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 $\mathbb{A}^2$ dargestellt und Darstellungen anderer geometrischer Objekte daraus abgeleitet. Geraden können etwa 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. 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 $u \in \mathbb{R}$. 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 $u < 0$.

  Als *Teilverhältnis* bezeichnet man die Größe $TV(A, B; P) = \frac{u}{1-u}$.

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.
- 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. Letztere bezeichnen wir auch als *normierte Koordinaten* und schreiben $\overline{P}$. 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, was die früher beschriebene Dualität von Punkten und Geraden in Sätzen der projektiven Geometrie plausibel macht. Die Punkte, für deren homogene Koordinate $p_z = 0$ gilt, liegen auf der Ferngeraden, denn deren homogene Koordinaten lauteten ja gerade $(0 : 0 : 1)$.

Wir bezeichnen diese Erweiterung der affinen Ebene $\mathbb{A}^2$ um die Punkte der Ferngeraden als projektive Ebene $\mathbb{P}^2$. Bezeichnen wir dual dazu mit $\left(\mathbb{P}^2\right)^*$ den Raum der projektiven Geraden, so lassen sich die weiter oben untersuchten geometrischen Beziehungen nennerfrei durch Skalar-, Vektor- und Spatproduktoperationen im $\mathbb{R}^3$ beschreiben.

- 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 $\mathbb{R}^3$.

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

$$g = (a_yb_z - a_zb_y, a_zb_x - a_xb_z, a_xb_y - a_yb_x) = A \times B$$

- 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 $\mathbb{R}^3$ 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

$$h = P \times O_g = (-p_zg_2 : p_zg_1 : p_xg_2 - p_yg_1)$$

  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 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 zunächst 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.

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 $g = \{(p_x, p_y) : g_1p_x + g_2p_y + g_3 = 0\}$ als *Instanzen* dieser Klassen definieren. Die Koordinatenwerte für Punkte und Geraden sind dabei aus einem gemeinsamen Grundbereich K zu wählen, von dem wir voraussetzen, dass es sich um einen Körper handelt, also für diesen Datentyp die arithmetischen Operationen $+ \; - \; * \; /$ sowie ein boolesches Prädikat `boolean iszero()` definiert sind.

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 Objektzustands 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 $g.c$ also 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 $g \in$ `Line` ein Objekt vom Typ Gerade, so steht $g.\mathtt{c}$ für das Koordinatenattribut von g. Wie oben haben wir zu unterscheiden zwischen

- dem Attribut $g.\mathtt{c}$ selbst als informatischem Begriff,
- der Berechnungsvorschrift, nach welcher sich der Attributwert aus anderen Attributwerten berechnet,
- konkreten Attributwerten $g.\mathtt{c} \in C(L)$ oder $g.\mathtt{c}(t) \in C(L)$,
- und der Attributwertfunktion $g.\mathtt{c} \in F(C(L))$, welche die zeitliche Existenz von g in ihrer Gesamtheit beschreibt.

Aus dem Kontext der weiteren Ausführungen wird deutlich, in welchem Sinne $g.\mathtt{c}$ jeweils aufzufassen ist. $C(L)$ ist dabei der Bereich der (für Geraden zulässigen) Koordinatenwerte, $F(C(L))$ der zugehörige Funktionenraum. Beide hängen nicht von g selbst ab, sondern nur vom geometrischen Typ $g \in$ `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.

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 daherkommen. 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.

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

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

bestimmt, welches in unserem Fall die einfache Form $\texttt{pp\_line.DG}(c_1, c_2) = \texttt{isequal}(c_1, c_2)$ 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

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

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**

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

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

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

gilt.

Beispiele:

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

legt einen freien Punkt an. Die zugehörige Degenerationsbedingung für $m \in C(M)$ ist leer: $\texttt{freePoint.DG}(m) = \texttt{false}$. (Präziser: Die Degenerationsbedingung hängt vom gewählten Modell für Mausparameter ab)

`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 $\texttt{is\_parallel}(o_a, o_b)$ der zugehörigen geometrischen Objekte 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**.*

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 $\widetilde{w}_K.\texttt{DG}$ ergibt sich wie oben als oder-Verknüpfung der einzelnen $w_i.\texttt{DG}\left(u^{(i)}.\texttt{c}\right)$

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

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 $o.\texttt{c} = w.\texttt{c}(o_1.\texttt{c}, \ldots, o_n.\texttt{c})$. Dabei sind jedesmal dieselben Rechnungen $w.\texttt{c}$ mit je anderen Werten, also eine allgemeine Berechnungsvorschrift auszuführen.

Ist allgemeiner

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

eine Konstruktionsbeschreibung, so berechnen sich die Koordinatenwerte einer speziellen Realisierung der Endkonfiguration $O_E = \mathcal{K}(O_S)$ – Auführbarkeit vorausgesetzt – nach einer Berechnungsvorschrift, die sich als Zusammensetzung $\widetilde{w}_N.\texttt{c} \circ \cdots \circ \widetilde{w}_1.\texttt{c}$ 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 stattdessen eingeschränkte Klassen von Berechnungsvorschriften betrachten, auf die sich unsere Anwendungen bisher immer reduzieren ließen: Die Koordinaten von $o.\texttt{c}$ sollen sich durch arithmetische Operationen aus den Koordinaten von $o_1.\texttt{c}, \ldots, o_n.\texttt{c}$ berechnen lassen.

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);
```

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

Diese Berechnungsvorschrift kann durch eine entsprechende Termumformung weiter vereinfacht werden zu

```
ratsimp(ma);
```

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

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);
```

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

```
ratsimp(M);
```

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

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);
```

Alle Berechnungsvorschriften präsentieren sich als komplizierte rationale Ausdrücke in den Eingangsvariablen. Der letzte rationale Ausdruck vereinfacht 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 γ_a der Parameterobjekte enthalten.

Ersetzen wir diese Kerne durch Variablen $X = (x_a)$, so wird die Berechnungsvorschrift jeder einzelnen Koordinate durch einen rationalen Ausdruck $A(X) \in k(X)$ gegeben und wir können nach Vereinfachungsmöglichkeiten dieses Ausdrucks zu einem Ausdruck $A'(X)$ in $k(X)$ fragen. Die Berechnungsvorschrift 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 $\texttt{subst}(A, s)$ und $\texttt{subst}(A', s)$ berechnungsäquivalent sind, d. h. dasselbe berechnen. Im Rahmen eines CAS kann die Umwandlung eines rationalen Ausdrucks in eine Berechnungsvorschrift durch Ersetzen der Variablen durch Zahlenwerte mit dem Substitionsoperator 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) abbricht.

Wir werden die Möglichkeiten dieses Ansatzes zunächst an einigen Beispielen betrachten und uns später für die Beweiskraft der entsprechenden Rechnungen interessieren. Dazu benötigen wir zunächst Implementierungen der Berechnungsvorschriften der verschiedenen Konstruktionswerkzeuge in einem CAS, die in der Lage sind, diese Berechnungen auch auf symbolischen Eingabewerten auszuführen. Wir hatten gesehen hatten, dass sich bei diesen Berechnungen rationale Ausdrücke in $k(X)$ als Koordinatenwerte ergeben, die sich durch Termumformungen weiter vereinfachen lassen. 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`) effizient berechnen lassen, wollen wir weiter voraussetzen, dass in den Berechnungsfunktionen bei symoblischen Eingabewerten 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 $w.\texttt{c}$. 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 $w.\texttt{c}$ 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, $R = K[x_a : a \in I]$ der Polynomring über K in diesen Variablen und $Q(R)$ dessen Quotientenkörper. Ist $C = C(T)$ der Wertebereich eines geometrischen oder Parametertyps T über dem Grundbereich K, so bezeichnen wir mit $\overline{C} = C \otimes_K R$ den Wertebereich, den man durch Erweiterung $K \to R$ des Grundbereichs erhält.

Praktisch bedeutet dies, dass nicht nur Elemente aus K, sondern aus R an allen Stellen als Koordinaten erlaubt werden. Da die Berechnungsvorschriften nur arithmetische Operationen enthalten, sind sie auch auf solchen Eingaben – sofern es sich nicht um über $w.\texttt{DG}$ beschriebene degenerierte Lagen handelt – ausführbar und die Ergebnisse liegen in $\overline{C} \otimes_R Q(R)$.

Definition 7 *Als* universelle Realisierung eines Objekts *o vom Typ T bezeichnen wir jede Realisierung* $o.\texttt{u} \in \overline{C}$ *mit einem Satz „frischer“ Variablen, so dass sich für jede spezielle Realisierung von o die Koordinatendarstellung durch geeignete Variablenspezifikation* $s = \{x_a \to \gamma_a\}$ *mit* $\gamma_a \in K$ *ergibt:* $o.\texttt{c} = \texttt{subst}(o.\texttt{u}, s)$.

„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 über dem erweiterten Koeffizientenbereich R.

Anmerkung: Genauer muss gefordert werden, dass sich nicht nur spezielle Realisierungen von o mit Koordinaten aus K ableiten lassen, sondern auch Realisierungen, deren Koordinaten in Erweiterungen von K liegen. Der Polynomring R hat die universelle Eigenschaft, dass jede (endliche) Erweiterung von K als Spezialisierung der Erweiterung $K \to R$ hergestellt werden kann. Auf diese Feinheiten soll hier nicht eingegangen werden. 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. 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 Realisierungen mit Koordinaten aus K wertet diese Bedingung stets zu einem der booleschen Werte `true` oder `false` aus. Das ist für universelle Realisierungen 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 für alle Variablenbelegungen zu `true` auswertet. Wir wollen im Weiteren voraussetzen, dass $w.\mathtt{DGF}$ keine Tautologie ist, der Konstruktionsschritt also für wenigstens eine Belegung aus K ausführbar ist.

Wenn in der Berechnungsvorschrift für w nur arithmetische Operationen verwendet werden, erhalten wir für die Koordinaten $w.\mathtt{uc} = o.\mathtt{c}$ 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 Realisierung von O zu einer speziellen Realisierung von o führt, deren Koordinaten sich durch die Variablenspezifikation $s = \{x_a \to \gamma_a\}$ aus $w.\mathtt{uc}$ 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 $w.\mathtt{uc}$ *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 $o_{m+i}.\mathtt{uc}$ der Objekte $o_{m+i},\ i = 1, \ldots, N$.

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

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

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

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

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

Kurz, in der universellen Formel $w_2.\mathtt{uc}$ 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 voraussetzen wollen.

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

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

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

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

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 $\mathcal{K}$ ergibt sich

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

Wir setzen wieder voraus, dass dies keine Tautologie ist, da sonst $\mathcal{K}$ auf keiner Startkonfiguration ausführbar wäre.

Definition 9 *Eine Realisierung $B_E = (o_1.\mathtt{u}, \ldots, o_m.\mathtt{u}, o_{m+1}.\mathtt{uc}, \ldots, o_{m+N}.\mathtt{uc})$ 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 $o_i.\mathtt{uc}$ der Realisierungen der abgeleiteten Objekte als deren* universelle Formeln *und $\mathcal{K}.\mathtt{DGF}$ als* universelle Degenerationsbedingung *der Konstruktion.*

Bei diesen Betrachtungen ist genau zwischen rationalen Ausdrücken und rationalen Funktionen zu unterscheiden. Der Simplifikationsprozess rationaler Ausdrücke liefert äquivalente in $Q(R)$ 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.

Auf dem betrachteten symbolischen Niveau bedeutet die Ausführung der Konstruktion auf der universellen Startkonfiguration, dass in der universellen Formel

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

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 $M.\mathtt{uc}$ 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 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

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

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

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 Realisierung C_E der Endkonfiguration von $\mathcal{K}$, also eine solche mit $\mathcal{K}.\texttt{DG}(C_E) = \texttt{false}$ (Ausführbarkeit der Konstruktionsvorschrift), ist $\phi.\texttt{DG}(C_E) = \texttt{false}$ (Bestimmtheit der Schlussfolgerung) und $\phi.\texttt{c}(C_E) = \texttt{true}$:

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

(C) besteht aus zwei Teilen

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

und

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

wobei (C.1) in der Kontraposition auch als

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

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

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

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 effizient 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 $A_E = (o_1.\texttt{u}, \ldots, o_{m+N}.\texttt{u})$ 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 $A_S = (o_1.\texttt{u}, \ldots, o_m.\texttt{u})$ der unabhängigen Objekte vorkommen und die Variablen aus Y in den universellen Realisierungen $o_{m+1}.\texttt{u}, \ldots, o_{m+N}.\texttt{u}$ 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 Realisierung C_E der Endkonfiguration $\phi(C_E) = \texttt{true}$ gilt:

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

Sei – etwas detaillierter als im letzten Abschnitt –

- $A_i = (o_1.\mathtt{u}, \ldots, o_m.\mathtt{u}, o_{m+1}.\mathtt{u}, \ldots, o_{m+i}.\mathtt{u})$,
- $B_i = (o_1.\mathtt{u}, \ldots, o_m.\mathtt{u}, o_{m+1}.\mathtt{uc}, \ldots, o_{m+i}.\mathtt{uc})$ 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 Realisierung der Startkonfiguration 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 Realisierung 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 $i \geq 0$

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

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 $o_{m+i+1}.\mathtt{uc}(X) = \mathtt{subst}(w_{i+1}.\mathtt{uc}(X, Y), Y \to Y(X))$ durch Substitution $X \to X_0$ (rechte Seite). Diese Eigenschaft wird gerade durch die Kommutativität des folgenden Diagramms ausgedrückt

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

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 $w.\mathtt{uc}$ 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 $w.\mathtt{DGF}(X, Y)$ gilt also 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 erhalten wir daraus die Tautologie

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

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

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

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

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

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

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

In allen bisherigen Beispielen war $\phi.\texttt{c}$ 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 $\phi.\texttt{c} = \texttt{iszero}(\psi)$ mit

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

ein solcher atomarer Ausdruck ist. Setzen wir $\psi.\texttt{uc} = \psi(A_E)$, so ist $\phi.\texttt{uc} = \texttt{iszero}(\psi.\texttt{uc})$ die universelle Formel der geometrischen Eigenschaft ϕ. Wir können $\psi.\texttt{uc} \in R$ annehmen, da ein rationaler Ausdruck genau dann null wird, wenn das Zählerpolynom null wird. $\psi.\texttt{uc}$ 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 $\psi.\texttt{uc}(X, Y) = \psi(A_E)$ durch Spezifikation $Y \to Y(X)$. Eine weitere Simplifikation vereinfachte dann bereits diesen rationalen *Ausdruck* zu null, woraus folgt, dass $\texttt{iszero}(\psi(B_E))$ bereits als boolesche *Formel* unabhängig von Variablenbelegungen zu $\texttt{true}$ simplifiziert.

Von ähnlicher Struktur sind die Degenerationsbedingungen $w_i.\texttt{DGF}$. Mehrere und-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 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.`DGF` *als* `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 ϕ.`c` $=$ `iszero`(ψ) *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 Realisierung C_S der Startkonfiguration von $\mathcal{K}$ die Bedingung ϕ auf der zugehörigen Realisierung 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 $A_S(X) = (o_1.\mathtt{u}, \ldots, o_m.\mathtt{u})$ der Startkonfiguration,
- eine universelle Realisierung $A_E(X,Y) = (o_1.\mathtt{u}, \ldots, o_{m+N}.\mathtt{u})$ der Endkonfiguration,
- universelle Formeln $B_E(X) =$ `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 $\Psi'(X) = \psi(B_E) =$ `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 Realisierungen – die Voraussetzungen des Satzes sind widersprüchlich.

   Sei in diesem Fall i minimal mit der Eigenschaft, dass `subst`$(w_i$.`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.`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 Realisierungen 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 Realisierungen 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 1 *(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 Realisierungen der Startkonfiguration, wenn Ψ' als rationale Funktion zu null vereinfacht werden kann.

Zulässige spezielle Realisierungen C_S 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.