(\frac{\pi}{5})$ absolvieren:

```
p:subst(sin(x)=z,sinexpand(sin(5*x)));
```

$$16\,z^5 - 20\,z^3 + 5\,z$$

```
factor(p);
```

$$z\,\left(16\,z^4 - 20\,z^2 + 5\right)$$

Also ist der zweite Faktor $q = 16\,z^4 - 20\,z^2 + 5$ das Minimalpolynom von $\sin\left(\frac{\pi}{5}\right)$.

Der Ring der algebraischen Zahlen

Für die beiden algebraischen Zahlen $\sqrt{2}$ und $\sqrt{3}$ ist es nicht schwer, durch geeignete Umformungen ein Polynom zu finden, das deren Summe $a := \sqrt{2} + \sqrt{3}$ als Nullstelle hat:

$$a^2 = 5 + 2\sqrt{6} \Rightarrow (a^2-5)^2 = 24 \Rightarrow a^4 - 10\,a^2 + 1 = 0$$

a ist also Nullstelle des (in diesem Fall bereits irreduziblen) Polynoms $p(x) = x^4 - 10x^2 + 1$. Es stellt sich heraus, dass dies auch allgemein gilt:

Satz 4 *Die Summe und das Produkt zweier algebraischer Zahlen ist wieder eine algebraische Zahl.*

Vor dem allgemeinen Beweis des Satzes wollen wir die Aussage an einem etwas komplizierteren Beispiel studieren, in dem die auszuführenden Umformungen nicht so offensichtlich sind.

Betrachten wir dazu die beiden Zahlen $a = \sqrt{2}$ und $b = \sqrt[3]{5}$ und versuchen, ein Polynom $p := \sum_{i=0}^{n} r_i\,x^i$ zu konstruieren, das $c = a + b$ als Nullstelle hat, d.h. so dass $\sum_{i=0}^{n} r_i\,c^i = 0$ gilt. Um geeignete Koeffizienten r_i zu finden, berechnen wir zunächst die Potenzen $c^i = (a+b)^i$ als Ausdrücke in a und b. Dazu sind die binomischen Formeln sowie die Ersetzungen $\{a^2 \to 2, b^3 \to 5\}$ anzuwenden, die sich aus den charakteristischen Polynomen von a bzw. b unmittelbar ergeben. Eine Rechnung mit Maxima ergibt

```
tellrat(a^2=2,b^3=5);
l:makelist(ratsimp((a+b)^n),n,0,10),algebraic;
```

$$
\left[\begin{array}{l}
1,\\
b+a,\\
b^2+2\,a\,b+2,\\
3\,a\,b^2+6\,b+2\,a+5,\\
12\,b^2+(8\,a+5)\,b+20\,a+4,\\
(20\,a+5)\,b^2+(25\,a+20)\,b+4\,a+100,\\
(30\,a+60)\,b^2+(24\,a+150)\,b+200\,a+33,\\
(84\,a+210)\,b^2+(350\,a+81)\,b+183\,a+700,\\
(560\,a+249)\,b^2+(264\,a+1400)\,b+1120\,a+1416,\\
(513\,a+2520)\,b^2+(2520\,a+1944)\,b+4216\,a+3485,\\
(5040\,a+2970)\,b^2+(6160\,a+8525)\,b+6050\,a+21032
\end{array}\right]
$$

Wir sehen, dass sich alle Potenzen als Linearkombinationen von sechs Termen 1, a, b, b^2, $a\,b$, $a\,b^2$ darstellen lassen. Mehr als 6 solcher Ausdrücke sind dann sicher linear abhängig. Finden wir für unser Beispiel eine solche Abhängigkeitsrelation, indem wir eine Linearkombination von 7 verschiedenen Potenzen $(a+b)^i$ mit unbestimmten Koeffizienten aufstellen und diese dann so bestimmen, dass die 6 Koeffizienten vor 1, a, b, b^2, $a\,b$, $a\,b^2$ alle verschwinden. Leider bietet MAXIMA keine bequeme Möglichkeit, die Koeffizienten eines Polynoms zu extrahieren.

```
p:sum(concat(r,i)*x^i,i,0,6);
p1:ratsimp(subst(x=a+b,f)),algebraic;
```

$$
\begin{array}{l}
\left((30\,a+60)\,b^2+(24\,a+150)\,b+200\,a+33\right)\,r_6\\
+\left((20\,a+5)\,b^2+(25\,a+20)\,b+4\,a+100\right)\,r_5+\left(12\,b^2+(8\,a+5)\,b+20\,a+4\right)\,r_4\\
+\left(3\,a\,b^2+6\,b+2\,a+5\right)\,r_3+\left(b^2+2\,a\,b+2\right)\,r_2+(b+a)\,r_1+r_0
\end{array}
$$

Dieses Polynom muss zunächst in die distributive Normalform gebracht werden, um im zweiten Schritt rekursiv nach Potenzen von a und b zu sortieren. p_1 wird dabei als Polynom im Ring $((\mathbb{Q}[r_0,\ldots,r_6])\,[b])\,[a]$ interpretiert.

```
p2:expand(p1);
coeffs:map(lambda([u],makelist(coeff(u,b,j),j,0,2)),
     makelist(coeff(p2,a,i),i,0,1));
sys:flatten(coeffs);
```

$$
\begin{array}{l}
r_0+2r_2+5r_3+4r_4+100r_5+33r_6,\\
r_1+6r_3+5r_4+20r_5+150r_6,\\
r_2+12r_4+5r_5+60r_6,\\
r_1+2r_3+20r_4+4r_5+200r_6,\\
2r_2+8r_4+25r_5+24r_6,\\
3r_3+20r_5+30r_6
\end{array}
$$

Das ist ein homogenes lineares Gleichungssystem, dessen Lösung von einem Parameter abhängt.

```
vars:makelist(concat(r,i),i,0,6);
sol:solve(sys,vars);
```

$$\left[\ [r_0 = 17\,\%r_4, r_1 = -60\,\%r_4, r_2 = 12\,\%r_4, r_3 = -10\,\%r_4, r_4 = -6\,\%r_4, r_5 = 0, r_6 = \%r_4]\ \right]$$

Ein Polynom mit der Nullstelle $c = a + b$ lautet also

```
pp:subst(%r4=1,subst(sol[1],p));
```

$$x^6 \ - \ 6\,x^4 \ - \ 10\,x^3 \ + \ 12\,x^2 \ - \ 60\,x \ + \ 17$$

Testen wir schließlich noch, ob dieses Polynom irreduzibel ist:

```
factor(pp);
```

Damit wissen wir, dass es sich bei diesem Polynom sogar um das Minimalpolynom von $c = a + b$ handelt.

Beweis: Der Beweis des Satzes geht vollkommen analog. Sind $\alpha_1, \ldots, \alpha_s$ algebraische Zahlen über k vom Grad $d_1, \ldots, d_s$, so ergeben sich aus den entsprechenden Minimalpolynomen

$$p_i(x) = x^{d_i} - q_i(x)$$

Ersetzungsformeln

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

die es erlauben, jeden polynomialen Ausdruck aus $R := k[\alpha_1, \ldots, \alpha_s]$ in dessen **reduzierte Form** zu transformieren, d.h. ihn als Linearkombination der $D := d_1 \cdots d_s$ Produkte aus der Menge $T_{red} := \{\alpha_1^{j_1} \cdots \alpha_s^{j_s} \ : \ 0 \le j_i < d_i\}$ zu schreiben.

Ist nun $c \in R$ ein solcher polynomialer Ausdruck (also etwa Summe oder Produkt zweier algebraischer Zahlen), so kann man wie in obigem Beispiel eine nichttriviale lineare Abhängigkeitsrelation zwischen den Potenzen c^i, $i = 0, \ldots, D$ finden und erhält damit ein Polynom $p(x) \in k[x]$, dessen Nullstelle c ist. $\square$

Das gefundene Polynom muss allerdings nicht unbedingt das Minimalpolynom sein, da es in Faktoren zerfallen kann.

Aus dem im Beweis verwendeten konstruktiven Ansatz kann man sogar eine weitergehende Aussage ableiten:

Folgerung 2 *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.*

Allerdings bilden diese reduzierten Formen nur im Falle $s = 1$ eine kanonische Form (und natürlich nur, wenn die Elemente aus k in einer kanonischen Form darstellbar sind), da zwischen den verschiedenen algebraischen Zahlen $\alpha_1, \ldots, \alpha_s$ algebraische Abhängigkeitsrelationen bestehen können, die lineare Abhängigkeitsrelationen in der Menge T_{red} nach sich ziehen.

Beispiel: $\alpha_1 = \sqrt{2} + \sqrt{3},\ \alpha_2 = \sqrt{6}$. Es gilt $\alpha_1^2 - 2\,\alpha_2 - 5 = 0$.

Das Identifikationsproblem kann also für algebraische Zahlen so nicht gelöst werden.

Die Inverse einer algebraischen Zahl

Von einfachen algebraischen Zahlen wie etwa $1+\sqrt{2}$ oder $\sqrt{2}+\sqrt{3}$ wissen wir, dass man die jeweilige Inverse dazu recht einfach darstellen kann, wenn man mit einer auf geeignete Weise definierten *konjugierten* Zahl erweitert. So gilt etwa

$$\frac{1}{1+\sqrt{2}} = \frac{1-\sqrt{2}}{1-2} = \sqrt{2} - 1$$

und

$$\frac{1}{\sqrt{2}+\sqrt{3}} = \frac{\sqrt{3}-\sqrt{2}}{3-2} = \sqrt{3}-\sqrt{2}$$

Damit kann man in diesen Fällen auch die Inverse einer algebraischen Zahl (und damit beliebige Quotienten) als k-lineare Kombination der Produkte aus T_{red} darstellen. Es stellt sich die Frage, ob man auch kompliziertere rationale Ausdrücke mit algebraischen Zahlen auf ähnliche Weise vereinfachen kann. Wie sieht es z.B. mit

$$\frac{1}{\sqrt{2}+\sqrt{3}+\sqrt{5}}$$

aus?

Axiom liefert als Ergebnis sofort

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

Für die anderen Systeme sind dazu spezielle Funktionen, Schalter und/oder Pakete notwendig, so in Maple die Funktion `rationalize` aus der gleichnamigen Bibliothek und in MuPAD die Funktion `radsimp`. In Reduce muss der Schalter `rationalize` eingeschaltet sein. In Maxima benötigt man noch intimere Systemkenntnisse: Es ist die rationale Normalform im Kontext `algebraic` zu berechnen:

```
a:sqrt(2)+sqrt(3)+sqrt(5);
ratsimp(1/a), algebraic;
```

$$-\frac{\sqrt{2}\,\sqrt{3}\,\sqrt{5}-2\,\sqrt{3}-3\,\sqrt{2}}{12}$$

Untersuchen wir, auf welchem Wege sich eine solche Darstellung finden ließe. Wie man leicht ermittelt, hat $a=\sqrt{2}+\sqrt{3}+\sqrt{5}$ das charakteristische Polynom

$$p(x) = x^8 - 40\,x^6 + 352\,x^4 - 960\,x^2 + 576,$$

d.h. es gilt $a^8 - 40\,a^6 + 352\,a^4 - 960\,a^2 + 576 = 0$ oder

$$a^{-1} = \frac{-a^7 + 40\,a^5 - 352\,a^3 + 960\,a}{576}\,.$$

Ist das charakteristische Polynom bekannt, so ist es also nicht schwer, a^{-1} zu berechnen. Es werden einzig noch Ringoperationen benötigt, um den Ausdruck zu vereinfachen:

$$a^{-1} = \texttt{subst}\left(a=\sqrt{2}+\sqrt{3}+\sqrt{5}, \frac{-a^7+40a^5-352a^3+960a}{576}\right)$$

Das gilt auch allgemein:

Satz 5 *Ist $Q(x)\in k[x]$ das Minimalpolynom der algebraischen Zahl $a\neq 0$, so gilt*

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

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

$a \neq 0$ besitzt also stets eine Inverse, die sich polynomial durch Potenzen von a und damit als Linearkombination von Elementen aus T_{red} darstellen lässt, wenn wie oben $a \in k[\alpha_1, \ldots, \alpha_s]$ gilt. Aus dem Beweis ergibt sich, dass $Q(x)$ nicht unbedingt das Minimalpolynom sein muss, sondern wir nur von der Eigenschaft $Q(0) \neq 0$ Gebrauch machen.

Für eine algebraische Zahl a lässt sich eine Darstellung von $\frac{P(a)}{Q(a)}$ als k-lineare Kombination in $k[a]$ ebenfalls mit einem Ansatz mit unbestimmten Koeffizienten finden. Sei beispielsweise $a = \sqrt[5]{2}$ und $p = \frac{a}{a+2}$. Dann können wir den Ansatz $p = \sum_{i=0}^{4} r_i\, a^i$ mit unbestimmten Koeffizienten wählen und diese so bestimmen, dass $p{\cdot}(a+2) - a \in k[a]$ das Nullpolynom ist.

```
tellrat(a^5=2);
p:sum(concat(r,i)*a^i,i,0,4);
p0:ratsimp(p*(a+2)-a),algebraic;
p1:expand(p0); /* distributive Normalform */
sys:makelist(coeff(p1,a,i),i,0,4);
vars:makelist(concat(r,i),i,0,4);
sol:solve(sys,vars);
pp:subst(sol[1],p);
/* Probe */ ratsimp(pp*(a+2)),algebraic;
```

Für $k > 1$ kann es nichttriviale Linearkombinationen von Elementen aus T_{red} geben, die 0 ergeben. Beispiel: $a = \sqrt{2} + \sqrt{3},\ b = \sqrt{6}$. Es gilt $c = a^2 - 2\,b = 5$.

Die entsprechende Berechnung von $Q(x)$ für c mit MAXIMA aus den Minimalpolynomen $a^4 - 10\,a^2 + 1$ und $b^2 - 6$ führt zu folgendem Ergebnis:

```
tellrat(a^4-10*a^2+1=0,b^2=6);
p:sum(concat(r,i)*x^i,i,0,3);
p0:expand(ratsimp(subst(x=a^2-2*b,p))),algebraic;
coeffs:map(lambda([u],makelist(coeff(u,b,j),j,0,1)),
     makelist(coeff(p0,a,i),i,0,3));
sys:flatten(coeffs);
vars:makelist(concat(r,i),i,0,3);
sol:solve(sys,vars);
q:subst(%r4=1, subst(sol[1],p));
```

$$x^3 - 15\,x^2 - 21\,x + 355$$

```
factor(q);
```

$$(x^2 - 10\,x - 71)\,(x - 5)$$

c ist Nullstelle des zweiten Faktors, denn es gilt $c = 5$. Beide Polynome, $Q_1 = x^3 - 15\,x^2 - 21\,x + 355$ und $Q_2 = x - 5$, liefern dieselbe Inverse

$$c^{-1} = -\frac{1}{-5} = \frac{1}{5} \qquad \text{bzw.} \qquad c^{-1} = -\frac{1}{355}\left(c^2 - 15\,c - 21\right) = -\frac{-71}{355} = \frac{1}{5}\,.$$

Generell kann jedes Polynom $Q(x)$ mit $Q(c) = 0$ und nicht verschwindendem Absolutglied verwendet werden. Existiert so ein Polynom, so folgt zugleich $c \neq 0$. Wird nur ein Polynom $Q_0(x)$ gefunden, aus dem ein Faktor x abgespalten werden kann, so muss geprüft werden, ob vielleicht c eine nichttriviale Linearkombination aus T_{red} zu null ist. Das kann oft schon numerisch widerlegt werden. In diesem Fall hat Q_0 eine Darstellung $Q_0(x) = x^s\,Q_1(x)$ mit $Q_1(0) \neq 0$ und wegen $Q_1(c) = 0$ kann dieser Faktor verwendet werden. Für den Fall $c = 0$ ist c^{-1} natürlich nicht definiert.

Folgerung 3 *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.*

Natürlich ist es müßig, in jedem Fall erst das Minimalpolynom des jeweiligen Nenners zu bestimmen. Wir können stattdessen versuchen, diese Darstellung als k-lineare Kombination von Elementen aus T_{red} durch einen Ansatz mit unbestimmten Koeffizienten zu ermitteln.

Betrachten wir dazu den Ausdruck

$$u = \frac{\sqrt[3]{4} - 2\,\sqrt[3]{2} + 1}{\sqrt[3]{4} + 2\,\sqrt[3]{2} + 1} = \frac{a^2 - 2\,a + 1}{a^2 + 2\,a + 1}$$

mit $a = \sqrt[3]{2}$, d.h. $a^3 = 2$.

MAXIMA liefert

```
tellrat(a^3=2);
u:(a^2-2*a+1)/(a^2+2*a+1);
ratsimp(u),algebraic;
```

$$\frac{4\,a - 5}{3}$$

Offensichtlich ist $(a - 1)$ der „richtige“ Term, mit dem man u erweitern muss, um den Nenner in einen rationalen Ausdruck zu verwandeln.

Der Ansatz

$$u = \frac{a^2 - 2\,a + 1}{a^2 + 2\,a + 1} = a^2 r_2 + a r_1 + r_0$$

mit unbestimmten Koeffizienten r_2, r_1, r_0 liefert

```
f:(a^2*r2+a*r1+r0);
p0:expand(ratsimp((a^2-2*a+1)-(a^2+2*a+1)*f)),algebraic;
sys:makelist(coeff(p0,a,i),i,0,2);
sol:solve(sys,[r0,r1,r2]);
```

$$\left[\left[r_0 = -\frac{5}{3}, r_1 = \frac{4}{3}, r_2 = 0\right]\right]$$

und damit als Ergebnis unserer Vereinfachung

```
subst(sol[1],f);
```

$$\frac{4a - 5}{3}$$

Für den Fall $k = 1$, also die Hinzunahme einer einzelnen algebraischen Zahl α, gilt der folgende Satz

Satz 6 *Ist α 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 α 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.*

Zur Berechnung der reduzierten Form von A reicht es aus, ein d-dimensionales lineares Gleichungssystem mit Koeffizienten in k zu lösen.

Beweis: Sei $p(x) = x^d - q(x),\ \deg(q) < d$ das Minimalpolynom von α.

Wir müssen zum Beweis des Satzes nur zeigen, dass unser Verfahren zur Berechnung der reduzierten Form von Ausdrücken $A = \frac{P(\alpha)}{Q(\alpha)} \in R$, das wir oben an einem Beispiel demonstriert haben, stets zum Ziel führt. Es ist dazu das lineare Gleichungssystem in $r_0, \ldots, r_{d-1}$ zu lösen, das man aus

$$P(\alpha) = Q(\alpha) \cdot \left(\sum_{i=0}^{d-1} r_i \alpha^i \right)$$

nach (algebraischer) Ersetzung $\{\alpha^d \ \Rightarrow\ q(\alpha)\}$ durch Koeffizientenvergleich erhält.

Bei diesem Gleichungssystem handelt es sich um ein inhomogenes System von d linearen Gleichungen mit d Unbekannten. Dessen zugehöriges homogenes System

$$0 = Q(\alpha) \cdot \left(\sum_{i=0}^{d-1} r_i \alpha^i \right)$$

besitzt nur die triviale Lösung, da wegen $Q(\alpha) \neq 0$ jede nichttriviale Lösung zu einem Polynom $R(x) = \sum_{i=0}^{d-1} r_i x^i$ vom Grad $< d$ mit Nullstelle α führt.

Folglich hat (nach dem Lösungskriterium aus der linearen Algebra) das inhomogene System für jede rechte Seite genau eine Lösung, d.h. die Koeffizienten $r_0, \ldots, r_{d-1}$ lassen sich eindeutig bestimmen. □

Kapitel 3

Geometrische Sätze vom rationalen konstruktiven Typ

3.1 Grundlegende geometrische Zusammenhänge in koordinatengeometrischer Interpretation

Für die Visualisierung geometrischer Konfigurationen spielt die Darstellung durch Koordinaten eine zentrale Rolle. Im klassischen Zugang der ebenen Geometrie werden dazu Punkte P durch Koordinaten (p_x, p_y) im Punktraum $\mathbb{A}^2_K$ dargestellt mit Koordinatenwerten $p_x, p_y \in K$ aus einem Grundbereich K, von dem wir voraussetzen, dass dies ein Körper ist. Darstellungen anderer geometrischer Objekte können daraus abgeleitet werden. So können etwa Geraden durch zwei Punkte, ein Kreis durch Zentrum und Peripheriepunkt gegeben werden.

Eine kompakte Geradendarstellung ergibt sich durch Tripel $g = (g_1, g_2, g_3)$, welches für die Gerade $\{(p_x, p_y) : g_1 p_x + g_2 p_y + g_3 = 0\}$ steht. Ein solches Tripel bezeichnet man als *homogene Koordinaten* der Geraden g. Zueinander proportionale Tripel beschreiben dieselbe Gerade g – wir schreiben deshalb auch $g = (g_1 : g_2 : g_3)$ – und für (echte) Geraden dürfen g_1 und g_2 nicht gleichzeitig verschwinden. Es gibt genau eine „unechte“ Gerade, diese hat die homogenen Koordinaten $l_0 = (0 : 0 : 1)$. Wir sehen später, dass dies genau die *Ferngerade* der affinen Ebene ist.

Die wichtigsten geometrischen Eigenschaften von Punkten und Geraden spiegeln sich dann in den folgenden Formeln wider:

- A, B, C sind *kollinear*, d. h. liegen auf einer gemeinsamen Geraden g genau dann, wenn das homogene lineare Gleichungssystem

  $$\begin{aligned} g_1 a_x + g_2 a_y + g_3 &= 0 \\ g_1 b_x + g_2 b_y + g_3 &= 0 \\ g_1 c_x + g_2 c_y + g_3 &= 0 \end{aligned}$$

  eine nichttriviale Lösung in (g_1, g_2, g_3) besitzt, d. h. wenn

  $$\begin{pmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{pmatrix} = 0$$

  gilt.

- Analog gehen drei Geraden g, h, k durch einen gemeinsamen Punkt $P = (p_x, p_y)$ genau dann,

wenn das lineare Gleichungssystem

$$\begin{aligned} g_1 p_x + g_2 p_y + g_3 &= 0 \\ h_1 p_x + h_2 p_y + h_3 &= 0 \\ k_1 p_x + k_2 p_y + k_3 &= 0 \end{aligned}$$

eine Lösung in (p_x, p_y) besitzt. Das ist genau dann der Fall, wenn die zugehörige Koeffizientenmatrix denselben Rang hat wie die erweiterte Koeffizientenmatrix. Da dieser Rang höchstens 2 sein kann, muss also

$$\det \begin{pmatrix} g_1 & g_2 & g_3 \\ h_1 & h_2 & h_3 \\ k_1 & k_2 & k_3 \end{pmatrix} = 0 \tag{1}$$

gelten. Ist der Rang der Koeffizientenmatrix gleich 2, so hat das System dann eine eindeutig bestimmte Lösung, die etwa nach der Cramerschen Regel berechnet werden kann. Ist ihr Rang dagegen gleich 1, d. h. sind ihre drei Zeilen (g_1, g_2), (h_1, h_2) und (k_1, k_2) zueinander proportional, so sind die drei Geraden g, h, k zueinander parallel, schneiden sich also „im Unendlichen“ oder fallen zusammen.

Gleichung (1) bedeutet also, dass die drei Geraden g, h, k durch einen gemeinsamen Punkt gehen oder aber parallel zueinander sind. Geraden mit dieser Eigenschaft bezeichnen wir als *konkurrente Geraden*.

- Für die Parameter einer Geraden durch zwei Punkte A, B erhalten wir aus der Zwei-Punkte-Gleichung
$$(g_1, g_2, g_3) = (b_y - a_y,\ a_x - b_x,\ a_y b_x - a_x b_y)$$
- Zwei Geraden g, h sind parallel genau dann, wenn $g_1 h_2 - h_1 g_2 = 0$ gilt, d. h. ihre Normalenvektoren (g_1, g_2) und (h_1, h_2) zueinander parallel sind.
- Die Parameter der Parallelen h zu g durch einen Punkt P ergeben sich durch Adjustieren des Absolutglieds von g als
$$(h_1, h_2, h_3) = (g_1, g_2, -(g_1 p_x + g_2 p_y)) \ .$$
- Die Koordinaten des Schnittpunkts P zweier Geraden g, h berechnet sich als Lösung des entsprechenden Gleichungssystems nach der Cramerschen Regel zu
$$(p_x, p_y) = \left(\frac{g_2 h_3 - g_3 h_2}{d}, \frac{g_3 h_1 - g_1 h_3}{d} \right) \quad \text{mit} \quad d = g_1 h_2 - h_1 g_2$$
Die beiden Geraden schneiden sich stets dann, wenn $d \neq 0$ gilt, also die beiden Geraden nicht parallel sind.
- Ein Punkt P auf der Geraden $g = AB$ hat die Koordinaten
$$(p_x, p_y) = ((1-u)\, a_x + u\, b_x,\ (1-u)\, a_y + u\, b_y)$$
für ein geeignetes $u \in K$. Diese Beziehung ergibt sich aus der Vektorgleichung von Ortsvektoren
$$\overrightarrow{OP} = \overrightarrow{OA} + \overrightarrow{AP} = \overrightarrow{OA} + u\,\overrightarrow{AB} = \overrightarrow{OA} + u\,\left(\overrightarrow{OB} - \overrightarrow{OA}\right) = (1-u)\,\overrightarrow{OA} + u\,\overrightarrow{OB}$$
und gilt für alle Punkte $P \in g(AB)$, wobei u aus der Beziehung $\overrightarrow{AP} = u\,\overrightarrow{AB}$ eindeutig bestimmt ist. Wir bezeichnen $u = GP(A, B; P)$ als *Gleiterparameter*. Liegt P im Inneren der Strecke $\overline{AB}$, so gilt $0 < u < 1$, für Punkte P jenseits von B gilt $u > 1$ und für Punkte jenseits von A schließlich $u < 0$. Für den Mittelpunkt M der Strecke $\overline{AB}$ gilt $GP(A, B; M) = \frac{1}{2}$.
Als *Teilverhältnis* bezeichnet man die Größe $TV(A, B; P) = \frac{u}{1-u}$. Für den Mittelpunkt M der Strecke $\overline{AB}$ gilt $TV(A, B; M) = 1$.

Auch Begriffe aus der Euklidschen Geometrie lassen sich symbolisch durch entsprechende Koordinaten ausdrücken:

- So ergibt sich der Abstand zwischen den Punkten A, B aus der Formel

  $$d(A,B) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}.$$

  Da es sich dabei nicht um einen arithmetischen Ausdruck handelt, werden wir mit dem *Abstandsquadrat* $sqrdist(A,B) = d(A,B)^2$ arbeiten. Während $d(A,B)$ selbst nicht in K liegen muss, gilt $d(A,B)^2 \in K$.

- Zwei Geraden g, h sind orthogonal genau dann, wenn ihre Normalenvektoren (g_1, g_2) und (h_1, h_2) senkrecht aufeinander stehen, d. h. für das entsprechende Skalarprodukt

  $$g_1 h_1 + g_2 h_2 = 0$$

  gilt.

- Schließlich lässt sich das Lot h von P auf die Gerade g als

  $$(h_1, h_2, h_3) = (g_2, -g_1, g_1 p_y - g_2 p_x)$$

  ausdrücken.

Eine Reihe solcher Funktionen sind für das CAS Maxima in der Datei `geoprover.maxima` zusammengestellt.

Homogene Punktkoordinaten

Bei der Betrachtung der Konkurrenz dreier Geraden können wir statt nach Lösungen (p_x, p_y) des inhomogenen Gleichungssystems

$$\begin{aligned} g_1 p_x + g_2 p_y + g_3 &= 0 \\ h_1 p_x + h_2 p_y + h_3 &= 0 \\ k_1 p_x + k_2 p_y + k_3 &= 0 \end{aligned}$$

auch nach Lösungen (p_x, p_y, p_z) des homogenen Gleichungssystems

$$\begin{aligned} g_1 p_x + g_2 p_y + g_3 p_z &= 0 \\ h_1 p_x + h_2 p_y + h_3 p_z &= 0 \\ k_1 p_x + k_2 p_y + k_3 p_z &= 0 \end{aligned}$$

mit $p_z = 1$ fragen. Da Lösungen homogener Gleichungssysteme durch einen skalaren Faktor variiert werden können, reicht die Existenz von Lösungen mit $p_z \neq 0$ aus.

Solche Koordinaten $P = (p_x, p_y, p_z)$ bezeichnet man als *homogene* oder *projektive Punktkoordinaten*. Sie sind — wie die homogenen Geradenkoordinaten — nur bis auf einen skalaren Faktor verschieden null eindeutig bestimmt, wobei den affinen Koordinaten (p_x, p_y) die projektiven Koordinaten $(p_x, p_y, 1)$ entsprechen. P liegt auf der Geraden g genau dann, wenn

$$g_1 p_x + g_2 p_y + g_3 p_z = 0$$

gilt. An dieser Formel sieht man schon, dass Punkt- und Geradenkoordinaten in zueinander dualer Weise eingehen. Eine solche Dualität von Punkten und Geraden kann in vielen in Sätzen der projektiven Geometrie beobachtet werden. Die Punkte, für deren homogene Koordinate $p_z = 0$ gilt, liegen auf der Ferngeraden $l_0 = (0 : 0 : 1)$.

Wir bezeichnen diese Erweiterung der affinen Ebene $\mathbb{A}^2$ um die Punkte der Ferngeraden als projektive Ebene $\mathbb{P}^2$. Die weiter oben untersuchten geometrischen Beziehungen lassen sich nennerfrei durch Skalar-, Vektor- und Spatproduktoperationen in K^3 beschreiben, womit sich gewisse Berechnungsanomalien, die sich sonst aus möglichen Divisionen durch null ergeben, systematisch vermieden werden können.

- A, B, C in homogenen Punktkoordinaten sind kollinear $\iff$ $\det \begin{pmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{pmatrix} = 0$

  Notation: $\mathrm{sp}(A, B, C) = 0$ (Spatprodukt)

- Analog sind drei Geraden g, h, k konkurrent $\iff$ $\mathrm{sp}(g, h, k) = 0$

- Punkt P und Gerade g sind inzident $\iff$ $p_x g_1 + p_y g_2 + p_z g_3 = 0$

  Notation: $P * g = 0$ (Skalarprodukt)

- Für den Schnittpunkt P zweier Geraden g, h können wir die frühere Formel nennerfrei interpretieren:

$$\begin{aligned} P &= (g_2h_3 - g_3h_2, g_3h_1 - g_1h_3, g_1h_2 - g_2h_1) = \left(\begin{vmatrix} g_2 & g_3 \\ h_2 & h_3 \end{vmatrix}, \begin{vmatrix} g_3 & g_1 \\ h_3 & h_1 \end{vmatrix}, \begin{vmatrix} g_1 & g_2 \\ h_1 & h_2 \end{vmatrix} \right) \\ &= (g_1, g_2, g_3) \times (h_1, h_2, h_3) = g \times h \end{aligned}$$

  Das sind genau die Koordinaten des Vektorprodukts zweier Vektoren im K^3.

- Die Gleichung einer Geraden durch zwei in homogenen Koordinaten gegebene (verschiedene) Punkte A, B lautet analog

$$g = (a_y b_z - a_z b_y, a_z b_x - a_x b_z, a_x b_y - a_y b_x) = A \times B$$

- A, B, C sind genau dann kollinear, wenn A inzident zur Geraden durch B und C ist, also $A * (B \times C) = 0$ gilt. Dies stimmt wegen des bekannten Zusammenhangs $\mathrm{sp}(A, B, C) = A * (B \times C)$ zwischen Spat-, Vektor- und Skalarprodukt im K^3 mit obiger Determinantenformel überein.

Homogene Punkt- bzw. Geradenkoordinaten sind genau dann *nicht zulässig*, wenn sich $(0 : 0 : 0)$ ergibt. Aus der Formel für die Koordinaten des Schnittpunkts zweier Geraden g, h ist ersichtlich, dass sich nicht zulässige Koordinaten genau dann ergeben, wenn die Koordinaten von g und h proportional sind, d. h. wenn g und h identisch sind.

Analog ergeben sich nicht zulässige Geradenkoordinaten für die Verbindungsgerade zweier Punkte A und B genau dann, wenn $A = B$ gilt.

Auch Parallelität kann man ausdrücken, wenn berücksichtigt wird, dass diese Größen nicht projektiv invariant sind, d. h. bei ihrer Definition die Ferngerade $l_0 = (0 : 0 : 1)$ eine Rolle spielen muss:

- Zwei Geraden g, h sind genau dann parallel, wenn sie sich auf der Ferngeraden l_0 schneiden, d. h. $\mathrm{sp}(g, h, l_0) = 0$ gilt. Das stimmt mit unserer weiter oben hergeleiteten Formel überein.

  Die Koordinaten des Fernpunkts F_g der Geraden g ergeben sich aus der Formel $F_g = g \times l_0 = (-g_2 : g_1 : 0)$.

- Die Gerade h durch P, die parallel zu g verläuft, ergibt sich als Verbindung des Fernpunkts $F_g = (-g_2 : g_1 : 0)$ der Geraden g und P zu $h = P \times F_g$.

- Alle Senkrechten zur Geraden g gehen durch den gemeinsamen Fernpunkt $O_g = (g_1 : g_2 : 0)$, so dass sich die Senkrechte h zu g durch P als

  $$h = P \times O_g = (-p_z g_2 : p_z g_1 : p_x g_2 - p_y g_1)$$

  in Übereinstimmung mit der früher gefundenen Darstellung ergibt. O_g wird auch als *Orthogonalpunkt* von g bezeichnet.

Mit Parallelen kann man aus einem Standardframe ein ganzes affines Koordinatensystem gewinnen. Als *Standardframe* bezeichnet man ein Punkte-Quadrupel (E_0, E_1, E_2, E_3) der projektiven Ebene, von denen keine drei auf einer Geraden liegen. Als Ursprung E_0, Fernpunkt E_1 der x-Achse, Fernpunkt E_2 der y-Achse und Einheitspunkt E_3 bestimmen diese vier Punkte ein (affinen) Koordinatensystem so dass $E_0 = (0 : 0 : 1)$, $E_1 = (1 : 0 : 0)$, $E_2 = (0 : 1 : 0)$, $E_3 = (1 : 1 : 1)$ gilt. Der Einheitspunkt E_3 mit den (affinen) Koordinaten $(1, 1)$ bestimmt die beiden Koordinateneinheiten, da die Parallelen zur x- bzw. y-Achse durch E_3 die y- bzw. x-Achse in den Koordinaten-Einheiten schneiden.

Die hier beschriebenen Vorteile homogener Koordinaten veranlassen einige Designer von DGS, diese zur Darstellung von Punkten zu verwenden. Mit Blick auf die Abweichungen von den aus der Schule bekannten Notationen der analytischen Geometrie sowie der Probleme vor allem der Darstellung von Teilverhältnissen werden wir im Weiteren mit affinen Punktkoordinaten arbeiten.

3.2 Zur Algorithmisierung geometrischer Konstruktionen. Analytische Geometrie mit dem Computer

Wir können auf der Basis der im Abschnitt 3.1 hergeleiteten Beziehungen nun in einer klassischen Programmiersprache (die an dieser Stelle noch nicht über die Fähigkeit zur Symbolverarbeitung verfügen muss) Funktionen schreiben, die in der Lage sind, Beziehungen in durch konkrete Koordinatenwerte vorgegebenen geometrischen Konfigurationen zu überprüfen oder gesuchte Größen auszurechnen. Entsprechende Funktionen sind auch die Grundlage für dynamische Geometriesysteme, mit denen entsprechende Konfigurationen grafisch dargestellt werden können.

> *Anmerkung:* Während es im Abschnitt 3.1 zunächst um Beschreibungen in Koordinatenschreibweise über einem Grundbereich K ging, die eine *geometrische Interpretation* zulassen, sind nun Beschreibungen gefragt, die sich auch effektiv berechnen lassen. Hierfür ist es erforderlich, zunächst den Grundbereich (ggf.) auf einen effektiv berechenbaren Teilbereich $k \subset K$ einzuschränken.

In einer objektorientierten Sprache können wir die elementaren geometrischen Objekte Punkt und Gerade als *Klassen* `Point` und `Line` umsetzen und Punkte $P(p_x, p_y)$ bzw. Geraden $g = \{(p_x, p_y) : g_1 p_x + g_2 p_y + g_3 = 0\}$ als *Instanzen* dieser Klassen definieren.

> *Anmerkung:* Für praktische Zwecke des Designs eines DGS wird man dabei etwa den numerischen Datentyp `double` für den Bereich k der Koordinatenwerte verwenden. Wir verlassen damit allerdings den Bereich des exakten Rechnens in Richtung der Computerzahlen, für die nicht einmal festgestellt werden kann, ob ein Ausdruck *genau* zu null auswertet, ob also ein Punkt P *genau* auf einer Geraden AB liegt.
>
> Dies kann vermieden werden, wenn man etwa über dem Grundbereich $k = \mathbb{Q}$ rationaler Zahlen in exakter Bruchdarstellung rechnet. Dies hat aber andere fundamentale Probleme: In einer solchen Koordinatengeometrie gibt es zum Beispiel keine gleichseitigen Dreiecke.

Wir wollen deshalb über den Grundkörper k zusätzlich voraussetzen, dass ein boolesches Prädikat `boolean iszero(a)` für $a \in k$ definiert ist[1].

[1] Für $k =$ `double` wird man etwa – jenseits aller strenger Semantik – `iszero(a)` $=$ `is` $\left(|a| < 10^{-3}\right)$ definieren.

Geometrische Grundkonstruktionen können wir in diesem Kontext als Funktionen auffassen, die aus gegebenen Objekten neue konstruieren.

1) Die Gerade durch zwei Punkte P und Q

```
public static Line pp_line(Point p, Point q) {
  return new Line(q.y-p.y, p.x-q.x, p.y*q.x-p.x*q.y);
}
```

P und Q sind dabei als formale Parameter vom Typ `Point` Container für die aktuellen Koordinaten, der Rückgabewert der Funktion vom Typ `Line` der Container für die berechneten Koodinaten des davon abhängenden Objekts.

2) Analog können wir den Schnittpunkt zweier Geraden berechnen, wobei die zu definierende Funktion mit einer Ausnahme abbricht, wenn kein bzw. kein eindeutig bestimmter Schnittpunkt existiert.

```
public static Point intersection_point(Line g, Line h) {
  double d = g.a*h.b-g.b*h.a;
  if (iszero(d)) throw new GeoException("Geraden sind parallel");
  return new Point((g.b*h.c - g.c*h.b)/d,(g.c*h.a - g.a*h.c)/d);
}
```

Auch hier sind g und h formale Parameter, diesmal vom Typ `Line`.

3) Für das Lot l von einem Punkt P auf eine Gerade g erhalten wir analog

```
public static Line ortho_line(Point p, Line g) {
  return new Line(g.b, -g.a, g.a*p.y - g.b*p.x);
}
```

und für die Parallele zu einer Geraden g durch einen Punkt P

```
public static Line par_line(Point p, Line g) {
  return new Line(g.a, g.b, -(g.a*p.x + g.b*p.y));
}
```

Das *Abstandsquadrat* ergibt sich schließlich als

```
public static double sqrdist(Point p, Point q) {
  return (p.x-q.x)*(p.x-q.x) + (p.y-q.y)*(p.y-q.y);
}
```

4) Neben freien Punkten, die mit dem Punktkonstruktor erzeugt werden können, sind auch Punkte auf vorgegebenen Geraden (*Geradengleiter*) oder Kreisen (*Kreisgleiter*) interessant. Einen Geradengleiter auf einer durch zwei Punkte gegebenen Geraden kann man etwa durch ein variables Teilverhältnis festlegen:

```
public static Point varpoint(Point P, Point Q, double u) {
  return new Point((1-u)*p.x+u*q.x,(1-u)*p.y+u*q.y);
}
```

Insbesondere liefert

```
Point midpoint(Point P,Point Q) { return varpoint(P,Q,1/2); }
```

den Mittelpunkt der Strecke PQ.

5) Komplexere geometrische Konstruktionen (Makros) können aus nacheinander ausgeführten Grundkonstruktionen zusammengesetzt werden. Dem entsprechen auf der Seite der Programmiersprachen zusammengesetzte Funktionen. So findet man etwa den Fußpunkt des Lots vom Punkt P auf die Gerade a als

```
public static Point pedalpoint(Point P, Line a) {
  return intersection_point(ortho_line(P,a),a);
}
```

6) Schließlich kann man testen, ob für gewisse Konfigurationen geometrische Bedingungen erfüllt sind. So kann man etwa testen, ob zwei gegebene Geraden g und h parallel oder orthogonal sind, indem man prüft, ob $g_1h_2 - g_2h_1$ bzw. $g_1h_1 + g_2h_2$ verschwindet, oder ob ein Punkt P auf einer Geraden g liegt. Entsprechende Funktionen haben folgende Spezifikation:

```
public static boolean is_parallel(Line g, Line h) {
  return iszero(g.a*h.b-h.a*g.b);
}
```

bzw.

```
public static boolean is_orthogonal(Line g, Line h) {
  return iszero(g.a*h.a+g.b*h.b);
}

public static boolean is_point_on_line(Point P, Line g) {
  return iszero(g.a*P.x+g.b*P.y+g.c);
}
```

Auch kompliziertere Bedingungen, die wir im letzten Paragraph hergeleitet hatten, kann man auf diese Weise prüfen, so z. B., ob drei gegebene Punkte P, Q, R *kollinear* oder drei gegebene Geraden a, b, c *konkurrent* sind. Die entsprechenden Funktionen `is_collinear` und `is_concurrent` lassen sich leicht angeben.

Mit diesem Arsenal kann man die Koordinaten auch komplizierterer geometrischer Objekte bestimmen und entsprechende geometrische Sätze in konkreten Konfigurationen überprüfen.

Beispiel 1: Der Satz vom Schnittpunkt der Mittelsenkrechten. Die Funktion

```
static boolean CircumCenter_Test1(Point A, Point B, Point C) {
  return is_concurrent(p_bisector(A,B), p_bisector(B,C), p_bisector(C,A));
}
```

prüft, ob für ein Dreieck, das durch seine Eckpunkt(koordinaten) A, B, C gegeben ist, die drei Mittelsenkrechten durch einen gemeinsamen Punkt gehen. Alternativ kann man wie im elementargeometrischen Beweis dieses Satzes auch zuerst die Koordinaten des Schnittpunkt zweier der Mittelsenkrechten bestimmen und dann zeigen, dass dieser Punkt auf der dritten Mittelsenkrechten liegt:

```
static boolean CircumCenter_Test2(Point A, Point B, Point C) {
  Point M = intersection_point(p_bisector(A,B), p_bisector(B,C));
  return on_line(M, p_bisector(C,A));
}
```

Beispiel 2: Der Satz von der Eulerschen Geraden:

```
static boolean EulerLine_Test(Point A, Point B, Point C) {
  Point M = intersection_point(p_bisector(A,B), p_bisector(B,C));
  Point H = intersection_point(altitude(A,B,C), altitude(B,C,A));
  Point S = intersection_point(median(A,B,C), median(B,C,A));
  return is_collinear(M,H,S);
}
```

3.3 Zum grundsätzlichen Aufbau einer dynamischen Geometrie-Software (DGS)

Wir wollen uns an dieser Stelle zunächst auf die Betrachtung von Punkten und Geraden in der Ebene $\mathcal{E}$ beschränken und die grundlegenden informatischen Begriffe entwickeln, die für das Verständnis einer DGS erforderlich sind, sowie deren Verhältnis zu anderen Begriffen und Konzepten der Informatik insgesamt herausarbeiten.

Im letzten Abschnitt hatten wir bereits gesehen, dass sich das Attribute und Methoden bündelnde Klassen- und Instanzenkonzept des objektorientierten Programmierens gut für DGS eignet. Es erlaubt die Kapselung durch Koordinaten gegebener geometrischer Gebilde in neuen Sinneinheiten.

Die in der Informatik übliche Unterscheidung von abstrakter Identität eines Objekts und dessen sich im Laufe der Zeit ändernden Objekt*zustands* spielt für DGS eine wichtige Rolle. Wie bei Variablen haben wir dabei zu unterscheiden zwischen dem Objekt als Container des Zustands (dieser wird in den **Attributen** des Objekt dargestellt) und dem sich über die Zeit ändernden Zustand selbst (also den **Attributwerten**). Ist $g.c$ ein Attribut eines Objekts g mit Werten in einem Bereich C, so wird in der Attributdeklaration `public CType c` von g Speicherplatz für das Attribut `g.c` reserviert, der dann konkrete Werte $g.c \in C$ aufnehmen kann. Dieser Wert kann sich über die Zeit ändern, was als $g.c(t) \in C$ für einen Zeitparameter $t \in \mathcal{T}$ oder gleich als **Attributwertfunktion** $g.c : \mathcal{T} \to C$ dargestellt werden kann.

Attributwerte können also einmal als spezielle Werte aus einem Wertebereich C, zum anderen als Werte aus einem Funktionenraum $F(C) = \mathrm{Map}(\mathcal{T}, C)$ verstanden werden. Arithmetische Operationen auf C lassen sich zu solchen auf $F(C)$ fortsetzen, indem etwa für Funktionen $f, g \in F(C)$ die Summe $f + g \in F(C)$ punktweise durch $(f + g)(c) = f(c) + g(c)$ definiert wird. Der Definitionsbereich $D(f + g)$ dieser Verknüpfung ergibt sich als Durchschnitt der Definitionsbereiche $D(f) \cap D(g)$ bzw. im Fall eines Quotienten als $D(f/g) = D(f) \cap D(g) \setminus V(g = 0)$. Dabei kann es passieren, dass der Definitionsbereich der Verknüpfung leer ist, etwa im Fall, dass f und g disjunkte Definitionsbereiche haben.

Das Bewegen geometrischer Objekte kann die Bewegung anderer geometrischer Gebilde zur Folge haben – Zustandsänderungen propagieren also durch einen gerichteten azyklischen Graphen von Abhängigkeiten. Diese Abhängigkeiten werden durch Berechnungsvorschriften beschrieben, nach denen die Koordinaten des abhängigen Objekts aus denen der Vorgängerobjekte bestimmt werden, so dass es sich um Abhängigkeiten zwischen den Objekten selbst handelt. Wir führen deshalb die folgende begriffliche Unterscheidung ein:

Definition 1 *Die Klassen* `Point` *und* `Line` *bezeichnen wir als* ***geometrische Typen****, Instanzen einer solchen Klassen als* ***geometrische Objekte****. Jedes solche geometrische Objekt hat eine abstrakte Identität, auf die wir über den Namen des Objekts zugreifen, so dass wir voraussetzen, dass der Name eines geometrischen Objekts nicht veränderbar ist. Jeden Zustand eines solchen geometrischen Objekts im Laufe seines Lebenszyklus bezeichnen wir als* ***spezielle Realisierung des geometrischen Objekts****.*

Eine solche spezielle Realisierung ist also eine konkrete Ausprägung eines geometrischen Objekts in Raum und Zeit. Es kann dessen Zustand zu einem gewissen Zeitpunkt bestimmt (statische

Betrachtung der Verhältnisse in C) oder aber die Änderung des Zustands (dynamische Betrachtung der Verhältnisse in $F(C)$) untersucht werden.

Für jeden geometrischen Typ T spezifizieren wir ein spezielles Attribut c, welches die Koordinaten des jeweiligen Objekts enthält, und bezeichnen dieses als *Koordinatenattribut*. Ist etwa $g \in \texttt{Line}$ ein Objekt vom Typ Gerade, so steht $g.\texttt{c}$ für das Koordinatenattribut von g. Wie oben haben wir zu unterscheiden zwischen

- dem Attribut $g.\texttt{c}$ selbst als informatischem Begriff,
- der Berechnungsvorschrift, nach welcher sich der Attributwert aus anderen Attributwerten berechnet sowie
- konkreten Attributwerten $g.\texttt{c} \in C(L)$ oder $g.\texttt{c}(t) \in C(L)$.

Aus dem Kontext der weiteren Ausführungen wird deutlich, in welchem Sinne $g.\texttt{c}$ jeweils aufzufassen ist. $C(L)$ ist dabei der Bereich der (für Geraden zulässigen) Koordinatenwerte. Dieser hängt nicht von g selbst ab, sondern nur vom geometrischen Typ $g \in \texttt{Line}$ und natürlich vom gewählten **Koordinatenmodell**. Das Modell der homogenen Geradenkoordinaten lässt sich als Menge von Äquivalenzklassen

$$C(L) = \left(K^3 \setminus \{(0,0,0)\}\right) / \sim$$

bzgl. der Relation

$$(x,y,z) \sim (x',y',z') \iff \exists\, c \in K^* : x' = c\,x,\ y' = c\,y,\ z' = c\,z$$

oder kurz als Menge der nichttrivialen Orbits der Aktion der multiplikativen Gruppe K^* auf K^3 beschreiben. Ein solches Koordinatenmodell basiert stets auf einem **Grundbereich** K, aus welchem die Werte der Koordinaten kommen. Wir wollen stets voraussetzen, dass K ein algebraisch abgeschlossener Körper mit $char(K) = 0$ ist.

> *Anmerkung:* Im Weiteren unterscheiden wird zwischen dem Körper k, aus welchem die Koordinaten *spezieller Realisierungen* geometrischer Objekte stammen, und dem Koordinatenbereich K *potenziell möglicher* Realisierungen geometrischer Objekte im gewählten Koordinatenmodell. Eine solche Unterscheidung ist erforderlich, da das gewählte Koordinatenmodell, in welchem die entsprechenden Konstruktionen *interpretiert* werden, nicht notwendig berechenbar sein muss.
>
> So führt etwa $K = \mathbb{R}$ wegen Überabzählbarkeit auf ein nicht berechenbares Koordinatenmodell, $K = \mathbb{Q}$ dagegen zwar auf ein berechenbares Koordinatenmodell, aber zu einer Geometrie, in der es nicht einmal gleichseitige Dreiecke gibt.
>
> Die Koordinaten aller praktisch auftretenden speziellen Realisierungen geometrischer Objekte müssen allerdings effektiv berechenbar sein und damit aus einem effektiv berechenbaren Teilbereich $k \subset K$ kommen.

In praktischen Implementierungen wird $C(L) = k^3$ gesetzt, also mit Tripeln gearbeitet, und die weitere Struktur nur semantisch berücksichtigt.

DGS sind als grafische Anwendungen sinnvollerweise nach dem MVC-Konzept aufgebaut. Mausaktionen werden vom Controler an das Modell weitergegeben, dort die entsprechenden Berechnungen aktualisiert, und schließlich über den Controller (oder auch direkt über Event-Steuerung) an den View der Befehl zu einem `(re)paint` gegeben. In diesem Kontext ist g als geometrisches Objekt auf der Modell-Seite, die spezielle Realisierung $g(t)$ auf der View-Seite zu finden.

Geometrische Objekte werden durch entsprechende Konstruktionswerkzeuge Schritt für Schritt erzeugt, welche auf der Seite der Informatik als Funktionen zu definieren sind. Für jede solche

Konstruktion haben wir zwischen der Beschreibung dieser Konstruktion und deren Ausführung (und dynamischen Veränderung) zu unterscheiden. Eine Konstruktion muss erst vollständig beschrieben sein, ehe sie (erfolgreich oder auch erfolglos) ausgeführt werden kann. Wir müssen also – wie in anderen Bereichen der Informatik auch – zwischen *Designzeit* und *Laufzeit* unterscheiden.

Zur Designzeit wird eine Konstruktionsbeschreibung mit Hilfe vorhandener Konstruktionswerkzeuge nach entsprechenden Konstruktionsregeln erstellt. Die (zur Laufzeit wesentliche) Dynamik neu konstruierter geometrischer Objekte ergibt sich aus der Dynamik der geometrischen Objekte, welche an der Konstituierung beteiligt waren, und der Spezifik des Konstruktionswerkzeugs selbst. Diese ist in der Berechnungsvorschrift kodiert, nach welcher sich der Attributwert des Koordinatenattributs des neuen Objekts aus den Attributwerten der Koordinatenattribute der Vorgängerobjekte berechnet. Diese Berechnungsvorschrift wird zur Designzeit mit dem Koordinatenattribut des neuen Objekts verbunden und zur Laufzeit ausgeführt. Wir finden hier also die klassische informatische Unterscheidung zwischen Definition und Ausführung einer Berechnungsvorschrift wieder.

Konstruktionswerkzeuge sind prototypisch von zwei verschiedenen Arten.

Prototyp 1: `Point H = pedalpoint(Point P, Line a)`

Mit diesem Aufruf des Werkzeugs `pedalpoint` wird aus zwei vorhandenen geometrischen Objekten $P \in$ `Point` und $a \in$ `Line` ein neues geometrisches Objekt $H \in$ `Point` erzeugt. Der Attributwert des Koordinatenattributs $H.\mathtt{c}$ von H berechnet sich dabei aus den Attributwerten der Koordinatenattribute $P.\mathtt{c}$ und $a.\mathtt{c}$ nach einer Berechnungsvorschrift $\phi : C(P) \times C(L) \to \ C(P)$ als $H.\mathtt{c} = \phi(P.\mathtt{c}, a.\mathtt{c})$, die Dynamik $H.\mathtt{c}(t)$ aus den Dynamiken $P.\mathtt{c}(t)$ und $a.\mathtt{c}(t)$ als Zusammensetzung $H.\mathtt{c}(t) = \phi(P.\mathtt{c}(t), a.\mathtt{c}(t))$. Die Dynamik von H ist also durch die Dynamiken von P und a *vollständig* determiniert.

Prototyp 2: `Point M = varpoint(Point A, Point B, X u)`

Hier hängt M noch zusätzlich von einem Stellparameter u ab, dessen Natur X wir nun näher spezifizieren wollen. Ein Vergleich mit Prototyp 1 zeigt, dass u in derselben Doppelbedeutung wie ein geometrisches Objekt auch auftritt: Einerseits als abstrakte Identität eines Stellparameters, welcher selbst eine zeitliche Dynamik hat, und andererseits als Eingangsparameter, welcher die Dynamik von M beeinflusst. Die im Vergleich zu Prototyp 1 scheinbar eigenständige Dynamik von M kann also im Stellparameter gekapselt werden, so dass auch hier die Dynamik von M wie im Prototyp 1 *vollständig* durch die Dynamiken von A, B und u determiniert ist.

X steht also für einen weiteren Datentyp eines Stellparameters `SP` mit eigenem (eindimensionalem) Wertebereich $C(S)$. In Analogie zu den geometrischen Objekten bezeichnen wir einen solchen Typ als **Parametertyp** und Instanzen u dieses Typs als **Parameterobjekte**. Ein solches Parameterobjekt hat einerseits eine abstrakte Identität und andererseits einen zeitlichen Verlauf $u.\mathtt{c} \in F(C(S))$ mit Werten aus dem Parameterbereich $C(S)$. $u.\mathtt{c}(t) \in C(S)$ bezeichnen wir in Analogie zur Notation für geometrische Objekte als *spezielle Realisierung des Parameterobjekts*.

Wir können uns die Entkopplung von der visuellen Darstellung als „Bewegen mit der Maus“ etwa als Schieberegler vorstellen, dessen Schieben eine Bewegung von M auf der Geraden AB nach sich zieht. Eine solche Trennung ist auch bei freien Punkten sinnvoll, da für diese ebenfalls eine mittelbare Steuerung der Bewegung denkbar wäre. Das ist auch praktisch so: Die Koordinaten des Mauszeigers werden auf der View-Seite in Fensterkoordinaten abgegriffen und auf der Modell-Seite in Weltkoordinaten umgerechnet. Allerdings haben freie Punkte zwei Freiheitsgrade, so dass hierfür noch ein zweiter Parametertyp `MP` mit einem Parameterbereich $C(M)$ (M wie Mausparameter) benötigt wird.

Mit diesen zusätzlichen Definitionen können wir nun eine einheitliche Definition eines Konstruktionswerkzeugs geben, welche davon ausgeht, dass genügend Parameterobjekte zur Verfügung stehen, aber selbst das Anlegen eines einzigen freien Punktes durch ein Konstruktionswerkzeug geschieht (und so ist es ja praktisch auch). Die gesamte Dynamik der Konstruktion ist in den Parameterobjekten gekapselt, von denen wir voraussetzen wollen, dass sie vor allen Konstruktionsschritten angelegt werden. Die weiteren Schritte sind dann deterministisch.

Definition 2 *Seien $T_1, \ldots, T_n$ geometrische oder Parametertypen, T_a ein geometrischer Typ und $C_1, \ldots, C_n, C_a$ die zugehörigen Wertebereiche. Als* ***Konstruktionswerkzeug*** *w bezeichnen wir eine (informatische) Funktion der Signatur*

$$w : (T_1 \times \ldots \times T_n) \to T_a$$

zusammen mit einer Berechnungsvorschrift

$$w.\mathtt{c} : (C_1 \times \ldots \times C_n) \to C_a,$$

so dass sich für Objekte o_i vom Typ T_i und $o_a = w(o_1, \ldots, o_n)$ vom Typ T_a der Attributwert des Koordinatenattributs $o_a.\mathtt{c}$ einer speziellen Realisierung von o_a als

$$o_a.\mathtt{c} = w.\mathtt{c}(o_1.\mathtt{c}, \ldots, o_n.\mathtt{c}) \in C_a$$

aus den Attributwerten des Koordinatenattribute einer speziellen Realisierung der vorgegebenen Objekte $o_1, \ldots, o_n$ berechnet.
Weiter sei

$$\widetilde{w} = \mathrm{id} \times w : (T_1 \times \ldots \times T_n) \to (T_1 \times \ldots \times T_n \times T_a)$$

der Graph von w, also die Abbildung, welche die Aufrufargumente mit in das Rückgabetupel aufnimmt, und $\widetilde{w}.\mathtt{c}$ entsprechend der Graph von $w.\mathtt{c}$.

$w.\mathtt{c}$ induziert eine Funktion $F(C_1)\times\ldots\times F(C_n) \to F(C_a)$, welche die Dynamik von o_a in Abhängigkeit der Dynamiken von $o_1, \ldots, o_n$ beschreibt.
Eine weitere Feinheit haben wir noch nicht berücksichtigt: Ist z. B.

```
Line g = pp_line(Point A, Point B)
```

das Konstruktionswerkzeug, welches zu zwei gegebenen Punkten die Gerade durch diese Punkte konstruiert, so ist für spezielle Realisierungen der Punkte A und B diese Gerade nur definiert, wenn diese nicht zusammenfallen. Die Berechnungsvorschrift

$$\mathtt{pp\_line.c} : C(P) \times C(P) \longrightarrow C(L)$$

ist also nur eine partiell definierte Funktion. Die Ausnahmemenge wird durch ein boolesches Prädikat

$$\mathtt{pp\_line.DG} : C(P) \times C(P) \longrightarrow \mathtt{Boolean}$$

bestimmt, welches in unserem Fall die einfache Form $\mathtt{pp\_line.DG}(c_1, c_2) = \mathtt{isequal}(c_1, c_2)$ hat. Hierbei ist `Boolean` der Wertebereich des Datentyps `boolean`. In diesem (und so auch in den meisten Fällen) ist `isequal` sogar eine *geometrische Eigenschaft*, denn sie kann als (mathematische) Berechnungsvorschrift der informatischen Funktion

$$\mathtt{isequal} : \mathtt{Point} \times \mathtt{Point} \longrightarrow \mathtt{boolean}$$

interpretiert werden, nach der berechnet wird, ob die Attributwerte der Koordinatenattribute spezieller Realisierungen zweier Punkte zusammenfallen.

Definition 3 *Zu jedem Konstruktionswerkzeug w gibt es also weiter eine* **Degenerationsbedingung**

$$w.\mathtt{DG} : (C_1 \times \ldots \times C_n) \to \mathtt{Boolean},$$

so dass die Berechnungsvorschrift $w.\mathtt{c}$ genau dann auf einer Realisierung von $(o_1, \ldots, o_n)$ ausgeführt werden kann, wenn

$$w.\mathtt{DG}(o_1.\mathtt{c}, \ldots, o_n.\mathtt{c}) = \mathtt{false}$$

gilt.

Beispiele:

`freePoint: MP` $\to$ `Point`

legt einen freien Punkt an. Die zugehörige Degenerationsbedingung für $m \in C(M)$ ist leer: $\texttt{freePoint.DG}(m) = \texttt{false}$[2].

`pp_line: Point`$\times$`Point` $\to$ `Line`

erzeugt die Gerade durch zwei gegebene Punkte. Die zugehörige Degenerationsbedingung für $c_1, c_2 \in C(P)$ lautet $\texttt{pp\_line.DG}(c_1, c_2) = \texttt{isequal}(c_1, c_2)$.

`intersection_point: Line`$\times$`Line` $\to$ `Point`

erzeugt den Schnittpunkt zweier Geraden. Die zugehörige Degenerationsbedingung für $a, b \in C(L)$ ist $\texttt{intersection\_point.DG}(a, b) = \texttt{iszero}(a_1\, b_2 - b_1\, a_2)$. Diese Bedingung lässt sich ebenfalls als geometrische Bedingung $\texttt{is\_parallel}(a, b)$ anschreiben.

Konstruktionswerkzeuge werden eingesetzt, um mit ihnen eine geometrische Konfiguration schrittweise aufzubauen, wobei neu erzeugte geometrische Objekte von bereits vorhandenen sowie von Parameterobjekten abhängen. Diese Abhängigkeitsverhältnisse sind bei der (Neu)-Berechnung der Attributwerte der Koordinatenattribute zu berücksichtigen, was sich intern durch Ereignispropagation modellieren lässt.

Diese Abhängigkeiten lassen sich durch einen (endlichen) gerichteten azyklischen Graphen (DAG) darstellen, der aus der Konstruktionsbeschreibung extrahiert werden kann, also bereits zur Designzeit bekannt ist. Als *Konstruktionsbeschreibung* wird deshalb der prinzipielle Vorgang des Erstellens einer geometrischen Konfiguration zur Designzeit bezeichnet, als *Realisierung der Konfiguration* eine Folge spezieller Realisierungen geometrischer Objekte, welche nach dieser Beschreibung zur Konstruktionszeit erzeugt werden, sofern dies überhaupt möglich ist. Es kann Konstruktionsbeschreibungen geben, die sich nicht realisieren lassen, etwa weil sie in einem Zwischenschritt stets zu degenerierten Lagen führen, in denen der nächste Konstruktionsschritt nicht mehr ausführbar ist. Die Aussage über die prinzipielle Nichtrealisierbarkeit einer geometrischen Konfiguration ist ihrerseits wieder ein geometrischer Satz.

Definition 4 *Als* ***Konfiguration*** *bezeichnen wir einen azyklischen gerichteten Abhängigkeitsgraphen $\Gamma = (O, E)$ mit Knotenmenge $O = (o_1, \ldots, o_m)$ und Kantenmenge E, wobei die Knotenmenge eine Menge von geometrischen und Parameterobjekten ist. Wir setzen voraus, dass $(o_i, o_j) \in E \Rightarrow i < j$ gilt, also nur „spätere“ von „früheren“ Objekten abhängen. $T(o)$ bezeichnet den Typ des Objekts $o \in O$. Weiter setzen wir voraus, dass Parameterobjekte keine eingehenden Kanten haben, also zwischen Parameterobjekten keine Abhängigkeiten bestehen.*

Wir bezeichnen $\Gamma = (A, \emptyset)$ als ***Parameterkonfiguration****, wenn A ausschließlich Parameterobjekte enthält.*

Damit können wir nun den Begriff der Konstruktionsbeschreibung schrittweise herleiten.

Definition 5 *Sei*

- *$O = (o_1, \ldots, o_m)$ eine Sequenz geometrischer und Parameterobjekte und $\Gamma = (O, E)$ ein Konfiguration,*
- *$w : (T_1 \times \ldots \times T_n) \to T_{m+1}$ ein Konstruktionswerkzeug und*
- *$u \in O^n$ mit $T(u_i) = T_i$ eine Auswahlfunktion auf O für eine typgerechte Belegung der Aufrufparameter. (w ist* ***auf*** *Γ* ***anwendbar****, wenn es eine solche Belegung gibt.)*

Die Ergänzung von O um das geometrische Objekt $o_{m+1} = w(u_1, u_2, \ldots, u_n)$ vom Typ T_a und die Ergänzung von Γ zu $\Gamma' = (O \cup \{o_{m+1}\}, E \cup ((u_i, o_{m+1}), i = 1, \ldots, n))$ bezeichnen wir als ***Konstruktionsschritt****.*

[2] Präziser: Die Degenerationsbedingung hängt vom gewählten Modell für Mausparameter ab.

Es ist sinnvoll, als Belegung u auch Belegungen mit Dopplungen zuzulassen. So kann etwa das Konstruktionswerkzeug `altitude`(A, B, C), welches die Höhe durch A im Dreieck ABC konstruiert, auch für $A = B$ sinnvoll angewendet werden, um eine Senkrechte in B zu errichten.

Für die Laufzeiteigenschaft eines Konstruktionsschritts halten wir fest: Ein Konstruktionsschritt ist auf einer speziellen Realisierung von Γ **ausführbar**, wenn $w.\texttt{DG}(u_1.\texttt{c}, \ldots, u_n.\texttt{c}) = \texttt{false}$ gilt. Wir wollen annehmen, dass die Ausführung der Konstruktion mit einer Ausnahme abbricht, wenn der Schritt nicht ausführbar ist.

In all diesen Definitionen ist es möglich, den Begriff des Konstruktionsschritts und des Konstruktionswerkzeugs so zu fassen, dass statt eines einzelnen geometrischen Objekts o_{m+1} gleich ein ganzes Tupel $(o_{m+1}, \ldots, o_{m+k})$ neuer Objekte konstruiert wird, deren Typen $(T_{m+1}, \ldots, T_{m+k})$ vorgegeben sind. Ein solches Konstruktionswerkzeug bezeichnen wir als **verallgemeinertes Konstruktionswerkzeug**.

Definition 6 *Als **Konstruktionsbeschreibung** $\mathcal{K}$ bezeichnen wir eine Folge von Konstruktionsschritten, welche die **Startkonfiguration** $\Gamma_S = (O_S, E_S)$ mit $O_S = (o_1, \ldots, o_m)$ durch endlich viele konsekutive Konstruktionsschritte*

$$\widetilde{w}^{(N)} : \Gamma_S = \Gamma_0 \xrightarrow{\widetilde{w}_1} \Gamma_1 \xrightarrow{\widetilde{w}_2} \ldots \xrightarrow{\widetilde{w}_N} \Gamma_N = \Gamma_E \qquad (\mathcal{K})$$

*in die **Endkonfiguration** $\Gamma_E = (O_E, E_E)$ mit $O_E = (o_1, \ldots, o_{m+N})$ überführt. Wir schreiben auch kurz $O_E = \mathcal{K}(O_S)$.*

Die dynamischen Freiheitsgrade einer Konstruktionsbeschreibung werden allein durch deren Parameterobjekte bestimmt, sind also bereits in der Startkonfiguration festgelegt, da nach Definition in einem Konstruktionsschritt ausschließlich geometrische Objekte erzeugt werden.

Die Definition zeigt, dass zusammen mit $(\mathcal{K})$ auch jede Teilfolge

$$\widetilde{w}^{(i)} : \Gamma_S = \Gamma_0 \xrightarrow{\widetilde{w}_1} \Gamma_1 \xrightarrow{\widetilde{w}_2} \ldots \xrightarrow{\widetilde{w}_i} \Gamma_i$$

eine Konstruktionsbeschreibung ist. Wir bezeichnen sie als *Teil- oder Zwischenkonstruktion* $\mathcal{K}^{(i)}$.

Konstruktionsbeschreibungen werden wir in der formalen Notation eines **geometrischen Linearprogramms** (GLP) angeben. Ein solches Programm enthält in den ersten Zeilen die Startkonfiguration. Danach folgt die Angabe der einzelnen Konstruktionsschritte, wobei die Angabe der Belegung des entsprechenden Konstruktionswerkzeugs über Bezeichner für die konstruierten geometrischen Objekte ohne Vorwärtsreferenzen erfolgt.

Die Beschreibung der Konstruktion eines durch drei freie Punkte aufgespannten Dreiecks lässt sich damit wie folgt beschreiben:

```
Start(MP U1, U2, U3);
Point A = freePoint(U1);
Point B = freePoint(U2);
Point C = freePoint(U3);
Line a = pp_line(B,C);
Line b = pp_line(A,C);
Line c = pp_line(A,B);
```

Sind in solchen GLP als Abkürzungen geschachtelte Funktionsaufrufe wie folgt erlaubt

```
Point F = intersection_point(pp_line(B,C),ortho_line(A,pp_line(B,C)));
```

so sprechen wir von der *schwachen GLP-Notation*. Sind als Parameter in einem Konstruktionswerkzeug nur Bezeichner zugelassen, so sprechen wir von der *strengen GLP-Notation*.

In obigem Beispiel ist F der Lotfußpunkt der Höhe durch A im Dreieck ABC. Jedes GLP in schwacher Notation kann durch Einführung *anonymer geometrischer Objekte* in die strenge Notation überführt werden. Obiger Schritt kann wie folgt in eine Sequenz von Konstruktionsschritten *entschachtelt* werden:

```
Line _l1 = pp_line(B,C);
Line _l2 = ortho_line(A,_l1);
Point F = intersection_point(_l1,_l2);
```

Untersuchen wir nun genauer, wann eine Konstruktionsbeschreibung $\mathcal{K}$ auf einer (zulässigen) speziellen Realisierung der Startkonfiguration O_S ausführbar ist. Die Konstruktionsbeschreibung ist genau dann *nicht* ausführbar, wenn für ein $i > 0$ gilt, dass die Konstruktionsbeschreibung $\mathcal{K}^{(i-1)}$ ausführbar ist, $\mathcal{K}^{(i)}$ aber nicht mehr. Sei $u^{(i)}$ die Parameterauswahl der Anwendung des Werkzeugs w_i im Schritt i. Wir können (und wollen) $u^{(i)} \in O_E^n$ annehmen, da alle vor Schritt i konstruierten Objekte auch in der Endkonfiguration enthalten sind. Als Degenerationsbedingung im Schritt i ergibt sich $w_i.\mathtt{DG}\left(u^{(i)}.\mathtt{c}\right)$, als vollständige Degenerationsbedingung die Disjunktion

$$\bigvee_{i>0} w_i.\mathtt{DG}\left(u^{(i)}.\mathtt{c}\right)$$

dieser Bedingungen, wobei davon ausgegangen wird, dass dieser Ausdruck *kurz* (short) nach wachsendem i ausgewertet wird, da die in $w_i.\mathtt{DG}\left(u^{(i)}.\mathtt{c}\right)$ eingehenden Argumente zur Auswertung dieser Formel vorher berechnet werden, die entsprechenden Konstruktionsschritte auf dieser speziellen Realisierung der Konfiguration dazu ausführbar sein müssen.

Jede Konstruktionsbeschreibung ist zugleich ein verallgemeinertes Konstruktionswerkzeug, was die Einführung des Konzepts von **Makros** ermöglicht: Für ein verallgemeinertes Konstruktionswerkzeug $w : T_1 \times \cdots \times T_n \to U_1 \times \cdots \times U_m$ kann man w aus dem Abbildungsgraphen

$$\widetilde{w} = id \times w : T_1 \times \cdots \times T_n \to T_1 \times \cdots \times T_n \times U_1 \times \cdots \times U_m$$

von w durch Projektion auf die U-Komponenten gewinnen. Dasselbe gilt für die Berechnungsvorschrift $\widetilde{w}.\mathtt{c}$. Ist nun

$$\Gamma_S = \Gamma_0 \xrightarrow{\widetilde{w}_1} \Gamma_1 \xrightarrow{\widetilde{w}_2} \ldots \xrightarrow{\widetilde{w}_N} \Gamma_N = \Gamma_E$$

die Folge von Konstruktionsschritten der Konstruktionsbeschreibung $\mathcal{K}$, so ist

$$\widetilde{w}_K = \widetilde{w}_N \circ \cdots \circ \widetilde{w}_1 : \prod (T(o) \mid o \in O_S) \longrightarrow \prod (T(o) \mid o \in O_E)$$

das zugehörige verallgemeinerte Konstruktionswerkzeug. Die Degenerationsbedingung $\widetilde{w}_K.\mathtt{DG}$ ergibt sich wie oben als oder-Verknüpfung der einzelnen $w_i.\mathtt{DG}\left(u^{(i)}.\mathtt{c}\right)$

In einer DGS müssen zur bildlichen dynamischen Darstellung die Koordinaten der speziellen Realisierungen geometrischer Objekte nach dieser Konstruktionsbeschreibung nach jedem Mausevent neu berechnet werden. Da mit den eingeführten geometrischen Objekten weitere Attribute wie Farbe, Linienstärke, Bezeichnung usw. verbunden sind, müssen die entsprechenden Objekte in einer zur Laufzeit zu verwaltenden *Symboltabelle* aufgesammelt werden. Interpretierte Sprachen sind hierfür besser geeignet als compilierte, da dort bereits eine solche Symboltabelle angelegt ist. In compilierenden Programmiersystemen wie Java wird die Symboltabelle im Prinzip nur zur Übersetzungszeit benötigt. Im Design eines DGS auf dieser Basis muss also eine solche Symboltabelle (mit ihrer zentralen Eigenschaft der eindeutigen Repräsentation) softwareseitig angelegt und verwaltet werden.

Andererseits legt das vielfache Ausführen der mit einer Konstruktion verbundenen stets gleichen Berechnungsvorschriften auf verschiedenen (Maus)-Eingabewerten nahe, den entsprechenden Code zur Laufzeit „on the fly“ zu compilieren, wenn ein solches Feature vom verwendeten Programmiersystem angeboten wird. Weiter ist zu berücksichtigen, dass spezielle Parameterwerte zu degenerierten Situationen führen können, in denen einzelne spezielle Realisierungen von Objekten der jeweiligen Konstruktion nicht existieren und damit davon abhängende spezielle Realisierungen weiterer geometrischer Objekte auf einem ganzen Zweig des Abhängigkeitsgraphen nicht konstruiert werden können. Als Vorstufe davon können einzelne spezielle Realisierungen von Objekte zwar konstruierbar sein, aber außerhalb der Zeichenfläche liegen. Dies muss innerhalb der DGS abgefangen werden.

3.4 Symbolische analytische Geometrie

> Einen geometrischen Satz in einem Koordinatenmodell über einem Körper K zu beweisen, mit dessen Elementen möglicherweise nicht effektiv gerechnet werden kann (etwa die reellen oder komplexen Zahlen), bedeutet zu beweisen, dass dieser Satz für alle (zulässigen) *Interpretationen* innerhalb dieses Koordinatenmodells erfüllt ist.
>
> Eine Konstruktionsbeschreibung kann nur über speziellen Realisierungen geometrischer Objekte mit Koordinaten aus einem effektiv berechenbaren Körper $k \subset K$ ausgeführt werden, die Aussage eines geometrischen Satzes bezieht sich aber auf *alle* zulässigen Realisierungen geometrischer Objekte innerhalb des gegebenen Koordinatenmodells. Diese Rechnungen können nicht *ausgeführt*, sondern nur über K (in einer K-Theorie) *interpretiert* werden. Wir nennen jede solche Realisierung eine *spezielle Interpretation* des geometrischen Objekts.

Ist das geometrische Objekt $o = w(o_1, \ldots, o_n)$ mit dem Konstruktionswerkzeug w konstruiert worden, so berechnet sich der Wert des Koordinatenattributs einer speziellen Realisierung von o aus den aktuellen Werten der Koordinatenattribute einer speziellen Realisierung von $o_1, \ldots, o_n$ als $o.\mathtt{c} = w.\mathtt{c}(o_1.\mathtt{c}, \ldots, o_n.\mathtt{c})$. Dabei sind jedesmal dieselben Rechnungen $w.\mathtt{c}$ mit je anderen Werten, also eine allgemeine *Berechnungsvorschrift* auszuführen.

> *Anmerkung:* Ausdrucksmächtigkeit und Eigenschaften solcher Berechnungsvorschriften hängen vom gewählten *Berechnungsmodell* ab, das wesentlich durch mathematische Eigenschaften des gewählten Grundbereichs k für Koordinatenwerte sowie die informatischen Eigenschaften des Programmiermodells bestimmt wird.

Ist

$$\mathcal{K} : \Gamma_S = \Gamma_0 \xrightarrow{\widetilde{w}_1} \Gamma_1 \xrightarrow{\widetilde{w}_2} \ldots \xrightarrow{\widetilde{w}_N} \Gamma_N = \Gamma_E$$

eine Konstruktionsbeschreibung, so berechnen sich die Koordinatenwerte einer speziellen Realisierung der Endkonfiguration $O_E = \mathcal{K}(O_S)$ – Ausführbarkeit vorausgesetzt – nach einer Berechnungsvorschrift, die sich als Hintereinanderausführung $\widetilde{w}_N.\mathtt{c} \circ \cdots \circ \widetilde{w}_1.\mathtt{c}$ der Berechnungsvorschriften der einzelnen Konstruktionsschritte ergibt.

Damit entsteht die Frage, ob sich diese häufig auszuführenden Berechnungsvorschriften durch gleichwertige, aber effizienter auszuführende ersetzen lassen. Aus Sicht der Informatik wäre insbesondere über eine compilierte Form nachzudenken, was eine DGS mit entsprechenden Fähigkeiten zur inkrementellen Übersetzung erfordert. Dieser Ansatz soll uns hier nicht weiter interessieren.

Wir wollen ein Berechnungsmodell betrachten, auf das sich unsere Anwendungen bisher immer reduzieren ließen: Die Koordinaten von $o.\mathtt{c}$ lassen sich durch *arithmetische Operationen* aus den Koordinaten von $o_1.\mathtt{c}, \ldots, o_n.\mathtt{c}$ berechnen lassen.

Derartige Berechnungen lassen sich auch auf *symbolischen Eingaben* ausführen. Setzen wir etwa in der (MAXIMA-Implementierung der) Funktion `p_bisector` die Koordinaten der einzelnen Punkte mit Unbestimmten an, so erhalten wir daraus die Berechnungsvorschrift für die Mittelsenkrechte von $\overline{AB}$

```
A:Point(ax,ay); B:Point(bx,by); C:Point(cx,cy);
ma:p_bisector(B,C);
```

$$\left[b_x - c_x,\ b_y - c_y,\ \left(\frac{c_y}{2} + \frac{b_y}{2}\right)(c_y - b_y) - (b_x - c_x)\left(\frac{c_x}{2} + \frac{b_x}{2}\right)\right]$$

als *arithmetische Formel*, die durch eine entsprechende Termumformung weiter vereinfacht werden kann zu

```
ratsimp(ma);
```

$$\left[b_x - c_x,\ b_y - c_y,\ \frac{1}{2}\left(c_y^2 + c_x^2 - b_y^2 - b_x^2\right)\right].$$

Ähnliche Vereinfachungen ergeben sich für die Berechnungsvorschrift des Schnittpunkts M der Mittelsenkrechten m_a und m_b:

```
mb:p_bisector(A,C);
M:intersection_point(ma,mb);
```

$$\left[\begin{array}{c} \dfrac{\begin{array}{c}(b_y - c_y)\left((a_x - c_x)\left(\frac{a_x}{2} + \frac{c_x}{2}\right) + (a_y - c_y)\left(\frac{a_y}{2} + \frac{c_y}{2}\right)\right) - \\ (a_y - c_y)\left((b_x - c_x)\left(\frac{b_x}{2} + \frac{c_x}{2}\right) + (b_y - c_y)\left(\frac{b_y}{2} + \frac{c_y}{2}\right)\right)\end{array}}{a_x\,b_y - a_y\,b_x - a_x\,c_y + a_y\,c_x + b_x\,c_y - b_y\,c_x}, \\ \\ -\dfrac{\begin{array}{c}(b_x - c_x)\left((a_x - c_x)\left(\frac{a_x}{2} + \frac{c_x}{2}\right) + (a_y - c_y)\left(\frac{a_y}{2} + \frac{c_y}{2}\right)\right) - \\ (a_x - c_x)\left((b_x - c_x)\left(\frac{b_x}{2} + \frac{c_x}{2}\right) + (b_y - c_y)\left(\frac{b_y}{2} + \frac{c_y}{2}\right)\right)\end{array}}{a_x\,b_y - a_y\,b_x - a_x\,c_y + a_y\,c_x + b_x\,c_y - b_y\,c_x} \end{array}\right]$$

```
ratsimp(M);
```

$$\left[\begin{array}{c} -\dfrac{\begin{array}{c}-{a_x}^2\,b_y + {a_x}^2\,c_y - {a_y}^2\,b_y + {a_y}^2\,c_y + a_y\,{b_x}^2 + a_y\,{b_y}^2 - \\ a_y\,{c_x}^2 - a_y\,{c_y}^2 - b{x}^2\,c_y - {b_y}^2\,c_y + b_y\,{c_x}^2 + b_y\,{c_y}^2\end{array}}{2\,a_x\,b_y - 2\,a_y\,b_x - 2\,a_x\,c_y + 2\,a_y\,c_x + 2\,b_x\,c_y - 2\,b_y\,c_x}, \\ \\ \dfrac{\begin{array}{c}-{a_x}^2\,b_x + {a_x}^2\,c_x + a_x\,{b_x}^2 + a_x\,{b_y}^2 - a_x\,{c_x}^2 - a_x\,{c_y}^2 - \\ {a_y}^2\,b_x + {a_y}^2\,c_x - {b_x}^2\,c_x + b_x\,{c_x}^2 + b_x\,{c_y}^2 - {b_y}^2\,c_x\end{array}}{2\,a_x\,b_y - 2\,a_y\,b_x - 2\,a_x\,c_y + 2\,a_y\,c_x + 2\,b_x\,c_y - 2\,b_y\,c_x} \end{array}\right]$$

Nun bestimmen wir noch die Berechnungsvorschrift für m_c und den Test der Bedingung, dass M auf m_c liegt:

```
mc:p_bisector(A,B);
result:on_line(M,mc);
ratsimp(result);
```

In diesem Berechnungsmodell lassen sich alle Berechnungsvorschriften für die Koordinaten geometrischer Objekte sowie die Schlussfolgerung des geometrischen Satzes als komplizierte rationale Ausdrücke $A(X) \in k(X)$ in den Eingangsvariablen X darstellen. Der letzte rationale Ausdruck vereinfacht sogar als rationaler Ausdruck zu null. Da in eine Konstruktion nur endlich viele Parameterobjekte eingehen und diese die einzige Quelle der Dynamik der Konstruktion sind, lässt sich auch im allgemeinen Fall jede Berechnungsvorschrift durch arithmetische Operationen aus Kernen aufbauen, die nur die Koordinatenwerte der Parameterobjekte enthalten.

> Es bleibt zu untersuchen, was derartige *Berechnungen* im Koordinatenmodell über $k(X)$ mit *Beweisen* geometrischer Sätze im Koordinatenmodell über K zu tun haben.

Der Ausdruck $A(X)$ ergibt sich, wenn wir die entsprechende Berechnungsvorschrift mit Variablen $X = (x_a)$ als Koordinatenwerten aufrufen und die Berechnungen in $k(X)$ ausführen. In der Theorie rationaler Ausdrücke in $k(X)$ können wir auch nach Vereinfachungsmöglichkeiten des Ausdrucks $A(X)$ zu einem Ausdruck $A'(X)$ in $k(X)$ fragen. Die Berechnungsvorschrift für $\gamma_a \in K$ lässt sich aus A durch die Substitution $s = \{x_a \to \gamma_a\}$ zurückgewinnen. Die Vereinfachung des Ausdrucks A zum Ausdruck A' in $k(X)$ ist natürlich nur dann interessant, wenn die zugehörigen Berechnungsvorschriften `subst`(A, s) und `subst`(A', s) berechnungsäquivalent sind, d. h. dasselbe berechnen. Im Rahmen eines CAS kann die Umwandlung eines rationalen Ausdrucks in eine Berechnungsvorschrift für effektive Koordinatenwerte $\gamma_a \in k$ durch Ersetzen der Variablen durch Zahlenwerte mit dem Substitutionsoperator erreicht werden, da bei der Auswertung des Ausdrucks mit Zahlenwerten die entsprechenden Operationen als Funktions*aufrufe* interpretiert werden. Dabei kann es allerdings geschehen, dass die Berechnung mit einem Fehler (Ausnahme) „Division durch null“ abbricht.

Wir werden die Möglichkeiten dieses Ansatzes zunächst an einigen Beispielen betrachten und uns später für die Beweiskraft der entsprechenden Rechnungen interessieren. Dazu verwenden wir die GeoProver-Implementierungen der Berechnungsvorschriften der verschiedenen Konstruktionswerkzeuge in Maxima, mit denen sich die entsprechenden Berechnungen auch auf symbolischen Eingabewerten ausführen lassen. Die sich bei diesen Berechnungen ergebenden Ausdrücke in $k(X)$ lassen sich durch Termumformungen weiter vereinfachen. Da für rationale Ausdrücke mit der *rationalen Normalform* stets eine „einfachste“ Darstellung existiert, die sich in allen CAS durch entsprechende Transformationsfunktionen (in Maxima durch die Funktion `ratsimp`) effektiv berechnen lassen, wollen wir weiter voraussetzen, dass in den Berechnungsfunktionen über $k(X)$ stets diese rationale Normalform berechnet wird.

Beispiele: Siehe `geo-1.txt`

Die Ausdrücke A bzw. A' sind *universelle Formeln* der zum Werkzeug w gehörenden Berechnungsvorschrift $w.\mathtt{c}$. Wir wollen diesen Begriff nun präziser einführen.

Betrachten wir dazu den Konstruktionsschritt $o = w(o_1, \ldots, o_n)$, der aus gegebenen Objekten $o_1, \ldots, o_n$ das neue Objekt o erzeugt. Da in die Berechnung $w.\mathtt{c}$ die Koordinaten der aktuellen Realisierungen der Aufrufobjekte eingehen, muss zunächst ein Mechanismus eingeführt werden, diese Parameter durch Variable zu ersetzen. Mit diesem Ersetzungsmechanismus erzeugen wir *universelle Realisierungen* der gegebenen Objekte $o_1, \ldots, o_n$, aus denen sich alle anderen Realisierungen ableiten lassen.

Sei dazu $X = [x_a : a \in I]$ ein Satz von abzählbar vielen Variablen, $R = k[X]$ der Polynomring über k in diesen Variablen und $k(X)$ dessen Quotientenkörper. Ist $C = C(T)$ der Wertebereich eines geometrischen oder Parametertyps T über dem Grundbereich k, so bezeichnen wir mit $C_X(T) = C \otimes_k k(X)$ den Wertebereich, den man durch Erweiterung $k \to k(X)$ des Grundbereichs erhält.

Praktisch bedeutet dies, dass nicht nur Elemente aus k, sondern aus $k(X)$ an allen Stellen als Koordinaten erlaubt werden. Da die Berechnungsvorschriften nur arithmetische Operationen enthalten, sind sie auch auf solchen Eingaben ausführbar und die Ergebnisse liegen in $C_X(T)$, außer wenn durch einen rationalen Ausdruck $r(X)$ dividiert wird, der als rationaler Ausdruck gleich null ist. Dies kann aber mit Blick auf die starke Nulltestereigenschaft von `ratsimp` effektiv bestimmt werden. In diesem Fall ist außerdem $r(\gamma) = 0$ für jede Substitution $s = \{x_a \to \gamma_a\}$ mit $\gamma_a \in K$, die entsprechende Berechnungsvorschrift also auch für keine speziellen geometrischen Objekte über K interpretiertbar.

Der Polynomring R hat die universelle Eigenschaft, dass sich jede Substitution $s = \{x_a \to \gamma_a\}$ mit $\gamma_a \in K$ eindeutig zu einem Ringhomomorphismus $R \to K$ fortsetzen lässt, arithmetische Berechnungen in R also universeller Prototyp der Interpretationen von Rechnungen über K sind. Ähnliches gilt für $\frac{f(X)}{g(X)} \in k(X)$, wenn $g(\gamma) \neq 0$ ist.

Definition 7 *Als* universelle Realisierung eines Objekts *o vom Typ T bezeichnen wir jede Realisierung* $o.\mathtt{u} \in C_X$ *mit einem Satz „frischer“ Variablen, so dass sich für jede spezielle Interpretation von o die Koordinatendarstellung durch geeignete Variablenspezifikation* $s = \{x_a \to \gamma_a\}$ *mit*

$\gamma_a \in K$ *ergibt:* $o.\mathtt{c} = \mathtt{subst}(o.\mathtt{u}, s)$.
„Frische" Variablen heißt dabei: Ist

$$\mathrm{var}(o.\mathtt{u}) = \{a \in I \,|\, x_a \textit{ kommt in } o.\mathtt{u} \textit{ wirklich vor.}\}$$

die Menge der Indizes der in $o.\mathtt{u}$ *wirklich vorkommenden Variablen, so sind diese Mengen für alle universellen Realisierungen aller Objekte paarweise disjunkt. Da in einer endlichen Beschreibung nur eine endlich Anzahl universeller Realisierungen endlich vieler Objekte verwendet werden kann,* I *aber als abzählbar unendlich vorausgesetzt wurde, ist eine solche Variablenauswahl immer möglich.*

Eine universelle Realisierung eines geometrischen Objekts o ist selbst wieder eine spezielle Realisierung von o im Koordinatenmodell über $k(X)$. Die Idee einer *universellen Realisierung eines geometrischen Objekts* folgt dem Konzept des allgemeinen Punkts einer Varietät in der algebraischen Geometrie.

Wir können nun den Konstruktionsschritt $o = w(o_1, \ldots, o_n)$ mit universellen Realisierungen $o_1.\mathtt{u}, \ldots, o_n.\mathtt{u}$ von $O = (o_1, \ldots, o_n)$ ausführen und erhalten eine *universelle Formel*

$$w.\mathtt{uf} = w.\mathtt{u}(o_1.\mathtt{u}, \ldots, o_n.\mathtt{u}) \in C_X(T(o))\,.$$

Zur Ausführbarkeit des Konstruktionsschritts muss die Bedingung $w.\mathtt{DGF} = w.\mathtt{DG}(o_1.\mathtt{u}, \ldots, o_n.\mathtt{u})$ untersucht werden. Für spezielle Interpretationen von O mit Koordinaten aus k wertet diese Bedingung stets zu einem der booleschen Werte `true` oder `false` aus. Das ist für universelle Realisierungen[3] nicht mehr der Fall, da der entstehende Ausdruck $w.\mathtt{DGF}$ Variablen enthält, also eine boolesche Formel und kein boolescher Ausdruck mehr ist. Der Wertebereich dieser Formeln ist also `BooleanExpression` und nicht mehr `Boolean`. Als *Tautologie* bezeichnen wir jede boolesche Formel, die über $k(X)$ zu `true` auswertet. Wir wollen im Weiteren voraussetzen, dass $w.\mathtt{DGF}$ keine Tautologie ist.

Wenn in der Berechnungsvorschrift für w nur arithmetische Operationen verwendet werden, erhalten wir für die Koordinaten $w.\mathtt{uf} = o.\mathtt{c}$ des Ergebnisses dieses Konstruktionsschritts *rationale* Ausdrücke in $\{x_a \,|\, a \in \cup_i \mathrm{var}(o_i.\mathtt{u})\}$. Diese Ausdrücke stellen *universelle Formeln* in dem Sinne dar, dass jede Ausführung des Konstruktionsschritts mit einer *zulässigen* speziellen Interpretation von O zu einer speziellen Interpretation von o führt, deren Koordinaten sich durch die Variablenspezifikation $s = \{x_a \to \gamma_a\}$ aus $w.\mathtt{uf}$ berechnen lassen, für die $o_i.\mathtt{c} = \mathtt{subst}(o_i.\mathtt{u}, s),\ i = 1, \ldots, n$, gilt. Ausführbarkeit bedeutet dabei, dass $w.\mathtt{DG}\,(o_1.\mathtt{c}, \ldots, o_n.\mathtt{c}) = \mathtt{subst}(w.\mathtt{DGF}, s) = \mathtt{false}$ gilt.

Definition 8 *$w.\mathtt{uf}$ bezeichnen wir als die* universelle Formel *des Konstruktionswerkzeugs w, $w.\mathtt{DGF}$ als dessen* universelle Degenerationsbedingung.

Ist

$$\Gamma_S = \Gamma_0 \xrightarrow{\widetilde{w}_1} \Gamma_1 \xrightarrow{\widetilde{w}_2} \ldots \xrightarrow{\widetilde{w}_N} \Gamma_N = \Gamma_E$$

die Konstruktionsbeschreibung einer Konstruktion $\mathcal{K}$, so können wir diese Überlegungen auf jeden einzelnen Konstruktionsschritt anwenden.

Sei dazu $\Gamma_S = (O_S, E_S)$ mit $O_S = (o_1, \ldots, o_m)$ die Startkonfiguration, $\Gamma_E = (O_E, E_E)$ mit $O_E = (o_1, \ldots, o_{m+N})$ die Endkonfiguration und universelle Realisierungen $A_E = (o_1.\mathtt{u}, \ldots, o_{m+N}.\mathtt{u})$ für die Objekte aus O_E fixiert.

Wir beginnen mit der universellen Realisierung $o_1.\mathtt{u}, \ldots, o_m.\mathtt{u}$ der Startkonfiguration und analysieren die Berechnungen der daraus Schritt für Schritt zu konstruierenden Realisierungen $o_{m+i}.\mathtt{uf}$ der Objekte $o_{m+i},\ i = 1, \ldots, N$ im Koordinatenmodell über $k(X)$.

[3] Auch für Interpretationen der Bedingung über K ist Auswertbarkeit nicht garantiert, wenn die vorkommenden booleschen Prädikate für die speziellen Koordinatenwerte aus K nicht berechenbar sind.

$o_{m+1}.\mathtt{uf}$ berechnet sich aus der universellen Formel $w_1.\mathtt{uf}$, ist aber selbst keine universelle Realisierung mehr, sondern ergibt sich aus der universellen Realisierung $o_{m+1}.\mathtt{u}$ durch die Substitution

$$s_1 = \{x_a \to w_1.\mathtt{uf}_a\} \text{ für } x_a \in \mathrm{var}(o_{m+1}.\mathtt{u})\,,$$

wobei $w_1.\mathtt{uf}_a$ den zum Index a gehörenden Teil von $w_1.\mathtt{uf}$ bezeichnet. Die Berechnung ist ausführbar, wenn $w_1.\mathtt{DGF}$ keine Tautologie ist.

In der universellen Formel $w_2.\mathtt{uf}$ wird statt der konstruierten Realisierung $o_{m+1}.\mathtt{uf}$ die universelle Realisierung $o_{m+1}.\mathtt{u}$ verwendet. $o_{m+2}.\mathtt{uf}$ ergibt sich also, wenn in der universellen Formel $w_2.\mathtt{uf}$ die Substitution s_1 ausgeführt wird:

$$o_{m+2}.\mathtt{uf} = \mathtt{subst}(w_2.\mathtt{uf}, s_1)$$

Kurz, in der universellen Formel $w_2.\mathtt{uf}$ haben wir die Variablen $x_a \in \mathrm{var}(o_{m+1}.\mathtt{u})$ durch rationale Ausdrücke zu ersetzen. Die Berechnung ist ausführbar, wenn $\mathtt{subst}(w_2.\mathtt{DGF}, s_1)$ keine Tautologie ist, was wir zunächst voraussetzen wollen.

$o_{m+2}.\mathtt{uf}$ ergibt sich auch als Folge der Substitutionen

$$s_2 = \{x_b \to w_2.\mathtt{uf}_b\} \text{ für } x_b \in \mathrm{var}(o_{m+2}.\mathtt{u})\,,$$

mit der die Variablen in der universellen Realisierung $o_{m+2}.\mathtt{u}$ durch die universellen Formeln $w_2.\mathtt{uf}$ ersetzt werden, und s_1, mit der die Variablen $x_a \in \mathrm{var}(o_{m+1}.\mathtt{u})$ ersetzt werden.

$$o_{m+2}.\mathtt{uf} = \mathtt{subst}(o_{m+2}.\mathtt{u}, s_2, s_1)$$

Dies setzt sich über die anderen Konstruktionsschritte fort, so dass sich die Koordinaten der Realisierung der Endkonfiguration O_E aus der universellen Realisierung von O_E durch die Substitutionen $s_N, s_{N-1}, \ldots, s_1$ ergeben, mit denen rückwärts Schritt für Schritt für die freien Variablen der universellen Realisierungen der abhängigen Objekte die rationalen Ausdrücke eingesetzt werden, welche sich im jeweils vorherigen Konstruktionsschritt ergeben haben. Ersetzt man aber in einem rationalen Ausdruck einzelne Variablen durch andere rationale Ausdrücke, so entsteht als Ergebnis wieder ein rationaler Ausdruck (oder, wenn sich der Nullausdruck als Hauptnenner ergibt, ein Fehler). Durch eine solche Konstruktion wird der Bereich der rationalen Ausdrücke also nicht verlassen.

Als universelle Degenerationsbedingung der gesamten Konstruktion $\mathcal{K}$ im Koordinatenmodell über $k(X)$ ergibt sich

$$\mathcal{K}.\mathtt{DGF} = \bigvee_{i=1}^{N} \mathtt{subst}(w_i.\mathtt{DGF}, s_{i-1}, \ldots, s_1)\,.$$

Die Konstruktion $\mathcal{K}$ ist im Koordinatenmodell über $k(X)$ ausführbar, wenn dies keine Tautologie ist.

Definition 9 *Eine Realisierung $B_E = (o_1.\mathtt{u}, \ldots, o_m.\mathtt{u}, o_{m+1}.\mathtt{uf}, \ldots, o_{m+N}.\mathtt{uf})$ der Endkonfiguration $O_E = \mathcal{K}(O_S)$, die mit einer universellen Realisierung von O_S startet, bezeichnen wir als* universelle Realisierung der Konstruktion $\mathcal{K}$*, die dabei berechneten Formeln $o_i.\mathtt{uf}$ der Realisierungen der abgeleiteten Objekte als deren* universelle Formeln *und $\mathcal{K}.\mathtt{DGF}$ als* universelle Degenerationsbedingung *der Konstruktion.*

Der Simplifikationsprozess rationaler Ausdrücke liefert äquivalente in $k(X)$ Ausdrücke, die nicht unbedingt dieselbe rationale Funktion darstellen, da letztere verschiedene Definitionsbereiche haben können. Klar ist nur, dass die rationalen Funktionen der vereinfachten Ausdrücke für alle Variablenspezifikationen, welche für die unsimplifizierte Funktion nicht zu einem Fehlerabbruch führen, denselben Wert liefern wie diese.

Sehen wir uns als Beispiel noch einmal die Konstruktion $\mathcal{K}$ des Umkreismittelpunkts M des Dreiecks ABC als Schnittpunkt der Mittelsenkrechten $m = m_a$, $n = m_b$ an. Wir überführen dazu zunächst den Aufruf

```
Point M:=intersection_point(p_bisector(B,C), p_bisector(A,C))
```

in ein GLP:

```
Start(Point A,B,C);
Line g1:=p_bisector(B,C);
Line g2:=p_bisector(A,C);
Point M:=intersection_point(g1,g2);
```

Als universelle Realisierung der Startkonfiguration setzen wir $A.\mathtt{u} = (a_x, a_y)$, $B.\mathtt{u} = (b_x, b_y)$, $C.\mathtt{u} = (c_x, c_y)$ und ergänzen dies mit $g_1.\mathtt{u} = (m_1, m_2, m_3)$, $g_2.\mathtt{u} = (n_1, n_2, n_3)$, $M.\mathtt{u} = (m_x, m_y)$ zu einer universellen Realisierung der Endkonfiguration.

Im Koordinatenmodell über $k(X)$ bedeutet die Ausführung der Konstruktion auf der universellen Startkonfiguration, dass in der universellen Formel

$$w.\mathtt{uf} = \left(\frac{m_2n_3 - m_3n_2}{m_1n_2 - m_2n_1}, \frac{m_3n_1 - m_1n_3}{m_1n_2 - m_2n_1}\right)$$

des Konstruktionswerkzeugs `intersection_point`, nach der sich die Koordinaten des Schnittpunkts der universellen Realisierungen $g_1.\mathtt{u}$ und $g_2.\mathtt{u}$ der beiden Geraden g_1, g_2 berechnen, die Variablen von m_i, n_i durch die entsprechenden rationalen Ausdrücke

$$s_1 = \left(m_1 = b_x - c_x,\ m_2 = b_y - c_y,\ m_3 = \frac{1}{2}\left(-b_x^2 - b_y^2 + c_x^2 + c_y^2\right)\right)$$

bzw.

$$s_2 = \left(n_1 = -a_x + c_x,\ n_2 = -a_y + c_y,\ n_3 = \frac{1}{2}\left(a_x^2 + a_y^2 - c_x^2 - c_y^2\right)\right)$$

zu ersetzen sind. Diese Ausdrücke sind ihrerseits während des Aufrufs

```
p_bisector(B,C) == ortho_line(midPoint(B,C),pp_line(B,C))
```

in einem Substitutions- und Simplifikationsprozess (der Tiefe 2) nach demselben Prinzip entstanden. Wir erhalten in diesem Fall als universelle Formel $M.\mathtt{uf}$ für den Umkreismittelpunkt M die früher berechnete Formel.

Die entsprechenden universellen Degenerationsbedingungen lauten

$$\mathtt{intersection\_point.DGF} = (m_1n_2 - m_2n_1 = 0)$$

sowie

$$\mathtt{p\_bisector.DGF} = (u_x = v_x) \wedge (u_y = v_y)\,.$$

Daraus ergibt sich

$$\begin{aligned}\mathcal{K}.\mathtt{DGF} = &((b_x - c_x)(a_y - c_y) - (a_x - c_x)(b_y - c_y) = 0)\\ &\vee ((b_x = c_x) \wedge (b_y = c_y)) \vee ((a_x = c_x) \wedge (a_y = c_y))\ ,\end{aligned}$$

was sich im Koordinatenmodell über $k(X)$ zu

$$(b_x - c_x)(a_y - c_y) - (a_x - c_x)(b_y - c_y) = 0 \tag{B}$$

vereinfachen lässt, da für $(b_x = c_x) \wedge (b_y = c_y) = \mathtt{true}$ die erste Klammer zu $(0 = 0)$ und damit ebenfalls zu `true` auswertet. Dasselbe gilt für den ersten und dritten Ausdruck der oder-Verknüpfung.

(B) ist die universelle Formel der geometrischen Bedingung `is_collinear(A,B,C)` und als Verschwinden eines polynomialen Ausdrucks in den Koordinaten der universellen Realisierungen von A, B, C angeschrieben. Formeln ähnlicher Gestalt ergeben sich aus anderen geometrischen Bedingungen. Untersuchen wir etwa, ob M tatsächlich der Umkreismittelpunkt ist, so sind die booleschen Formeln

```
(sqrdist(M,A)=sqrdist(M,B)) and (sqrdist(M,A)=sqrdist(M,C));
```

auf obiger Konfiguration $\mathcal{K}$ auszuwerten. Auch diese können als Verschwinden polynomialer Ausdrücke

```
is(sqrdist(M,A)-sqrdist(M,B)=0) and is(sqrdist(M,A)-sqrdist(M,C)=0);
```

angeschrieben werden. Für obige universelle Realisierung der Endkonfiguration (A, B, C, g_1, g_2, M) ergibt sich

$$a_x^2 - 2\,m_x\,a_x + a_y^2 - 2\,m_y\,a_y - b_x^2 + 2\,m_x\,b_x - b_y^2 + 2\,m_y\,b_y$$

als universelle Formel für `sqrdist(M,A)-sqrdist(M,B)`, die im Koordinatenmodell über $k(X)$ zu null vereinfacht, wenn die oben berechneten Ausdrücke für M eingesetzt werden.

Definition 10 *Seien $T_1, \ldots, T_n$ geometrische Typen. Eine (informatische) Funktion*

$$\phi : T_1 \times \cdots \times T_n \to \texttt{boolean}$$

zusammen mit einer Berechnungsvorschrift

$$\phi.\texttt{c} : C(T_1) \times \cdots \times C(T_n) \to \texttt{Boolean}$$

und einer Degenerationsbedingung

$$\phi.\texttt{DG} : C(T_1) \times \cdots \times C(T_n) \to \texttt{Boolean},$$

so dass die Berechnungsvorschrift auf allen Tupeln $(c_1, \ldots, c_n)$ ausführbar ist, für die

$$\phi.\texttt{DG}(c_1, \ldots, c_n) = \texttt{false}$$

gilt, bezeichnen wir als geometrische Eigenschaft.
Sind $o_1 \in T_1, \ldots, o_n \in T_n$ Objekte der richtigen Typen und $o_1.\texttt{u}, \ldots, o_n.\texttt{u}$ deren universelle Realisierungen, so bezeichnen wir

$$\phi.\texttt{uf} = \phi.\texttt{c}(o_1.\texttt{u}, \ldots, o_n.\texttt{u}) \in \texttt{BooleanExpression}$$

als die zugehörige universelle Formel *dieser geometrischen Eigenschaft und*

$$\phi.\texttt{DGF} = \phi.\texttt{DG}(o_1.\texttt{u}, \ldots, o_n.\texttt{u}) \in \texttt{BooleanExpression}$$

als die zugehörige universelle Degenerationsbedingung *dieser Eigenschaft.*

Sowohl die zu beweisenden Aussagen der bisher betrachteten geometrischen Sätze als auch die Degenerationsbedingungen sind solche geometrischen Eigenschaften.

Wir können damit die typische Konstellation für eine ganze Klasse geometrischer Sätze genauer beschreiben:

> *Wenn* sich eine gewisse Menge geometrischer Objekte in einer durch eine Konstruktionsbeschreibung $\mathcal{K}$ definierten Abhängigkeitsrelation befindet, *dann* gilt für diese Objekte eine zusätzliche geometrische Eigenschaft ϕ.

Genauer: Für jede zulässige spezielle Interpretation C_E der Endkonfiguration von $\mathcal{K}$ im Koordinatenmodell über K, also eine solche mit $\mathcal{K}.\mathtt{DG}(C_E) = \mathtt{false}$ (Ausführbarkeit der Konstruktionsvorschrift), ist $\phi.\mathtt{DG}(C_E) = \mathtt{false}$ (Bestimmtheit der Schlussfolgerung) und $\phi.\mathtt{c}(C_E) = \mathtt{true}$:

$$\forall\ C_E\Big(\mathtt{not}\ \mathcal{K}.\mathtt{DG}(C_E) \Rightarrow\ (\mathtt{not}\ \phi.\mathtt{DG}(C_E)\ \wedge\ \phi.\mathtt{c}(C_E))\Big) \tag{C}$$

(C) besteht aus zwei Teilen

$$\forall\ C_E\Big(\mathtt{not}\ \mathcal{K}.\mathtt{DG}(C_E) \Rightarrow \mathtt{not}\ \phi.\mathtt{DG}(C_E)\Big) \tag{C.1}$$

und

$$\forall\ C_E\Big(\mathtt{not}\ \mathcal{K}.\mathtt{DG}(C_E) \Rightarrow \phi.\mathtt{c}(C_E)\Big)\,, \tag{C.2}$$

wobei (C.1) in der Kontraposition auch als

$$\forall\ C_E\Big(\phi.\mathtt{DG}(C_E) \Rightarrow \mathcal{K}.\mathtt{DG}(C_E)\Big) \tag{C.3}$$

angeschrieben werden kann. Dies ist selbst wieder ein – gewöhnlich wesentlich einfacher zu beweisender – geometrischer Satz; meist ist die Degenerationsbedingung $\phi.\mathtt{DG}$ sowieso leer. Wir wollen voraussetzen, dass dieser Satz gilt, so dass sich unsere Behauptung auf (C.2) reduziert, was äquivalent auch als

$$\forall\ C_E\Big(\mathcal{K}.\mathtt{DG}(C_E)\ \vee\ \phi.\mathtt{c}(C_E)\Big) \tag{C.4}$$

angeschrieben werden kann – eine spezielle Realisierung der gegebenen Konfiguration ist entweder degeneriert für die Konstruktionsvorschrift *oder* die Schlussfolgerung gilt. Sätze dieser Struktur bezeichnen wir als *geometrische Sätze vom konstruktiven Typ.*

Abschließend sei angemerkt, dass die bisher betrachteten Beispiele immer vom Modell der affinen Punktkoordinaten ausgingen. Für das Modell mit homogenen Punktkoordinaten lässt sich sogar erreichen, dass alle universellen Formeln polynomial sind, da durch Skalieren mit einem entsprechenden Faktor in den universellen Formeln immer Nennerfreiheit erreicht werden kann. Da die Klasse der polynomialen Ausdrücke nicht verlassen wird, wenn man in polynomialen Ausdrücken Variablen durch Polynome ersetzt, gelten dieselben Ausführungen wie oben, wenn man „rationale Ausdrücke“ durch „polynomiale Ausdrücke“ ersetzt. Dabei können überhaupt keine rechnerischen Ausnahmen mehr auftreten, sondern nur die geometrische Ausnahme, dass sich $(0 : 0 : 0)$ für die Koordinaten einer speziellen Realisierung ergibt. Nach dem Satz über starke Nulltester für Polynome lassen sich solche Fragen effektiv entscheiden.

3.5 Der Mechanisierungssatz für geometrische Sätze vom rationalen konstruktiven Typ

Untersuchen wir nun die Beweiskraft der ausgeführten symbolischen Berechnungen für Sätze vom konstruktiven Typ näher. Sei dazu $\mathcal{K}$ eine Konfiguration, die durch eine Konstruktionsbeschreibung

$$\mathcal{K} : \Gamma_S = \Gamma_0 \xrightarrow{\widetilde{w}_1} \Gamma_1 \xrightarrow{\widetilde{w}_2} \ldots \xrightarrow{\widetilde{w}_N} \Gamma_N = \Gamma_E$$

erzeugt wird, wie bisher $O_S = (o_1, \ldots, o_m)$ die Start- und $O_E = (o_1, \ldots, o_{m+N})$ die Endkonfiguration.

Wir fixieren wieder universelle Realisierungen $A_E = (o_1.\mathtt{u}, \ldots, o_{m+N}.\mathtt{u})$ der Objekte aus O_E und unterteilen die Menge der dabei eingeführten Variablen in zwei disjunkte Teilmengen X und Y,

wobei die Variablen aus X in den universellen Realisierungen $A_S = (o_1.\mathtt{u}, \ldots, o_m.\mathtt{u})$ der unabhängigen Objekte vorkommen und die Variablen aus Y in den universellen Realisierungen $o_{m+1}.\mathtt{u}, \ldots, o_{m+N}.\mathtt{u}$ der abgeleiteten Objekte.

Sei schließlich ϕ eine geometrische Eigenschaft, die auf dieser Konfiguration „allgemein“ gelten soll, d. h. es ist zu zeigen, dass für jede zulässige spezielle Interpretation C_E der Endkonfiguration $\phi(C_E) = \mathtt{true}$ gilt:

$$\forall\, C_E \Big(\mathcal{K}.\mathtt{DG}(C_E) \;\vee\; \phi.\mathtt{c}(C_E) \Big) \tag{C.4}$$

Sei – etwas detaillierter als im letzten Abschnitt –

- $A_i = (o_1.\mathtt{u}, \ldots, o_m.\mathtt{u}, o_{m+1}.\mathtt{u}, \ldots, o_{m+i}.\mathtt{u})$,
- $B_i = (o_1.\mathtt{u}, \ldots, o_m.\mathtt{u}, o_{m+1}.\mathtt{uf}, \ldots, o_{m+i}.\mathtt{uf})$ die aus der universellen Startkonfiguration erzeugte Realisierung (universelle Formeln) der Teilkonstruktion $\mathcal{K}^{(i)}$,
- $C_S = (o_1.\mathtt{c}, \ldots, o_m.\mathtt{c})$ eine zulässige spezielle Interpretation der Startkonfiguration über K und
- $C_i = (o_1.\mathtt{c}, \ldots, o_m.\mathtt{c}, o_{m+1}.\mathtt{c}, \ldots, o_{m+i}.\mathtt{c})$ die aus C_S erzeugte spezielle Interpretation der Teilkonstruktion $\mathcal{K}^{(i)}$.

B_i entsteht aus A_i durch die Substitutionen $s_i, \ldots, s_1$ wie im letzten Abschnitt beschrieben, die zu einer Variablensubstitution $Y \to Y(X)$ zusammengefasst werden können.

Sei weiter $X \to X_0$ die Variablensubstitution, die A_S in C_S überführt, so dass also $o_i.\mathtt{c} = \mathtt{subst}(o_i.\mathtt{u}, X \to X_0)$ für $i = 1, \ldots, m$ gilt. Die Existenz von $X \to X_0$ ergibt sich aus der Universalitätseigenschaft von A_S.

Wir zeigen mit Induktion, dass für $i \geq 0$

$$o_{m+i+1}.\mathtt{c} = \mathtt{subst}(o_{m+i+1}.\mathtt{uf}, X \to X_0) \tag{A}$$

gilt, d. h. dass es egal ist, ob die Koordinaten von $o_{m+i+1}.\mathtt{c}$ aus den speziellen Koordinaten von C_i mit der Berechnungsvorschrift $w_{i+1}.\mathtt{c}$ bestimmt werden (linke Seite) oder aus der universellen Formel $o_{m+i+1}.\mathtt{uf}(X) = \mathtt{subst}(w_{i+1}.\mathtt{uf}(X, Y), Y \to Y(X))$ durch Substitution $X \to X_0$ (rechte Seite). Diese Eigenschaft wird gerade durch die Kommutativität des folgenden Diagramms ausgedrückt

$$\begin{array}{ccc} B_i & \xrightarrow{w_{i+1}.\mathtt{c}} & o_{m+i+1}.\mathtt{uf} \\ {\scriptstyle X\to X_0}\big\downarrow & & \big\downarrow{\scriptstyle X\to X_0} \\ C_i & \xrightarrow{w_{i+1}.\mathtt{c}} & o_{m+i+1}.\mathtt{c} \end{array}$$

und bedeutet im Kern, dass es für zwei Ausdrücke $A, B \in Q(R)$ egal ist, ob sie zuerst arithmetisch zu $A \circ B$ verknüpft (obere Zeile) und dann die Variablen mit $X \to X_0$ ersetzt werden, oder ob diese Ersetzungen unmittelbar für A, B erfolgen und dann die Verknüpfung in K berechnet wird:

$$\mathtt{subst}(A \circ B, X \to X_0) = \mathtt{subst}(A, X \to X_0) \circ \mathtt{subst}(B, X \to X_0)\,,$$

wobei $\circ$ für eine arithmetische Operation steht. Wegen der Universalitätseigenschaft von R lässt sich $X \to X_0$ aber eindeutig zu einem Ringhomomorphismus $\pi : R \to K$ fortsetzen und die obige Eigenschaft ist gerade die Operationstreue von π für $\circ \in \{+, -, \cdot\}$.

Etwas mehr arbeiten muss man für die Division: es muss garantiert werden, dass bei der Substitution $X \to X_0$ keine Nenner zu null werden. Wir verfolgen den Weg auftretender Nenner deshalb genauer. Ist w eines der eingesetzten Konstruktionswerkzeuge und $s(X, Y)$ ein in der universellen Formel $w.\mathtt{uf}$ vorkommender Nenner, so muss $w.\mathtt{DG}$ ausschließen, dass Belegungen

(X_0, Y_0) mit $s(X_0, Y_0) = 0$ zulässig sind. Auf der Ebene der universellen Degenerationsbedingung $w.\mathtt{DGF}(X, Y)$ *als Boolescher Ausdruck* über $Q(R)$ gilt aber vereinbarungsgemäß die Tautologie $\mathtt{not}\ w.\mathtt{DGF}(X, Y) \Rightarrow s(X, Y) \neq 0$ oder äquivalent dazu $s(X, Y) = 0 \Rightarrow w.\mathtt{DGF}(X, Y)$.

Kumuliert über alle eingesetzten Konstruktionswerkzeuge ($s(X, Y)$ ist Nenner in *einem* der vorkommenden Konstruktionswerkzeuge) erhalten wir daraus die Tautologie

$$s(X, Y) = 0 \Rightarrow \bigvee_{i>0} w_i.\mathtt{DGF}(X, Y)$$

Boolescher Ausdrücke über $Q(R)$. In den universellen Formeln der Elemente von B_i wird $s(X, Y)$ durch $s(X, Y(X))$ ersetzt. Für diese Formeln gilt dann die Tautologie

$$s(X, Y(X)) = 0 \Rightarrow \bigvee_{i>0} \mathtt{subst}(w_i.\mathtt{DGF}(X, Y), Y \to Y(X))$$

Boolescher Ausdrücke über $Q(R)$. Letzteres ist aber genau die universelle Degenerationsbedingung $\mathcal{K}.\mathtt{DGF}(X)$ der Konstruktionsbeschreibung $\mathcal{K}$. Für jede Belegung $X \to X_0$ mit $s(X_0, Y(X_0)) = 0$ gilt also $\mathcal{K}.\mathtt{DGF}(X_0) = \mathtt{true}$, so dass eine solche Realisierung nicht zulässig ist.

Zum Beweis der geometrischen Aussage, dass für jede zulässige spezielle Interpretation C_E der Endkonfiguration $\phi(C_E) = \mathtt{true}$ gilt, betrachten wir das folgende kommutative Diagramm:

$$\begin{array}{ccccc} A_E & \xrightarrow{Y\to Y(X)} & B_E & \xrightarrow{X\to X_0} & C_E \\ \downarrow{\scriptstyle \phi.\mathtt{c}} & & \downarrow{\scriptstyle \phi.\mathtt{c}} & & \downarrow{\scriptstyle \phi.\mathtt{c}} \\ \phi.\mathtt{c}(A_E)(X, Y) & \xrightarrow{Y\to Y(X)} & \phi.\mathtt{c}(B_E)(X) & \xrightarrow{X\to X_0} & \phi.\mathtt{c}(C_E) \end{array}$$

Die oben geführten Beweise verwenden wiederum implizit die Kommutativität dieses Diagramms, d. h. dass der konkrete boolesche Wert $\phi.\mathtt{c}(C_E) \in \mathtt{Boolean}$ gleichberechtigt als Ergebnis der Anwendung von ϕ auf C_E oder aus der booleschen Formel $\phi.\mathtt{c}(B_E)(X)$ durch die Variablenspezifikation $X \to X_0$ und nachfolgende Interpretation des nun variablenfreien booleschen Ausdrucks als boolescher Wert in einer Prädikatenlogik über K gewonnen werden kann. Hierfür müssen wir natürlich zusätzliche Annahmen über die Struktur von $\phi.\mathtt{c}$ treffen.

In allen bisherigen Beispielen war $\phi.\mathtt{c}$ stets eine Formel der Prädikatenlogik 1. Stufe, deren Atome als Verschwinden des Ergebnisses einer gewissen arithmetischen Berechnung angeschrieben werden konnten. Nach den Auswerteregeln prädikatenlogischer Formeln können wir uns in der weiteren Betrachtung auf den Fall beschränken, dass $\phi.\mathtt{c} = \mathtt{iszero}(\psi)$ mit

$$\psi : C(T_1) \times \cdots \times C(T_n) \to K$$

ein solcher atomarer Ausdruck ist. Setzen wir $\psi.\mathtt{uf} = \psi(A_E)$, so ist $\phi.\mathtt{uf} = \mathtt{iszero}(\psi.\mathtt{uf})$ die universelle Formel der geometrischen Eigenschaft ϕ. Wir können $\psi.\mathtt{uf} \in R$ annehmen, da ein rationaler Ausdruck genau dann null wird, wenn das Zählerpolynom null wird. $\psi.\mathtt{uf}$ bezeichnen wir als *polynomiale universelle Formel* der geometrischen Eigenschaft ϕ. Wir erhalten damit das folgende Beweisschema-Diagramm

$$\begin{array}{ccccc} A_E & \xrightarrow{Y\to Y(X)} & B_E & \xrightarrow{X\to X_0} & C_E \\ \downarrow{\scriptstyle \psi} & & \downarrow{\scriptstyle \psi} & & \downarrow{\scriptstyle \psi} \\ \psi(A_E) & \xrightarrow{Y\to Y(X)} & \psi(B_E) & \xrightarrow{X\to X_0} & \psi(C_E) \end{array} \tag{BS.1}$$

das aus denselben Gründen wie oben kommutativ ist. Insbesondere ergibt sich $\psi(B_E)$ aus dem universellen Polynom $\psi.\mathtt{uf}(X, Y) = \psi(A_E)$ durch Spezifikation $Y \to Y(X)$. Eine weitere Simplifikation vereinfachte dann bereits diesen rationalen *Ausdruck* zu null, woraus folgt, dass $\mathtt{iszero}(\psi(B_E))$ bereits als boolesche *Formel* unabhängig von Variablenbelegungen zu $\mathtt{true}$ simplifiziert.

Von ähnlicher Struktur sind die Degenerationsbedingungen $w_i.\texttt{DGF}$. Mehrere oder-verknüpfte Bedingungen lassen sich durch Aufmultiplizieren zu einer zusammenfassen, da ein Produkt nur dann verschieden null ist, wenn alle Faktoren verschieden null sind:

$$\texttt{iszero}(p_1) \vee \cdots \vee \texttt{iszero}(p_k) \iff \texttt{iszero}(p_1 \cdot \ldots \cdot p_k)\,.$$

In praktischen Anwendungen arbeitet man aus Performancegründen dennoch mit mehreren Polynomen.

Kern der gesamten Argumentation ist also die Kommutativität der oben angeführten Diagramme, was äquivalent zur Vertauschbarkeit von Variablensubstitution und Berechnung der entsprechenden universellen Formeln ist. In allen bisher betrachteten Beispielen ist dies durch den polynomialen oder rationalen Charakter der universellen Formeln gewährleistet, da Variablensubstitutionen operationstreu sind, d. h. mit den arithmetischen Operationen in R bzw. im Quotientenkörper $Q(R)$ kommutieren. Wir führen deshalb die folgenden Begriffe ein.

Definition 11 *Einen Konstruktionsschritt $o = w(o_1, \ldots, o_n)$, dessen universelle Formel aus rationalen oder sogar polynomialen Ausdrücken in den Variablen einer universellen Realisierung besteht, bezeichnen wir als* rational *bzw.* polynomial.

Lässt sich $w.\texttt{DGF}$ als $\texttt{iszero}(d(X,Y))$ mit $d(X,Y) \in R$ darstellen, so bezeichnen wir dies als Konstruktionsschritt mit polynomialen Degenerationsbedingungen.

Eine Konstruktionsbeschreibung $\mathcal{K}$ bezeichnen wir als polynomial *bzw.* rational*, wenn sie aus polynomialen bzw. rationalen Konstruktionsschritten mit polynomialen Degenerationsbedingungen aufgebaut ist.*

Eine geometrische Bedingung ϕ, deren universelle Formel sich in der Form $\phi.\texttt{c} = \texttt{iszero}(\psi)$ mit $\psi \in R$ darstellen lässt, bezeichnen wir als polynomiale Bedingung.

Als geometrischen Satz vom rationalen konstruktiven Typ *bezeichnen wir eine rationale Konstruktionsbeschreibung $\mathcal{K}$ zusammen mit einer auf der Endkonfiguration von $\mathcal{K}$ gegebenen polynomialen Bedingung ϕ.*

Wir sagen, dass der Satz gilt*, wenn für jede zulässige spezielle Interpretation C_S der Startkonfiguration von $\mathcal{K}$ über K die Bedingung ϕ auf der zugehörigen Interpretation C_E der Endkonfiguration zu* `true` *auswertet.*

Wie ausgeführt ist zu beachten, dass diese Definitionen nicht nur vom Charakter der Konstruktionswerkzeuge abhängen, sondern auch vom verwendeten Koordinatenmodell. Im Modell der affinen Punktkoordinaten etwa sind einige der bisher betrachteten Konstruktionswerkzeuge nur rationale Werkzeuge, im Modell der homogenen Punktkoordinaten dagegen alle Konstruktionswerkzeuge polynomial.

Alle bisher betrachteten geometrischen Sätze waren Sätze vom rationalen konstruktiven Typ. Ein solcher Satz $(\mathcal{K}, \phi)$ wird durch die folgenden Formeln begleitet:

- Eine universelle Realisierung $A_S(X) = (o_1.\texttt{u}, \ldots, o_m.\texttt{u})$ der Startkonfiguration,
- eine universelle Realisierung $A_E(X,Y) = (o_1.\texttt{u}, \ldots, o_{m+N}.\texttt{u})$ der Endkonfiguration,
- universelle Formeln $B_E(X) = \texttt{subst}(A_E, Y \to Y(X))$ für die Realisierung von $\mathcal{K}$ auf der universellen Realisierung A_S der Startkonfiguration,
- die universelle Formel $\Psi(X,Y) = \psi(A_E)$ der Behauptung ϕ sowie
- das Resultat der Substitution $\Psi'(X) = \psi(B_E) = \texttt{subst}(\Psi(X,Y), Y \to Y(X))$.

Für die ausgeführten symbolischen Rechnungen gibt es folgende Alternativen:

1. A_S selbst ist für $\mathcal{K}$ nicht zulässig, d. h. $\mathcal{K}$.`DGF` vereinfacht als `BooleanExpression` zu `true`. Es gibt also auch keine zulässigen speziellen Interpretationen – die Voraussetzungen des Satzes sind widersprüchlich.

   Sei in diesem Fall i minimal mit der Eigenschaft, dass $\texttt{subst}(w_i.\texttt{DGF}, Y \to Y(X))$ zu `true` vereinfacht. Damit kann die Teilkonstruktion $\mathcal{K}^{(i-1)}$ ausgeführt werden und A_S ist für $\mathcal{K}^{(i-1)}$ zulässig.

   Die Nichtausführbarkeit der Konstruktion lässt sich damit als geometrischer Satz vom konstruktiven Typ mit der Konstruktionsbeschreibung $\mathcal{K}^{(i-1)}$ und der Behauptung $w_i.\texttt{DG}$ formulieren: *Wenn* man wie angegeben bis zum Schritt $i-1$ konstruiert, so landet man immer in einer degenerierten Lage des Konstruktionswerkzeugs w_i.

2. Die zurückgegebene rationale Funktion $\Psi'(X)$ simplifiziert zu null.

   Dann gilt die geometrische Aussage für alle zulässigen speziellen Interpretationen C_E der Endkonfiguration, da sich der Wert von ψ auf C_E durch Variablenspezifikation $X \to X_0$ aus Ψ' ergibt. Der Satz ist allgemeingültig.

3. Die zurückgegebene rationale Funktion $\Psi'(X)$ simplifiziert nicht zu null.

   Dann gilt die Aussage für fast alle zulässigen speziellen Interpretationen der Endkonfiguration nicht. Der Satz ist in der formulierten Form nicht allgemeingültig.

   Die Aussage gilt nur unter Zusatzbedingungen, die analytisch das Verschwinden des Zählerpolynoms des simplifizierten Ausdrucks nach sich ziehen müssen.

   Gelingt es, diese Bedingung als geometrische Eigenschaft zu identifizieren, dann lässt sich ein entsprechender geometrischer Satz formulieren. Er ist aber nicht vom konstruktiven Typ.

Ist $d_i(X,Y)$ die polynomiale universelle Formel der Degenerationsbedingung des Konstruktionswerkzeugs w_i, also $w_i.\texttt{DGF} = \texttt{iszero}(d_i)$, so gilt

$$\begin{aligned}\mathcal{K}.\texttt{DGF} &= \bigvee_{i>0} \texttt{subst}(w_i.\texttt{DGF}, Y \to Y(X)) = \bigvee_{i>0} \texttt{iszero}(\texttt{subst}(d_i, Y \to Y(X))) \\ &= \texttt{iszero}(\texttt{subst}(d_1 \cdot \ldots \cdot d_N, Y \to Y(X)))\,.\end{aligned}$$

Wir haben damit den folgenden Mechanisierungssatz bewiesen:

Satz 7 *(Über das mechanisierte Beweisen geometrischer Sätze vom rationalen konstruktiven Typ) Sei $(\mathcal{K}, \phi)$ ein geometrischer Satz vom rationalen konstruktiven Typ,*

- *$A_E(X,Y)$ eine universelle Realisierung der Endkonfiguration,*
- *$B_E(X) = \texttt{subst}(A_E, Y \to Y(X))$ das Ergebnis der Anwendung von $\mathcal{K}$ auf eine universelle Realisierung A_S der Startkonfiguration,*
- *$\Psi' = \texttt{subst}(\psi(A_E), Y \to Y(X)) \in Q(R)$,*
- *$d_i \in R$ Polynome, so dass $w_i.\texttt{DGF} = \texttt{iszero}(d_i(X,Y))$ gilt und*
- *$d = d_1 \cdot \ldots \cdot d_N$.*

Der Satz ist genau dann gültig, d. h. gilt für alle zulässigen speziellen Interpretationen der Startkonfiguration im Koordinatenmodell über K, wenn Ψ' als rationale Funktion in $k(X)$ zu null vereinfacht werden kann.

Zulässige spezielle Interpretationen C_S über K sind genau diejenigen, welche aus $A_S = B_S$ durch eine Variablenspezifikation $X \to X_0$ gewonnen werden können, für die $\texttt{subst}(d_i, Y \to Y(X), X \to X_0) \neq 0$ für $i = 1, \ldots, N$ oder kurz $\texttt{subst}(d, Y \to Y(X), X \to X_0) \neq 0$ gilt.

Kapitel 4

Geometrische Sätze vom Gleichungstyp

4.1 Vor- und Nachbedingungen. Erste Beispiele für Sätze vom Gleichungstyp

Der bisher betrachtete Mechanisierungsansatz für Geometrietheoreme ging davon aus, dass man einen gegebenen Satz der Geometrie in eine konstruktive Form überführen kann. Das ist aber eine für geometrische Aussagen eher untypische Situation. Die meisten geometrischen Sätze gehen von einer bestimmten (konstruktiv gegebenen) geometrischen Konfiguration $\mathcal{K}$ aus und behaupten dann:

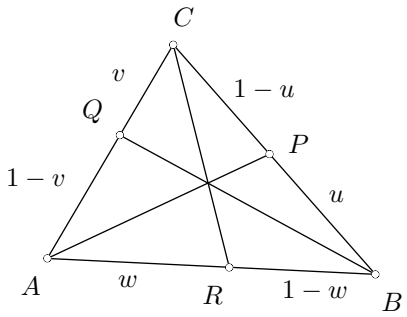
> *Wenn* in $\mathcal{K}$ zusätzlich gewisse geometrische Eigenschaften erfüllt sind, *dann* ergeben sich daraus gewisse andere geometrische Eigenschaften als Konsequenzen.

So behauptet etwa der

Satz 8 (Satz des Ceva) *Gegeben sei ein Dreieck $xABC$ und Punkte P, Q, R auf den Dreiecksseiten mit Teilverhältnissen u, v, w.*
Die Ecktransversalen AP, BQ und CR gehen genau dann durch einen gemeinsamen Punkt, wenn

$$u\,v\,w = (1-u)\,(1-v)\,(1-w)$$

gilt.



In der im vorigen Abschnitt entwickelten Terminologie lässt sich $\mathcal{K}$ wie folgt beschreiben:

```
Start(Point A,B,C; SP U,V,W);
Point P = varpoint(B,C,U);
Point Q = varpoint(C,A,V);
Point R = varpoint(A,B,W);
```

Die universellen Formeln der Konfiguration $\mathcal{K}$ in GeoProver-Notation ergeben sich unmittelbar als

```
A:Point(ax,ay); B:Point(bx,by); C:Point(cx,cy);
P:varpoint(B,C,u); Q:varpoint(C,A,v); R:varpoint(A,B,w);
```

Die universelle Degenerationsbedingung ergibt sich als $\mathcal{K}.\texttt{DGF} = \texttt{is\_collinear}(A.\texttt{u}, B.\texttt{u}, C.\texttt{u})$.

Die weitere geometrische Voraussetzung sowie die Behauptung des Satzes von Ceva sind dann in den Formeln

```
poly:is_concurrent(pp_line(A,P),pp_line(B,Q),pp_line(C,R));
con:u*v*w-(1-u)*(1-v)*(1-w);
```

kodiert.

In der gegebenen universellen Konfiguration $\mathcal{K}$, in deren universellen Formeln die sechs Koordinaten von A, B, C und die Gleiterparameter u, v, w als Variablen vorkommen, ist also zu zeigen, dass die zulässigen speziellen Interpretationen von $\mathcal{K}$ über K, für die *poly* verschwindet, und jene, für die *con* verschwindet, zusammenfallen.

Hierfür sind die Nullstellenmengen der Polynome *poly* (der Voraussetzung) und *con* (der Schlussfolgerung = conclusio) über K miteinander zu vergleichen. Wir wollen hierfür und im Weiteren voraussetzen, dass K ein algebraisch abgeschlossener Körper mit $char(K) = 0$ ist, dass also jedes univariate Polynom $f(x) \in k[x]$ mit Koeffizienten in unserem berechenbaren Grundbereich k auch Nullstellen über $K \supset k$ besitzt. Wir betrachten also ausschließlich geometrische Sätze in einem Koordinatenmodell über einem algebraisch abgeschlossenen Körper.

In obigem Beispiel lässt sich *poly* als $-f_1^2 \cdot f_2$ mit

$$\begin{aligned} f_1 &= a_x\, b_y - a_x\, c_y - a_y\, b_x + a_y\, c_x + b_x\, c_y - b_y\, c_x \\ f_2 &= 2\, u\, v\, w - u\, v - u\, w + u - v\, w + v + w - 1 \end{aligned}$$

darstellen. Also gilt

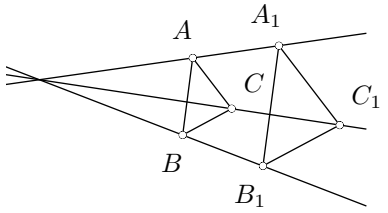
$$\texttt{iszero}(poly) \iff \texttt{iszero}(f_1) \vee \texttt{iszero}(f_2)\,.$$

Analysiert man die beiden anderen Faktoren genauer, so erkennt man, dass f_1 die universelle Formel der Bedingung `is_collinear`(A,B,C) ist und $f_2 = con$ gilt, so dass

$$\texttt{iszero}(poly) \iff \mathcal{K}.\texttt{DGF} \vee \texttt{iszero}(con)$$

gilt. Damit ist der Satz von Ceva entsprechend der im letzten Abschnitt entwickelten Methodik bewiesen.

Ähnlich können wir auch die Aussage des folgenden Satzes formulieren.



Satz 9 (Affiner Satz von Desargue, Teil 2)
Sind in den Dreiecken ABC und $A_1B_1C_1$ entsprechende Dreiecksseiten zueinander parallel, also $AB \parallel A_1B_1$, $AC \parallel A_1C_1$ und $BC \parallel B_1C_1$, so sind die Geraden AA_1, BB_1 und CC_1 konkurrent.

Die universellen Formeln der Beschreibung der zu Grunde liegenden Konfiguration

$$\mathcal{K} = \texttt{Start}(\texttt{Point}\ A, B, C, A_1, B_1, C_1)$$

bestehen aus genau den universellen Realisierungen der 6 Punkte A, B, C, A_1, B_1, C_1.

```
A:Point(ax,ay); B:Point(bx,by); C:Point(cx,cy);
A1:Point(dx,dy); B1:Point(ex,ey); C1:Point(fx,fy);
```

Die Voraussetzung lässt sich in den drei universellen Formeln

```
polys: [
is_parallel(pp_line(A,B),pp_line(A1,B1)),
is_parallel(pp_line(B,C),pp_line(B1,C1)),
is_parallel(pp_line(A,C),pp_line(A1,C1)) ];
```

kodieren, die Behauptung in der universellen Formel

```
con:is_concurrent(pp_line(A,A1),pp_line(B,B1),pp_line(C,C1));
```

Die Voraussetzung entspricht hier also einer Menge von drei Polynomen

$$\begin{aligned}
&a_x\,d_y - a_x\,e_y - a_y\,d_x + a_y\,e_x - b_x\,d_y + b_x\,e_y + b_y\,d_x - b_y\,e_x\\
&b_x\,e_y - b_x\,f_y - b_y\,e_x + b_y\,f_x - c_x\,e_y + c_x\,f_y + c_y\,e_x - c_y\,f_x\\
&a_x\,d_y - a_x\,f_y - a_y\,d_x + a_y\,f_x - c_x\,d_y + c_x\,f_y + c_y\,d_x - c_y\,f_x\,,
\end{aligned}$$

die Behauptung dem deutlich komplizierteren Polynom

$$\begin{aligned}
&-a_x\,b_x\,c_y\,d_y + a_x\,b_x\,c_y\,e_y + a_x\,b_x\,d_y\,f_y - a_x\,b_x\,e_y\,f_y + a_x\,b_y\,c_x\,d_y - a_x\,b_y\,c_x\,f_y - a_x\,b_y\,c_y\,e_x +\\
&a_x\,b_y\,c_y\,f_x - a_x\,b_y\,d_y\,f_x + a_x\,b_y\,e_x\,f_y - a_x\,c_x\,d_y\,e_y + a_x\,c_x\,e_y\,f_y + a_x\,c_y\,d_y\,e_x - a_x\,c_y\,e_y\,f_x -\\
&a_x\,d_y\,e_x\,f_y + a_x\,d_y\,e_y\,f_x - a_y\,b_x\,c_x\,e_y + a_y\,b_x\,c_x\,f_y + a_y\,b_x\,c_y\,d_x - a_y\,b_x\,c_y\,f_x - a_y\,b_x\,d_x\,f_y +\\
&a_y\,b_x\,e_y\,f_x - a_y\,b_y\,c_x\,d_x + a_y\,b_y\,c_x\,e_x + a_y\,b_y\,d_x\,f_x - a_y\,b_y\,e_x\,f_x + a_y\,c_x\,d_x\,e_y - a_y\,c_x\,e_x\,f_y -\\
&a_y\,c_y\,d_x\,e_x + a_y\,c_y\,e_x\,f_x + a_y\,d_x\,e_x\,f_y - a_y\,d_x\,e_y\,f_x + b_x\,c_x\,d_y\,e_y - b_x\,c_x\,d_y\,f_y - b_x\,c_y\,d_x\,e_y +\\
&b_x\,c_y\,d_y\,f_x + b_x\,d_x\,e_y\,f_y - b_x\,d_y\,e_y\,f_x + b_y\,c_x\,d_x\,f_y - b_y\,c_x\,d_y\,e_x + b_y\,c_y\,d_x\,e_x - b_y\,c_y\,d_x\,f_x -\\
&b_y\,d_x\,e_x\,f_y + b_y\,d_y\,e_x\,f_x - c_x\,d_x\,e_y\,f_y + c_x\,d_y\,e_x\,f_y + c_y\,d_x\,e_y\,f_x - c_y\,d_y\,e_x\,f_x
\end{aligned}$$

mit 48 Summanden vom Grad 4. Dass jede gemeinsame Nullstelle der Polynome *polys* eine Nullstelle von *con* ist, ist hier nicht mehr so einfach zu erkennen. Der Satz ist jedoch offensichtlich ein Satz der affinen Geometrie, so dass wir zur Algebraisierung ein beliebiges affines Koordinatensystem verwenden können, bzw. alternativ bei vorgegebenem Koordinatensystem durch eine geeignete affine Transformation die Konfiguration unter Erhaltung aller geometrischen Eigenschaften in eine solche transformieren, für die

```
A:Point(0,0); B:Point(0,1); C:Point(1,0);
```

gilt[1]. Mit diesen speziellen Koordinaten erhalten wir

$$\texttt{polys} = [\,d_x - e_x, -e_x - e_y + f_x + f_y, -d_y + f_y\,]$$

und als Behauptung

$$\texttt{con} = -d_x\,e_x\,f_y - d_x\,e_y\,f_y + d_x\,f_y + d_y\,e_x\,f_x + d_y\,e_x\,f_y - d_y\,e_x.$$

polys ist ein homogenes lineares Gleichungssystem, welches sich einfach lösen lässt

```
sol:solve(polys,[fy, ex, fx, dx, ey, dy]);
```

$$[[f_y = r_4, e_x = r_6, f_x = r_6 + r_5 - r_4, d_x = r_6 + r_5 - r_4, e_y = r_5, d_y = r_5]]$$

Die Lösungsmenge hängt von drei Parametern (r_4, r_4, r_6) ab. In die Behauptung eingesetzt erhalten wir

```
expand(subst(sol[1],con));
```

$$0$$

Der entstehende Ausdruck vereinfacht bereits als Polynom in (r_4, r_4, r_6) zu null, woraus die Behauptung des Satzes von Desargue für alle zulässigen speziellen Interpretationen folgt.

Wir hätten auch wie im letzten Abschnitt die Variablen $X = (d_x, d_y, e_x, e_y, f_x, f_y)$ in abhängige $Y = (e_y, f_x, f_y)$ und unabhängige $U = (d_x, d_y, e_x)$ unterteilen und die abhängigen durch die unabhängigen ausdrücken können:

[1] Durch diese Einschränkung der speziellen Interpretationen ist automatisch die Nichtdegenerationsbedingung `is_collinear`(A, B, C) erfüllt, die den Beweisgang „im Allgemeinen“ erheblich verkompliziert.

```
sol:solve(polys,[ey,fx,fy]);
```

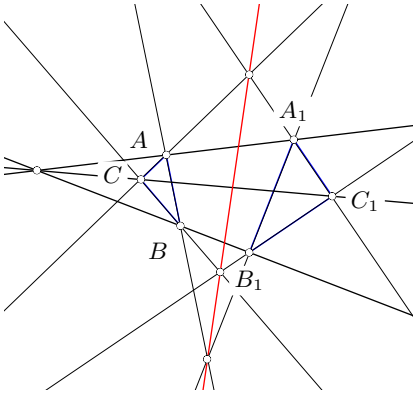
$$[[e_y = d_y, f_x = d_x, f_y = e_x + d_y - d_x]]$$

sol entspricht der Ersetzungsformel $Y \to Y(X)$. Setzen wir dies in die universelle Formel der Behauptung *con* ein, so ergibt sich ebenfalls

```
expand(subst(sol[1],con));
```

$$0$$

Für den **allgemeinen Satz von Desargue** können wir ähnlich argumentieren. Wir beginnen wieder mit der Konfiguration




```
A:Point(ax,ay);
B:Point(bx,by);
C:Point(cx,cy);
A1:Point(dx,dy);
B1:Point(ex,ey);
C1:Point(fx,fy);
```

und erweitern diese um die universellen Formeln der Schnittpunktkoordinaten

```
X:intersection_point(pp_line(A,B),pp_line(A1,B1));
Y:intersection_point(pp_line(A,C),pp_line(A1,C1));
Z:intersection_point(pp_line(B,C),pp_line(B1,C1));
```

Voraussetzung und Behauptung des Satzes lassen sich dann in den rationalen Ausdrücken

```
poly:is_concurrent(pp_line(A,A1),pp_line(B,B1),pp_line(C,C1));
con:is_collinear(X,Y,Z);
```

kodieren. Während *poly* ein irreduzibles Polynom vierten Grades mit 48 Termen ist, hat *con* eine deutlich komplexere Struktur. Der Nenner ist das Produkt der Degenerationsbedingungen

$$\texttt{is\_parallel}(\texttt{pp\_line}(A,B),\texttt{pp\_line}(A_1,B_1)) \text{ usw.}$$

des für die Konstruktion von X, Y und Z angewendeten Konstruktionswerkzeugs.

Da der Satz ein Satz (mindestens) der affinen Geometrie ist, können wir durch Auswahl des Koordinatensystems wieder die Rechnung vereinfachen. Wir setzen

```
A1:Point(0,0); B1:Point(0,1); C1:Point(1,0);
```

und erhalten nun

$$con = \frac{(a_y\,b_x - a_x\,c_y + a_x\,b_x\,c_y - a_y\,b_x\,c_x + a_x\,b_y\,c_y - a_y\,b_x\,c_y)\,(a_x\,b_y - a_y\,b_x - a_x\,c_y + a_y\,c_x + b_x\,c_y - b_y\,c_x)}{(b_x + b_y - c_x - c_y)\,(a_x - b_x)\,(a_y - c_y)}$$

Der Nenner ist wieder gerade die Degenerationsbedingung des Werkzeugs in den reduzierten Koordinaten, der Zähler lässt sich als $poly \cdot \texttt{is\_collinear}(A,B,C)$ darstellen. Damit gilt also auch hier $poly(X_0) = 0 \iff con(X_0) = 0$ für jede zulässige spezielle Interpretation $X \to X_0$ der Grundkonfiguration über K.

Aufgabe: Formulieren und untersuchen Sie auf dieselbe Weise den Satz von Pappus.

Wir sehen also, dass die algebraische Formulierung allgemeinerer geometrischer Sätze mit den Formeln der universellen Realisierung B_E einer konstruktiv erzeugten Basiskonfiguration $\mathcal{K}$ startet,

zusätzliche geometrische Voraussetzungen in einer Menge $F = \{f_1, \ldots, f_m\} \subset R$ von Polynomen kodiert, so dass die Behauptung ebenfalls in einem Polynom $g \in R$ ausgedrückt werden kann. Den geometrischen Satz zu beweisen bedeutet zu zeigen, dass für jede zulässige spezielle Interpretation von $\mathcal{K}$ im Koordinatenmodell über K, also jede zulässige Variablenspezifikation $X \to X_0 \in K^n$, für die $F(X_0) = 0$ gilt, auch $g(X_0) = 0$ folgt:

$$\forall\ X_0 \in K^n\ \left(\mathcal{K}.\texttt{DGF}(X_0) \neq 0 \Rightarrow (F(X_0) = 0 \Rightarrow g(X_0) = 0)\right).$$

4.2 Weitere geometrische Typen

Abstände

Einen weiteren geometrischen Typ `Distance` hatten wir bereits implizit als Rückgabetyp der Funktion `sqrdist` eingeführt. Für ihn gilt $C(D) = K$.

Kreise

Mit dem in diesem Kapitel entwickelten Ansatz können wir auch nichtlineare geometrische Bedingungen erfassen. Diese Bedingungen treten auf, wenn geometrische Linien mehrere Schnittpunkte haben (Schnitt zweier Kreise, Schnitt von Kreis und Gerade) oder wenn es mehrere Geraden mit einer gegebenen Eigenschaft gibt (Parallelenpaar zu einer gegebenen Geraden in gegebenem Abstand, Winkelhalbierendenpaar eines gegebenen Winkels). Diese können ihrem Wesen nach nicht konstruktiver Natur sein, da nicht in jedem Schritt ein *eindeutig bestimmtes* geometrisches Objekt konstruiert wird.

Betrachten wir zunächst, wie sich die eben aufgelisteten geometrischen Bedingungen durch die Koordinaten der an ihnen beteiligten geometrischen Objekte beschreiben lassen.

Ähnlich wie eine Gerade kann man einen Kreis durch die Koordinaten seines Mittelpunkts $M = (m_x, m_y)$ und eines Punkts $P = (p_x, p_y)$ auf der Peripherie beschreiben. Statt des Radius r verwenden wir dessen Quadrat $r^2 = (p_x - m_x)^2 + (p_y - m_y)^2$, den *Sqradius*, um im Bereich polynomialer Ausdrücke zu bleiben.

Als Objekt der analytischen Geometrie wird ein solcher Kreis durch seine Gleichung

$$(x - m_x)^2 + (y - m_y)^2 - r^2 = (x^2 + y^2) + c_2\, x + c_3\, y + c_4$$

mit $c_2 = -2\, m_x,\ c_3 = -2\, m_y,\ c_4 = m_x^2 + m_y^2 - r^2$ gegeben, die durch die drei Parameter $c = (c_2, c_3, c_4)$ eindeutig charakterisiert werden kann. Umgekehrt lassen sich aus den Kreisparametern (c_2, c_3, c_4) auch Zentrum (m_x, m_y) und der Sqradius r^2 unmittelbar polynomial zurückgewinnen.

Im GeoProver-Paket stehen dafür

> die Funktionen `circle_center(c:Circle):Point`
> und `circle_sqradius(c:Circle):Scalar`,
> der Konstruktor `pc_circle(M:Point, P:Point):Circle`
> und die polynomiale Bedingung `on_circle(P:Point, c:Circle)`

zur Verfügung. `pc_cricle` bestimmt die Koordinaten eines Kreises mit Zentrum M und einem gegebenen Punkt A auf der Peripherie und `on_circle` gibt für einen Punkt $P(p_x, p_y)$ den Wert des Ausdrucks $(p_x^2 + p_y^2) + c_2\, p_x + c_3\, p_y + c_4$ zurück.

Betrachten wir nun den Kreis, der durch drei vorgegebene Punkte A, B, C verläuft. Universelle Formeln für die Koordinaten des Mittelpunkts $M = (m_x, m_y)$ dieses Kreises können wir aus den Gleichungen

```
A:Point(ax,ay); B:Point(bx,by); C:Point(cx,cy); M:Point(mx,my);
polys:[ sqrdist(M,A)-sqrdist(M,B), sqrdist(M,A)-sqrdist(M,C) ];
```

bestimmen, wobei für A, B, C universelle Realisierungen eingesetzt sind. Das entsprechende Gleichungssystem

$$2\,m_x\,(-a_x+b_x)+2\,m_y\,(-a_y+b_y)+a_x^2+a_y^2-b_x^2-b_y^2=0,$$
$$2\,m_x\,(-a_x+c_x)+2\,m_y\,(-a_y+c_y)+a_x^2+a_y^2-c_x^2-c_y^2=0$$

ist linear in (m_x, m_y) und besitzt eine eindeutige Lösung genau dann, wenn die zugehörige Determinante, die wieder einmal genau mit der Kollinearitätsbedingung `is_collinear(A,B,C)` übereinstimmt, verschieden von 0 ist.

```
solve(polys,[mx,my]);
```

ergibt dann die universellen Formeln für die Mittelpunktskoordinaten, die wir nun fest als Prozedur zur weiteren Verwendung einbrennen können. Alternativ hätten wir den Mittelpunkt natürlich auch wie bisher berechnen können:

```
M:intersection_point(p_bisector(B,C), p_bisector(C,A));
```

Für den Kreis durch drei vorgegebene Punkte A, B, C erhalten wir

$$\begin{aligned}
M &= \left(\frac{\Delta_1}{\Delta}, \frac{\Delta_2}{\Delta}, \frac{\Delta_3}{\Delta}\right) \text{ mit}\\
\Delta_1 &= -{a_x}^2\,b_y + {a_x}^2\,c_y - {a_y}^2\,b_y + {a_y}^2\,c_y + a_y\,{b_x}^2 + a_y\,{b_y}^2 - a_y\,{c_x}^2 - a_y\,{c_y}^2 - {b_x}^2\,c_y\\
&\quad - {b_y}^2\,c_y + b_y\,{c_x}^2 + b_y\,{c_y}^2\\
\Delta_2 &= {a_x}^2\,b_x - {a_x}^2\,c_x - a_x\,{b_x}^2 - a_x\,{b_y}^2 + a_x\,{c_x}^2 + a_x\,{c_y}^2 + {a_y}^2\,b_x - {a_y}^2\,c_x + {b_x}^2\,c_x\\
&\quad - b_x\,{c_x}^2 - b_x\,{c_y}^2 + {b_y}^2\,c_x\\
\Delta_3 &= -{a_x}^2\,b_x\,c_y + {a_x}^2\,b_y\,c_x + a_x\,{b_x}^2\,c_y + a_x\,{b_y}^2\,c_y - a_x\,b_y\,{c_x}^2 - a_x\,b_y\,{c_y}^2 - {a_y}^2\,b_x\,c_y\\
&\quad + {a_y}^2\,b_y\,c_x - a_y\,{b_x}^2\,c_x + a_y\,b_x\,{c_x}^2 + a_y\,b_x\,{c_y}^2 - a_y\,{b_y}^2\,c_x\\
\Delta &= a_x\,b_y - a_x\,c_y - a_y\,b_x + a_y\,c_x + b_x\,c_y - b_y\,c_x
\end{aligned}$$

Die Parameter dieses Kreises sind rationale Funktionen der Parameter der Punkte A, B, C, wobei im Nenner das Polynom $\Delta = \texttt{is\_collinear}(A, B, C)$ auftritt. Um solche Nenner zu vermeiden, wollen wir ähnlich wie für Geraden auch für Kreise homogene Koordinaten verwenden, d. h. einen Kreis durch ein Tupel $c = (c_1 : c_2 : c_3 : c_4)$ charakterisieren, das für die Punktmenge

$$\big\{(x, y)\ :\ c_1(x^2 + y^2) + c_2\,x + c_3\,y + c_4 = 0\big\}$$

steht. Als homogene Koordinaten für einen Kreis $c = \texttt{p3\_circle}(A, B, C)$ ergibt sich damit $c = (\Delta : \Delta_1 : \Delta_2 : \Delta_3)$ mit den oben berechneten Polynomen.

Wir wollen als nächstes charakterisieren, wann vier Punkte A, B, C, D auf einer gemeinsamen Kreislinie liegen. Mit den oben berechneten Koordinaten des Umkreismittelpunkts M von $\Delta\,ABC$ ergibt sich die gesuchte Bedingung aus

```
D:Point(dx,dy);
on_circle(D,p3_circle(A,B,C));
```

dessen Zähler ein Polynom vierten Grades ist, welches die entsprechende Bedingung beschreibt.

Satz 10 *Vier Punkte $A = (a_x, a_y), B = (b_x, b_y), C = (c_x, c_y), D = (d_x, d_y)$ liegen genau dann auf einem gemeinsamen Kreis, wenn*

$$\begin{aligned}
pol_1 &= {a_x}^2\,b_x\,c_y - {a_x}^2\,b_x\,d_y - {a_x}^2\,b_y\,c_x + {a_x}^2\,b_y\,d_x + {a_x}^2\,c_x\,d_y - {a_x}^2\,c_y\,d_x - a_x\,{b_x}^2\,c_y +\\
&\quad a_x\,{b_x}^2\,d_y - a_x\,{b_y}^2\,c_y + a_x\,{b_y}^2\,d_y + a_x\,b_y\,{c_x}^2 + a_x\,b_y\,{c_y}^2 - a_x\,b_y\,{d_x}^2 - a_x\,b_y\,{d_y}^2 - a_x\,{c_x}^2\,d_y -
\end{aligned}$$

$$a_x\,{c_y}^2\,d_y+a_x\,c_y\,{d_x}^2+a_x\,c_y\,{d_y}^2+{a_y}^2\,b_x\,c_y-{a_y}^2\,b_x\,d_y-{a_y}^2\,b_y\,c_x+{a_y}^2\,b_y\,d_x+{a_y}^2\,c_x\,d_y-{a_y}^2\,c_y\,d_x+a_y\,{b_x}^2\,c_x-a_y\,{b_x}^2\,d_x-a_y\,b_x\,{c_x}^2-a_y\,b_x\,{c_y}^2+a_y\,b_x\,{d_x}^2+a_y\,b_x\,{d_y}^2+a_y\,{b_y}^2\,c_x-a_y\,{b_y}^2\,d_x+a_y\,{c_x}^2\,d_x-a_y\,c_x\,{d_x}^2-a_y\,c_x\,{d_y}^2+a_y\,{c_y}^2\,d_x-{b_x}^2\,c_x\,d_y+{b_x}^2\,c_y\,d_x+b_x\,{c_x}^2\,d_y+b_x\,{c_y}^2\,d_y-b_x\,c_y\,{d_x}^2-b_x\,c_y\,{d_y}^2-{b_y}^2\,c_x\,d_y+{b_y}^2\,c_y\,d_x-b_y\,{c_x}^2\,d_x+b_y\,c_x\,{d_x}^2+b_y\,c_x\,{d_y}^2-b_y\,{c_y}^2\,d_x$$

verschwindet.

Diese Bedingung kann man auch als Determinante schreiben: Die vier Punkte A, B, C, D liegen genau dann auf einem gemeinsamen Kreis, wenn es eine nichttriviale Lösung $x = (x_1, x_2, x_3, x_4)$ des Gleichungssystems

$$\begin{aligned}
x_1\,(a_x^2+a_y^2)+x_2\,a_x+x_3\,a_y+x_4&=0\\
x_1\,(b_x^2+b_y^2)+x_2\,b_x+x_3\,b_y+x_4&=0\\
x_1\,(c_x^2+c_y^2)+x_2\,c_x+x_3\,c_y+x_4&=0\\
x_1\,(d_x^2+d_y^2)+x_2\,d_x+x_3\,d_y+x_4&=0
\end{aligned}$$

gibt, d. h. wenn

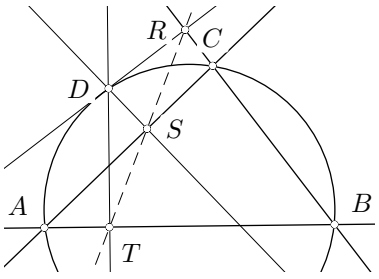
$$\begin{vmatrix} (a_x^2+a_y^2) & a_x & a_y & 1\\ (b_x^2+b_y^2) & b_x & b_y & 1\\ (c_x^2+c_y^2) & c_x & c_y & 1\\ (d_x^2+d_y^2) & d_x & d_y & 1 \end{vmatrix} = 0$$

gilt. Wir können diese universelle Formel als Prozedur

$$\texttt{is\_concyclic}(A,B,C,D) = \texttt{num}(\texttt{on\_circle}(D, \texttt{p3\_circle}(A,B,C)))$$

zur weiteren Verwendung in unsere Programmsammlung aufnehmen.

Damit können wir bereits den folgenden Satz beweisen:



Satz 11 (Satz von der Simsonschen Geraden) *Ist D ein Punkt auf dem Umkreis des Dreiecks ABC, so liegen die Lotfußpunkte R, S, T von D auf die Dreiecksseiten oder deren Verlängerungen auf einer Geraden.*

Wir gehen wieder von den universellen Formeln einer Konfiguration $\mathcal{K}$ mit freien Punkten A, B, C, D und daraus konstruierten Lotfußpunkten R, S, T aus.

```
A:Point(ax,ay); B:Point(bx,by); C:Point(cx,cy); D:Point(dx,dy);
R:pedalpoint(D,pp_line(B,C));
S:pedalpoint(D,pp_line(A,C));
T:pedalpoint(D,pp_line(A,B));
```

Die universelle Formel der geometrischen Eigenschaft, dass D auf dem Umkreis von $\Delta\,ABC$ liegt, ist genau das oben hergeleitete Polynom pol_1 vierten Grades. Die universelle Formel der Behauptung

```
con:is_collinear(R,S,T);
```

ist eine rationale Funktion, deren Zähler ein Polynom pol_2 8. Grades mit 576 Summanden ist und dessen Nenner pol_3 aus dem Produkt

$$pol_3 = \texttt{sqrdist}(A,B)\cdot\texttt{sqrdist}(B,C)\cdot\texttt{sqrdist}(A,C)$$

besteht.

In einem Koordinatenmodell über $K = \mathbb{R}$ gilt $\texttt{sqrdist}(B,C) = 0 \iff B = C$, so dass $\texttt{sqrdist}(B,C) = 0$ eine geometrisch degenerierte Lage kodiert. Das ist in einem Koordinatenmodell über $K = \mathbb{C}$ nicht mehr so, so dass eine Geometrie über diesem Grundbereich andere Eigenschaften hat als die uns vertrauten. Suchen wir nach der Quelle der Nenner $\texttt{sqrdist}(B,C) = b_x^2 - 2\,b_x\,c_x + b_y^2 - 2\,b_y\,c_y + c_x^2 + c_y^2$, so werden wir schnell im Konstruktionswerkzeug fündig, mit dem R konstruiert wurde. In der Tat lässt sich ein Lot nur dann fällen, wenn $B \neq C$ ist, da sonst die Richtung der Basisgeraden unbestimmt ist. Zur Bestimmung des Lotfußpunkts muss aber weiterhin das Lot die Basisgerade schneiden – ein in der reellen Geometrie offensichtlicher Tatbestand, der in der komplexen Geometrie nicht mehr allgemein erfüllt ist: Die Berechnungen zeigen, dass das Lot auf BC genau dann parallel zu dieser Basisgeraden ist, wenn $\texttt{sqrdist}(B,C) = 0$ gilt. Das ergibt sich auch aus unseren früheren Betrachtungen zu homogenen Koordinaten und der Einbettung der affinen in die projektive Ebene – die Lotgerade h durch einen Punkt P zu einer gegebenen Geraden $g = (g_1 : g_2 : g_3)$ kann als Gerade durch P und den Orthogonalpunkt $O_g = (g_1 : g_2 : 0)$ der Geraden g bestimmt werden. Diese ist genau dann parallel zu g, wenn h durch den Fernpunkt $F_g = (-g_2 : g_1 : 0)$ der Geraden g geht, wenn also $O_g = F_g$ gilt. Das ist aber genau dann der Fall, wenn die entsprechenden homogenen Koordinaten proportional sind, wenn also $g_1^2 + g_2^2 = 0$ gilt. g ist also eine Gerade, die auf sich selbst senkrecht steht. Solche (notwendig komplexen) Geraden bezeichnet man auch als *isotrope* Geraden. Im Affinen gibt es keine Lotfußpunkte auf solche Geraden. Die korrekte Degenerationsbedingung des Werkzeugs `pedalpoint(D,g)` ist also in der Tat $g_1^2 + g_2^2 = 0$.

pol_2 lässt sich seinerseits in das Produkt

$$pol_2 = \texttt{is\_collinear}(A,B,C)^2 \cdot pol_1$$

zerlegen, woraus der geforderte Beweis nunmehr leicht abzuleiten ist. Aus dem Beweis

$$\texttt{not}\ \mathcal{K}.\texttt{DG} \Rightarrow \big(\texttt{is\_collinear}(R,S,T) \iff \texttt{iszero}(pol_1) \iff \texttt{is\_concyclic}(A,B,C,D)\big)$$

folgt, dass nicht nur der Satz selbst richtig ist, sondern auch dessen Umkehrung gilt: Sind die Lotfußpunkte R, S, T kollinear, so liegt D auf dem Umkreis von $\Delta\, ABC$.

Winkel und Winkelhalbierende

(Orientierte) Winkel sind als eigenständige Objekte im GeoProver-Paket 1.3 noch nicht enthalten und werden sinnvoll als Tripel von Punkten $\texttt{Angle}(A,B,C)$ aufzufassen sein, wobei B der Scheitelpunkt ist und A und C Punkte auf den Schenkeln des Winkels sind.

Davon zu unterscheiden ist das **Winkelmaß** $m(\alpha)$ des orientierten Winkels zwischen den Geraden $g = BA$ und $h = BC$. Als Maß werden wir den Tangens der Winkelgröße verwenden. Zwischen diesem und den Anstiegen $\tan(\alpha_g) = -g_1/g_2$ bzw. $\tan(\alpha_h) = -h_1/h_2$ der Geraden $g = (g_1, g_2, g_3)$ und $h = (h_1, h_2, h_3)$ besteht folgender Zusammenhang:

$$\tan(\angle(g,h)) = \tan(\alpha_h - \alpha_g) = \frac{\tan(\alpha_h) - \tan(\alpha_g)}{1 + \tan(\alpha_h)\tan(\alpha_g)} = \frac{g_1\,h_2 - g_2\,h_1}{g_1\,h_1 + g_2\,h_2}$$

Die Degenerationsbedingung $g_1\,h_1 + g_2\,h_2 = 0$ entspricht gerade der Orthogonalität der beiden Geraden, die hier aber in keiner Weise einer geometrisch degenerierten Situation entspricht. Wir arbeiten deshalb mit einer homogenen Variante $(w_1 : w_2)$ des Winkelmaßes von α, so dass $\tan(\alpha) = \frac{w_1}{w_2}$ gilt und $(1:0)$ die Winkelgröße $90°$ repräsentiert.

Wir haben so einen weiteren geometrischen Typ `Angle` mit einem entsprechenden Koordinatenmodell $C(A)$ definiert. Im GeoProver-Paket geben die Konstruktionswerkzeuge $\texttt{l2\_angle}(g,h)$ und $\texttt{p3\_angle}(A,B,C)$ jeweils dieses Winkelmaß des orientierten Winkels zwischen dem (geordneten) Geradenpaar $(g = AB, h = BC)$ zurück. Zwei Winkel $(v_1 : v_2)$ und $(w_1 : w_2)$ sind genau

dann gleich, wenn $v_1\,w_2 - v_2\,w_1 = 0$ gilt. Damit lässt sich eine Vergleichsfunktion `eq_angle`(v, w) definieren.

Für die Geraden $g = (g_1, g_2, g_3)$ und $h = (h_1, h_2, h_3)$ ergeben sich die Geradenkoordinaten einer Winkelhalbierenden $w = (w_1, w_2, w_3)$ als Nullstellen von

```
sys:eq_angle(l2_angle(g,w),l2_angle(w,h));
```

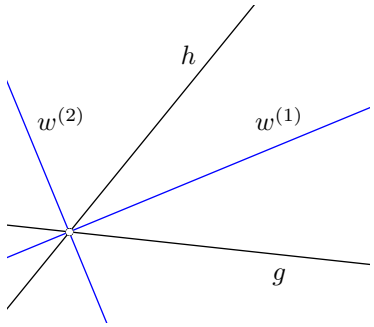
$$(g_1\,h_2 + g_2\,h_1)\,\left(w_2^2 - w_1^2\right) + 2\,(g_1\,h_1 - g_2\,h_2)\,w_1\,w_2$$

Dies ist eine quadratische Gleichung, die bei vorgegebenen (g, h) zwei Lösungen hat.

In der Formel spielen die dritten Komponenten der Geradenkoordinaten keine Rolle, da es beim Winkelvergleich nur um Geradenanstiege geht. w_3 kann aus der Bedingung `is_concurrent`(g, h, w) als

$$w_3 = \frac{g_1\,h_3\,w_2 - g_2\,h_3\,w_1 - g_3\,h_1\,w_2 + g_3\,h_2\,w_1}{g_1\,h_2 - g_2\,h_1}$$

bestimmt werden.



Im Weiteren wollen wir annehmen, dass die Geraden durch den Ursprung gehen, also $g_3 = h_3 = w_3 = 0$ gilt. Für (w_1, w_2) ergibt sich

$$w_2 = w_1\,\frac{-\,(g_1\,h_1 - g_2\,h_2) \pm \sqrt{(g_1^2 + g_2^2)\;(h_1^2 + h_2^2)}}{g_1\,h_2 + g_2\,h_1}\,,$$

was auf die homogenen Geradenkoordinaten

$$\left((g_1\,h_2 + g_2\,h_1)\,,\; -\,(g_1\,h_1 - g_2\,h_2) \pm \sqrt{(g_1^2 + g_2^2)\;(h_1^2 + h_2^2)},\; 0\right)$$

führt. Dieser Ausdruck beschreibt zwei zueinander senkrechte Geraden $w^{(1)}$ und $w^{(2)}$, deren Parameter sich allerdings, wie die Koordinaten der Schnittpunkte zweier Kreise, im Allgemeinen nicht rational durch die Koordinaten der Ausgangsgeraden ausdrücken lassen. Auch lassen sich die beiden Winkelhalbierenden algebraisch nicht voneinander unterscheiden.

Aufgabe: Weisen Sie nach, dass die beiden durch die Formeln gegebenen Geraden $w^{(1)}$ und $w^{(2)}$ senkrecht aufeinander stehen.

Als polynomiale geometrische Eigenschaft lässt sich jedoch die Bedingung ausdrücken, dass ein Punkt $P(p_x, p_y)$ auf einer der beiden Winkelhalbierenden liegt, wie sich aus folgender Rechnung ergibt:

```
A:Point(ax,ay); B:Point(bx,by); C=Point(cx,cy); P:Point(px,py);
sys:eq_angle(p3_angle(A,B,P),p3_angle(P,B,C));
```

sys ist ein Polynom vom Grad 2 in p_x, p_y, das für allgemeine Punkte A, B, C 56 Terme enthält, und die universelle Formel der polynomialen geometrischen Eigenschaft, dass P auf einer der Winkelhalbierenden von $\angle ABC$ liegt. Im GeoProver-Paket ist diese universelle Formel als `on_bisector`(P, A, B, C) implementiert.

Aufgabe: Vier Punkte A, B, C, D liegen nach der Umkehrung des Peripheriewinkelsatzes auf einem Kreis, wenn B und D auf demselben Bogen $\overset{\frown}{AC}$ liegen und $|\angle ABC| = |\angle ADC|$ gilt.

Zeigen Sie, dass diese Bedingung genau auf das oben hergeleitete Polynom vierten Grades führt.

Diskutieren Sie den Fall, dass B und D auf unterschiedlichen Bögen $\overset{\frown}{AC}$ liegen. (Hinweis: Beachten Sie, dass $\tan(-\alpha) = -\tan(\alpha)$ gilt, da mit *orientierten* Winkeln gerechnet wird.)

4.3 Geometrische Sätze vom Gleichungstyp

Im letzten Kapitel hatten wir geometrische Sätze studiert, deren algebraische Formulierung mit den Formeln der universellen Realisierung B_E einer konstruktiv erzeugten Basiskonfiguration $\mathcal{K}$ startet, zusätzliche geometrische Voraussetzungen, eine Nichtdegenerationsbedingung sowie die Behauptung in einer Menge $F = \{f_1, \ldots, f_m\} \subset R$ von Polynomen, einem Polynom $h \in R$ sowie einem Polynom $g \in R$ kodiert, so dass der Beweis des geometrischen Satzes in einem Koordinatenmodell über einem algebraisch abgeschlossenen Körper K mit $char(K) = 0$ auf eine Aussage

$$\forall\ X_0 \in K^n\ \ (\ F(X_0) = 0\ \wedge\ h(X_0) \neq 0 \Rightarrow g(X_0) = 0\ )$$

über Nullstellen von Polynomen bzw. rationalen Funktionen zurückgeführt werden kann.

Die Problemstellung

Wir wollen dies noch etwas einschränken und voraussetzen, dass folgende Größen gegeben sind:

- eine konstruktive *polynomiale* geometrische Konfiguration $\mathcal{K}$ mit einer universellen Realisierung $B_E(X)$ der Endkonfiguration, die aus einer universellen Startkonfiguration $A_S(X)$ gewonnen wurde, was wir im Weiteren als *allgemeine geometrische Basiskonfiguration* (AGK) bezeichnen;
- eine Menge polynomialer geometrischer Bedingungen mit Polynomen

  $$F = \{f_1, \ldots, f_m\} \subset R = k[X]$$

  in den Variablen X als universelle Formeln, die weitere implizite Abhängigkeiten zwischen den Elementen von B_E beschreiben und die wir als *allgemeine geometrische Voraussetzungen* (AGV) bezeichnen;
- eine (oft nicht explizit bekannte) polynomiale Nichtdegenerationsbedingung mit universeller Formel $h \in R$ für die Gültigkeit der Schlussfolgerung, die wenigstens die Ausführbarkeit von $\mathcal{K}$ garantiert, sowie
- eine polynomiale geometrische Bedingung mit universeller Formel $g \in R$, welche die Folgerung des Satzes algebraisch kodiert.

Definition 12 *Als* Geometriesatz vom Gleichungstyp *bezeichnen wir dann eine Aussage der folgenden Gestalt:*

> *Für jede spezielle Interpretation C_E von $\mathcal{K}$ über K, welche durch eine (für $\mathcal{K}$ zulässige) Substitution $X \to X_0$ erzeugt wurde, nicht degeneriert ist (für welche also $h(X_0) \neq 0$ gilt) und die AGV erfüllt (wenn also X_0 eine gemeinsame Nullstelle der Polynome $f \in F$ ist), gilt die geometrische Folgerung (d. h. X_0 ist auch eine Nullstelle von g).*
>
> $$\forall\ X_0 \in K^n\ \left(\bigwedge_{f\in F} f(X_0) = 0 \wedge h(X_0) \neq 0\ \Rightarrow\ g(X_0) = 0\right) \qquad (G_1)$$
>
> *oder anders*
>
> $$\forall\ X_0 \in V(F)\ \ (h(X_0) \neq 0\ \Rightarrow\ g(X_0) = 0)\ , \qquad (G_2)$$
>
> *wobei $V(F) = \{X_0 \in K^n \mid \forall f \in F\ f(X_0) = 0\}$ die Menge der gemeinsamen Nullstellen der Polynome $f \in F$ über K bezeichnet.*

Wir schreiben für einen solchen Satz auch kurz $(F/h \Rightarrow g)$.

Geometriesätze vom Gleichungstyp und Ideale. Zwei Beispiele

Obwohl die genaue Bestimmung von $V(F)$ den Übergang zu algebraischen Erweiterungen erfordert, lassen sich eine Reihe solcher Sätzen bereits durch algebraische Überlegungen beweisen, die nur Umformungen von Polynomen über dem Grundkörper k, also nur rationale Operationen erfordern.

Satz 12 *Ist $I = I(F) \subset R$ das von F erzeugte Ideal und $g \in I$ bzw. $g \equiv 0 \pmod I$, so gilt der Satz $(F \Rightarrow g)$.*

Dies folgt unmittelbar aus der Definition des Idealbegriffs. Das Ideal

$$I(F) = \{p_1 f_1 + p_2 f_2 + \cdots + p_m f_m \mid p_1, \ldots, p_m \in R\}$$

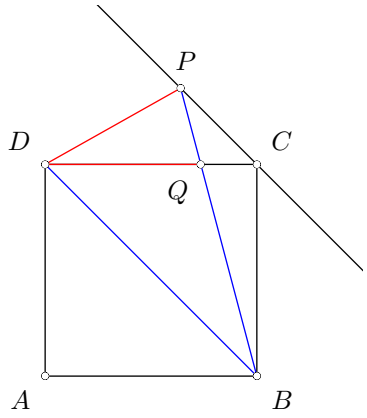
ist die Menge aller polynomialen Kombinationen der Elemente $f \in F$, so dass $V(F) = V(I)$ gilt. Damit verschwindet jedes Polynom $g \in I$ auf allen Nullstellen $X_0 \in V(F)$.
Die Modulo-Relation für Ideale ist definiert durch

$$g_1 \equiv g_2 \pmod I \iff g_1 - g_2 \in I$$

und ist eine Kongruenzrelation auf R, welche die bekannte Modulo-Relation auf den ganzen Zahlen verallgemeinert.

Betrachten wir dazu zunächst folgendes Beispiel.

Satz 13 (Beispiel von Arnon) *Gegeben sei ein Quadrat $ABCD$ und der Punkt P auf der Parallelen zu BD durch C mit $|BP| = |BD|$. Sei ferner Q der Schnittpunkt von CD und BP. Dann ist $|DP| = |DQ|$.*



Für die Basiskonfiguration wählen wir ein Koordinatensystem, so dass die Eckpunkte des Quadrats die Koordinaten

```
A:Point(0,0); B:Point(1,0);
C:Point(1,1); D:Point(0,1);
```

haben.
Diese Basiskonfiguration erweitern wir um einen freien Punkt P und den Geradengleiter $Q \in CD$. Die Koordinaten der universellen Realisierung erfordern drei Variable

```
P:Point(px,py); Q:varpoint(C,D,qs);
```

Die AGV lassen sich durch ein System $F = \{f_1, f_2, f_3\}$ von drei Polynomen anschreiben

```
[f1,f2,f3]:[ on_line(P,par_line(C,pp_line(B,D))),
  sqrdist(B,D)-sqrdist(B,P), on_line(Q,pp_line(B,P)) ];
```

$$\left[p_x + p_y - 2,\ -p_x^2 + 2\,p_x - p_y^2 + 1,\ 1 - p_y\,q_s - p_x\right] ,$$

aus denen sich alle zulässigen Werte $X = (p_x, p_y, q_s)$ mit $F(X) = 0$ bestimmen lassen. Lösen wir die erste Gleichung nach $p_y = 2 - p_x$ auf und setzen dies in die zweite und dritte Gleichung ein, so ergibt sich

$$\left[-2\,p_x^2 + 6\,p_x - 3,\ p_x\,q_s - 2\,q_s - p_x + 1\right] .$$

Das erste Polynom ergibt nach Reskalierung $f_4 = p_x^2 - 3\,p_x - \frac{3}{2} \in I = I(F)$. Daraus lassen sich zwei (nicht mehr rationale) Werte für p_x bestimmen. Setzt man jeden dieser Werte in die dritte Gleichung ein, so kann dazu stets ein eindeutig bestimmter Wert q_s ermittelt werden.

```
xvars:[px,py,qs];
sol:solve([f1,f2,f3],xvars);
```

$$\left[p_x = \frac{1}{2}\left(3 \pm \sqrt{3}\right),\ p_y = \frac{1}{2}\left(1 \mp \sqrt{3}\right),\ q_s = 2 \pm \sqrt{3}\right]$$

Die Behauptung

```
g:sqrdist(D,Q) - sqrdist(D,P);
```

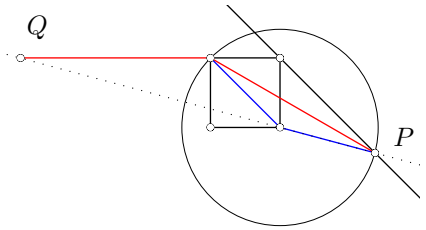
$$g = -p_x^2 - p_y^2 + 2\,p_y + q_s^2 - 2\,q_s$$

gilt für all diese Werte, wie die Rechnung

```
ratsimp(map(lambda([u],subst(u,g)),sol));
```

$$[0, 0]$$

zeigt. Jedoch wurde dabei wiederum direkt mit algebraischen Zahlen gerechnet, der Grundbereich k also verlassen.



Es sei noch angemerkt, dass die Rechnung *zwei* Lösungen für P liefert. Das ist auch plausibel, weil $|BP| = |BD|$ irgend etwas mit dem Kreis um B durch D zu tun hat und dieser Kreis natürlich *zwei* Schnittpunkte mit der Parallelen durch C hat. In der Abbildung ist die zweite Lösung zu sehen.

Gehen wir vom Ideal

$$I = I(F) = \left(f_1 = p_x + p_y - 2, f_2 = -p_x^2 + 2\,p_x - p_y^2 + 1, f_3 = 1 - p_y\,q_s - p_x\right)$$

aus, das vom Polynomsystem $F = \{f_1, f_2, f_3\}$ erzeugt wird, so sehen wir, dass

$$p_y \equiv 2 - p_x \pmod{I}$$

gilt. Wir können diese Kongruenz verwenden, um die erste Gleichung zu orientieren und die algebraische Ersetzungsregel $p_y \to 2 - p_x$ für das Rechnen $\pmod{I}$ auf die anderen Idealerzeuger f_2 und f_3 anzuwenden.

```
[f4,f5]:expand(subst(py=2-px,[f2,f3]));
```

Damit ändert sich das Ideal I nicht, da wir wie im Euklidschen Algorithmus f_2 und f_3 nur um polynomiale Vielfache von f_1 abändern. Wir erhalten

$$I = I(F) = \left(f_1 = p_y + p_x - 2, f_4 = -2\,p_x^2 + 6\,p_x - 3, f_5 = p_x\,q_s - 2\,q_s - p_x + 1\right)$$

Aus der Orientierung von f_4 ergibt sich eine weitere algebraische Ersetzungsregel $p_x^2 \to 3\,p_x - \frac{3}{2}$ für das Rechnen $\pmod{I}$. Mit diesen beiden Ersetzungsregeln lässt sich $g(p_x, p_y, q_s)$ (und jedes andere Polynom) so weit reduzieren, dass es keine Summanden mehr enthält, die monomiale Vielfache von p_y oder p_x^2 sind.

```
g1:expand(subst(py=2-px,g));
g2:expand(subst(px^2=3*px-3/2,g1));
```

$$g_1 = -2\,p_x^2 + 2\,p_x + q_s^2 - 2\,q_s$$
$$g_2 = q_s^2 - 2\,q_s - 4\,p_x + 3\,.$$

Es gilt $g \equiv g_1 \equiv g_2 \pmod I$ und g_2 hat die Gestalt $g_2 = -4\,p_x + \left(q_s^2 - 2\,q_s + 3\right)$. Wir haben bisher stets so orientiert, dass größere durch kleinere Terme bzgl. der lexikographischen Termordnung ersetzt wurden, für die $p_y > q_s > p_x$ gilt. Leider lässt sich f_5 bzgl. dieser Termordnung nur so orientieren, dass $p_x\,q_s$ ersetzt wird, was uns für g_2 nicht weiterhilft.

Nun gilt aber $f_5 = (p_x - 2)\,q_s - p_x + 1$ und

$$(p_x - 2)\,(2\,p_x - 2) = 2\,p_x^2 - 6\,p_x + 4 = 1 - f_4 \equiv 1 \pmod I,$$

so dass sich $(2\,p_x - 2)\,f_5 \to f_6 = q_s - 2\,p_x + 1 \in I$ mit den bisherigen beiden Ersetzungsregeln reduzieren lässt zur Idealbasis

$$I = I(F) = \left(f_1 = p_y + p_x - 2, f_6 = q_s - 2\,p_x + 1, f_4 = -2\,p_x^2 + 6\,p_x - 3\right)$$

und die Gleichung f_6 zu einer dritten algebraischen Ersetzungsregel $q_s \to 2\,p_x - 1$. Mit diesen Ersetzungsregeln lässt sich g_2 zu null reduzieren

```
g3:expand(subst(qs=2*px-1,g2));
expand(subst(px^2=3*px-3/2,g3));
```

womit $g \in I$ gezeigt und damit der geometrische Satz bewiesen ist.

Im Zuge der Untersuchungen haben wir mit

$$G = \left\{p_y + p_x - 2,\ q_s - 2\,p_x + 1,\ p_x^2 - 3\,p_x + \frac{3}{2}\right\}$$

eine andere Basis des Ideals $I = I(F)$ konstruiert, die unmittelbar erlaubt, alle monomialen Vielfachen von p_y, q_s und p_x^2 zu ersetzen und so zu jedem Polynom $g(p_x, p_y, q_s)$ ein Polynom $g' = a\,p_x + b$ mit $g \equiv g' \pmod I$ zu konstruieren. G ist sogar eine *Gröbnerbasis* (bzgl. der lexikografischen Termordnung), das Polynom g' *Normalform* von g bzgl. der Basis G.

Für ein gegebenes System von Polynomen $F \subset k[X]$ und eine polynomiale Bedingung $g \in k[X]$ kann man die Begriffe *Gröberbasis* $G = \texttt{GBasis}(F, X)$ und *Normalform* $g_0 = \texttt{NF}(g, F, X)$ definieren und diese können mit dem `grobner`-Paket von MAXIMA auch effektiv berechnet werden. Eine genaue Definition geben wir später. In diesem Abschnitt benötigen wir nur zwei Eigenschaften:

(1) Die Gröbnerbasis G ist eine Basis des Ideals $I = I(F)$ (mit weiteren Eigenschaften).

(2) Es gilt $g_0 \equiv g \pmod I$, also insbesondere $g \in I$ für $g_0 = 0$.

Damit können wir den **Beweis für das Arnon-Beispiel** nun kurz wie folgt führen:

```
load("grobner");
xvars:[py,qs,px];
G:poly_reduced_grobner([f1,f2,f3],xvars);
poly_normal_form(g,G,xvars);
```

$$G = \left[q_s - 2\,p_x + 1,\ p_y + p_x - 2,\ p_x^2 - 3\,p_x + \frac{3}{2}\right]$$
$$g_1 = 0$$

Allgemein gilt

Satz 14 *Ist X die Menge der Variablen in der Problemstellung, G eine Basis von F und die Normalform $\texttt{NF}(g, G, X) = 0$, so gilt der Satz $(F \Rightarrow g)$.*

Das Arnon-Beispiel war besonders einfach, weil in ihm alle Variablen durch Gleichungen gebunden sind. Für allgemeinere Beweisschemata läuft der vorgestellte Ansatz meist nicht so glatt durch.

Sehen wir uns als Nächstes den **Satz vom Schnittpunkt der Winkelhalbierenden im Dreieck** an. Wir hatten bereits gesehen, dass es zu jedem Winkel grundsätzlich zwei Winkelhalbierende gibt, deren Koordinaten – ähnlich wie Schnittpunkte von Kreisen – nicht mehr in k liegen und die sich algebraisch nicht voneinander unterscheiden lassen. Wir formulieren deshalb den Satz wie folgt:

Satz 15 (Satz vom Schnittpunkt der Winkelhalbierenden im Dreieck) *Im Dreieck ABC liegt ein gemeinsamer Punkt P einer der Winkelhalbierenden des Winkels bei A und einer der Winkelhalbierenden des Winkels bei B auch auf einer der Winkelhalbierenden des Winkels bei C.*

Wir bekommen auf diese Weise vier Schnittpunkte und jeder solche Punkt ist von den drei Dreiecksseiten gleichweit entfernt, also Mittelpunkt des Inkreises oder eines der drei Ankreise des Dreiecks ABC (Übungsaufgabe).

Wir beginnen wieder mit einer allgemeinen geometrischen Basiskonfiguration. Zur Vereinfachung der Rechnungen wählen wir das Koordinatensystem so, dass $A(0,0)$ und $B(1,0)$ gilt. Daraus können wir die AGK

```
A:Point(0,0); B:Point(1,0); C:Point(cx,cy); P:Point(px,py);
```

sowie die AGV als Polynomsystem $F = \{f_1, f_2\}$

```
polys:[ on_bisector(P,B,A,C), on_bisector(P,C,B,A) ];
```

und die Behauptung als

```
g:on_bisector(P,A,C,B);
```

anschreiben. Berechnen wir auch für dieses Beispiel die Gröbnerbasis

```
vars:[px,py,cx,cy];
G:poly_reduced_grobner(polys,vars);
```

$$2\,c_x^2\,p_y - 2\,c_y\,c_x\,p_x - 2\,c_x\,p_y + c_y\,c_x + \underline{c_y\,p_x^2} - c_y\,p_y^2$$

$$\frac{1}{2}\,c_y - p_y + c_x\,p_y - c_y\,p_x + \underline{p_x\,p_y}$$

$$2\,c_x\,p_y^2 - \frac{1}{2}\,c_x\,c_y^2 - \frac{1}{2}\,c_y^2\,p_x + c_y\,p_y^3 + \frac{1}{2}\,c_y^2 - 2\,c_x^2\,p_y^2 - c_y^2\,p_y^2 - c_y\,p_y + c_x\,c_y\,p_y + \underline{c_x\,c_y^2\,p_x} - c_x^2\,c_y\,p_y$$

$$c_x^2\,c_y\,p_y^2 - 2\,c_x^2\,p_y^3 - c_x\,c_y\,p_y^2 + 2\,c_x\,p_y^3 + c_y^3\,p_y^2 - \frac{1}{4}\,c_y^3 - 2\,c_y^2\,p_y^3 + c_y^2\,p_y + \underline{c_y\,p_y^4} - c_y\,p_y^2$$

Aus zwei Polynomen sind hier vier geworden. Eine Gröbnerbasis kann deutlich mehr Elemente enthalten als die Ausgangsbasis. Auch werden in der Ausgabe die Monome leider nicht lexikographisch angeordnet, so dass die Leitmonome nicht sofort zu erkennen sind, die in den Ersetzungsregeln verwendet werden.

Für das `grobner`-Paket von MAXIMA kann eine geordnete Liste der Monome sowie das Leitmonom mit den folgenden beiden Funktionen bestimmt werden:

```
terms(p,vars):=block([poly_return_term_list:true], poly_normal_form(p,[],vars));
lm(p,vars):=first(terms(p,vars));
```

Die Rechnung

```
poly_normal_form(g,G,vars);
```

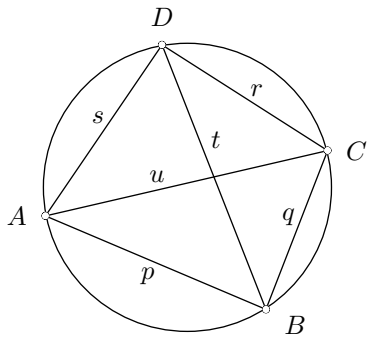
$$0$$

zeigt, dass g im Ideal $I = I(F) = I(G)$ liegt und damit der Satz $(F \Rightarrow g)$ gilt.

4.4 Weitere Beispiele

Satz 16 (Satz des Ptolemäus) *Das Viereck $ABCD$ ist genau dann ein Sehnenviereck, wenn für die Seitenlängen p, q, r, s und die Diagonalenlängen t, u die Beziehung $p\,r + q\,s = t\,u$ gilt.*

Dieser Satz macht in seiner Formulierung zum einen von der speziellen Lage der Punkte auf der Kreislinie Gebrauch (ansonsten kann man die Viereckssseiten nicht von den Diagonalen unterscheiden) und verwendet zum anderen mit den Seitenlängen Größen, die sich nicht polynomial durch die Koordinaten der Eckpunkte ausdrücken lassen.
Eine zur Ausgangsbedingung äquivalente Aussage, die besser in unser Konzept passt, erhält man, wenn man die Behauptung wie folgt umformt:



$$(p\,r)^2 + (q\,s)^2 - (t\,u)^2 = 2\,p\,q\,r\,s$$
$$((p\,r)^2 + (q\,s)^2 - (t\,u)^2)^2 - (2\,p\,q\,r\,s)^2 = 0$$

Ausmultiplizieren liefert die Gleichung

$$p^4r^4 - 2\,p^2q^2r^2s^2 - 2\,p^2r^2t^2u^2 + q^4s^4 - 2\,q^2s^2t^2u^2 + t^4u^4 = 0\,,$$

in der nur noch Quadrate von Seitenlängen vorkommen und die zudem nunmehr symmetrisch in den drei Paaren gegenüberliegender Strecken ist.

Ersetzen wir diese Quadrate durch die entsprechenden Abstandsquadrate

```
pp:((p*r)^2+(q*s)^2-(t*u)^2)^2 - (2*p*q*r*s)^2;

A:Point(ax,ay); B:Point(bx,by); C:Point(cx,cy); D:Point(dx,dy);
uhu:[
  p^2 = sqrdist(A,B), q^2 = sqrdist(B,C), r^2 = sqrdist(C,D),
  s^2 = sqrdist(D,A), t^2 = sqrdist(A,C), u^2 = sqrdist(B,D) ];

pol4:normal(subst(uhu,pp));
```

so erhalten wir ein Polynom pol_4 8. Grades mit 962 Summanden, das im Wesentlichen das Quadrat des Polynoms pol_1 ist ($pol_4 = -4\,pol_1^2$).

Schnittpunkte von zwei Kreisen. Die Potenzgerade

Im Allgemeinen lassen sich die Schnittpunktkoordinaten zweier Kreise oder von Kreis und Gerade nicht rational durch die Ausgangskoordinaten ausdrücken. So erhalten wir etwa für die Schnittpunkte zweier Kreise mit den Radien r_1 und r_2 und dem Mittelpunktsabstand $2\,u$ als Bedingung an die Koordinaten des Schnittpunkts $P = (p_x, p_y)$ das Gleichungssystem

```
P:Point(px,py);
polys:[
       on_circle(P,pc_circle(Point(-u,0),Point(-u-r1,0))),
       on_circle(P,pc_circle(Point(u,0),Point(u+r2,0))) ];
```

$$\left[-{r_1}^2 + u^2 + 2\,u\,{p_x} + {p_x}^2 + {p_y}^2, -{r_2}^2 + u^2 - 2\,u\,{p_x} + {p_x}^2 + {p_y}^2\right]\,,$$

zu dessen Lösung

```
sol:=solve(polys,px,py,IgnoreSpecialCases);
```

$$p_x = \frac{{r_1}^2 - {r_2}^2}{4\,u},\ p_y = \pm\frac{\sqrt{(r_1+r_2-2\,u)\,(r_1+r_2+2\,u)\,(r_1-r_2+2\,u)\,(r_2-r_1+2\,u)}}{4\,u}$$

bereits Wurzelausdrücke erforderlich sind.

Das ist auch verständlich, denn es kann vorkommen, dass zwei Kreise in der reellen Geometrie keine gemeinsamen Punkte haben – dies ist genau dann der Fall, wenn der Radikand des Wurzelausdrucks negativ wird. Wir betrachten dazu in GeoGebra zwei Kreise, bestimmen deren Schnittpunkte und konstruieren deren Verbindungsgerade. Verändern wir die Größe eines Kreises, so dass keine (reellen) Schnittpunkte mehr existieren, verschwindet auch die Gerade. Verändern wir die Größe weiter, so dass wieder (reelle) Schnittpunkte existieren, so werden diese und alle davon abhängenden Objekte auch wieder angezeigt. Im Algebrafenster sehen wir, dass die entsprechenden geometrischen Stücke in der Zwischenzeit als *undefiniert* behandelt werden.

Interessanterweise ist allerdings p_x ein *rationaler* Ausdruck, so dass auch im Fall eines negativen Radikanden die imaginären Schnittpunkte P_1 und P_2 dieselbe *reelle* x-Koordinate haben. Mehr noch, verbindet man diese beiden imaginären Punkte, so ergibt sich (in den hier gewählten Koordinaten) eine *reelle* Gerade senkrecht zur x-Achse. Diese Gerade wird auch als Potenzgerade der beiden Kreise bezeichnet.

Definition 13 *Sind k_1 und k_2 zwei Kreise mit den Mittelpunkten M_1 und M_2 und den Radien r_1 und r_2, so bezeichnet man die Menge derjenigen Punkte P, für die*

$$\texttt{sqrdist}(M_1,P) - \texttt{sqrdist}(M_2,P) = r_1^2 - r_2^2$$

oder gleichbedeutend

$$\texttt{sqrdist}(M_1,P) - r_1^2 = \texttt{sqrdist}(M_2,P) - r_2^2$$

gilt, als die Potenzgerade $\texttt{radical\_axis}(k_1,k_2)$ *dieser beiden Kreise.*
Die Größe $p(P,k_1) = \texttt{sqrdist}(M_1,P) - r_1^2$ bezeichnet man auch als die Potenz *des Punkts P bzgl. des Kreises k_1.*

Die Potenzgerade zweier Kreise ist also der geometrische Ort aller Punkte, die bzgl. der beiden Kreise dieselbe Potenz haben. Nach Satz des Pythagoras ist für einen Punkt P außerhalb des Kreises k die Potenz $p(P,k)$ gerade gleich dem Quadrat der Länge eines der Tangentenabschnitte von P an den Kreis k.

Ist P ein solcher Punkt und X der Lotfußpunkt von P auf die Zentrale M_1M_2, so liegt nach dem Satz des Pythagoras auch X auf der Potenzgeraden. Die Potenzgerade ist also genau die Senkrechte auf der Zentralen im Punkt X. Schneiden sich die beiden Kreise, so liegen die Schnittpunkte ebenfalls auf der Potenzgeraden. Da sich die Koordinaten von X rational durch die Kreisparameter ausdrücken lassen, ist auch $\texttt{radical\_axis}(k_1,k_2)$ ein rationales Konstruktionswerkzeug, obwohl sich die Koordinaten der Schnittpunkte der beiden Kreise selbst nicht rational durch die Kreisparameter ausdrücken lassen. Die Geradengleichung der Potenzgeraden der beiden Kreise ergibt sich sofort aus den Kreisgleichungen

$$\begin{aligned} k_1:&\ (x^2+y^2) + a_1\,x + a_2\,y + a_3 = 0\\ k_2:&\ (x^2+y^2) + b_1\,x + b_2\,y + b_3 = 0 \end{aligned}$$

als deren Differenz $(a_1-b_1)\,x + (a_2-b_2)\,y + a_3 - b_3 = 0$.

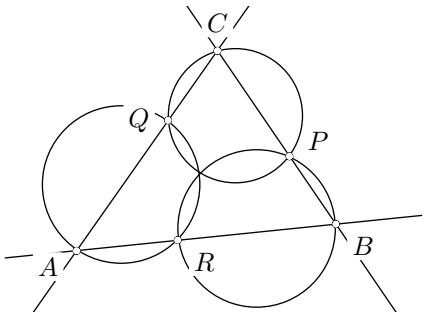
Aufgabe:

a) Geben Sie eine Konstruktionsbeschreibung für die Potenzgerade, wenn sich die beiden Kreise nicht schneiden.

b) Zeigen Sie, dass die drei Potenzgeraden, die sich zu drei gegebenen Kreisen paarweise konstruieren lassen, konkurrent sind.

Kreise mit vorgegebenem Schnittpunkt

Betrachten wir dazu den **Satz vom Miquelschen Punkt**. Die polynomiale geometrische Grundkonfiguration (Dreieck ABC mit Punkten R, S, T auf den Dreiecksseiten) lässt sich wie folgt beschreiben:




```
A:Point(ax,ay);
B:Point(bx,by);
C:Point(cx,cy);
P:varpoint(B,C,u1);
Q:varpoint(A,C,u2);
R:varpoint(A,B,u3);
```

Zum Beweis des Satzes bestimmen wir (zur Vereinfachung wieder mit einem speziell positionierten Koordinatensystem) die Koordinaten des Schnittpunkts $S = (s_x, s_y)$ zweier der Kreise und zeigen, dass dieser Punkt auch auf dem dritten Kreis liegt:

```
A:Point(0,0); B:Point(0,by); C:Point(cx,cy);
P:varpoint(B,C,u1); Q:varpoint(A,C,u2); R:varpoint(A,B,u3);
c1:p3_circle(R,Q,A); c2:p3_circle(R,P,B); c3:p3_circle(P,Q,C);

S:Point(sx,sy);
polys:[ on_circle(S,c1), on_circle(S,c2) ];
sol:solve(polys,[sx,sy]);
```

Wir erhalten zwei Lösungen, deren Koordinaten jeweils rationale Funktionen in den Parametern sind. Eine der beiden Lösungen ist der Punkt R, von dem wir bereits wissen, dass er zu beiden Kreisen gehört. Der zweite Punkt ist der gesuchte Schnittpunkt S (seine rationalen Koordinaten haben 10 bzw. 26 Zählerterme und 42 Nennerterme). Er liegt auf dem dritten Kreis, wie die folgende Rechnung zeigt.

```
S0:subst(sol[1],S);
on_circle(S0,c3);
```

Es ist nicht verwunderlich, dass die Schnittpunktkoordinaten von S rationale Funktionen in den Parametern sind, wenn einer der beiden Schnittpunkte bekannt ist, da die Schnittpunktbedingung auf eine Gleichung zweiten Grades hinausläuft.

Aufgabe: Finden Sie allgemein Formeln für folgende Funktionen:

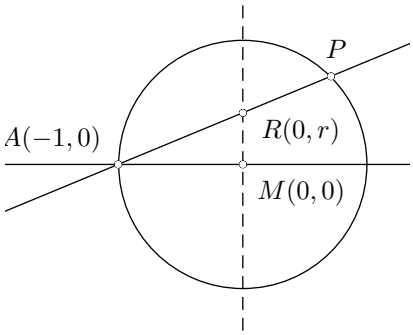
- `other_cl_point(P:Point,c:Circle,l:Line):Point`, die zu gegebenem Kreis c und gegebener Gerade l den zweiten Schnittpunkt von c und l findet, wenn P der andere Schnittpunkt ist.
- `other_cc_point(P:Point,c1:Circle,c2:Circle):Point`, die zu zwei gegebenen Kreisen c_1 und c_2 den zweiten Schnittpunkt von c_1 und c_2 findet, wenn P der andere Schnittpunkt ist.

Kreisgleiter

Sätze über Punkte auf einer Kreislinie $k(M, r)$ lassen sich oft auch als Sätze vom konstruktiven Typ anschreiben, wenn eine rationale Parametrisierung der Punkte auf dem Kreis als Kreisgleiter verwendet wird. Diese ergibt sich, wenn wir durch einen bekannten Punkt auf der Kreislinie Sekanten zeichnen und die Koordinaten des jeweils zweiten Schnittpunkts in Abhängigkeit vom Anstieg dieser Sekante ausdrücken.

Für den Einheitskreis $x^2+y^2=1$ und den Referenzpunkt $(-1,0)$ ergibt sich für die Koordinaten eines Kreisgleiters folgende Formel

$$P = \left[\frac{r^2-1}{r^2+1},\ \frac{2\,r}{r^2+1}\right]$$



Jeder Punkt $P \neq A$ auf dem Kreis kann auf diese Weise eindeutig dargestellt werden. Dass eine *vollständige* Parametrisierung der Kreislinie durch Werte $r \in \mathbb{R}$ nicht möglich ist, hat topologische Gründe – die Kreislinie ist kompakt, der Parameterbereich $r \in \mathbb{R}$ dagegen nicht.

Auch in diesem Fall kann der fehlende Punkt A mit $r=\infty$ in Verbindung gebracht werden, so dass eine vollständige Parametrisierung der Kreislinie durch Punkte $(r:s) \in \mathbb{P}^1$ auf der (kompakten) projektiven Geraden möglich ist – in homogenen Koordinaten $P = \left(\left(r^2-s^2\right) : (2\,r\,s) : \left(r^2+s^2\right)\right)$.

Sind M und A Punkte in allgemeiner Lage, so ergeben sich nur leicht kompliziertere Formeln, die in der GeoProver-Funktion `circle_slider`(M,A,u) implementiert sind.

Damit können wir einen konstruktiven Beweis des **Satzes von der Simsonschen Geraden** geben:

```
M:Point(0,0);
A:Point(0,1);
B:circle_slider(M,A,u1);
C:circle_slider(M,A,u2);
D:circle_slider(M,A,u3);
R:pedalpoint(D,pp_line(B,C));
S:pedalpoint(D,pp_line(A,C));
T:pedalpoint(D,pp_line(A,B));
is_collinear(R,S,T);
```

4.5 Normalformen und Gröbnerbasen

Idealbasen und Normalformen

Wir untersuchen nun die die Problemstellung näher, für ein gegebenes Polynom $g \in R = k[X]$ mit $X = (x_1,\ldots,x_n)$ zu entscheiden, ob es in einem gegebenen Ideal $I = I(F) \subset R$ enthalten ist, welches von einer Menge $F = \{f_1,\ldots,f_m\}$ erzeugt wird. Dieses Problem wird als **Idealenthaltenseinsproblem** bezeichnet.

Wir hatten dazu in den bisherigen geometrischen Beispielen Rechnungen $\pmod I$ ausgeführt, indem wir die Polynome aus F als Gleichungen im Faktorring $S = R/I$ betrachtet, orientiert und als algebraische Ersetzungsregeln angewendet hatten. Basis einer solchen Orientierung war eine Termordnung auf $T(X)$, bzgl. welcher die Summanden eines Polynoms $f \in R$ fallend geordnet werden, um dieses in distributiver Normalform darzustellen.

> *Im Folgenden setzen wir voraus, dass auf $T(X)$ eine Termordnung fixiert ist, ohne dies jedesmal separat zu erwähnen. In den meisten hier betrachteten Anwendungen ist dies die lexikografische Termordnung bzgl. der Variablenordnung auf X, welche durch die Angabe einer Variablenliste definiert ist.*

Für $f = \sum_a c_a\,X^a \neq 0$ und $X^{a_0} = \max\left(X^a : c_a \neq 0\right)$ definieren 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

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

Wir wenden diesen Zugang auf die Untersuchung der Frage an, ob die Polynome $f = -x\,z + y^2$ und $g = x\,y - z$ im Ideal $I = I(y - x^2, z - x^3)$ enthalten sind. Bzgl. der lexikographischen Termordnung mit $z > y > x$ lassen sich aus den Basispolynomen Kongruenzrelationen

$$y \equiv x^2 \pmod{I} \text{ und } z \equiv x^3 \pmod{I}$$

und daraus algebraische Ersetzungsregeln $y \to x^2$ und $z \to x^3$ herleiten und auf f und g anwenden.

$$\begin{array}{rcccl} f & \equiv & -x \cdot x^3 + (x^2)^2 & = & 0 \pmod{I} \\ g & \equiv & x \cdot x^2 - x^3 & = & 0 \pmod{I} \end{array}$$

In beiden Fällen können wir $f, g \in I$ schließen.

Dieses Vorgehen ist auch allgemein möglich. Wir orientieren die Polynome $f \in F$ entsprechend und leiten daraus algebraische Ersetzungsregeln her, die monomiale Vielfache des Leitterms $lt(f)$ durch geeignete monomiale Vielfache des Reduktums $red(f)$ ersetzen:

$$lt(f) \mapsto -lc(f)^{-1}\, red(f).$$

Beispiel: Die Basis $B_1 = \left\{ f_1 = x^2 + x\,y + y^2,\ f_2 = x\,z + y\,z,\ f_3 = y^3 - z^3 \right\}$ des Ideals $I = I(B_1)$ liefert (bzgl. der lexikographischen Termordnung mit $x < y < z$) das Ersetzungssystem

$$x^2 \mapsto -x\,y - y^2,\ x\,z \mapsto -y\,z,\ y^3 \mapsto z^3\,.$$

Wenden wir die Regeln in der genannten Reihenfolge auf das Polynom $g = x^2y^2 + x^2z^2 + y^2z^2$ an, so erhalten wir nacheinander

$$\begin{aligned} g &\mapsto x^2z^2 - x\,y^3 - y^4 + y^2z^2 \mapsto -x\,y^3 - x\,y\,z^2 - y^4 \mapsto -x\,y\,z^2 - x\,z^3 - y^4 \\ &\mapsto -x\,z^3 - y^4 + y^2z^2 \mapsto -y^4 + y^2z^2 + y\,z^3 \mapsto y^2z^2 \end{aligned}$$

Ein solches Verfahren ist im `grobner`-Paket von MAXIMA in der Funktion `poly_normal_form` implementiert.

```
load("grobner");
vars:[x,y,z]; [f1,f2,f3]:[x^2+x*y+y^2, x*z+y*z, y^3-z^3];
map(lambda([u],lm(u,vars)),[f1,f2,f3]);
g:x^2*y^2+x^2*z^2+y^2*z^2;
poly_normal_form(g,[f1,f2,f3],vars);
```

$$y^2z^2$$

Das aus B abgeleitete Ersetzungssystem erlaubt es also, alle Terme aus dem Monoidideal

$$\Sigma(B) := \{x^\alpha \ : \ \exists\, f \in B : lt(f) \,|\, x^\alpha\}$$

durch eine Linearkombination „kleinerer“ Terme zu ersetzen. Diese Terme bezeichnen wir deshalb auch als *Nichtstandardterme*, die verbleibenden Terme $T(X) \setminus \Sigma(B)$ dagegen als die *Standardterme* bzgl. B.

Beispiel: Es kommt auf die Reihenfolge beim Berechnen der Normalform an:

$$B_2 := \left\{ u\,x - y^2, u\,y - z^2, u\,z - x^2 \right\}$$

und Reduktion des Monoms $u^2 x\,y\,z$.

```
vars:[u,x,y,z];
B2:[u*x-y^2,u*y-z^2,u*z-x^2];
g:u^2*x*y*z;
poly_normal_form(g,[B2[1],B2[2]],vars);
```

$$y^2 z^3$$

```
poly_normal_form(g,[B2[1],B2[3]],vars);
```

$$x^3 z^2$$

Verschiedene Pfade liefern eine der Normalformen y^2z^3, z^3x^2 oder x^3z^2. Begriff des *Reduktionspfads.*

Der folgende Algorithmus **NF** nimmt in einem Polynom $f \in R$ so lange Ersetzungen vor, bis der Leitterm des entstehenden Polynoms ein Standardterm bzgl. B ist:

NF(f : Polynom, B : Basis) : Polynom

Input: Polynom $f \in R$, endliche Menge $B \subset R$.
Output: Polynom $f' \in R$ mit $f \equiv f' \pmod{I(B)}$ und $f' = 0$ oder $lt(f') \notin \Sigma(B)$.

`while` $(f \neq 0)$ `and` $(M := \{b \in B \ : \ lt(b) \,|\, lt(f)\} \neq \emptyset)$ `do`
  `choose` $b \in M$
  $f := f - \frac{lm(f)}{lm(b)} b$
`return` f

Dieser Algorithmus terminiert offensichtlich, weil die Folge der Leitmonome der in den einzelnen Schritten entstehenden Zwischenergebnisse eine streng monoton fallende Folge von Monomen ergibt, die wegen der Wohlfundiertheit der Termordnung endlich sein muss.

Ähnlich wie im Erweiterten Euklidschen Algorithmus kann man den Normalforn-Algorithmus so modifizieren, dass sogar eine Darstellung von $f' - f \in I(B)$ als polynomiale Kombination der Basiselemente zurückgegeben wird. In obigem Beispiel etwa erhalten wir bei der Berechnung von $g' := \mathtt{NF}(g, B)$ nacheinander

$$\begin{aligned}
g \mapsto g_1 &= g - y^2 f_1 &&= x^2z^2 - x\,y^3 - y^4 + y^2z^2\\
\mapsto g_2 &= g_1 - z^2 f_1 &&= -x\,y^3 - x\,y\,z^2 - y^4\\
\mapsto g_3 &= g_2 + x f_3 &&= -x\,y\,z^2 - x\,z^3 - y^4\\
\mapsto g_4 &= g_3 + y\,z\,f_2 &&= -x\,z^3 - y^4 + y^2z^2\\
\mapsto g_5 &= g_4 + z^2 f_2 &&= -y^4 + y^2z^2 + y\,z^3\\
\mapsto g_6 &= g_5 + y f_3 &&= y^2z^2 = g'
\end{aligned}$$

also $g = (y^2 + z^2) f_1 + (-y\,z - z^2)f_2 + (-x + y) f_3 + g'$.

Allgemein lassen sich die Kofaktoren während der Reduktion auf dieselbe Weise in einem Vektor $(v_1, \ldots, v_m)$ aufsammeln:

NFwithRelations(f: Polynom, B: Basis): (Polynom,Vektor)

Input: Polynom $f \in R$, endliche Menge $B = \{b_1, \ldots, b_m\} \subset R$
Output: Polynom $f' \in R$ mit $f' = 0$ oder $lt(f') \notin \Sigma(B)$ und Vektor $v = (v_1, \ldots, v_m)$ mit $f = \sum_i v_i\, b_i + f'$.

`for` $i = 1, \ldots, m$ `do` $v_i := 0$

```
while (f ≠ 0) and (M := {b ∈ B : lt(b) | lt(f)} ≠ ∅) do
  choose b_i ∈ M
  f := f − (lm(f)/lm(b_i)) b_i
  v_i := v_i + lm(f)/lm(b_i)
return (f, v)
```

Diese Darstellung als polynomiale Kombination der Basisvektoren hat eine weitere wichtige Eigenschaft; sie kommt ohne „große“ intermediäre Terme aus:

Satz 17 *Sei $B = \{b_1, \ldots, b_m\} \subset R$ eine endliche Menge von Polynomen. Dann liefert der Algorithmus **NFwithRelations** für jedes Polynom $f \in R$ nach endlich vielen Schritten eine Darstellung*

$$f = v_1\, b_1 + \ldots + v_m\, b_m + r$$

mit $v_1, \ldots, v_m, r \in R$, in der $r = 0$ oder $lt(r) \notin \Sigma(B)$ und $lt(f) \geq lt(v_i)\, lt(b_i)$ für alle i gilt.

Beweis: Da `NFwithRelations` nur eine Modifikation von `NF` ist, muss nur die letzte Aussage

$$lt(f) \geq lt(v_i)\, lt(b_i) \tag{1}$$

bewiesen werden.
Sind f' und $f'' = f - \frac{lm(f')}{lm(b_i)} b_i$ zwei intermediäre Terme, so unterscheiden sich die Vektoren v für f' und f'' nur im Summand mit dem Term $m = \frac{lt(f')}{lt(b_i)}$, für den aber gilt $lt(f) \geq lt(f') = m \cdot lt(b_i)$. Aussage (1) gilt also für alle Terme von h_i und folglich auch für den Leitterm. □

Mit dem Algorithmus `NF` können wir schon eine Antwort auf die erste Hälfte des Idealenthaltenseins-Problems geben:

Satz 18 *Seien R, f und B wie in obigem Satz. Ist $\mathtt{NF}(f, B) = 0$, so gilt $f \in I(B)$.*

Die Umkehrung dieses Satzes gilt nicht, d. h. es kann durchaus Elemente $f \in I$ geben, für die $\mathtt{NF}(f, B) \neq 0$ gilt. Mehr noch kann das Ergebnis vom gewählten Reduktionspfad abhängen. Der Satz kann dahingehend verschärft werden, dass $\mathtt{NF}(f, B) = 0$ für einen einzigen Reduktionspfad ausreicht, um auf $f \in I$ zu schließen. Jedoch auch von dieser Verschärfung gilt die Umkehrung nicht: Wir werden später auf systematische Weise Polynome $f \in I$ konstruieren, die als Summe von Standardtermen überhaupt nicht weiter reduziert werden können.

Der bisher betrachtete Normalform-Algorithmus beendet seine Arbeit, wenn ein Standardterm als Leitterm erzeugt worden ist. Dabei kann es sein, dass immer noch Nichtstandardterme im Reduktum des erzeugten Polynoms auftreten, die ebenfalls noch reduziert werden können. Einen entsprechenden Algorithmus, der **NF** noch rekursiv auf das Reduktum anwendet, bezeichnet man als **totalen Normalform-Algorithmus**.

TNF(f: Polynom, B: Basis): Polynom

Input:	Polynom $f \in R$, endliche Menge $B \subset R$
Output:	Polynom $f' \in R$ mit $f \equiv f' \pmod{I(B)}$ und $f' = 0$ oder $T(f') \cap \Sigma(B) = \emptyset$

```
f:=NF(f,B)
if f = 0 then return f else return lm(f) + TNF(red(f), B)
```

Für diesen Algorithmus kann man ebenfalls wieder eine Variante angeben, die $f - f'$ als polynomiale Kombination der Basiselemente $b \in B$ darstellt. `poly_normal_form` implementiert diesen Algorithmus.

Monoidideale und Dickson-Lemma

Für ein Ideal $I \subset R$ bezeichnen wir mit

$$\Sigma = \Sigma(I) = \{lt(f) \in T \,:\, f \in I \setminus \{0\}\}$$

die Teilmenge der als Leitterm von Polynomen $f \in I$ vorkommenden Terme. $T = T(X)$ ist bzgl. der Termmultiplikation ein Monoid und Σ ein *Monoidideal* (d. h. es gilt $\Sigma \cdot T = \Sigma$).

Definition 14 *Eine Teilmenge $\Sigma_0 = \{X^{a_1}, \ldots, X^{a_m}\}$ eines Monoidideals Σ bezeichnet man als* Basis, *wenn $\Sigma_0 \cdot T = \Sigma$ gilt und als* Minimalbasis, *wenn Σ_0 minimal bzgl. Inklusion mit dieser Eigenschaft ist.*
Wir schreiben in diesem Fall $\Sigma = (X^{a_1}, \ldots, X^{a_m})$.

Satz 19 *Jedes Monoidideal $\Sigma \subset T$ hat eine eindeutig bestimmte Minimalbasis. Diese besteht genau aus den $X^\alpha \in \Sigma$, die minimal in Σ bzgl. der Teilbarkeitsrelation sind, d. h. für die*

$$X^\beta \in \Sigma, X^\beta \,|\, X^\alpha \;\Rightarrow\; X^\beta = X^\alpha$$

gilt. Für diese Menge schreiben wir $Gen(\Sigma)$.

Beispiel: $\Sigma = (x^4y^2, x^3y^4, x^2y^5)$. Grafische Darstellung im $\mathbb{N}^2$.
Vereinigung, Produkt und Durchschnitt von Monoididealen $\Sigma_1 = \Sigma(G_1)$ und $\Sigma_2 = \Sigma(G_2)$ sind wieder Monoidideale. Erzeugendensysteme für Vereinigung und Produkt lassen nach den folgenden Regeln auf einfache Weise aus G_1 und G_2 berechnen.

1. $\Sigma_1 \cup \Sigma_2 = \Sigma(G_1 \cup G_2)$
2. $\Sigma_1 \cdot \Sigma_2 = \Sigma(G_1 \cdot G_2)$

Die Berechnung eines Erzeugendensystems des Durchschnitts ist etwas komplizierter.
Beispiel: $I(x^3y, y^4) \cap I(x^5, x^2y^2) = I(x^2y^4, x^3y^2, x^5y)$
Aufgabe: Finden Sie eine allgemeine Formel für die Erzeugenden des Durchschnitts zweier Monoidideale.

Satz 20 (Dickson-Lemma) *Jedes Monoidideal $\Sigma \subset T = T(x_1, \ldots, x_n)$ besitzt eine endliche Basis.*

Beweis: Wir führen den Beweis mit Induktion nach n. Für $n = 1$ ist die Aussage offensichtlich. Für den Induktionsschritt sei $T' = T(x_1, \ldots, x_{n-1})$. Nach Induktionsvoraussetzung wissen wir, dass jedes Monoidideal in T' eine endliche Basis besitzt und wir wollen dies für Monoidideale $\Sigma \subset T$ zeigen. Mit $X = (x_1, \ldots, x_{n-1})$ und $y = x_n$ kann jedes $M \in T$ (eindeutig) als $M = X^\alpha y^m$ dargestellt werden.
Betrachten wir zunächst

$$\Sigma' := \{X^\alpha \;:\; \exists\, m > 0 \; X^\alpha y^m \in \Sigma\}.$$

Diese Menge ist ein Monoidideal (warum?) in T', hat also eine endliche Basis

$$B := \{X^{\alpha_i} \;:\; i = 1, \ldots, k\},$$

wobei für geeignete m_i stets $X^{\alpha_i} y^{m_i} \in \Sigma$ gilt.
Sei $m := max(m_i \,:\, i = 1, \ldots, k)$ und für $l \geq 0$

$$\Sigma_l := \{X^\alpha \;:\; X^\alpha y^l \in \Sigma\}.$$

Diese Mengen, die man als die „Scheiben“ von Σ in y-Richtung verstehen kann, sind ebenfalls Monoidideale in T' (warum?) und es gilt

$$l_1 < l_2 \Rightarrow \Sigma_{l_1} \subset \Sigma_{l_2}$$

sowie $\Sigma_l = \Sigma'$ für $l \geq m$. Jedes der kleineren ($l \leq m$) Monoidideale hat wiederum eine endliche Basis

$$B_l := \{X^{\alpha_l(i)} \ : \ i = 1, \ldots, k_l\},$$

so dass nach Definition

$$C_l := \{X^{\alpha_l(i)} y^l \ : \ i = 1, \ldots, k_l\} \subset \Sigma$$

gilt.

Wir behaupten nun, dass die Vereinigung $C := \bigcup_{l \leq m} C_l \subset \Sigma$ eine endliche Basis von Σ ist. Nach der Basiseigenschaft ist dazu nur zu zeigen, dass jedes Monom $M \in \Sigma$ durch eines aus C teilbar ist. Das ist nach Konstruktion aber offensichtlich. $\square$

Zum besseren Verständnis des Beweises wollen wir ihn noch einmal an einem Beispiel nachvollziehen:

$\Sigma = (x^4y^2, x^3y^4, x^2y^5)$ Dann ist $\Sigma' = (x^2), m = 5$ und für die einzelnen „Scheiben“ erhalten wir $\Sigma_0 = \Sigma_1 = \{0\}, \Sigma_2 = \Sigma_3 = (x^4), \Sigma_4 = (x^3)$. Nach dem Beweisschema erhalten wir

$$C = \{x^4y^2, x^4y^3, x^3y^4, x^2y^5\},$$

wobei das Monom x^4y^3 in einer Minimalbasis überflüssig ist.

Interreduktion

Wir kehren zu Polynomidealen zurück und betrachten das folgende System von Polynomen

$$B_3 := \left\{ f_1 = x^2 + y + z - 3,\ f_2 = x + y^2 + z - 3,\ f_3 = x + y + z^2 - 3 \right\},$$

so erkennen wir, dass die daraus ableitbaren Ersetzungsregeln nicht unabhängig voneinander sind. Das liegt daran, dass das Basiselement f_1 dafür verwendet werden kann, die anderen beiden Elemente zu reduzieren. Wir erhalten entsprechend

$$\begin{aligned} f_4 &:= NF(f_1, \{f_3\}) = y^2 + 2\,y\,z^2 - 5\,y + z^4 - 6\,z^2 + z + 6 \\ f_5 &:= NF(f_2, \{f_3\}) = y^2 - y - z^2 + z \end{aligned}$$

Im Allgemeinen können wir ähnlich vorgehen und erhalten ein Ergebnis, das der Triangulierung einer Matrix im Gaussverfahren entspricht: Solange in der Basis ein Element enthalten ist, das bzgl. der anderen reduziert werden kann, führen wir diese Reduktion aus und ersetzen das alte Basiselement durch das neue (oder lassen es weg, wenn die Reduktion 0 ergeben hat).

Eine Basis B mit der Eigenschaft

$$\forall\, f \in B \ : \ lt(f) \notin \Sigma(B - \{f\})$$

heißt *reduziert*. Der folgende Algorithmus `Interreduce` berechnet aus einer beliebigen Basis eine reduzierte Basis desselben Ideals. Er ist als `poly_reduction` im `grobner`-Paket von MAXIMA implementiert.

Interreduce(B: Basis): Basis

Input: Basis $B = \{b_1, \ldots, b_m\} \subset R$

Output: Basis B' mit $I(B) = I(B')$ und $|B'| = |Gen(\Sigma(B'))|$

`while exists` $f \in B,\ lt(f) \in \Sigma\,(B - \{f\})$ `do`

$B = B - \{f\}$
$f' = NF(f, B)$
`if` $f' \neq 0$ `then` $B = B \cup \{f'\}$
`return` B

Satz 21 *Der Algorithmus* ***Interreduce*** *terminiert, wenn* $(T, <)$ *eine noethersche Termordnung ist, und erfüllt die gegebene Spezifikation.*

Beweis: Sei $B' = B - \{f\}$. Wegen $f' = NF(f, B') \equiv f \pmod{B'}$ gilt offensichtlich $I(B' \cup \{f\}) = I(B' \cup \{f'\})$, so dass nur die Termination zu beweisen ist.

Nach Auswahl von f gilt $\Sigma(B) = \Sigma(B')$ und $f' = 0$ oder $lt(f') \notin \Sigma(B')$. Im zweiten Fall ist $\Sigma' = \Sigma(B' \cup \{f'\})$ eine echte Obermenge von $\Sigma = \Sigma(B)$.

Würde der Algorithmus nicht terminieren, so gäbe es also eine unendliche Kette $\Sigma_1 \subset \Sigma_2 \subset \ldots$ von echt ineinander enthaltenen Monoididealen. Dies widerspricht aber dem Dicksonlemma. □

Bemerkung: Die Eigenschaft, dass es sich um eine *wohlfundierte* Termordnung handelt, wurde nur für die Termination von NF benötigt; die `while`-Schleife terminiert allein auf Grund des Dicksonlemmas.

In obigem Beispiel erhalten wir nacheinander

```
[f1,f2,f3]:[x^2 + y + z - 3, x + y^2 + z - 3, x + y + z^2 - 3];
f4:poly_normal_form(f1,[f2,f3],vars);
```

$$y^4 + 2y^2z - 6y^2 + y + z^2 - 5z + 6$$

```
f5:poly_normal_form(f2,[f3,f4],vars);
```

$$y^2 - y - z^2 + z$$

```
f5:poly_normal_form(f2,[f3,f4],vars);
```

$$2\,y\,z^2 - 4\,y + z^4 - 5\,z^2 + 6$$

also

$$B = \{f_1, f_2, f_3\} \mapsto \{f_2, f_3, f_4\} \mapsto \{f_3, f_4, f_5\} \mapsto \{f_3, f_5, f_6\}$$

mit den Monoididealen

$$\Sigma(x) \subset \Sigma(x, y^4) \subset \Sigma(x, y^2) \subset \Sigma(x, y^2, y\,z^2)$$

Die reduzierte Basis B heißt *vollständig interreduziert*, wenn alle Terme in den Basiselementen bis auf den Leitterm Standardterme sind.

$$\forall b \in B \quad T(red(b)) \cap \Sigma(B) = \emptyset.$$

Eine solche Basis kann aus einer interreduzierten gewonnen werden, indem das Reduktum jedes Basiselements durch dessen totale Normalform ersetzt wird.

Aufgabe: Überführen Sie die Idealbasis

$$B_5 := \{w + x + y + z, w\,x + x\,y + y\,z + z\,w, w\,x\,y + x\,y\,z + y\,z\,w + z\,w\,x, w\,x\,y\,z - 1\}$$

(bzgl. der lexikografischen Ordnung $w > x > y > z$) in eine entsprechende reduzierte Form.

Gröbnerbasen und Hilberts Basissatz

Die systematische Vergrößerung von $\Sigma(B)$ im Zuge des Interreduktionsprozesses kann man verallgemeinern: Wir können beliebige Polynome $f \in I$ mit $lt(f) \notin \Sigma(B)$ mit demselben Erfolg zu B hinzunehmen, um die „Reduktionskraft“ von B zu verstärken.

Dafür müssen wir nicht einmal von einer Idealbasis ausgehen, sondern können diesen Prozess mit $B = \emptyset$ als Startmenge beginnen. In jedem Schritt ergänzen wir B um ein Element $0 \neq f \in I$ mit $lt(f) \notin \Sigma(B)$, so lange das möglich ist. Wir bekommen dabei eine Kette $\Sigma_0 \subset \Sigma_1 \subset \ldots$ von echt wachsenden Monoididealen, was nach dem Dicksonlemma nach endlich viele Schritten zu einer endlichen Menge G von Polynomen mit der Eigenschaft $\Sigma(G) = \Sigma(I)$ führt.

Es handelt sich dabei um besondere Teilmengen von I, denn selbst für eine Basis B des Ideals I gilt zwar $\Sigma(B) \subseteq \Sigma(I)$, es muss aber nicht unbedingt $\Sigma(B) = \Sigma(I)$ gelten.

Beispiel: $I = I(f_1 = x^3 - 2\,x\,y, f_2 = x^2\,y - 2\,y^2 + x)$. Dann gilt $B = \{f_1, f_2\}, \Sigma(B) = (x^3, x^2y)$, aber wegen $x^2 = x\,f_2 - y\,f_1 \in I$ außerdem wenigstens $x^2 \in \Sigma(I)$.

Definition 15 *Eine Teilmenge $G \subset I$ des Ideals I mit $\Sigma(G) = \Sigma(I)$ heißt* Gröbnerbasis *von I.*

Da Σ als Monoidideal endlich erzeugt ist, hat das Ideal I auch Gröbnerbasen mit endlich vielen Elementen. Jede Teilmenge $G' \subset G$, deren Leitterme noch immer Σ erzeugen, ist ebenfalls eine Gröbnerbasis von I. Eine Gröbnerbasis G heißt *minimal*, wenn $\{lt(g) \mid g \in G\} = Gen(\Sigma)$ gilt.

Ist G darüber hinaus vollständig interreduziert, so heißt G *minimale reduzierte Gröbnerbasis*. Wir werden später sehen, dass eine solche Gröbnerbasis bei vorgegebenem Ideal I und vorgegebener Termordnung $(T, <)$ eindeutig bestimmt ist.

Obwohl unsere Argumentation nicht konstruktiv war, haben wir oben gezeigt, dass jedes Ideal eine Gröbnerbasis hat. Es bleibt noch der Begriff „Basis“ zu rechtfertigen, d. h. zu zeigen, dass G wirklich eine Basis des Ideals I ist.

Satz 22 *Jede endliche Gröbnerbasis $G = \{g_1, \ldots, g_r\} \subset I$ ist eine Basis des Ideals I.*

Beweis: Wegen $G \subset I$ bleibt nur zu zeigen, dass jedes Element $f \in I$ auch tatsächlich als polynomiale Kombination dieser Polynome darstellbar ist. Berechnen wir die Normalform mit Relationen von f bzgl. G, erhalten wir eine Darstellung

$$f = p_1 g_1 + \ldots + p_r g_r + q$$

mit einem Polynom q, das entweder Null ist oder einen Leitterm $lt(q) \notin \Sigma(G) = \Sigma(I)$ hat. Da letzteres wegen $q \in I$ nicht möglich ist, folgt $q = 0$ und damit $f \in I(G)$. $\square$

Wir sagen deshalb auch ohne Bezug auf ein Ideal, dass G eine Gröbnerbasis ist, wenn G dies bzgl. des Ideals $I = I(G)$ ist.

Einer der zentralen Sätze der kommutativen Algebra ergibt sich nun als einfache Folgerung:

Folgerung 4 (Hilberts Basissatz) *Jedes Ideal $I \subset R$ besitzt eine endliche Basis.*

Da es sich beim Dickson-Lemma, auf dem der obige Beweis beruht, um eine reine Existenzaussage handelt, die uns kein konstruktives Verfahren zum Auffinden einer solchen Gröbnerbasis in die Hand gibt, bleibt die Möglichkeit des praktischen Umgangs mit diesem Begriff zunächst im Dunkeln. Bevor wir dieses aufhellen, wollen wir die Nützlichkeit des eingeführten Begriffs auch für andere konstruktive Fragestellungen zusammentragen.

Erste Eigenschaften von Gröbnerbasen

Gröbnerbasen erlauben eine eindeutige Antwort auf das Idealenthaltenseinsproblem:

Satz 23 *Sei $G = \{g_1, \ldots, g_r\}$ eine Gröbnerbasis des Ideals I. Dann gilt*

$$f \in I \;\Leftrightarrow\; NF(f, G) = 0.$$

Beweis: Wir hatten bereits gesehen, dass $NF(f, G) = 0 \;\Rightarrow f \in I$ für jede Idealbasis von I gilt, so dass die umgekehrte Richtung zu zeigen bleibt. Nehmen wir also an, es gäbe ein $f \in I$ mit $NF(f, G) = r \neq 0$. Dann ist einerseits $lt(r) \notin \Sigma(G) = \Sigma(I)$, andererseits $f \equiv r \pmod I$, d. h. $r \in I$, ein Widerspruch. □

Folgerung 5 *Für eine Gröbnerbasis G und ein Polynom $f \in R$ ist $TNF(f, G)$ unabhängig vom gewählten Reduktionspfad.*

Beweis: Sind r_1 und r_2 zwei totale Normalformen, die unterschiedlichen Reduktionspfaden entsprechen, so gilt wegen $f \equiv TNF(f, G) \pmod{I(G)}$ auch $r = r_1 - r_2 \in I$. Da aber beide Normalformen Linearkombinationen aus Standardtermen sind, so auch r. Also muss $r = 0$ sein, da sonst $lt(r) \notin \Sigma(G) = \Sigma(I)$ wäre. □

S-Polynome

Wir wollen nun gezielt Beispiele von Polynomen konstruieren, die im von der Basis B erzeugten Ideal liegen, aber deren Leitterm nicht in $\Sigma(B)$ enthalten ist. Einen Weg zur Konstruktion solcher Polynome hatten wir im Beispiel

$$B_2 = \left\{ f_1 = u\,x - y^2, f_2 = u\,y - z^2, f_3 = u\,z - x^2 \right\}$$

gesehen: $u^2\,x\,y\,z$ lässt sich auf zwei verschiedenen Pfaden jeweils zur Normalform y^2z^3 oder x^3z^2 reduzieren. Demzufolge ist $f = x^3z^2 - y^2z^3 \in I$, aber $lt(f) = x^3z^2 \notin \Sigma(B_2)$. Weitere kleinste Gegenbeispiele können für B_2 auf folgende Weise konstruiert werden:

$$\begin{aligned}
s_{12} &= y \cdot f_1 - x \cdot f_2 = (u\,x\,y - y^3) - (u\,x\,y - x\,z^2) = x\,z^2 - y^3 \\
s_{13} &= z \cdot f_1 - x \cdot f_3 = (u\,x\,z - y^2 z) - (u\,x\,z - x^3) = x^3 - y^2 z \\
s_{23} &= z \cdot f_2 - y \cdot f_3 = (u\,y\,z - z^3) - (u\,y\,z - x^2 y) = x^2 y - z^3
\end{aligned}$$

In jedem der drei Beispiele kann man dem Ergebnis nicht mehr ansehen, wie es als Linearkombination der Basiselemente entstanden ist, da sich diese Kombination nur durch Hinzufügen zweier gleicher Terme mit entgegengesetztem Vorzeichen ergibt, die in der fixierten Termordnung *größer* als die verbleibenden Terme sind. Ein solches Element hat nur dann eine verschwindende Normalform, wenn es einen zweiten Weg zu seiner Darstellung als Element von I gibt, die *ohne Termüberschreitung* auskommt.

Das allgemeine Schema der Konstruktion solcher Elemente suggeriert die folgende

Definition 16 *Seien $f, g \in R$ zwei nichttriviale Polynome und $m = lcm(lt(f), lt(g))$ das kleinste gemeinsame Vielfache der Leitterme der beiden Polynome. Dann bezeichnen wir das Polynom*

$$S(f, g) := \frac{m}{lm(f)} f - \frac{m}{lm(g)} g = \frac{m}{lm(f)} red(f) - \frac{m}{lm(g)} red(g),$$

das die kleinste monomiale Kombination aus f und g ist, in der sich die beiden Kopfterme gegenseitig wegheben, als das S-Polynom *von f und g.*

Die Funktion `poly_s_polynomial` im `grobner`-Paket von MAXIMA implementiert die Berechnung solcher S-Polynome.

Ein S-Polynom, falls es nicht verschwindet, besitzt einen Leitterm, der echt kleiner ist als der erwartete Leitterm $m = lcm(lt(f), lt(g))$. Auf diese Weise entstehen Polynome, deren Leitterm möglicherweise nicht in $\Sigma(B)$ enthalten ist. Durch Hinzunahme dieser Polynome in die Basis vergrößern wir also $\Sigma(B)$. Fügen wir etwa im Beispiel B_2 zur Basis die drei neu erzeugten Polynome hinzu,

```
vars:[u,x,y,z];
[f1,f2,f3]:[u*x-y^2,u*y-z^2,u*z-x^2];
f4:poly_s_polynomial(f1,f2,vars);
f5:poly_s_polynomial(f1,f3,vars);
f6:poly_s_polynomial(f2,f3,vars);
```

so erhalten wir eine neue Basis $B_{2a} = \{f_1, f_2, f_3, f_4, f_5, f_6\}$ mit

$$\Sigma(B_{2a}) = \left(u\,x, u\,y, u\,z, x\,z^2, x^3, x^2y\right) \supset \Sigma(B_2)\,.$$

Bzgl. dieser neuen Basis ist offensichtlich

$$NF(S(f_i, f_j), B_{2a}) = 0$$

für $(i, j) \in \{(1,2), (1,3), (2,3)\}$. Andererseits können zwischen den neuen und alten Elementen neue S-Polynome konstruiert werden:

```
s14:poly_s_polynomial(f1,f4,vars);
```

$$s_{14} = z^2\,f_1 - u\,f_4 = (u\,x\,z^2 - y^2z^2) - (u\,x\,z^2 - u\,y^3) = u\,y^3 - y^2z^2$$

Im Gegensatz zu obigen Polynomen ist dieses Polynom nicht bereits in Normalform bzgl. B_{2a}. Es gilt

```
poly_normal_form(s14,[f1,f2,f3,f4,f5,f6],vars);
```

$$s_{14} \mapsto_{f_2} 0$$

also $NF(s_{14}, B_{2a}) = 0$. Damit entsteht aus diesem Polynom kein neues Polynom aus I mit bis dahin unbekanntem Leitterm. Dasselbe gilt für die S-Polynome $s_{ij}, i \in \{1,2,3\}, j \in \{4,5,6\}$. Die restlichen S-Polynome liefern

$$\begin{aligned}
s_{45} &= -x^2y^3 + y^2z^3 \mapsto_{f_6} 0\\
s_{46} &= -x\,y^4 + z^5 =: f_7\\
s_{56} &= x\,z^3 - y^3z \mapsto_{f_4} 0
\end{aligned}$$

Wir erhalten also ein weiteres Polynom $f_7 \in I$ mit bis dahin unbekanntem Leitterm $x\,y^4$ und schließlich wegen

$$s_{47} = y^4f_4 + z^2f_7 = -y^7 + z^7 =: f_8$$

ein letztes solches Polynom. Die Normalformen aller weiteren S-Polynome verschwinden, so dass wir auf einfachem Wege keine neuen Polynome $f \in I$ mit $lt(f) \notin \Sigma(G)$ für $G = \{f_1, \ldots, f_8\}$ konstruieren können. Weiter unten werden wir sehen, dass G in der Tat bereits eine Gröbnerbasis ist.

Betrachten wir unser zweites Beispiel: Für $I(B_3)$ hatten wir bereits eine neue Idealbasis

$$B_{3b} := \{f_3, f_5, f_6\} = \left\{x + y + z^2 - 3, y^2 - y - z^2 + z, 2\,y\,z^2 - 4\,y + z^4 - 5\,z^2 + 6\right\}$$

konstruiert. Hier liefert nur $NF(S(f_5, f_6), B_{3b})$ ein neues nichttriviales Polynom f_7:

$$\begin{aligned}
S(f_5, f_6) = z^2\,f_5 - \frac{1}{2}\,y\,f_6 &= 2\,y^2 - \frac{1}{2}\,y\,z^4 + \frac{3}{2}\,y\,z^2 - 3\,y - z^4 + z^3\\
&\mapsto_{f_5} -\frac{1}{2}\,y\,z^4 + \frac{3}{2}\,y\,z^2 - y - z^4 + z^3 + 2\,z^2 - 2\,z\\
&\mapsto_{z^2f_6} \frac{1}{2}\,y\,z^2 - y + \frac{1}{4}\,z^6 - \frac{9}{4}\,z^4 + z^3 + \frac{7}{2}\,z^2 - 2\,z\\
&\mapsto_{f_6} f_7 := \frac{1}{4}\left(z^6 - 10\,z^4 + 4\,z^3 + 19\,z^2 - 8\,z - 6\right)
\end{aligned}$$

Alle S-Polynome, die man seinerseits mit f_7 bilden kann, reduzieren zu null. Auch hier gilt, wie wir nun zeigen wollen, dass $G := \{f_3, f_5, f_6, f_7\}$ bereits eine Gröbnerbasis ist.

Der Charakterisierungssatz für Gröbnerbasen

Satz 24 (Charakterisierungssatz für Gröbnerbasen) *Die folgenden Bedingungen an eine Basis G eines Ideals $I \subset R$ (bzgl. einer auf $T(X)$ fixierten Termordnung) sind äquivalent:*

1. *G ist eine Gröbnerbasis von I, d. h. es gilt $\Sigma(I) = \Sigma(G)$.*
2. *Für jedes Element $f \in I$ und jede Reduktionsstrategie gilt $NF(f, G) = 0$.*
3. *Für jedes Element $f \in I$ gibt es eine Reduktionsstrategie mit $NF(f, G) = 0$.*
4. *Für jedes Paar $g_1, g_2 \in G$ und jede Reduktionsstrategie gilt $NF(S(g_1, g_2), G) = 0$.*
5. *Für jedes Paar $g_1, g_2 \in G$ gibt es eine Reduktionsstrategie mit $NF(S(g_1, g_2), G) = 0$.*
6. *Jedes Element $f \in I$ hat eine Darstellung*
$$f = \sum_{g \in G} h_g g \quad \textit{mit} \quad \forall\, g\ (lt(f) \geq lt(h_g g)).$$
7. *Die Standardterme $N(G) := T(X) \setminus \Sigma(G)$ sind k-linear unabhängig* $\pmod I$.
8. *Die Standardterme $N(G)$ bilden eine k-Vektorraumbasis des Faktorrings R/I, d. h. jedes Element $f \in R$ besitzt eine eindeutige Darstellung*
$$f \equiv \sum_{m \in N(G)} c_m m \pmod I$$
mit $c_m \in k$.

Beweis: Wir hatten bereits gesehen, dass 1. $\Rightarrow$ 2. gilt. Da S-Polynome spezielle Elemente aus I sind, sind die Implikationen 2. $\Rightarrow$ 3. $\Rightarrow$ 5. und 2. $\Rightarrow$ 4. $\Rightarrow$ 5. trivial, so dass nur noch 5. $\Rightarrow$ 6. und 6. $\Rightarrow$ 1. zu zeigen bleibt.

5. $\Rightarrow$ 6.: Sei $f = \sum h_g g \in I$, aber $lt(f) < m := max(lt(h_g g))$. Im folgenden bezeichnet $f_1, f_2, \ldots$ Kombinationen $\sum h'_g g \in I$ mit $max(lt(h'_g g)) < m$. Zunächst ist

$$f = \sum\nolimits^{*} h_g g + f_1 = \sum\nolimits^{*} (lm(h_g) + red(h_g))\, g + f_1 = \sum\nolimits^{*} lm(h_g)\, g + f_2\,,$$

wobei sich die Summation $\sum^*$ nur über diejenigen $g \in G$ erstreckt, für die $lt(h_g g) = m$ gilt. Insbesondere ist $lt(g) \,|\, m$ für $g \in *$. Betrachten wir den Ausdruck unter dem Summenzeichen. Für jeden einzelnen Summanden gilt nach Konstruktion $lt(h_g)\, lt(g) = m > lt(f)$, also ist $\sum^* lc(h_g)\, lc(g) = 0$. Sei o. B. d. A $g_1 \in *$. Dann gilt $lt(g_1) \,|\, m$ und

$$\sum\nolimits^{*} lc(h_g)\, lc(g)\, \frac{m}{lm(g_1)}\, g_1 = 0\,.$$

m ist ein gemeinsames Vielfaches aller $lt(g)$, $g \in *$, also insbesondere ein Vielfaches von $m_g := lcm(lt(g_1), lt(g))$. Deshalb ist

$$\sum\nolimits^{*} lm(h_g)\, g = \sum\nolimits^{*} lc(h_g)\, lc(g)\, \frac{m}{lm(g_i)}\, g_i = \sum\nolimits^{*} lc(h_i)\, lc(g_i)\, \frac{m}{m_i}\, S(g_i, g_1)\,,$$

wenn man die oben hergeleitete Null-Summe hinzufügt. Da aber $lt(S(g_i, g_1)) < m_i$ und außerdem $NF(S(g_i, g_1), G) = 0$ gilt, erhalten wir auf diese Weise eine Darstellung von f als Kombination $\sum h'_g g$ mit $max(lt(h'_g g)) < m$. Noethersche Induktion beweist dann die Aussage 6.

6. $\Rightarrow$ 1.: Aus der Existenz der genannten Darstellung für $f \in I$ folgt $lt(f) = max_g(lt(h_g)\, lt(g))$, also $lt(f) \in \Sigma(G)$.

1. $\Rightarrow$ 7.: Wäre

$$f = \sum_{m \in N(G)} c_m m \equiv 0 \pmod I$$

eine nichttriviale Linearkombination von Standardmonomen aus I, dann wäre wegen $f \in I$ auch $lt(f) \in \Sigma(I) = \Sigma(G)$.

7. $\Rightarrow$ 8.: Jedes Nichtstandardmonom kann mit Hilfe von $TNF(\_, G)$ als Linearkombination von Standardmonomen dargestellt werden. Also bilden diese auch ein Erzeugendensystem.

8. $\Rightarrow$ 1.: Wäre $m \in \Sigma(I) \setminus \Sigma(G)$ und $f \in I$ mit $lt(f) = m$, so wäre $f' := TNF(f, G) \in I$ eine nichttriviale Linearkombination von Termen aus $N(G)$. $\square$

Der Charakterisierungssatz zeigt, dass die oben berechnete Menge $G = \{f_1, \ldots, f_8\}$ eine Gröbnerbasis von $I = I(B_2)$ ist, weil sie die Eigenschaft 5. erfüllt.

Wir wollen diesen Abschnitt mit einem Kriterium beschließen, das es erlaubt, manche S-Polynome nicht zu untersuchen.

Satz 25 *(Hauptsyzygienkriterium) Sind $f, g \in R$ nichttriviale Polynome mit teilerfremden Leittermen, so gilt $NF(S(f,g), \{f,g\}) = 0$.*

Beweis: Aus der Teilerfremdheit folgt $m = lcm(lt(f), lt(g)) = lt(f) \cdot lt(g)$ und somit

$$S(f,g) = \frac{m}{lm(f)}\, red(f) - \frac{m}{lm(g)}\, red(g) = \frac{1}{lc(f)\, lc(g)}\, (lm(g)\, red(f) - lm(f)\, red(g))$$

Führen wir nun die Substitutionen $lm(f) \mapsto -red(f)$ und $lm(g) \mapsto -red(g)$ aus, so erhalten wir 0. $\square$

Betrachten wir noch einmal die Rechnungen zum Beispiel

$$B_{3b} = \{f_3, f_5, f_6\} = \left\{x + y + z^2 - 3, y^2 - y - z^2 + z, 2\,y\,z^2 - 4\,y + z^4 - 5\,z^2 + 6\right\}.$$

Da die anderen Leittermpaare zueinander teilerfremd sind brauchen wir nach dem Hauptsyzygienkriterium nur $S(f_5, f_6)$ zu untersuchen, was in der Tat ein neues Polynom

$$f_7 := z^6 - 10\,z^4 + 4\,z^3 + 19\,z^2 - 8\,z - 6$$

liefert. Von diesem brauchen wir nur $S(f_6, f_7)$ zu untersuchen, was zu 0 reduziert.

Der Buchberger-Algorithmus

Ähnlich wie oben können wir auch im allgemeinen Fall versuchen, aus einer gegebenen Idealbasis B eine Gröbnerbasis zu konstruieren. Wir versuchen, nacheinander alle S-Polynome $S(f_i, f_j),\ f_i, f_j \in B$ vermöge B zu reduzieren. Ist $f := NF(S(f_i, f_j), B) \neq 0$, so wissen wir zumindest, dass $lt(f) \in \Sigma(I) \setminus \Sigma(B)$, d.h. $f \in I$ ein Element aus dem Ideal ist, dessen Leitterm man aus den bisher bekannten Basiselementen nicht herleiten kann. Fügen wir andererseits dieses Polynom zur Menge B hinzu, kann $S(f_i, f_j)$ nunmehr trivialerweise zu null reduziert werden. Durch die Hinzunahme des neuen Basiselements vergrößert sich andererseits die Anzahl der möglichen S-Polynome, so dass die Termination dieses Vorgehens nicht offensichtlich ist. Im Beispiel der Menge B_1 jedenfalls terminierte dieses Verfahren.

Die formale Spezifikation des entsprechenden Algorithmus, den man zu Ehren seines Erfinders den **Buchergeralgorithmus** nennt, sieht wie folgt aus:

GBasis(B: Basis): Basis

Input: Endliche Menge $B = \{f_1, \ldots, f_m\} \subset R$.

Output: Gröbnerbasis G des Ideals $I = I(B)$.

`G:=B;`
$P := \{(f_i, f_j) \,|\, 1 \leq i < j \leq m\}$;

```
While P ≠ ∅ do
  Choose p ∈ P; P := P \ {p};
  f := NF(S(p), G)
  if f ≠ 0 then
    P := P ∪ {(g, f) | g ∈ G};
    G := G ∪ {f};
return G;
```

Satz 26 *Der Algorithmus terminiert nach endlich vielen Schritten.*

Beweis: In jedem Schritt mit $f \neq 0$ wird $\Sigma(G)$ echt vergrößert. Nach dem Dicksonlemma sind aber nur endlich viele solche Schritte möglich. □

Satz 27 *Ist G eine Gröbnerbasis des Ideals I und $G' \subset G$ eine Teilmenge, so dass*

$$Gen(\Sigma(G)) = \{lt(g),\, g \in G'\}$$

gilt, so ist auch G' eine Gröbnerbasis von I. Eine solche Gröbnerbasis nennt man minimal.

Der Beweis dieses Satzes ergibt sich sofort aus der Definition einer Gröbnerbasis. Auf Grund der Eindeutigkeit der Minimalbasis des Monoidideals $\Sigma(I)$ ist die Menge $\{lt(g),\, g \in G'\}$ eindeutig bestimmt. Die Menge

$$G'' = \{lt(g) - TNF(lt(g), G'),\, g \in G'\} \subset I$$

bezeichnet man schließlich als *minimale reduzierte Gröbnerbasis*. Jedes dieser Polynome ist die Differenz zwischen einem minimalen Nichtstandardterm und dessen eindeutiger Darstellung $\pmod I$ als Linearkombination von (in der Termordnung kleineren) Standardtermen. Offensichtlich ist eine solche minimale reduzierte Gröbnerbasis eindeutig bestimmt. Die Funktion `poly_reduced_grobner` aus dem `grobner`-Paket von MAXIMA berechnet eine solche minimale reduzierte Gröbnerbasis.

Beispiel: Gröbnerbasisberechnung für die Idealbasis aus dem Beweis des Satzes vom Schnittpunkt der Winkelhalbierenden im Dreieck:

```
A:Point(0,0); B:Point(1,0); C:Point(cx,cy); P:Point(px,py);
 [f1,f2]:[ on_bisector(P,B,A,C), on_bisector(P,C,B,A) ];
vars:[px,py,cx,cy];
```

$$f_1 = \underline{c_y\, p_y^2} + 2\, c_x\, p_x\, p_y - c_y\, p_x^2$$
$$f_2 = \underline{-c_y\, p_y^2} - 2\, c_x\, p_x\, p_y + 2\, p_x\, p_y + 2\, c_x\, p_y - 2\, p_y + c_y\, p_x^2 - 2\, c_y\, p_x + c_y$$

An dieser Stelle ist $G = \{f_1, f_2\}$, $P = \{(f_1, f_2)\}$. Berechne $S(f_1, f_2)$:

```
f3:poly_s_polynomial(f1,f2,vars);
h3:poly_normal_form(f3,[f1,f2],vars);
```

$$h_3 = \underline{2\, p_x\, p_y} + 2\, c_x\, p_y - 2\, p_y - 2\, c_y\, p_x + c_y$$

An dieser Stelle ist $G = \{f_1, f_2, h_3\}$, $P = \{(f_1, h_3), (f_2, h_3)\}$. Berechne $S(f_1, h_3)$:

```
f4:poly_s_polynomial(f1,h3,vars);
h4:poly_normal_form(f4,[f1,f2,h3],vars);
```

$$\begin{aligned} h_4 = &\underline{-2\, c_y\, p_y^3} + 2\, c_y^2\, p_y^2 + 4\, c_x^2\, p_y^2 - 4\, c_x\, p_y^2 + 2\, c_x^2\, c_y\, p_y - 2\, c_x\, c_y\, p_y + 2\, c_y\, p_y \\ &- 2\, c_x\, c_y^2\, p_x + c_y^2\, p_x + c_x\, c_y^2 - c_y^2 \end{aligned}$$

An dieser Stelle ist $G = \{f_1, f_2, h_3, h_4\}$, $P = \{(f_2, h_3), (f_1, h_4), (f_2, h_4), (h_3, h_4)\}$. Berechne $S(f_2, h_3)$:

```
f5:poly_s_polynomial(f2,h3,vars);
h5:poly_normal_form(f5,[f1,f2,h3,h4],vars);
```

Dieses reduziert zu null, ebenso $S(f_1, h_4)$ und $S(f_2, h_4)$. Berechne $S(h_3, h_4)$:

```
f6:poly_s_polynomial(h3,h4,vars);
h6:poly_normal_form(f6,[f1,f2,h3,h4],vars);
```

$$\begin{aligned} h_6 =& \underline{-4\,c_y\,p_y^4} + 8\,c_y^2\,p_y^3 + 8\,c_x^2\,p_y^3 - 8\,c_x\,p_y^3 - 4\,c_y^3\,p_y^2 - 4\,c_x^2\,c_y\,p_y^2 + 4\,c_x\,c_y\,p_y^2 \\ & + 4\,c_y\,p_y^2 - 4\,c_y^2\,p_y + c_y^3 \end{aligned}$$

An dieser Stelle ist $G = \{f_1, f_2, h_3, h_4, h_8\}$, $P = \{(f_1, h_8), (f_2, h_8), (h_3, h_8), (h_4, h_8)\}$. Alle weiteren Paare reduzieren zu null, der Buchberger-Algorithmus terminiert mit $G = \{f_1, f_2, h_3, h_4, h_8\}$. Wegen $lt(f_1) = lt(f_2)$ kann f_1 weggelassen werden, denn $\Sigma(G)$ ändert sich dadurch nicht. Durch Interreduktion

```
G:poly_reduction([f2,h3,h4,h6],vars);
```

kann daraus schließlich die eindeutig bestimmte minimale Gröbnerbasis berechnet werden.

Kapitel 5

Anhang: Geometrietheoreme „im Allgemeinen“

5.1 Geometriebeweise. Weitere Probleme

Degenerationsbedingungen

Beim Beweis des Satzes über die Winkelhalbierenden haben wir eine für die Rechnung günstige AGK gewählt. Wären wir von einem Schnittpunkt der Winkelhalbierendenpaare durch B und C gestartet und hätten zeigen wollen, dass dieser auch auf dem Winkelhalbierendenpaar durch A liegt, so wären wir zunächst gescheitert. Auch diesmal enthält die Gröbnerbasis vier Elemente, aber g kann damit nicht zu null reduziert werden.

```
polys:[ on_bisector(P,C,B,A), on_bisector(P,A,C,B) ];
g:on_bisector(P,B,A,C);
G:poly_reduced_grobner(polys,vars);
g1:poly_normal_form(g,G,vars);
```

$$g_1 = 2\,p_y - c_y - 2\,c_x\,p_y + 2\,c_y\,p_x - 2\,p_x\,p_y$$

Wegen $g \equiv g_1 \not\equiv 0 \pmod{I}$ gilt sogar $g \notin I$. Dies ergibt sich aus der Gröbnerbasen-Eigenschaft von G. Allerdings reduziert

```
poly_normal_form(((cx-1)^2+cy^2)*g,G,vars);
```

zu null. Für das Polynom $h = (c_x - 1)^2 + c_y^2$ gilt also $h \cdot g \in I$. Offensichtlich ist der Satz $(F/h \Rightarrow g)$ zum Satz $(F \Rightarrow g \cdot h)$ äquivalent und unser Satz damit unter der Nichtdegenerationsbedingung h bewiesen. Dies gilt auch allgemein.

Satz 28 *Ist G eine Basis von F und* `NF`$(h^k\,g, G, X) = 0$ *für ein $k \geq 0$, so gilt der Satz $(F/h \Rightarrow g)$.*

Die Frage nach der geometrischen Relevanz dieser Nichtdegenerationsbedingung bleibt noch zu diskutieren. h ist genau die geometrische Bedingung `sqrdist`$(B, C) = 0$. Über den reellen Zahlen entspricht dies $B = C$, so dass die Gerade BC als ein Schenkel entartet. Im Allgemeinen bedeutet `sqrdist`$(B, C) = 0$, dass die Gerade $l = BC$ isotrop ist, also senkrecht auf sich selbst steht. In unserer ersten Rechnung waren wir von den Winkelhalbierenden durch A und B ausgegangen und die spezielle Wahl des Koordinatensystems hatte dafür gesorgt, dass A, B nicht isotrop war. Das „o. B. d. A.“ ist also möglicherweise doch nicht ganz unbedenklich, denn die Eigenschaft einer Geraden, isotrop zu sein, ist natürlich vom Koordinatensystem unabhängig. Wenn eine der drei Seiten des Dreiecks nicht isotrop ist, dann können wir unseren Beweis stets mit dieser Seite starten.

Es bleibt also offen, wie das mit den Winkelhalbierenden für Dreiecke ist, dessen Seiten sämtlich isotrop sind. Diesen Fall haben wir auch mit unserer ersten Berechnung (durch die spezielle Wahl der Koordinaten für A und B) nicht untersucht.

Wir wollen diese komplizierten Fragen der geometrischen Interpretation von auf algebraischem Weg gefundenen Nichtdegenerationsbedingungen hier nicht weiter verfolgen und uns der zweiten Frage zuwenden, wie derartige Nichtdegenerationsbedingungen algebraisch gefunden werden können. Eine systematische Suche könnte zum Beispiel nach der Menge

$$\{h \in R : h{\cdot}g \in I\}$$

oder allgemeiner

$$\left\{h \in R : \exists\, k \in \mathbb{N}\ h^k{\cdot}g \in I\right\}$$

aller Polynome fragen, die den „Beweis retten“. Es stellt sich heraus, dass diese Mengen Ideale sind und als Idealquotient bzw. stabiler Idealquotient berechnet werden können.

Ein indirekter Beweisansatz

Ein indirekter Beweisansatz ergibt sich aus folgender Überlegung: Ist g ein Polynom, das auf allen gemeinsamen Nullstellen von F verschwindet und t eine neue Variable, so hat offensichtlich das System $F = 1 - t{\cdot}g = 0$ keine gemeinsamen Lösungen. Ist umgekehrt X_0 eine Lösung mit $F(X_0) = 0$, aber $g(X_0) \neq 0$, so hat $F = 1 - t{\cdot}g = 0$ die Lösung (t_0, X_0) mit $t_0 = \frac{1}{g(X_0)}$.

$V(F, 1 - t{\cdot}g) = \emptyset$ ist aber nach dem Hilbertschen Nullstellensatz (über einem algebraisch abgeschlossenen Körper K) äquivalent dazu, dass das Ideal $I = I(F, 1 - t{\cdot}g) \subset k[t, X]$ das Einsideal ist, was wiederum durch eine Gröbnerbasisberechnung geprüft werden kann.

Nach dem Hilbertschen Nullstellensatz hat ein Gleichungssystem B über einem algebraisch abgeschlossenen Körper K genau dann eine nichttriviale Nullstellenmenge $V(B) \neq \emptyset$, wenn $I(B)$ nicht das Einsideal $I(1)$ ist. Da sich letzteres aber durch $\Sigma(I(1)) = T(X) = (1)$ auszeichnet, erhalten wir die folgende Äquivalenz von Aussagen:

Satz 29 *Gegeben sei ein polynomiales Gleichungssystem $B \subset R = k[x_1, \ldots, x_n]$ mit Koeffizienten aus einem Körper k und ein algebraisch abgeschlossener Erweiterungskörper K von k. Folgende Aussagen sind dann äquivalent:*

1. *$V_K(B) = \emptyset$, d. h. B hat keine gemeinsamen Nullstellen über K.*
2. *$I(B) = I(1)$ ist das Einsideal.*
3. *Jede Gröbnerbasis $G =$ `GBasis`(B) enthält ein konstantes Polynom.*
4. *$\{1\}$ ist die minimale reduzierte Gröbnerbasis von B.*

Satz 30 *Ist das Ideal $I(F, 1 - t{\cdot}g)$ das Einsideal, so gilt der Satz $(F \Rightarrow g)$.*

Für den Satz über die Winkelhalbierenden ergibt sich für diesen Zugang in der ersten Variante

```
polys:[ on_bisector(P,B,A,C), on_bisector(P,C,B,A) ];
g:on_bisector(P,A,C,B);
G:poly_reduced_grobner(append(polys,[1-t*g]),[t,px,py,cx,cy]);
```

unmittelbar das Einsideal, also $V(F) \subset V(g)$.

In der zweiten Variante

```
polys:[ on_bisector(P,C,B,A), on_bisector(P,A,C,B) ];
g:on_bisector(P,B,A,C);
G:poly_reduced_grobner(append(polys,[1-t*g]),[t,px,py,cx,cy]);
```

ergibt sich leider nicht das Einsideal, sondern eine komplizierte Gröbnerbasis G, deren Leitterme wir inspizieren können

```
map(lambda([u],lm(u,vars)),G);
```

Der dritte Leitterm ist c_x^2, das entsprechende Polynom $g_3 = c_y^2 + c_x^2 - 2\,c_x + 1 = c_y^2 + (c_x - 1)^2$ ein Polynom allein in den Variablen c_x, c_y, von deren Unabhängigkeit wir ausgegangen. Die Bedingung entspricht der Eigenschaft, dass die Gerade AC isotrop ist, also einer degenerierten Lage.

Dies ist außerdem genau das weiter oben verwendete Polynom h, so dass $V(F) \cap V(1-t{\cdot}g) \subset V(h)$ gilt und damit unser Satz ein weiteres Mal unter der Nichtdegenerationsbedingung h bewiesen ist. Dieses Ergebnis gilt auch allgemein

Satz 31 *Ist G eine Basis von $I = I(F, 1 - t{\cdot}g)$ und $\mathtt{NF}(h, G, [t, X]) = 0$ für ein $h \in R$, so gilt der Satz $(F/h \Rightarrow g)$.*

Beweis: Aus der Rechnung folgt $h \in I(F, 1-t{\cdot}g)$, also die Existenz einer polynomialen Darstellung in $k[t, X]$ der

$$h(X) = \sum_i p_i(X,t){\cdot}f(X) + p(X,t){\cdot}(1 - t{\cdot}g(X))\,. \qquad (*)$$

Gäbe es nun ein $X_0 \in \mathbb{A}^n$ mit $F(X_0) = 0$, aber $g(X_0) \neq 0$, so können wir diesen Wert in die Darstellung $(*)$ einsetzen:

$$h(X_0) = \sum_i p_i(X_0,t){\cdot}f(X_0) + p(X_0,t){\cdot}(1 - t{\cdot}g(X_0)) = p(X_0,t){\cdot}(1 - t{\cdot}g(X_0))$$

wegen $f(X_0) = 0$ für alle $f \in F$. Auf der linken Seite steht ein Polynom vom Grad 0 in t, auf der rechten Seite ein Polynom von positivem Grad in t. Die Annahme der Existenz eines solchen X_0 führt also auf einen Widerspruch.

Allgemein ist das Ideal der t-freien Polynome $I(F, 1 - t{\cdot}g) \cap k[X]$ gerade der *stabile Idealquotient*

$$I : g^\infty = \{h \in R : \exists n > 0\ (g^n{\cdot}h \in I)\}$$

und damit im Wesentlichen die Menge $\{h \in R : V(g{\cdot}h) \supset V(F)\}$ derjenigen Polynome, für welche der Satz $(F/h \Rightarrow g)$ gilt. □

5.2 Gültigkeit von Geometrietheoremen „im Allgemeinen“

Gröbnerbasen und das Eliminationsproblem

Viele konstruktive Fragestellungen der Algebra lassen sich auf Eliminationsprobleme zurückführen. Diese lassen sich ebenfalls mit Gröbnerbasen konstruktiv behandeln.

Sei $B \subset R = k[X]$ eine endliche Menge von Polynomen und die Menge der Variablen in zwei Teilmengen $X = (x_1, \ldots, x_k, y_1, \ldots, y_m)$ aufgeteilt. Wir fragen nach den Polynomen im Ideal $I = I(B)$, die $x_1, \ldots, x_k$ nicht enthalten, also nach einer Basis des *Eliminationsideals*

$$I' = I(B) \bigcap k[y_1, \ldots, y_m].$$

Zu dessen Berechnung wählen wir auf $T(X)$ eine Termordnung, in der jeder Term, der eine Variable x_i enthält, größer ist als jeder Term, der nur Variablen y_j enthält. Solche Termordnungen bezeichnet man als *Eliminationsordnungen* für $(x_1, \ldots, x_k)$, da ein Polynom $f(x_1, \ldots, x_k, y_1, \ldots, y_m)$ genau dann keine der Variablen $x_1, \ldots, x_k$ enthält, wenn dies für dessen Leitterm $lt(f)$ gilt.

Neben der lexikografischen Ordnung gibt es eine Reihe anderer Eliminationsordnungen, bzgl. derer sich Gröbnerbasen gewöhnlich schneller ausrechnen lassen.

Satz 32 *Ist $R = k[x_1, \ldots, x_k, y_1, \ldots, y_m]$ und $G = GBasis(B)$ eine (min. reduzierte) Gröbnerbasis des Polynomsystems $B \subset R$ bzgl. einer Eliminationsordnung für $(x_1, \ldots, x_k)$, so ist*

$$G' = \{g \in G : lt(g) \in T(y_1, \ldots, y_m)\}$$

eine (min. reduzierte) Gröbnerbasis des Eliminationsideals $I' = I(B) \bigcap k[y_1, \ldots, y_m]$.

Beweis: Offensichtlich gilt $G' \subset I'$. Ist $f \in I'$, so gilt $f \in I$ und folglich $NF(f, G) = 0$. Da bei der Reduktion aber nur solche $g \in G$ mit $lt(g) \leq lt(f)$, also $g \in G'$ herangezogen werden, gilt $NF(f, G') = 0$. Diese Eigenschaft charakterisiert aber Gröbnerbasen. □

Die lexikografische Termordnung ist eine Eliminationsordnung für jedes Anfangssegment der Variablen. Damit hat eine Gröbnerbasis bgzl. dieser Ordnung eine „Dreiecksgestalt“.

Folgerung 6 *Ist $R = k[x_1, \ldots, x_n]$ und $G = GBasis(B)$ eine (min. reduzierte) Gröbnerbasis von $B \subset R$ bzgl. der lexikografischen Termordnung, in welcher $x_1 > \cdots > x_n$ gilt, so ist*

$$G_i = \{g \in G : lt(g) \in T(x_i, \ldots, x_n)\}$$

eine (min. reduzierte) Gröbnerbasis des Eliminationsideals $I(B) \bigcap k[x_i, \ldots, x_n]$.

Unabhängige und abhängige Variablen

Wir haben bereits begonnen, auf einer heuristischen Basis zwischen unabhängigen und abhängigen Variablen zu unterscheiden. Dies wollen wir nun präzisieren. Im konstruktiven Fall war eine wesentliche Voraussetzung für die Gültigkeit des entsprechenden Mechanisierungssatzes die *algebraische* Unabhängigkeit der Variablen gewesen. Ist $I \subset R = k[X]$ ein Ideal, so heißt eine Teilmenge $U \subset X$ der Variablen *unabhängig* bzgl. I, wenn $I \cap k[U] = (0)$ gilt, d. h. I kein nicht triviales Polynom allein in den U-Variablen enthält.

Der Nachweis, dass eine vorgegebene Variablenmenge unabhängig ist, kann nach Definition auf ein Eliminationsproblem reduziert werden, das mit der Berechnung einer Gröbnerbasis konstruktiv gelöst werden kann:

Satz 33 *Ist G eine Gröbnerbasis von I bzgl. einer Eliminationsordnung für U und*

$$G' = \{g \in G : lt(g) \in k[U]\} ,$$

so ist U genau dann unabhängig bzgl. I, wenn $G' = \emptyset$ gilt.

Da degenerierte Lagen in diesem Sinne stets durch Bedingungen auf die ansonsten frei wählbaren Parameter charakterisiert sind, sind vor allem Polynome $h \in k[U]$ als Nichtdegenerationsbedingungen von geometrischer Relevanz. Genau ein solches Polynom haben wir für unser Beispiel oben (zweite Variante des Schnitts der Winkelhalbierenden) bereits in G selbst gefunden.

Generisch gültige Geometrietheoreme

Im Fall der Geometrietheoreme vom konstruktiven Typ haben wir die Menge der Variablen $X = (Y, U)$ in unabhängige und abhängige unterteilt und lineare Algebra über dem Körper $k(U)$ der rationalen Funktionen in U getrieben. Wir wollen deshalb nun auch für den allgemeinen Fall im Ring $S = k(U)[Y]$ der Polynome in Y mit rationalen Funktionen in U als Koeffizienten rechnen statt wie bisher im Ring $R = k[X] = k[Y, U]$. Wir wollen dabei voraussetzen, dass U eine maximale bzgl. $I = I(F)$ unabhängige Variablenmenge ist.

Betrachten wir zunächst einige Beispiele.

Beispiel 1: Satz vom Miquelschen Punkt.

Satz 34 *Sind im Dreieck ABC $P \in BC$, $Q \in AC$ und $R \in AB$ Punkte auf den Dreiecksseiten auf den Dreiecksseiten, so gehen die Umkreise der Dreiecke AQR, BPR und CPQ durch einen gemeinsamen Punkt.*

Wir wählen als AGK die folgende Basiskonfiguration aus freien Punkten A, B, C und Geradengleitern P, Q, R sowie S als weiteren freien Punkt, der durch die AGV als Schnittpunkt der Umkreise von AQR und BPR ausgezeichnet wird.

```
A:Point(0,0); B:Point(1,0); C:Point(cx,cy);
P:varpoint(B,C,ps); Q:varpoint(A,C,qs); R:varpoint(A,B,rs);
S:Point(sx,sy);

polys:[ is_concyclic(A,Q,R,S), is_concyclic(B,P,R,S) ];
g:is_concyclic(C,P,Q,S);
```

Die beiden AGV-Polynome legen nahe, die Variablen $Y = (s_x, s_y)$ als abhängig zu betrachten, die restlichen als unabhängig. Dies kann an dieser Stelle zwar nicht begründet werden, doch wir rechnen erst einmal los – vielleicht ergibt sich eine Begründung im Nachhinein.

```
yvars:[sx,sy]; uvars:[cx,cy,ps,qs,rs];
G:poly_reduced_grobner(polys,yvars);
poly_normal_form(g,G,yvars);
```

Leider haben wir keinen Erfolg, die Normalform reduziert nicht zu null. Der Grund ist allerdings klar, denn von den *zwei* Möglichkeiten für S ist eine der bereits bekannte Schnittpunkt R der beiden Kreise, der natürlich nicht auf dem dritten Kreis liegt. $V(F) = V_1 \cup V_2$ besteht also aus zwei Komponenten und der Satz gilt nur auf einer von ihnen.

Da die Gröbnerbasis eine recht einfache Struktur

$$\left[s_x - p_1(U)\, s_y - r_s,\ s_y^2 - p_2(U)\, s_y\right]$$

hat, lassen sich die beiden Lösungen $(s_x, s_y) \in \mathbb{A}^2_{k(U)}$ des zugehörigen Gleichungssystems mit $s_{y.1} = 0$ und $s_{y.2} = p_2(U)$ leicht bestimmen. Eine Lösung entspricht dem Schnittpunkt R der beiden Kreise, die andere dem Punkt S und erfüllt die Gleichung g:

```
sol:solve(G,yvars);
ratsimp(subst(sol[1],g));
ratsimp(subst(sol[2],g));
```

Die Nullstellenmenge $V(F) \subset \mathbb{A}^2_{k(U)}$ besteht in diesem Fall also aus zwei (nulldimensionalen über $k(U)$) Komponenten, wobei der Satz auf einer Komponente gilt, auf der zweiten (aus nahe liegenden Gründen) dagegen nicht. Es handelt sich dabei jedoch nicht um eine degenerierte Lage wie in früheren Beispielen, sondern um einen essentiell auszuschließenden Fall.

Beispiel 2: Satz vom Schnittpunkt der Winkelhalbierenden In der zweiten Variante unserer bisherigen Rechnungen ergibt sich mit der natürlichen Einteilung in abhängige $Y = (p_x, p_y)$ und unabhängige $U = (c_x, c_y)$ Variablen

```
A:Point(0,0); B:Point(1,0); C:Point(cx,cy); P:Point(px,py);
polys:[ on_bisector(P,C,B,A), on_bisector(P,A,C,B) ];
g:on_bisector(P,B,A,C);

yvars:[px,py]; uvars:[cx,cy];
G:poly_reduced_grobner(polys,yvars);
```

$$p_x + p_1(U)\,p_y^3 + p_2(U)\,p_y^2 + p_3(U)\,p_y + p_4(U)$$
$$p_y^4 + q_1(U)\,p_y^3 + q_2(U)\,p_y^2 + q_3(U)\,p_y + q_4(U)$$

Auch in diesem Fall ist das von F in $S = k(U)[p_x, p_y]$ erzeugte Ideal $I' = I(F)\cdot S = I(G)$ nulldimensional und hat genau vier Nullstellen in $\mathbb{A}^2_L$, dem zweidimensionalen affinen Raum über L, dem algebraischen Abschluss von $k(U)$. Diese entsprechen den vier („generischen“) Schnittpunkten der möglichen Auswahlen der Halbierenden von Innen- und Außenwinkel von $\angle ABC$ und $\angle BCA$. Jeder von ihnen liegt auf der („generischen“) Halbierenden entweder des Innen- oder des Außenwinkels von $\angle CAB$, da

```
poly_normal_form(g,G,yvars);
```

zu null reduziert und in S folglich $g \in I'$ gilt. Die Probleme mit der degenerierten Situation $\texttt{sqrdist}(B, C) = 0$ treten nicht auf.

Es bleibt zu untersuchen, was das mit der ursprünglichen geometrischen Fragestellung zu tun hat.

Beispiel 3: Die Simsonsche Gerade Als AGK wählen wir einen durch das Zentrum M und einen Punkt A auf der Peripherie definierten Kreis c, weitere freie Punkte B, C, D sowie Geradengleiter R, S, T.

```
M:Point(0,0); A:Point(0,1); c:pc_circle(M,A);
B:Point(bx,by); C:Point(cx,cy); D:Point(dx,dy);
R:varpoint(B,C,rs); S:varpoint(A,C,ss); T:varpoint(A,B,ts);
```

Als AGV ergeben sich sechs Bestimmungsgleichungen

```
polys:[
  on_circle(B,c), on_circle(C,c),on_circle(D,c),
  is_orthogonal(pp_line(A,B),pp_line(D,T)),
  is_orthogonal(pp_line(A,C),pp_line(D,S)),
  is_orthogonal(pp_line(B,C),pp_line(D,R)) ];
```

und als Behauptung das Polynom

```
g:is_collinear(R,S,T);
```

Die entsprechende Setzung der Variablen und Gröbnerbasisberechnung

```
yvars:[rs,ss,ts,by,cy,dy]; uvars:[bx,cx,dx];
G:poly_reduced_grobner(polys,yvars);
poly_normal_form(g,G,yvars);
```

reduziert g wieder zu null. Die Gröbnerbasis G besteht aus 6 Elementen und es ist $\Sigma(G) = (r_s, s_s, t_s, b_y^2, c_y^2, d_y^2)$. Auch in diesem Fall ist das Ideal $I' = I(F)\cdot k(U)[Y]$ nulldimensional über $k(U)$. Aus der Gröbnerbasis lesen wir ab, dass dieses System 8 „generische“ Lösungen besitzt und g auf allen diesen Punkten verschwindet, d. h. der Satz von der Simsonschen Geraden „generisch“ (d. h. in $\mathbb{A}^2_K$) gilt.

In allen bisher betrachteten Beispielen besteht die Gröbnerbasis aus Gleichungen vom Grad 1 und 2 in den abhängigen Variablen und erlaubt es, die abhängigen Variablen durch die unabhängigen auszudrücken, auch wenn es im Gegensatz zum linearen Fall mehrere, aber stets endlich viele Lösungen für Y gibt, die sich durch komplizierte *universelle Formeln* in den Parametern U ausdrücken lassen. Diese Formeln „leben“ nicht mehr in $k(U)$, sondern in einer algebraischen Erweiterung dieses Funktionenkörpers. In manchen Fällen, wie etwa beim Miquelschen Punkt,

ist darüber hinaus der Satz nicht in allen diesen „generischen Lösungen“ richtig, sondern einige müssen ausgeschlossen werden.

Wenn U eine bzgl. $I = I(F)$ maximale unabhängige Variablenmenge ist, so ist in jedem Fall das Erweiterungsideal $I' = I \cdot S$ ein nulldimensionales Ideal und S/I' ein endlichdimensionaler $k(U)$-Vektorraum. Dessen Dimension gibt an, wie viele Lösungen Y es über dem „generischen“ Tupel U gibt. Beides kann man aus einer Gröbnerbasis $G' = \texttt{GBasis}(F, Y)$ ablesen: Dimension null liegt vor, wenn die Anzahl der Standardterme $N(G', Y) \subset T(Y)$ endlich ist, die Vektorraumdimension stimmt dann mit dieser Anzahl überein, da $N(G', Y)$ eine $k(U)$-Vektorraumbasis von S/I' ist. Diese „Punkte“ im affinen Raum über $k(U)$ werden in der algebraischen Geometrie auch als *allgemeine Punkte* der zugehörigen Varietäten bezeichnet.

Untersuchen wir zunächst den Fall, dass der geometrische Satz auf allen diesen „generischen Lösungen“ gilt. Um dies zu testen, haben wir $\texttt{NF}(g, G', Y) = 0$ für die über S berechnete Gröbnerbasis G' geprüft. Der folgende Satz gibt Auskunft, was dieses Ergebnis mit unserer ursprünglichen geometrischen Fragestellung zu tun hat.

Satz 35 (Gültigkeit geometrischer Sätze vom Gleichungstyp „im Allgemeinen“)
Sei $[F(Y,U) \Rightarrow g(Y,U)]$ *ein Satz vom Gleichungstyp,* U *eine für diesen Satz maximale unabhängige Teilmenge der Variablen und* $S = k(U)[Y]$.

Sei weiter $G' = \texttt{GBasis}(F, Y)$ *eine über* S *berechnete Gröbnerbasis.*

Gilt $\texttt{NF}(g, G', Y) = 0$, *so gibt es eine (effektiv konstruierbare) Nichtdegenerationsbedingung* $h(U)$, *so dass* g *auf allen gemeinsamen Nullstellen* (Y_0, U_0) *von* F *mit* $h(U_0) \neq 0$ *verschwindet, also der Satz* $[F/h \Rightarrow g]$ *gilt.*

Beweis: $\texttt{NF}(g, G', Y) = 0$ bedeutet $g \in I' = I(F)\cdot S$ als Ideal in S. Also gibt es eine (mit `NFRelations` berechenbare) Darstellung

$$g = \sum_i p_i(Y,U)\, f_i(Y,U) \text{ in } S$$

mit $p_i(Y,U) \in k(U)[Y]$. Bilden wir den Hauptnenner $h(U)$ zu allen Nennern in allen Koeffizienten dieser Polynome, so gilt $p_i(Y,U) = \frac{q_i(Y,U)}{h(U)}$ mit *Polynomen* $q_i(Y,U) \in k[U][Y] = R$ und folglich

$$h(U)\cdot g = \sum_i q_i(Y,U)\, f_i(Y,U) \text{ in } R$$

und damit $h(U)\cdot g \in I$. $\square$

Auch die Frage, ob U wirklich eine maximale unabhängige Teilmenge der Variablen ist, kann an Hand der Gröbnerbasis G' entschieden werden. Ist $G' = \{1\}$ (bzw. enthält ein Polynom aus $k[U]$), so ist $I' = I(F)\cdot S$ das triviale Ideal und U war in Wirklichkeit algebraisch abhängig bzgl. I. Ansonsten ist U algebraisch unabhängig modulo I und maximal genau dann, wenn S/I' nulldimensional ist, d. h. G' zu jeder Y-Variablen ein Polynom enthält, dessen Leitterm eine reine Potenz in dieser Variablen ist.

$h(U)$ im Rahmen fertiger Gröbnerpakete aufzusammeln ist schwierig, algorithmisch aber prinzipiell möglich, wenn die polynomialen Darstellungen der $q \in G'$ durch die $f \in F$ im Zuge der `GBasis`-Rechnung und von g durch die $q \in G'$ im Zuge der `NF`-Rechnung aufgesammelt werden. Allerdings folgt allein aus der *Existenz* von $h \in k[U]$, dass eine „zufällige“ Wahl von U_0 „normalerweise“ zulässig ist. Genauer: Der Satz gilt auf einer nichttrivialen Zariski-offenen (und damit dichten) Teilmenge der Parameter U. Deshalb auch „Gültigkeit im Allgemeinen“.

Es ist auch plausibel, dass wenigstens für ein Radikalideal I' aus dem Nichtverschwinden von $NF(g, G', Y)$ folgt, dass g auf wenigstens einer der „allgemeinen“ Nullstellen von F nicht verschwindet und folglich nicht allgemeingültig ist.

Eine geometrische Interpretation des Satzes über die generische Gültigkeit von Sätzen von Gleichungstyp

Um den Zusammenhang zwischen diesen „allgemeinen“ Nullstellen und dem (uns eigentlich interessierenden) Nullstellengebilde $V_R(F)$ besser zu verstehen, müssen wir den Zusammenhang zwischen Nullstellen $V_R(F)$ von F über $R = k[Y,U]$ und $V = V_S(F)$ über $S = k(U)[Y]$ genauer studieren.

Der Polynomring S enthält R als Unterring. Das von F in S erzeugte Ideal $I' = I \cdot S$ bezeichnet man als das *Erweiterungsideal* von I. Umgekehrt können wir zu einem Ideal $I' \subset S$ das *Kontraktionsideal* $I' \cap R$ in R bilden, das aus allen rationalen Kombinationen der Erzeugenden besteht, „in denen sich die Nenner wegkürzen“. Sicherlich ist $I \subset I' \cap R$.

Die Elemente des Rings S sind Polynome in den Variablen Y mit rationalen Funktionen in U als Koeffizienten. Durch Hauptnennerbildung überzeugt man sich leicht, dass man jedes Element aus S in der Form $\frac{z(Y,U)}{n(U)}$ mit $z \in R$ und $n \in N = k[U] \setminus \{0\}$ darstellen kann. Die Menge N ist *multiplikativ*, d.h. $n_1, n_2 \in N$ impliziert $n_1 n_2 \in N$, womit die üblichen Rechenregeln für Brüche Anwendung finden können. Wir schreiben deshalb auch

$$S = N^{-1}R := \left\{ \frac{z}{n} \ : \ z \in R,\ n \in N \right\}$$

und nennen den Ring S die *Lokalisierung* von R nach der multiplikativen Menge N.

Zunächst wollen wir das Erweiterungsideal $I' = I \cdot S$ genauer beschreiben. Dessen Elemente lassen sich in der Form

$$u = \sum_{f \in F} \frac{z_f}{n_f} f = \frac{\sum h_f f}{n}$$

mit $z_f, h_f \in R,\ n_f, n \in N$ darstellen. Da $\sum h_f f$ genau die Elemente aus I sind, erkennen wir, dass I' aus genau den Elementen von S besteht, die sich in der Form $\frac{z}{n}$ mit $z \in I, n \in N$ darstellen lassen. Entsprechend besteht das Kontraktionsideal $I' \cap R$ aus allen solchen Elementen, die außerdem noch polynomial auch in U sind, d. h. für die zusätzlich z vollständig durch n teilbar ist:

$$I' \cap R = \{r \in R : \ \exists\, n \in N\, (n \cdot r \in I)\}.$$

Verschiedene Ideale können dasselbe Nullstellengebilde haben, etwa gilt $V(f) = V(f^2)$ für ein $f \in R$[1]. Es existiert jedoch eine eineindeutige Korrespondenz zwischen Nullstellenmengen (über einem algebraisch abgeschlossenen Körper) und *Radikalidealen*, siehe dazu den Korrespondenzsatz in der Vorlesung *Gröbnerbasen und Anwendungen*.

Jedes solche Radikalideal I kann als Durchschnitt endlich vieler Primideale $I = \cap P_\alpha$ dargestellt werden. Eine bzgl. I unabhängige Variablenmenge U muss allerdings bzgl. einer dieser Primkomponenten, die ja größer sind als I, nicht mehr unabhängig sein. Wir erkennen das daran, ob $P_\alpha \cap N = \emptyset$ gilt (in diesem Fall bleibt die Variablenmenge unabhängig) oder nicht (in diesem Fall gibt es eine algebraische Beziehung $h_\alpha(U) \in P_\alpha$ zwischen den eigentlich unabhängigen Variablen U). Komponenten der ersten Art nennen wir *generisch*, Komponenten der zweiten Art *speziell*. Zerlegen wir das Nullstellengebilde der Voraussetzungen eines Geometrietheorems in seine Komponenten, so entsprechen letztere gewissen, durch $h_\alpha(U) = 0$ beschriebenen, degenerierten Situationen.

Satz 36

(1) Sei I ein Radikalideal und $I = \cap P_\alpha$ dessen Zerlegung in Primkomponenten. Dann gilt

$$I' = I \cdot S = \bigcap (P_\alpha \cdot S)\,.$$

[1] Ideale entsprechen in diesem Sinne Nullstellen „mit Vielfachheiten“.

(2)

$$P_\alpha \cdot S = \begin{cases} \text{ein Primideal } Q_\alpha \subset S \text{ mit } Q_\alpha \cap R = P_\alpha & \textit{wenn } P_\alpha \cap N = \emptyset \\ (1) & \textit{wenn } P_\alpha \cap N \neq \emptyset \end{cases}$$

Beweis: (1) Wegen $I \subset \cap P_\alpha$ folgt $I \cdot S \subset \cap (P_\alpha \cdot S)$. Sei umgekehrt $u \in \cap (P_\alpha \cdot S)$. Nach geeigneter Hauptnennerbildung hat es eine Darstellung der Form $u = \frac{z}{n}$ mit $z \in \cap P_\alpha = I$, $n \in N$, also $u \in I \cdot S$.

(2) Sei $0 \neq s \in P_\alpha \cap N$. Dann ist $1 = \frac{s}{s} \in P_\alpha \cdot S$.

Sei $P_\alpha \cap N = \emptyset$ und $Q_\alpha := P_\alpha \cdot S$.

Q_α ist ein Primideal: Aus

$$u_1 u_2 = \frac{p_1}{n_1}\frac{p_2}{n_2} = \frac{p}{n} \in Q_\alpha$$

mit $p_1, p_2 \in R$, $p \in P_\alpha$, $n, n_1, n_2 \in N$ folgt $n \cdot p_1 p_2 \in P_\alpha$ und wegen $n \notin P_\alpha$ und dessen Primidealeigenschaft schließlich $p_1 p_2 \in P_\alpha$.

$Q_\alpha \cap R = \{u \in R : \exists\, n \in N\, (n \cdot u \in P_\alpha)\}$. Wie eben folgt dann bereits $u \in P_\alpha$. $\square$

Folgerung 7

$$I' \cap R = \cap\{P_\alpha \ : \ P_\alpha \cap N = \emptyset\}$$

Ist insbesondere g ein Polynom in R, $F \subset R$ eine Menge von Polynomen und $G' =$ `GBasis`(F, Y), so gilt `NF`$(g, G', Y) = 0$ *genau dann, wenn g auf allen* generischen *Komponenten von $V_R(F)$ verschwindet.*

Wir sagen in diesem Fall, dass der geometrische Satz bzgl. der unabhängigen Variablenmenge U *generisch richtig ist*, was folgendes heißt:

Satz 37 (Generische Gültigkeit geometrischer Sätze vom Gleichungstyp)

Sei $[F(Y,U) \ \Rightarrow \ g(Y,U)]$ ein Satz vom Gleichungstyp, U eine für diesen Satz maximale unabhängige Teilmenge der Variablen und $S = k(U)[Y]$.

Sei weiter $G' =$ `GBasis`(F, Y) eine über S berechnete Gröbnerbasis.

Gilt `NF`$(g, G', Y) = 0$*, so ist $[F \ \Rightarrow \ g]$ auf allen generischen Komponenten von $V_R(F)$ richtig.*

Die Aussage $g = 0$ ist höchstens auf speziellen Komponenten von $V(F)$ falsch, auf denen aber die Variablen U nicht mehr unabhängig sind. Jede solche Komponente enthält im definierenden Ideal ein Polynom $h_\alpha(U)$. Das Produkt dieser Polynome können wir als Nichtdegenerationsbedingung nehmen.

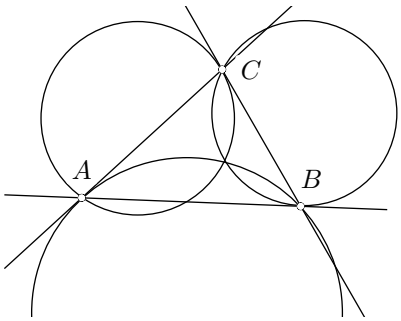
Ist umgekehrt $\mathtt{NF}(g, G', Y) \neq 0$, so verschwindet g auf einer der generischen Komponenten nicht, d. h. kann durch keine Nichtdegenerationsbedingung, die nur die unabhängigen Parameter enthält, gerettet werden.

Generell, wenn $\mathtt{NF}(g, G', Y)$ nicht verschwindet, wie oben im Beispiel *Miquelscher Punkt*, liefern uns Untersuchungen in S Aussagen, die das Verhalten auf speziellen Komponenten ausblenden. In diesem Beispiel hat $V(I)$ zwei generische Komponenten. Eine entspricht dem Schnittpunkt R der Kreise durch A, Q, R und B, P, R, die andere dem gesuchten Schnittpunkt S. Auf der ersteren Komponente gilt der Satz nicht, auf der zweiteren sehr wohl. Der Gröbnerfaktorisierer zeigt, dass $V(I)$ daneben noch eine weitere spezielle Komponenten hat, welche die Bedingung $c_1 = 0$ enthält und einer degenerierten Lagen entspricht. In einem solchen Fall ist aber der Kreis und damit auch der Schnittpunkt S nicht mehr eindeutig bestimmt, so dass wir auch nicht prüfen können, ob er auf dem dritten Kreis liegt.

5.3 Weitere Beispiele

Betrachten wir als weiteres Beispiel den **Satz vom Brocardschen Punkt**.

Satz 38 *Zum Dreieck ABC betrachten wir den Kreis durch A, der BC in C berührt, den Kreis durch B, der AC in A berührt und den Kreis durch C, der AB in B berührt. Diese drei Kreise gehen durch einen gemeinsamen Punkt, den* Brocardschen Punkt.



Wir beginnen mit einem speziellen Koordinatensystem, in dem die Eckpunkte A, B, C des Dreiecks sowie weitere freie Punkte M_1, M_2, M_3 und P gegeben sind, die durch die AGV als Mittelpunkte der drei Berührkreise c_1, c_2, c_3 sowie als Schnittpunkt von c_1 und c_2 spezifiziert werden.

```
A:Point(0,0); B:Point(1,0); C:Point(cx,cy); P:Point(px,py);
M1:Point(m1x,m1y); M2:Point(m2x,m2y); M3:Point(m3x,m3y);
c1:pc_circle(M1,A); c2:pc_circle(M2,B); c3:pc_circle(M3,C);
```

Die AGV $F = \{f_1, \ldots, f_8\}$ bestehen aus 8 Polynomen in 10 Variablen, wovon drei die Berührbedingungen und 5 die Lage der Punkte A, B, C, P auf den jeweiligen Kreisen kodieren.

```
polys1:[ is_cl_tangent(c1,pp_line(A,C)), is_cl_tangent(c2,pp_line(A,B)),
  is_cl_tangent(c3,pp_line(B,C)) ];

polys2:[ on_circle(B,c1), on_circle(C,c2), on_circle(A,c3),
  on_circle(P,c1), on_circle(P,c2) ];
```

Als Behauptung g ergibt sich

```
g: on_circle(P,c3);
```

$$-c_x^2 + 2\,m_{3x}\,c_x - c_y^2 + 2\,m_{3y}\,c_y + p_x^2 - 2\,m_{3x}\,p_x + -p_y^2 - 2\,m_{3y}\,p_y$$

Mit Blick auf die 8 Gleichungen setzen wir $U = (c_x, c_y)$ und die restlichen acht Variablen als abhängige Variablenmenge Y.

```
yvars:[px,py,m1y,m2y,m3y,m1x,m2x,m3x]; uvars:[cx,cy];
G:poly_reduced_grobner(polys,yvars);
```

Damit kommt Maxima nicht mehr zurecht. Leider würde g auch nicht zu null reduzieren, da wie im Satz vom Miquelschen Punkt neben der intendierten Lösung für P auch noch $P = B$ Nullstelle des Polynomsystems F ist.

Eine genauere Analyse zeigt außerdem, dass die drei Berührbedingungen $polys_1$ vollständige Quadrate sind, also unnötig komplizierte Polynome ergeben. Ist allgemein die Gerade $l = RS$ die Tangente an einen Kreis c mit dem Berührpunkt R, so hat auch der Berührpunkt R' der zweiten Tangente von S an den Kreis c rationale Koordinaten, so dass die quadratische Gleichung, aus welcher wir die Berührbedingung als deren Diskriminante gewonnen hatten, zwei *rationale* Lösungen hat. Damit muss die Diskriminante aber ein vollständiges Quadrat sein.

```
M:Point(0,0); R:Point(rx,ry); S:Point(sx,sy);
h:is_cl_tangent(pc_circle(M,R),pp_line(R,S));
factor(h);
```

$$4\,\left(r_x^2 - s_x\,r_x + r_y^2 - s_y\,r_y\right)^2$$

Wir können die drei Berührbedingungen durch die folgenden einfacheren Bedingungen ersetzen, dass der Berühr-Radius senkrecht auf der Tangenten steht.

```
polys1b:=[ is_orthogonal(pp_line(M1,A),pp_line(A,C)),
  is_orthogonal(pp_line(M2,B),pp_line(B,A)),
  is_orthogonal(pp_line(M3,C),pp_line(C,B)) ];
```

Die Gröbnerbasis dieses Systems

```
polys:append(polys1b, polys2);
G:poly_reduced_grobner(polys,yvars);
```

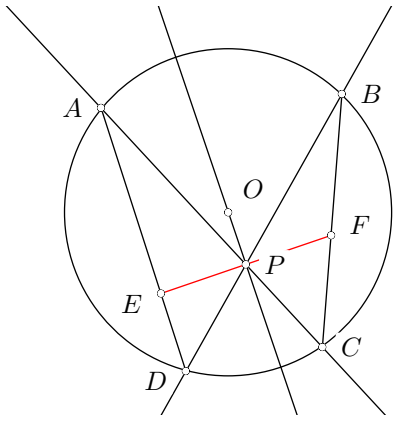
enthält ein Polynom $p_y^2 - q(U)\,p_y$, das sich wieder in zwei Linearfaktoren zerlegen lässt. Die folgende Rechnung zeigt, dass $V_S(G)$ damit wieder in zwei Komponenten zerfällt. Auf der einen, im Fall $P = B$, gilt der Satz nicht, auf der anderen ist er gültig.

```
sol:solve(G,yvars);
map(lambda([u],ratsimp(subst(u,g))),sol);
```

$$\left[c_x^2 - 2\,c_x + c_y^2 + 1, 0\right]$$

Aufgabe: Geben Sie einen konstruktiven Beweis dieses Satzes.

Betrachten wir zum Abschluss einen weiteren geometrischen Satz, den wir als Satz vom Gleichungstyp formulieren können.



Satz 39 (Schmetterlingstheorem) *Auf einer Kreislinie mit dem Mittelpunkt O seien vier Punkte A, B, C, D gegeben. P sei der Schnittpunkt von AC und BD. Die Senkrechte auf der Verbindungslinie OP schneide die Seiten AD und BC (die „Schmetterlingsflügel“) in den Punkten E und F. Dann gilt $|PE| = |PF|$.*

Zur Formulierung als geometrischer Satz vom Gleichungstyp sei O der Koordinatenursprung und $A = (1, 0)$ auf der x-Achse gelegen. $B = (b_x, b_y)$, $C = (c_x, c_y)$ und $D = (d_x, d_y)$ seien drei freie Punkte, die AGV bestimmen, dass sie auf dem Kreis c um O durch A liegen.

Dies ergibt drei Bestimmungsgleichungen, so dass wir $b_y, c_y, d_y \in Y$ und $b_x, c_x, d_x \in U$ wählen. $P = (p_x, p_y)$ ist der Schnittpunkt von AC und BD, was zwei Bestimmungsgleichungen für p_x, p_y ergibt, so dass wir $p_x, p_y \in Y$ wählen. Für die Punkte $F = (f_x, f_y)$ und $G = (g_x, g_y)$ ergeben sich ebenfalls je zwei Schnittbedingungen, so dass wir ebenfalls $f_x, f_y, g_x, g_y \in Y$ setzen.

Insgesamt erhalten wir folgende algebraische Formulierung

```
O:Point(0,0); A:Point(1,0); c:pc_circle(O,A);
B:Point(bx,by); C:Point(cx,cy); D:Point(dx,dy);
P:Point(px,py); E:Point(ex,ey); F:Point(fx,fy);
l:ortho_line(P,pp_line(O,P));

polys: [ on_circle(B,c), on_circle(C,c), on_circle(D,c),
  is_collinear(B,D,P), is_collinear(A,C,P),
  on_line(E,l), is_collinear(A,D,E),
  on_line(F,l), is_collinear(B,C,F) ];
g:sqrdist(P,E)-sqrdist(P,F);
```

Das Gleichungssystem hat eine recht überschaubare Struktur

$$\begin{aligned} polys = [\ & b_x^2 + b_y^2 - 1,\ c_x^2 + c_y^2 - 1,\ d_x^2 + d_y^2 - 1, \\ & b_x\, d_y - b_y\, d_x - b_x\, p_y + b_y\, p_x + d_x\, p_y - d_y\, p_x, \\ & c_y - p_y + c_x\, p_y - c_y\, p_x,\ p_x^2 - e_x\, p_x + p_y^2 - e_y\, p_y, \\ & d_y - e_y + d_x\, e_y - d_y\, e_x,\ p_x^2 - f_x\, p_x + p_y^2 - f_y\, p_y, \\ & b_x\, c_y - b_y\, c_x - b_x\, f_y + b_y\, f_x + c_x\, f_y - c_y\, f_x\ ] \end{aligned}$$

$$g = e_x^2 - 2\,p_x\,e_x + e_y^2 - 2\,p_y\,e_y - f_x^2 + 2\,p_x\,f_x - f_y^2 + 2\,p_y\,f_y$$

MAXIMAS `GBasis`-Algorithmus kommt damit aber nicht mehr zurecht.

```
yvars:[ex,ey,fx,fy,px,py,by,cy,dy]; uvars:[bx,cx,dx];
G:poly_reduced_grobner(polys,yvars);
```

Mit einer stärker konstruktiv formulierten AGK können einige der Polynome aus der AGV eliminiert werden. Wählen wir etwa E und F als Geradengleiter und P als Schnittpunkt der Geraden AC und BD, so benötigen wir nur 5 Gleichungen, um das Problem zu formulieren

```
O:Point(0,0); A:Point(1,0); c:pc_circle(O,A);
B:Point(bx,by); C:Point(cx,cy); D:Point(dx,dy);
E:varpoint(A,D,es); F:varpoint(B,C,fs);
P:intersection_point(pp_line(A,C), pp_line(B,D));
h:ortho_line(P,pp_line(O,P));

polys:[ on_line(E,h), on_line(F,h),
  on_circle(B,c), on_circle(C,c), on_circle(D,c) ];
g:num(ratsimp(sqrdist(P,E)-sqrdist(P,F)));

yvars:[es,fs,by,cy,dy]; uvars:[bx,cx,dx];
G:poly_reduced_grobner(polys,yvars);
```

Auch diese Rechnung geht mit MAXIMA leider nicht durch.

Mit einer noch stärker konstruktiv formulierten AGK können die AGV auf drei Bedingungen reduziert werden, die nach dem Hauptsyzygienkriterium bereits eine Gröbnerbasis bilden. Dies geht allerdings zu Lasten der Komplexität der zu beweisenden Schlussfolgerung g, die in distributiver Normalform ein Polynom vom Grad 22 mit 15 188 Summanden ist. Dennoch kann MAXIMA dessen Normalform berechnen und damit den Satz beweisen.

```
O:Point(0,0); A:Point(1,0); c:pc_circle(O,A);
B:Point(bx,by); C:Point(cx,cy); D:Point(dx,dy);
P:intersection_point(pp_line(A,C), pp_line(B,D));
h:ortho_line(P,pp_line(O,P));
```

```
E:intersection_point(pp_line(A,D),h);
F:intersection_point(pp_line(B,C),h);

polys:[ on_circle(B,c), on_circle(C,c), on_circle(D,c) ];
g:num(ratsimp(sqrdist(P,E)-sqrdist(P,F)));

yvars:[es,fs,by,cy,dy]; uvars:[bx,cx,dx];
G:poly_reduced_grobner(polys,yvars); poly_normal_form(g,G,yvars);
```

Aufgabe: Geben Sie ein konstruktives Beweisschema für diesen Satz an, indem Sie B, C, D als Kreisgleiter ansetzen, und untersuchen Sie, ob dieser Beweis durchgeht.