Reinhard Oldenburg

Bidirektionale Verknüpfung von Computeralgebra und dynamischer Geometrie

Zusammenfassung:

Das Lernen mit multiplen Repräsentationsformen kann von Lehr-Software unterstützt werden. Für die Verzahnung von Algebra und Geometrie wird das Konzept der bidirektionalen, dynamischen Verknüpfung von Computeralgebra und dynamischer Geometrie definiert. Die mathematischen, technischen und didaktischen Hintergründe werden dargestellt und anhand des Prototyps FeliX erläutert.

Abstract:

Educational software can support learning with multiple representations. Regarding the case of algebra and geometry the concept of a bi-directional link between dynamic geometry and computer algebra is introduced. Mathematical, didactial and technical foundations are considered and explained using the author's prototypical system FeliX.

1 Einleitung

Die Verwendung von multiplen Repräsentationsformen kann Lernen unterstützen (Edelmann 1996, S. 235ff; Anderson 1995, S. 103ff). In der Mathematikdidaktik wurde diese Erkenntnis intensiv angewendet. Die NCTM hat dem Thema der Repräsentationen das Yearbook 2001 gewidmet (Cuoco & Curcio 2001). Für die Bedeutung beim Problemlösen in Algebra und Geometrie siehe etwa (Zimmermann 2003, S. 11ff).

Auch in Hinblick auf den Computereinsatz ist das Thema bereits vielfach diskutiert (z.B. Tall 1991, Noss & Hoyles 1996). Bedauerlicherweise sind verschiedene Repräsentationsformen in mathematischer schulbezogener Software, wenn überhaupt vorhanden, in der Regel nicht gleichwertig. Beispielsweise können mit Computeralgebrasystemen (CAS) wie Derive Parabeln algebraisch und geometrisch behandelt werden, aber dabei ist die algebraische Seite ausgezeichnet, weil nur sie ein Operieren mit den Objekten erlaubt. Die graphische Darstellung dagegen ist nur eine abhängige Darstellung, mit ihr kann nicht interagiert werden. Bei konventionellen dynamischen Geometriesystemen (DGS) liegen die Verhältnisse umgekehrt: Primär ist die geometrische Darstellung, während die Termboxen rein abhängig sind.

In dieser Arbeit soll begründet werden, dass

- eine gleichberechtigte bidirektionale Integration von CAS und DGS wünschenswert ist
- eine solche Integration realisierbar ist
- sich durch diese Integration wertvolle didaktische Möglichkeiten eröffnen

Die gegenwärtige kommunikationslose Koexistenz von CAS und DGS hat schon mehrere Autoren herausgefordert. Schon bei (Schumann 1991) findet man erste Überlegungen zu einer Verbindung und der Wunsch nach einer algebraischen Bestimmung von Ortskurven. Erneuert wurde die Forderung nach einer Integration der verschiedenen "modes of solution" in (Schumann & Green 2000). Auf der ICTME5 (Borovcnik & Kautschisch 2002) fand ein Symposium zum Thema "Cooperation von CAS und DGS" statt. Dort hat Roanes eine Interfacesoftware vorgestellt, die es erlaubt, Sketchpad-Konstruktionen in Maple zu laden. Dies ermöglicht algebraische Berechungen rund um die Konstruktion, wie sie z.B. nötig sind, wenn die Gleichungen von Ortslinien gefunden werden sollen. Auch einige DGS haben die Idee aufgegriffen und sprechen von einer Integration von CAS und DGS (Geogebra (Hohenwater 2004), Geonext (Ehmann 2004), Lugares (Botana&Valcarce 2004)). Eine angemessen Diskussion dieser Systeme liegt jenseits des Rahmens dieses Aufsatzes, es wird jedoch auch so klar werden, dass keines der Systeme die unten genannten Funktionalitäten zur Verfügung stellt.

Hier sollen noch einige weitere Zitate belegen, dass der Wunsch nach Integration von DGS und CAS nicht nur beim Autor besteht.

> "It would be desirable to equip CAS with DGS elements....A two-window-technology – one for the algebraic side and the other for the geometric side – would be desirable." (Halverscheid 2004)
>
> "Already the promise of parallel processing may bring new possibilities, for instance in the simultaneous processing of several different representations." (Tall 1991, p. 245)

Interessant ist darüber hinaus, dass schon vor langer Zeit (Hölz 1994) der Wunsch von Schülern dokumentiert wurde, konstruierte Punkte noch weiteren Bedingungen zu unterwerfen – ein Wunsch, der zwar üblichem geometrischen Denken widerspricht, aber algebraisch sinnvoll ist.

Die bisher gegebenen Argumente für die Integration stützen sich auf allgemein-didaktische Überlegungen. Aber auch Beobachtungen aus dem eigenen Unterricht des Autors zeigen, dass Schüler den Mangel der bisherigen Softwaresituation klar erkennen. In zwei parallel unterrichteten neunten Klassen wurde die Parabel als Ortslinie eingeführt und zunächst auf Papier, dann mit dem DGS Euklid erkundet. Danach wurden Parabeln auch algebraisch beschrieben und via implizites Plotten in Derive gezeichnet. In Derive, nicht aber in DGS, wurde auch der Schritt zu anderen Kegelschnitten gemacht. In einem abschließenden Fragebogen zur Evaluation wurden die Schüler gefragt: "Welches Programm ist besser geeignet sich die Abstandsdefinition der Parabel zu veranschaulichen?". Alle Schüler entschieden sich für Euklid, weil, so eine typische Begründung, "man noch mit der Maus daran ziehen kann". Dabei haben die Schüler durchaus die Defizite im Vergleich zu CAS erkannt: "Die Parabelerzeugung bei Euklid gefällt mir besser, man kann allerdings Gleichungen nur schwer oder gar nicht ausrechnen."

Ein Blick in die didaktische Literatur zeigt einen großen Bereich von Themen, die sich sowohl mit CAS als auch mit DGS behandeln lassen: Kegelschnitte, Ortslinien, Bezierkurven, das Linsengesetzt, Ausgleichsgeraden, Gelenkmechanismen u.v.a.m. Die Bearbeitung solcher Fragestellung erfordert also die Wahl eines Werkzeugs, entweder durch den Lehrenden oder den Lernenden.

In der Regel verfügt der Lernende nicht über die nötige Kompetenz, aus einer gegebenen Problemstellung heraus sich für das richtige Werkzeug zu entscheiden. Diese

Kompetenz zu entwickeln ist zwar ein wünschenswertes Ziel, das aber nach meiner Erfahrung erst in der Sekundarstufe II in nennenswertem Umfang erreichbar ist und auch dort nur, wo die Problemstellung und mögliche Lösungen von den Schülern von vorneherein überblickt werden. Ein Problem beim Erwerb dieser Kompetenz ist die Fehler-Intoleranz der gegenwärtigen Situation: Wenn sich herausstellt, dass man zunächst ein ungünstiges Werkzeug gewählt hat, verliert die ganze bisherige Arbeit ihren Wert und man muss im anderen Werkzeug von vorne beginnen. Eine Integration von CAS und DGS könnte hier Abhilfe schaffen.

In dem der Lehrer für die Schüler das Werkzeug wählt, nimmt er ihnen nicht nur eine kognitiv anspruchsvolle Entscheidung weg, sondern legt auch fest, wie künftig mit den zu untersuchenden Objekten operiert werden kann. Eine Parabel in einem DGS unterstützt andere Operationen (z.B. am Brennpunkt ziehen, eine Tangente anlegen) als eine Parabel im CAS (z.B. Faktorisieren, Scheitelpunkt bestimmen...). Nach der Reifikationstheorie wie sie von Sfard gründlich diskutiert wurde (Sfard 1991) entstehen mathematische Objekte aus Prozessen mathematischen Handelns. Wenn also die Operationsmöglichkeiten je nach Lernumgebung variieren, müssen auch die von den Schülern geleisteten Konstruktionen differieren.

Es ist offensichtlich, dass ein Begriff umso flexibler und angemessener verwendet werden kann, je stärker seine Vernetzung mit anderen Begriffen ist, je reichhaltiger also die mit ihm verbundenen Operationsmöglichkeiten sind. Damit schließt sich der Bogen hin zum Wert der multiplen Repräsentationsformen. Es ist denkökonomisch, Sachverhalte, die invariant unter einem Wechsel sind, durch eine einheitliche kognitive Entität zu repräsentieren. Wenn die Schüler also die algebraische und die geometrische Repräsentationsform z.B. der Abstandseigenschaft der Parabel operierend erleben können, besteht die berechtigte Hoffnung auf ein horizontal vernetzendes Lernen.

1.1 Das postulierte Ziel

Ziel der hier skizzierten Entwicklung soll ein System sein, dass DGS und CAS bidirektional verknüpft in dem Sinne, dass man in einem Geometriefenster (das insbesondere einen Zugmodus bietet) und einem Algebrafenster (mit CAS-Funktionalität) parallel arbeiten kann, und Aktionen in einem Fenster das jeweils andere Fenster entsprechend beeinflussen, so dass beide als Repräsentation der gleichen mathematischen Situation gesehen werden können, diese Bidirektionalität soll gegeben sein auf

- der Ebene der Lage von Objekten resp. der Werte von Variablen
- und auf der Ebene der geometrischen Beziehungen bzw. algebraischen[1] Gleichungen zwischen den Objekten resp.Variablen.

Ein System, das dieser Beschreibung genügt, soll ADGS genannt werden (Algebraisches Dynamisches Geometrie-System). Schlussfolgerungen aus diesen Postulaten:

- Fehlende Konstruktionsmöglichkeiten (z.B. Kreisspiegelung, Objekte beliebig binden, Ortskurven als Objekte verwenden) können immer ausgeglichen werden, indem man die entsprechenden Gleichungen formuliert.

[1] Hierbei wird algebraisch im weiten Sinne verstanden, umfasst also z.B. auch transzendente Gleichungen.

- ADGS verfügen automatisch über diskrete Schalter, Gitterfang u.ä. weil sich das algebraisch formulieren lässt. Wenn Ungleichungen erlaubt sind, können auch Boolesche Punkte simuliert werden.
- In ADGS können halbfreie Punkte nicht nur explizit als Gleiter auftreten, sondern die Einschränkung der Bewegungsfreiheit kann beliebig tief verschachtelte Ursachen haben.
- ADGS können anders als herkömmliche DGS keinen Abhängigkeitsbaum zu Grunde legen. Auf der algebraischen Seite lassen sich selbstverständlich zirkulare Bedingzungen formulieren, beispielsweise eine geschlossene Kette von fünf Strecken fester Länge.
- In ADGS kann man an Punkten ziehen, die in herkömmlichen Systemen abhängig und damit nicht ziehbar wären. Das ergibt sich aus der vorhergehenden Beobachtung.
- In ADGS kann man an allen Objekten mit der Maus ziehen. Wenn das graphische user interface z.B. nur Maus-Reaktionen für Punkte vorsieht, setzt man einfach Punkte auf das zu ziehende Objekt und zieht an diesen Punkten.
- ADGS können nicht immer stetig sein, da algebraisch sich Gleichungen mit unstetigen Lösungen formulieren lassen.
- Ortskurven können in ADGS gut untersucht werden, weil die Infrastruktur eines CAS zur Verfügung steht.

Es ist wichtig zu verstehen, dass diese Eigenschaften aus dem obigen Postulat folgen. Ein System, das auch nur eine der Eigenschaften nicht erfüllt, kann dem Postulat nicht genügen. Es ist deshalb auch nahezu aussichtslos, ein herkömmliches DGS zu einem vollen ADGS aufbohren zu wollen.

1.2 Zum Primat der Algebra bei Computernutzung

Die reine Geometrie des abstrakten Raumes ist eine intellektuelle Konstruktion. Reale Geometrie muss immer repräsentiert sein, sei es im kognitiven Netz eines Menschen oder im Computer. Um letzteres zu erreichen, gibt es im wesentlichen zwei Zugänge:
Geometrisch-Symbolisch: Geometrische Objekte werden dabei durch abstrakte Symbole dargestellt. Für die lineare Geometrie wurde das in (Wang 1998, p. 296ff) ausgeführt. Dort gibt man sich eine Menge von Punkten vor und rechnet mit den Verbindungsvektoren rein symbolisch. Diese Operationen unterliegen dann algebraischen Regeln, die symbolisch manipuliert werden können. Dieser Ansatz leistet aber nicht, was wir wollen, denn die Objekte bleiben notwendig rein symbolisch. Um geometrische Objekte aber auf den Bildschirm eines PC zu bringen und mit der Maus manipulierbar zu machen, braucht man konkrete Koordinaten. Diese könnten natürlich parallel hinzugefügt werden, aber die beiden Repräsentationen wären dann doch disjunkt. Deshalb ziehen wir die Alternative vor:
Koordinatenzentriert: Objekte werden über ihre Koordinaten beschrieben. Die aktuellen Werte ihrer Koordinatenvariablen erlauben sofort, die Konstruktion am Bildschirm darzustellen. Dieser Ansatz lieget daher allen verbreiteten CAD- und DGS-Programmen zugrunde. Ein entscheidender Unterscheid zu diesen rein numerischen Systemen ist aber, dass in einem ADGS Variablen sowohl symbolisch verwendet werden, als auch stets ei-

nen aktuellen numerischen Wert besitzen. Diese explizite Kombination verschiedener Aspekte des Variablenbegriffs ist auch in herkömmlichen CAS nicht vorhanden (er kann dort allerdings simuliert werden).
Man beachte, dass beide Ansätze im wesentlichen stark algebraischer Natur sind. Dies ist kein Zufall, schließlich sind abstrakte Computer (bis auf die Endlichkeit des Speichers) äquivalent zu Turingmaschinen und diese wiederum zu dem algebraisch beschreibbaren Lambda-Kalkül. Aus dieser theoretischen Sicht ist klar, dass alle Mathematik, die in einem Computer repräsentiert werden soll, algebraisiert werden muss. Mit Blick auf die Geometrie spreche ich deshalb vom Primat der Algebra über die Geometrie in der Computernutzung (ähnliches gilt auch für die Stochastik, Topologie etc.). Die Kehrseite der Medaille ist, dass jeder, der mathematische Sachverhalte (nicht nur geometrische) am Computer modellieren will, algebraische Kompetenzen braucht. ADGS bieten Anlass zur Hoffnung, die algebraischen Kompetenzen zu stärken.

1.3 Algebra in herkömmlichen DGS

Konventionelle DGS (Euklid (Mechling 2004) oder Cinderella (Kortenkamp & Richter-Gebert 2004)) verwenden eine algebraische Modellierung der (Koordinaten-) Geometrie. Auch in den Teilen eines DGS, die sich der reinen synthetischen Geometrie verschrieben haben, scheinen diese algebraische Strukturen gelegentlich durch die geometrische Oberfläche, z.B. dann, wenn das Programm im Zugmodus Entscheidungen zur Auswahl eines Schnittpunktes unter mehreren Möglichkeiten treffen muss.
Aber moderne DGS bieten Funktionen an, die über die synthetische Geometrie hinaus gehen. Dies sind:
Berechnete Punkte: Damit kann man in Verbindung mit der Ortslinienfunktion z.B dynamische Funktionsgraphen zeichnen lassen. Wenn man in der Lage ist, die nötigen algebraischen Berechungen (von Hand oder mit einem CAS) durchzuführen, kann man auch Punkte auf Ortslinien setzen oder die Schnittpunkte von Ortslinien mit anderen Objekten erzeugen – Fähigkeiten, die den herkömmlichen DGS abgehen.
Strecken fester Länge: Diese Umsetzung der euklidischen Abstandsgleichung ist eine interessante Fähigkeit, denn sie bietet gegenüber Konstruktionen mit Hilfskreisen erweiterte Operationsmöglichkeiten, weil an beiden Endpunkten gezogen werden darf. Der Versuch, ein Gelenkviereck damit zu modellieren scheitert aber z.B. in Euklid, weil die Objekte zu eng verwandt wären.
Termfenster und Anzeige von Gleichungen: In Termfenstern können numerische Terme berechnet werden, ohne CAS im Hintergrund dürfen dabei aber keine symbolischen Variablen verwendet werden. Ähnliches gilt für die Anzeige der Gleichungen, bei denen in der Regel nur die aktuellen numerischen Koeffizienten der Gleichung angezeigt werden.
Ein ADGS sollte mit all diesen Beschränkungen aufräumen.

1.4 Programmierbarkeit

Traditionell besitzen DGS mit dem Makrokonzept eine eingeschränkte, aber leicht erlernbare Programmierbarkeit. Lediglich einige moderne DGS gehen darüber hinaus. DrGeo (Fernandes 2004) kann Scheme-Programme ausführen und Cinderella (Kortenkamp & Richter-Gebert 2004) wird in Zukunft Java-Methoden aufrufen können.

Ich bin nicht der Auffassung, dass die Programmerstellung für ein DGS zum regulären Inhalt des Mathematikunterrichts werden sollte, aber alle Erfahrung (von Emacs bis Office) lehrt, dass dort wo Programmiermöglichkeiten geboten werden, sich mehr nützliche Anwendungen finden als ursprünglich gedacht. Der ambitionierte Lehrer sollte die Möglichkeit haben, aus dem Werkzeug eine maßgeschneiderte Lernumgebung zu machen.

Beispiele für sinnvolle Anwendungen kleiner Steuerprogramme sind etwa Animationen oder die dynamische Ausführung von Algorithmen, die Daten der Konstruktion verwenden. So kann man zu einer Menge von Punkten eine Lösung des Traveling-Salesman-Problems oder einen minimalen aufspannenden Baum anzeigen lassen. Dies zeigt, dass auch der Informatikunterricht ein programmierbares DGS gut einsetzten kann.

1.5 Vision und Realität

Ziel dieses Aufsatzes ist die Konkretisierung der Vision einer Integration von CAS und DGS. Die grundlegenden Konzepte werden benannt, ihre mathematische Struktur wird geklärt und es werden didaktisch relevante Konsequenzen gezogen. In seinen zentralen Teilen spricht dieser Aufsatz deshalb nicht von einem bestimmten Programm. Im Hintergrund steht aber immer das System FeliX des Autos, das als gegenwärtig einzige Implementation eines ADGS die Realisierbarkeit der Konzepte beweist.

2 ADGS – von innen gesehen

In diesem Kapitel werden die verschiedenen Konzepte eines ADGS zusammengetragen und verbunden.

2.1 Objektklassen

Nach dem Primat der Algebra über die Geometrie, das durch die Verwendung des Computers bedingt ist, muss jedes Objekt algebraisch charakterisiert werden. In diesem Abschnitt geben wir eine (für praktische Systeme unvollständige) Liste der geometrischen Objektklassen und erläutern wie sie repräsentiert werden können.

Punkt: Wie Punkte repräsentiert werden sollten, hängt von der zu realisierenden Geometrie ab. Wir beschränken uns hier auf den Fall der Euklidischen Geometrie, so dass ein Punkt P durch seine reellwertigen Koordinatenvariablen $co(P)=x_P,y_P$ beschrieben wird. Der Grund für diese Wahl liegt vor allem darin, dass diese Form auch diejenige ist, die die Schüler kennen und verwenden. Für die Eingabe algebraischer Koordinaten durch Schüler kommen nur solche Koordinaten in Frage, bei der internen Verwendung einer anderen Darstellung müsste also ständig konvertiert werden. Damit wahrt man auch die Möglichkeit, den Schülern transparent zu machen, wie das System arbeitet.

Gerade: Geraden besitzen zwei Freiheitsgrade, eine gute Darstellung wäre daher z.B. mit den Koordinaten c und φ in $\cos(\varphi)x+\sin(\varphi)y=c$. Nachteilig daran ist vor allem, dass dies ein Verständnis die trigonometrischen Funktionen voraussetzt und dass algebraische Rechnungen wie z.B. die Elimination von Variablen erschwert werden. Deshalb erscheint es sinnvoll, die algebraische Beschreibung durch die homogenen Koordinaten a,b,c in $ax+by=c$ vorzunehmen.

Der zusätzliche Freiheitsgrad durch diese homogene Gleichung ist für einige Algorithmen ein Hindernis. Deshalb sollte die Gleichung $a^2+b^2=1$ als Eichfixierung (wie die Physiker sagen) in einer eigenen Liste *G* gespeichert werden.
Kreis: Radius und Mittelpunktkoordinaten sind hier die natürlichen Variablen.
Polygon: Eine Liste von Punkten oder x-y-Koordinatenpaaren erscheinen als natürliche Wahl.
Funktionsgraph, Parameterkurve, Implizite Kurve: Ob diese Objekttypen unterschieden werden ist allein eine Frage der Bequemlichkeit – für den Implementator wie den Nutzer. Wichtig scheint mir hier, dass das System die Klasse der darstellbaren Graphen/Kurven nicht künstlich einschränkt (etwa auf implizite Plots quadratischer Gleichungen (Kegelschnitte)), damit der Spiel- und Erkundungstrieb durch schöne Graphen belohnt werden kann.

Weitere Objektklassen sind denkbar (und können in FeliX vom kundigen Nutzer sogar selbst hinzugefügt werden), aber nicht dringend nötig. Man beachte, dass sich z.B. Kegelschnitte sehr gut als implizite Kurven behandeln lassen. Unter diesem Gesichtspunkt könnte man sogar auf Geraden und Kreise verzichten. Dies nicht zu tun, hat allein praktische Gründe: So können diese Objekte wesentlich schneller gezeichnet werden.

2.2 Konstruktion

Eine Konstruktion besteht aus einer Menge *O* von Objekten, wobei jedes einzelne Objekt eindeutig zu einer Objektklasse gehört. Die Menge der Variablen *V* der Konstruktion ist die Vereinigung aller Variablen der Objekten mit evtl. weiteren, "unsichtbaren" Variablen.
Darüber hinaus muss eine Konstruktion die Beziehung der Objekte untereinander beinhalten. Auf geometrischer Ebene kann das durch Relationen wie *ist_Mittelpunkt_von*, *liegt_auf*, *ist_senkrecht_zu* u.s.w. beschrieben werden. Außerdem kann man die Konstruktionsreihenfolge als Abhängigkeitsgraph kodieren und die Mathematica-Version von FeliX tut dies auch (nicht aber die MuPAD-Version). Es ist aber eine wichtige Einsicht, dass diese Informationen nur von sekundärer Relevanz sein können, da dem Benutzer die Freiheit eingeräumt werden soll, beliebige algebraische Relationen zu spezifizieren. Die zentrale Kodierung der Relationen erfolgt deshalb durch Gleichungen *E*, durch Ungleichungen *I* und durch Hilfsgleichungen *G*.

Die Hilfsgleichungen sind dabei solche, deren Gültigkeit nicht zwingend ist, die aber zu einer Normierung beitragen. Beispielsweise wird für jede Gerade die Gleichung $a^2 + b^2 = 1$ hinterlegt, um die Freiheit der homogenen Koordinaten einzuschränken. Einige numerische Routinen laufen effizienter und vor allem robuster, wenn man solche Hilfsgleichungen hinzunimmt.

Die Konstruktion sollte in Hinblick auf den Zugmodus auch speichern, welche Objekte wie bewegt werden können. Bei einer konventionellen DGS ergibt sich das automatisch aus dem Abhängigkeitsgraph. Nur die Basisobjekte und Gleiter können mit der Maus bewegt werden, die ersten uneingeschränkt, die zweiten nur eingeschränkt.

Eine solche Unterscheidung ist in einem algebraischen Setup nicht mehr statisch. Weil eine Gleichung nicht vorschreibt, wie sie aufgelöst werden soll, hat der Benutzer – und u.U. das System – hier viel Freiheit. Wie teilen deshalb die Objekte in Basisobjekte

B und Nichtbasisobjekte ein und sprechen auch von Basisvariablen und Nichtbasisvariablen.

Prinzipiell können alle Objekte mit der Maus bewegt werden, Basisobjekte haben aber ein hohes Beharrungsvermögen: Ein Basisobjekt wird nur bewegt, wenn es selbst gezogen wird, aber nicht als Folge der Bewegung eines anderen Objektes.

2.3 Konstruktionsoperationen

Hier wird erläutert, wie eine Konstruktion modifiziert werden kann, um eine neue Konstruktion zu erhalten. Die Operationen, die dies leisten, können sowohl über das GUI als auch über Befehle der CAS-Programmiersprache gegeben werden. Eine Typisierung:
Rein additive Operationen: Dies sind Operationen, die ein neues Objekt aufnehmen, ohne eine Beziehung zu den Objekten und Relationen der alten Konstruktion herzustellen.
Rein relationale Operationen: Diese lassen die Objektmenge unverändert, führen aber neue Relationen ein. Diese können Variablen eines einzigen Objekts oder auch mehrerer Objekte umfassen. Die Spezifikation der Operation kann symbolisch-geometrisch (z.B. Punk liegt auf Objekt, zwei Geraden sind orthogonal, etc....) oder algebraisch sein, und beide Spezifikationsmöglichkeiten müssen äquivalent sein.
Additiv-relationale Operationen: Sie kombinieren beide zuvor behandelten Typen. Es wird also ein neues Objekt erzeugt und seine Relationen mit den bisherigen Objekten wird festgelegt. Im Falle symbolisch-geometrischer Operationen entspricht das den Konstruktionsbefehlen für abhängige Elemente in konventionellen DGS. Insbesondere enthält diese Klasse von Operationen auch diejenigen, die Ortskurven erzeugen.
Löschen: Von Relationen oder auch von Objekten (und dann von allen Relationen, die dieses Objekt betreffen).
Konvertierung: Der Datentyp eines Objektes wird geändert. Beispielsweise aus einer parametrischen Kurve in eine implizite oder von einem Kreis in eine parametrische Kurve.

2.4 Konfiguration

Die Konfiguration einer Konstruktion gibt, geometrisch gesehen, an, wo die Objekte liegen, und algebraisch, welche Werte ihre Koordinaten haben.

Eine Konfiguration lässt sich z.B. durch eine Abbildung $i : V \to \mathrm{R}$ in die reellen Zahlen definieren, die jeder Variable ihren aktuellen Wert zuordnet. Diese Abbildung lässt sich durch Substitution auf beliebige Terme fortsetzen.

Variationsmöglichkeiten ergeben sich, wenn man die Wertemenge vergrößert, beispielsweise durch Hinzunahme eines Elementes für "nicht definiert" oder durch Verwendung komplexer Zahlen. Auf diese Variationsmöglichkeiten wird später noch genauer eingegangen.

2.5 Operationen

Ziehen
Wenn der Benutzer versucht, ein Objekt zu verziehen, bedeutet das, dass er die Werte der Variablen ändern will. Es ist nicht von vorneherein klar, ob das erfolgen kann, ohne die geltenden Relationen zu verletzen. Dies ist schon von den Gleitern in konventionel-

len DGS her bekannt. Um die gewünschten Werte für die Zugvariablen (die Variablen des Zugobjekts) zu erreichen oder ihnen zumindest nahe zu kommen, dürfen auch alle anderen Variablen bis auf die Basisvariablen (der Basisobjekte) verändert werden – aber nur so, dass die Relationen weiterhin gelten.
Unter diesen Bedingungen gilt einer der drei folgende Fälle für das Zugobjekt:

a) Das Objekt kann tatsächlich an den gewünschten Ort geschoben werden.
b) Das Objekt ist nur eingeschränkt beweglich und das System muss nach Maßgabe der Beweglichkeit eine Konfiguration auswählen, die dem Zugziel möglichst gut entspricht.
c) Das Objekt ist völlig unbeweglich, die Zuganforderung scheitert. Im einfachsten Fall tritt das ein, wenn E Gleichungen enthält, die die Koordinaten explizit festlegen.

Im Fall b) ist die Reaktion des Systems durch die Konstruktion in der Regel nicht eindeutig determiniert. In den Fällen a) und b) müssen möglicherweise auch andere, Nichtbasisvariablen geändert werden, um den Relationen genüge zu tun. Hier gibt es u.U. eine willkürlich Freiheit. Nach der Entscheidung, welche Variablen geändert werden, kann es immer noch sein, dass die Relationen die neuen Werte nicht eindeutig festlegen, sondern entweder

a) eine endlich Zahl von Lösungen besitzen. Ein Spezialfall hiervon ist z.B. die Frage der Auswahl von Schnittpunkten bei Kreisen und Kegelschnitten (Kortenkamp 1999).
b) eine unendliche Zahl von Lösungen besitzen.

Gewalt-Ziehen

Gleichungen sind wie Gesetze. Es kann den Wunsch geben, sie zu übertreten (Beispiel s.u.). Für den Zugmodus bedeutet dies, dass man die Ursachen, die einen gewöhnlichen Zugversuch scheitern lassen, schlicht ignoriert. Dies kann geschehen, indem man akzeptiert, dass einige Gleichungen aus E nicht erfüllt sind und/oder, dass auch Basispunkte verschoben werden. Diese Freiheiten müssen durch das System oder den Benutzer fest gelegt werden.

Setzen

Für das explizite Setzen von Objektkoordinaten eines Objektes mit einem Kommando der Algebra-Konsole gilt sinngemäß das gleiche wie für den Zugmodus. Allerdings kann man dabei nicht mehr davon ausgehen, dass die neue Konfiguration in der Nähe (in der metrischen Topologie des Konfigurationsraums) der alten liegt, so dass hier z.B. die Bedeutung eines stetigen Verhaltens eine gänzlich andere ist als im Zugmodus. Wenn die Koordinaten von Nicht-Basis-Punkten gesetzt werden, kann die Konfiguration inkonsistent werden (s.u).

Gleichungen hinzufügen

Das interaktive Hinzufügen von Gleichungen hat in herkömmlichen DGS keine Entsprechung. Einzig die Operation "Binden an Objekt" ist ein einfacher Spezialfall. Er entspricht dem hinzufügen der Gleichung zu E, die besagt, dass ein Punkt P auf dem Objekt liegt. Dem konventionellen DGS-Kommando "Punkt P an Kreis K binden" entspricht die

Hinzunahme der Gleichung $(x_P - x_K)^2 + (y_P - y_K)^2 = r_K{}^2$. Die Freiheit hier beliebige Gleichungen hinzuzufügen, erweitert das Problem erheblich: Eine konsistente Konfiguration kann nach dem Hinzufügen einer Gleichung inkonsistent sein.

Relaxation

Wir haben nunmehr zwei Fälle kennen gelernt, wie inkonsistente Konfigurationen entstehen können. Dies macht ein Werkzeug (Relaxation) notwendig, das versucht, eine inkonsistente Konfiguration konsistent zu machen. Natürlich brauchen die Gleichungen *E* keine Lösung zu besitzen, so dass dieser Versuch nicht in jedem Fall gelingen kann.

Im Gegensatz zum Zugmodus gibt es bei der Relaxation kein Zugziel. Bedingung ist lediglich, dass Basisvariablen nicht verändert werden dürfen. Die restlichen Variablen sollten natürlich so wenig wie möglich verändert werden, damit der Nutzer die neue Konfiguration als sinnvolle Modifikation seiner alten Konfiguration erkennt.

3 Realisierung als Optimierungsproblem

Die bisherige Beschreibung zeigt die Komplexität von ADGS. Die Realisierung wäre sicher aussichtslos, wenn man nicht die Leistungsfähigkeit des algebraischen Kalküls ausnutzen könnte. Diese aber ermöglicht verhältnismäßig einfache Lösungen. In diesem Abschnitt wir die Theorie der nichtlinearen Optimierung verwendet.

3.1 Die Optimierungsformulierung

Die Idee, Optimierungsalgorithmen einzusetzen, stammt ursprünglich aus der Beobachtung, dass nicht jeder Zug mit der Maus erfolgreich sein kann. Typisches klassisches Beispiel sind die halbfreien Punkte. Eine naheliegende Forderung ist, dass ein solcher Punkt dem mit der Maus angegeben Zugziel so nahe wie möglich kommt, ohne natürlich seine Bindung an einen Träger aufzugeben.

Dazu wird die Lösung eines reellen Gleichungssystems ersetzt durch die Minimierung der Fehlerquadratsumme. Zum Gleichungssystem $E = \{g_1 = 0, ..., g_m = 0\}$ wird die Zielfunktion $f = \sum_{i=1}^{m} g_i{}^2$ gebildet und diese minimiert.

Bei allen Zug- und Relatationsprozessen sollen, wie oben beschrieben, die Werte der Basisvariablen (mit evtl. Ausnahme des Zugpunktes) nicht geändert werden. Die Einsetzung ihrer aktuellen Werte in beliebige Funktionen liefert Funktionen, die nur noch von nicht-Basisvariablen (und den Variablen des Zugpunktes) abhängen, sie sollen durch ein Dach gekennzeichnet werden, z.B. $\hat{g}_i$.

Relaxation: Zuerst werden, wie eben beschrieben, die Werte der Basisvariablen in *f* eingesetzt, dann wird mit einem lokalen Optimierungsalgorithmus die resultierede Funktion $\hat{f}$ minimiert. Dabei dienen die bisherigen Variablenbelegungen als Startwert. Der Minimierungsalgorithmus ermittelt neue Werte für die Variablen, also die relaxierte Konfiguration.

Zugmodus bzw. Setzen: Diese Operationen versuchen, Variablen $x, y \in V$ auf neue Werte x_0, y_0 zu setzen. Ziel ist nun, (x, y) möglichst nahe an (x_0, y_0) heranzubringen, ohne die Gleichungen zu verletzen. Dies ist ein restringiertes Minimierungsproblem:

$$\min_{x \in F} (x - x_0)^2 + (y - y_0)^2 \text{ mit } F = \{x \mid \hat{g}_1(x) = 0, ..., \hat{g}_m(x) = 0\}$$

Für dieses restringierte Minimierungsproblem kann entweder ein spezialisierter lokaler Algorithmus verwendet werden, oder es kommt ein gewöhnlicher Optimierungsalgorithmus mit der Penalty-Term-Methode zum Einsatz. Diese besteht darin, die Funktion $(x - x_0)^2 + (y - y_0)^2 + \alpha \cdot \hat{f}$ zu minimieren, wobei α eine willkürliche große Zahl ist. Dabei muss α so groß gewählt werden, dass das Minimum nur in tolerabler Größe von der Lösungsmenge F von E abweicht. Es reicht dabei völlig aus, wenn die Genauigkeit so groß ist, dass sie nicht mehr als 1 Pixel in der Darstellung ausmacht.

Gewalt-Zugmodus: Man setzt (x, y) auf (x_0, y_0) und führt diese Werte auch in die Gleichungen ein. Dann substituiert man die Basisvariablen und minimiert $\hat{f}$.

Ungleichungen: Ein besonders attraktiver Aspekt des Optimierungsansatzes ist, dass bei Verwendung eines geeigneten Optimierungsalgorithmus auch Ungleichungen formuliert werden können, d.h. die zulässige Menge F kann zusätzlich durch Ungleichungen eingeschränkt werden:

$$F = \{x \mid \hat{g}_1(x) = 0, ..., \hat{g}_m(x) = 0, \hat{h}_1(x) \geq 0, ..., \hat{h}_k(x) \geq 0\}$$

Neben der expliziten Anwendung können solche Ungleichungen z.B. nützlich sein, um einen von mehren Schnittpunkten nichtlinearer Objekte auszuwählen.

3.2 Konsequenzen

Die bisherige Beschreibung lässt noch etliche Details der Implementierung offen. Dennoch können schon auf diesem Niveau der Präzisierung wichtige Schlussfolgerungen gezogen werden.

Quasistetigkeit: Durch die Verwendung eines *lokalen* Minimierungsalgorithmus ergibt sich in der Regel ein stetiges Zugverhalten. Bei schnellen Mausbewegungen kann man aber in ein anderes Minimum abgleiten. Je nach Natur der Gleichungen in E können aber Sprünge sogar erzwungen sein. Wenn man etwa für einen Punkt (x,y) die Gleichungen $x(x\text{-}1)=0$ und $y=0$ festlegt, kann er nur zwischen (0|0) und (1|0) hin und her springen.

Viele der Probleme, die in der Stetigkeitsdiskussion rund um Cinderella eine Rolle spielten, können gar nicht auftreten, denn es gilt die:

Existenzquantifizierung: Nur solche Züge sind möglich, die die Existenz von im jeweiligen Zug abhängigen Elementen erhalten.

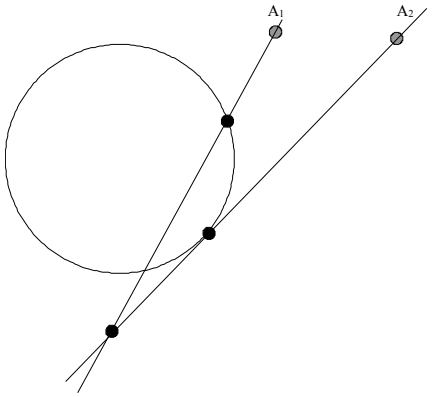


Bild 1

In Bild 1 ist das am Schnitt eines Kreises mit einer Geraden illustriert. Der graue Punkt A kann von Position A_1 aus nach rechts nur bis zur Position A_2 bewegt werden, weil sonst der Schnittpunkt mit dem Kreis nicht mehr existieren würde. Wird der Mauszeiger trotzdem weiter nach rechts bewegt, bleibt A hängen.
Durch dieses Verhalten werden natürlich alle Überlegungen zum Verhalten von verschwindenden und wieder auftauchenden (wo?) Objekten, wie sie in herkömmlichen DGS an der Tagesordnung sind, hinfällig. Es versteht sich, dass allein die Auswirkung dieses Verhaltens auf den Lernprozess nicht einfach vorherzusehen ist und kritischer didaktischer Reflexion bedarf. Sollte sich dabei herausstellen, dass die Auswirkungen überwiegend negativ sind, müssen die Koordinaten komplexifiziert werden (s.u.).
Im Gewalt-Zugmodus kann A beliebig weit bewegt werden. Der Schnittpunkt verschwindet dann nicht, sondern bleibt erhalten. Er nimmt eine mittlere Position zwischen Kreis und Gerade ein, um die Quadrat-Fehlersumme zu minimieren.

Projektionseffekte: Unter DGS-Entwicklern ist lange und ohne einheitliches Ergebnis diskutiert worden, wie man halbfreie Punkte auf Objekten mitziehen soll, wenn sich der Träger bewegt. Wenn der Träger ein Kreis oder eine Gerade ist, bietet sich die orthogonale Projektion von der letzten Position des Gleiters auf die neue Lage des Trägers an. Diese naheliegende Lösung führt aber zu dem bekannten Spiraleffekt: Wird ein halbfreier Punkt auf der Radiusstrecke eines Kreises positioniert und dann der Punkt auf dem Umfang des Kreises bewegt, wandert der Gleiter zum Kreismittelpunkt hin. Mit anderen Worten, die Projektionsstrategie erhält nicht das Teilungsverhältnis. Dies gilt auch für die Minimierungsstrategie, die sich in diesem Spezialfall sich auf die Orthogonalstrategie reduziert.

3.3 Komplexe Koordinaten

Konfigurationen von Basisobjekten, in denen das Gleichungssystem unlösbar wird, stellen ein generelles Problem dar. Der Gewalt-Zugmodus löst ein Teil dieser Probleme – allerdings mit Konsequenzen, deren didaktische Nützlichkeit noch zu bewerten ist.

In Cinderella wird zur Vermeidung von Unlösbarkeiten die gesamte Arithmetik komplex durchgeführt. Dies ist hier prinzipiell genau so möglich (wenn auch die beiden realen Implementierungen von FeliX das noch nicht tun), d.h. jede Variable x wird durch zwei reelle Variablen a,b mit $x=a+ib$ ersetzt. Die Fehlerquadratsumme ist dann anzusetzen als

$$f = \sum_{i=1}^{m} \overline{g_i} \cdot g_i$$

Dies ist dann eine reellwertige Funktion in reellen Variablen.

3.4 Technische Aspekte

Die Behandlung von kleinen Konstruktionen mit 10 bis 20 Objekten stellt softwaretechnisch kein Problem dar. Größere Probleme mit ca. 100 Objekten können mit state-of-the-Art Algorithmen ebenfalls noch behandelt werden. Dazu sind folgende Optimierungen, die für den Nutzer unsichtbar bleiben sollten, sinnvoll:

- Glättung nicht differenzierbarer Funktionen, insbesondere des Absolutbetrags
- Verminderung des Ausartungsgrades. Insbesondere die homogenen Koordinaten der Geraden sollten entweder durch Fixierung (z.B. Einheitsnormalenvektor) oder durch Umparametrisierung (trigonometrische Darstellung) entschärft werden.
- Da die meisten Gleichungen differenzierbar sind, sollte der Minimierungsalgorithmus dies ausnutzen können.
- Erzeugung von optimiertem Maschinencode zur Laufzeit.

In der MuPAD-Version von FeliX werden all diese Techniken eingesetzt.
Noch größere Konstruktionen sind mit allgemeinen Methoden nicht mehr beherrschbar. Es gibt aber eine große Klasse von Systemen, die auch dann noch effizient minimiert werden können. Entsprechende Optimierer, die das Gleichungssystem nach graphentheoretischen Methoden strukturieren, wurden in der Mathematica-Variante von FeliX ansatzweise gestestet, ihre gründliche Untersuchung steht aber noch aus.

4 Realisation als Differentialgleichungssystem

Die Bezeichnung "Dynamische Geometrie" weckt in ihrem ersten Wortteil physikalische Assoziationen. Ganz unberechtigt kann dies auch nicht sein, schließlich lassen sich viele geometrische Konstruktionen mit Stäben, Gelenken und Fäden so realisieren, dass an ihnen gezogen werden kann. Es zeigt sich, dass es gleich eine Vielzahl von Möglichkeiten gibt, Differentialgleichungen (DGL) zur Modellierung von dynamischer Geometrie einzusetzen. Nach den bisherigen Beobachtungen bei FeliX sind Optimierungsalgorithmen performanter. Zudem ist der gegenwärtige Sand der Forschung so, dass die Vielzahl der mit DGLs möglichen Zugstrategien noch nicht vollständig untersucht ist. Wir beschränken uns hier deshalb auf die Darlegung einiger Grundideen.

4.1 DGL erster Ordnung

Der einfachste Weg zu einem Differentialgleichungssystem besteht, nomen est omen, in der Differentiation. Dazu nennen wir die Variablen $V = \{q_1,\ldots,q_k,q_{k+1},q_{k+2},\ldots,q_n\}$, wobei die ersten *k* Variablen Nichtbasisvariablen sein sollen, q_{k+1},q_{k+2} sollen die mit der Maus gezogenen Variablen sein und der Rest sind Basisvariablen. In der alten Konfiguration waren die Werte der Variablen $\{q^0{}_1,\ldots,q^0{}_k,q^0{}_{k+1},q^0{}_{k+2},\ldots,q^0{}_n\}$. Der Zug mit der Maus versucht nun, q_{k+1},q_{k+2} auf neue Werte $q^1{}_{k+1},q^1{}_{k+2}$ zu bringen. In dieser einfachen Modellierung müssen wir annehmen, dass das auch möglich ist. Dann kann man annehmen, dass der Zug zwischen den Zeitpunkten *t*=0 und *t*=1 vor sich geht und dabei zu jedem Zeitpunkt gilt: $q_{k+1}(t) = q^0{}_{k+1}\cdot(1-t) + q^1{}_{k+1}\cdot t, \quad q_{k+2}(t) = q^0{}_{k+2}\cdot(1-t) + q^1{}_{k+2}\cdot t$.

Nun werden in *E* die in diesem Zug konstanten Variablen $q_{k+3},\ldots,q_n$ durch ihre Werte ersetzt, und für q_{k+1},q_{k+2} wird die obige lineare Interpolation eingeführt. Die restlichen Variablen werden als Funktionen der Zeit verstanden. Man erhält dann ein System $g_i(q_1,...,q_k,t) = 0, i = 1..n$ von Gleichungen. Dies wird durch totale Differentiation nach *t* in ein DGL-System verwandelt (der Punkt bezeichnet, wie üblich, Ableitung nach der Zeit):

$$\frac{\partial}{\partial t} g_i(q_1,...,q_k,t) + \sum_{j=1}^{k} \frac{\partial}{\partial q_j} g_i(q_1,...,q_k,t)\cdot \dot{q}_j = 0 \quad i = 1..n$$

Dieses System ist dann nach den $\dot{q}_i$ aufzulösen. Das resultierende System kann numerisch integriert werden, wobei die alte Konfiguration als Anfangswert dient. Die neue Konfiguration ergibt auch aus dem Wert der Funktionen $q_i(1)$ am Intervallende. Wenn die Ausgangskonfiguration *E* erfüllte, wird das auch die neue Konfiguration tun. Es können dann die üblichen Existenz- und Eindeutigkeitssätze zur Untersuchung herangezogen werden. Dabei stellt sich heraus, dass die Existenz einer eindeutigen differenzierbaren Lösung gewährleistet ist, solange die Jakobi-Matrix von *g* maximalen Rang hat. Diese Bedingung ist generisch erfüllt.

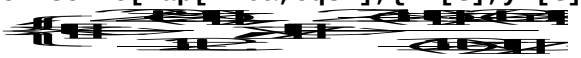
Zur Illustration zeigt das folgende Mathematica-Notebook die Berechung der Bewegung des Schnittpunktes S zweier Kreise von Radius 1. Der Mittelpunkt des ersten Kreises ist fest bei (1,0) während der Mittelpunkt des ersten Kreises von (0,0) längs der x-Achse nach (2,0) verschoben wird.

```
(* Dynamische Geometrie mit DGL
   Schnittpunkt  ist (x,y)
   Mittelpunkte M1=(x1,0) M2=(x2,0), Radius 1  *)

 eqs={(x-x1)^2+y^2==1,(x-x2)^2+y^2==1}

 (* Anfangskonfiguration *)
```

```
Conf0:={x1→0,x2→1, x→1/2,y→Sqrt[1-1/4]}
 eqs/.Conf0
 {True,True}
 (* Bewegung von M1 nach (2,0) bei t=0..1, M2 fest *)
 eqsM=Map[(#[[1]]-#[[2]])&,
          eqs/. { x→x[t], y→y[t], x1→2*t, x2→1} ]
```

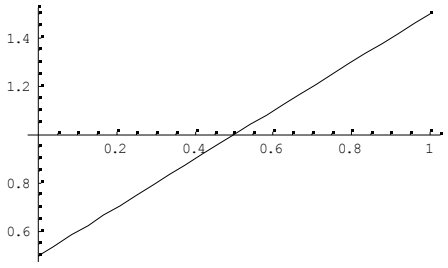
```
 eqsD=Map[Dt,eqsM]
```

```
 dsol=Solve[Map[#==0&,eqsD],{x'[t],y'[t]}]
```

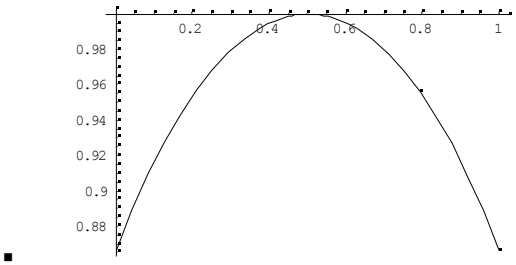


```
 deq=Map[(#[[1]]==#[[2]])&,dsol[[1]]]
```

```
 ndsol=NDSolve[Join[deq,{x[0]==(x/.Conf0),y[0]==
(y/.Conf0)}],{x[t],y[t]},{t,0,1}]

 {{x[t]→InterpolatingFunction[{{0.,1.}},<>][t],
   y[t]→InterpolatingFunction[{{0.,1.}},<>][t]}}
 ndsol[[1]]/.t→0
 {x[0]→0.5,y[0]→0.866025}
 ndsol[[1]]/.t→1
 {x[1]→1.5,y[1]→0.866025}
 Plot[Map[#[[2]]&,ndsol[[1]]][[1]],{t,0,1}]
```




```
  Plot[Map[#[[2]]&,ndsol[[1]]][[2]],{t,0,1}]
```



4.2 Systeme zweiter Ordnung: Hamiltonsche Formulierung

Das obige System nochmals zu differenzieren, erscheint wenig sinnvoll. Wenn man denn zur zweiten Ableitung greift, sollte man etwas in sich Interessantes verwirklichen: Physikalisches Verhalten. Dies wurde schon kurz (Oldenburg 2003) vorgestellt. Dabei wird den einzelnen Objekten ein Parameter "Masse" zugeordnet, der ihre Trägheit bei Bewegungen bestimmt. Die Unterscheidung zwischen Basisobjekten und anderen wird damit zu einer graduellen: Basisobjekte besitzen unendliche Masse. Im Gegensatz zu den Systemen erster Ordnung aus dem letzen Abschnitt braucht der Zielpunkt nicht konsistent zu sein: Die Maus übt nur eine Kraft in die Richtung des Ziels aus, und der Punkt folgt dem nur, soweit es seine Einschränkungen zulassen. Das Verhalten der Konstruktion gleicht dann der einer mechanischen, reibungsfreien Realisierung im schwerelosen Raum. Das hat interessante Konsequenzen. Beispielsweise kann man durch geeignetes Ziehen, die Konstruktion um ihren Schwerpunkt drehen. Stetiges Verhalten ist bei diesem Ansatz garantiert und bietet eine alternative Realisierung zur Modellierung in Cinderella. Allerdings reicht die bisherige Implementation noch nicht aus zu beurteilen, ob das resultierende Zugverhalten als intuitiv empfunden wird.

5 Kurven

Kurven sind bisher nur in der Klassifikation der Objekttypen als Funktionsgraphen, parametrisierte und implizite Kurven aufgetaucht. Sie sollten in einem ADGS ebenso wie Punkte first-class-citizens sein, d.h. sie sollten ebenso algebraisch wie geometrisch erzeuget werden können:

- Kurven werden algebraisch spezifiziert, in dem ihre Gleichung (Funktionsterm, Parameterfunktionen, implizite Gleichung) angegeben wird.
- Kurven werden geometrisch spezifiziert, in dem man eine geometrische Konstruktion erstellt, in der sich die Kurve ergibt als
    - Spur eines Punktes bei der Variation eines anderen Punktes längs eines Trägerobjektes (Ortskurven, traditionell in DGS).
    - möglicher Orbit eines Punktes, der in seiner Bewegungsfreiheit eingeschränkt ist (Relationenkurve, s. (Oldenburg 2004d))

Nach ihrer Erzeugung sollen Kurven wie andere Objekte manipuliert werden können, insbesondere sollen sie zum Ausgangspunkt weiterer Konstruktionen gemacht werden können. Dazu gehört

- Punkte an Kurven binden
- Schnitte von Kurven mit anderen Objekten
- Tangential- und Normalenvektoren
- Berechnung von Bogenlängen etc.

Diese Operationen sind verhältnismäßig einfach zu implementieren. Der Schnitt zweier Kurven beispielsweise verlangt nichts anderes, als einen Punkt zu erzeugen und für jede Kurve die passende Gleichung zu *E* hinzuzufügen.
Im folgenden beschränken wir uns auf Relationenkurven, da dies ein neues Konzept darstellen.

5.1 Relationenkurven

Schon in herkömmlichen DGS kann man beobachten, dass Punkte beim Ziehen auf drei verschiedene Arten reagieren: Entweder, sie lassen sich beliebig verschieben, oder sie sind unbeweglich, oder sie lassen sich eingeschränkt nur auf einer bestimmten Bahn verschieben. In einem konventionellen DGS kommt dieser dritte Fall nur vor, wenn der Punkt an ein Trägerobjekt (meist nur Gerade oder Kreis, evtl. Kegelschnitt oder Funktionsgraph) gebunden ist. In diesem Falle ist das Trägerobjekt schon da. Relationenkurven entstammen der umgekehrten Situation: Ein Punkt ist aufgrund von Gleichungen nur eingeschränkt beweglich und man fragt nach der Kurve, längs der er sich bewegen kann. Auch diese Frage hat natürlich zwei Gesichter: Geometrisch fragt man nach der Gestalt der Kurve, algebraisch nach ihrer Gleichung.

Lassen Sie uns das Beispiel betrachten, dass zwei Punkte mit Koordinaten (x_A,y_A) und (x_B,y_B) erzeugt worden seien, und wie einen weiteren Punkt (x,y) dadurch einschränken, dass wir verlangen, dass sein Abstand von A ebenso groß ist, wie sein Abstand von B: $(x-x_A)^2+(y-y_A)^2=(x-x_B)^2+(y-y_B)^2$. In diesem Fall genügt eine elementare Termumformung um zu sehen, dass (x,y) einer Geradengleichung genügt. Komplizierter wird es, wenn mehrere Gleichungen zusammen spielen.
Für diesen Fall habe ich den folgenden Pseudocode-Algorithmus veröffentlicht (Oldenburg 2004d):

```
Eingabe: Equations, BasicObjects, P (der zu verfolgende
Punkt)
V:= Variablen ohne Variablen von {BasicObjects,P}
Eliminiere Variablen V aus Equations
```

Dieser Algorithmus ist insofern unbestimmt, als die Elimination im letzten Schritt nicht präzisiert wird. Dies soll hier geschehen für den Fall, dass das Gleichungssystem allein polynomielle Gleichungen enthält (Viele Gleichungssysteme, die diese Beziehung verletzen, können auf eine solche Form gebracht werden, in dem man z.B. auftretende Wurzeln $\sqrt{A}$ durch eine neue Variable *v* ersetzt und die weitere Gleichung $v^2=A$ zu *E* hinzunimmt.).

Sei also jetzt $E=\{g_1=0,...,g_n=0\}$ rein polynomiell und die Variablen seien $\{x_1,...,x_k,x_{k+1},x_{k+2},...,x_m\}$, wobei die ersten k Variablen Nicht-Basisvariablen sein mögen, x_{k+1},x_{k+2} die Variablen des eingeschränkten Punktes, und die restlichen Basis-

variablen. Der Eliminationsschritt aus obigem Pseudoalgorithmus verlangt dann also, die Variablen $\{x_1,...,x_k\}$ zu eliminieren, und die restlichen zu behalten. Man erhält dann also eine Gleichung in x_{k+1}, x_{k+2}, die noch Basisvariablen als Parameter beinhalten kann.

Aus der Theorie der Gröbnerbasen ist bekannt (Cox et al 1997), dass diejenige Teilmenge *S* einer Gröbnerbasis zu *E* bzgl. der lexikographischen Termordnung mit $x_1 > x_2 > \cdots > x_m$, die genau aus den Elementen besteht, die die Variablen $\{x_1,...,x_k\}$ *nicht* mehr enthalten, eine Gröbnerbasis des entsprechenden Eliminationsideals bildet. Sie sind – nach der geometrischen Interpretation der Elimination – die Gleichungen derjenigen algebraischen Varietät, die durch Projektion der Varietät von *E* auf die Dimensionen der zu behaltenden Variablen entsteht. Die Polynommenge *S* des Eliminationsideals muss jetzt inspiziert werden:

- Wenn S nur aus einem Element besteht, das x_{k+1}, x_{k+2} enthält, ist man fertig: Die gesuchte Gleichung ist gefunden.
- Wenn S mehre Gleichungen enthält, die x_{k+1}, x_{k+2} verknüpfen, kann auch noch (mindestens) eine Basisvariable eliminiert werden. Dies kann sukzessive mit x_{k+3} beginnend durchgeführt werden.
- Wenn *S*={1}, bedeutet das zunächst, dass keine Lösung gefunden wurde. Dies kann allerdings oft daran liegen, dass nicht alle zu eliminierenden Variablen auch eliminiert werden konnte. Typische Ursache dafür sind die homogenen Koordinaten von Geraden. In diesem Fall sind von *a*,*b*,*c* nur *a*,*b* zu eliminieren. Die Elemente von *S* spalten dann einen Faktor ab, der *c* alleine enthält.

Die Umsetzung dieses Verfahrens liefert für die üblichen klassischen algebraischen Kurven aus Standardkonstruktionen die Gleichungen in typischerweise ein bis drei Sekunden.

Die Berechnung von traditionellen Ortskurven stellt einen Spezialfall der Berechung einer Relationenkurve dar, falls der unabhängige (Steuer-)Punkt der Ortskurvenkonstruktion als Nicht-Basisobjekt vereinbart ist (oder zumindest für die Berechnung temporär so behandelt wird). Allerdings ermöglichen traditionelle Ortslinienkonstruktionen auch einfachere, effizientere Zugänge, auf die hier aber nicht eingegangen werden soll.

6 Beispiele für die Anwendung von ADGS

6.1 Dynagraph

Goldenberg hat 1991 mit dem Programm Dynagraph eine alternative graphische Repräsentationsform von Funktionen in die didaktische Diskussion eingeführt, bei der der Kovariationsaspekt stark zum Tragen kommt: Mit der Maus wird das Funktionsargument auf einem Zahlenstrahl eingestellt und der Funktionswert erscheint auf einem anderen, parallelen Zahlenstrahl. In letzter Zeit wurde diese Idee durch DGS wieder belebt (Malle

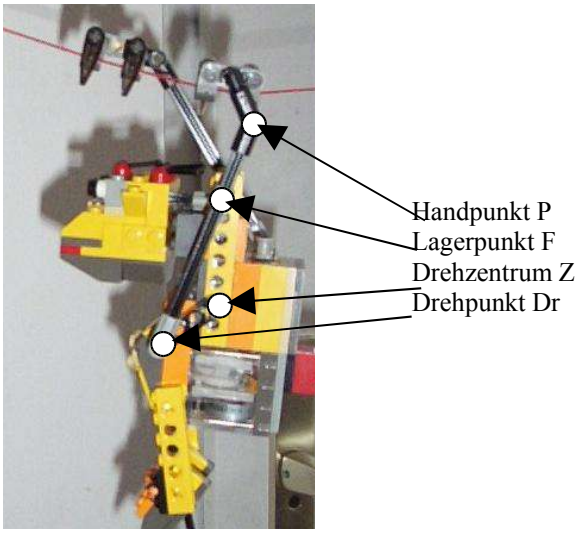
2000). In einem ADGS lassen sich solche Anwendungen sehr schnell konstruieren, wie der folgende Mathematica/FeliX-Code zeigt:

```
addObject["point",0,0,"X"];
addObject["point",0,2,"Y"];
addEquation[yc["X"]==0];
addEquation[yc["Y"]==2];
addEquation[xc["X"]^2==xc["Y"]];
```

Die beiden Punkten lassen sich nur auf der x-Achse, resp. auf der y=2-Parallelen dazu verschieben und es gilt stets eine quadratische Relation zwischen den Abszissen. Weil man in einem ADGS an allen Punkten ziehen kann, kommt neben der Funktion $x \mapsto x^2$ sofort auch die Umkehrfunktion dazu ins Spiel: Man kann am Punkt Y ziehen und sehen, welche x-Werte dazu passen. Dabei entdeckt man z.B., dass sich Y dem Versuch widersetzt, ihn zu negativen Werten hin zu ziehen.

6.2 Beispiel eines Schubkurbelgetriebes

Betrachten wir die Leistungsfähigkeit der Berechnung von Relationenkurven an einem Beispiel, das der Schüler-Realität entspringt. Das Photo zeigt das Schubkurbelgetriebe eines Lego-Kletterroboters. Dieser lässt sich natürlich auch mit einem herkömmlichen DGS realisieren, aber in einem ADGS kann man die Mechanik sowohl an Dr als auch P ziehen. Außerdem gelingt die Berechung der Bahnkurve von P symbolisch mit Parametern.



In der Mathematica-Version von FeliX kann die Konstruktion mit der Maus oder durch folgende Konstruktionsbefehle geschehen:

```
Z=addObject["point",1,-2,"Z"];
k=addObject["circlePR",Z,3,"k"];
Dr=addObject["point",1,-1,"Dr"];
F=addObject["point",1,2,"F"];
```

```
g=addObject["line2P",Dr,F,"g"];
addEquation["PointOn",Dr,k];
P=addObject["point",1,1,"P"];
addEquation["PointOn",P,g];
addEquation[distsqr[P,Dr]==100];
BasicObjects={Z,F};
```

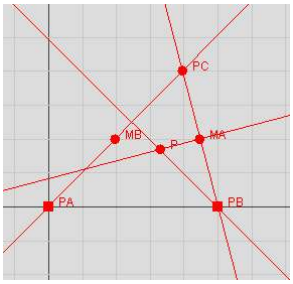
Die Relationen kurven von P genügt dann der Gleichung:

$$
\begin{aligned}
&8816\,x^2 + 416\,x^3 - 204\,x^4 - 4\,x^5 + x^6 + 1600\,x\,y - 832\,x^2\,y - 16\,x^3\,y + 8\,x^4\,y + \\
&7616\,y^2 + 416\,x\,y^2 - 396\,x^2\,y^2 - 8\,x^3\,y^2 + 3\,x^4\,y^2 - 832\,y^3 - 16\,x\,y^3 + 16\,x^2\,y^3 - \\
&192\,y^4 - 4\,x\,y^4 + 3\,x^2\,y^4 + 8\,y^5 + y^6 - 17632\,x\,xc[F] - 832\,x^2\,xc[F] + \\
&408\,x^3\,xc[F] + 8\,x^4\,xc[F] - 2\,x^5\,xc[F] - 1600\,y\,xc[F] + 64\,x\,y\,xc[F] + \\
&32\,x^2\,y\,xc[F] - 16\,x^3\,y\,xc[F] - 800\,y^2\,xc[F] + 384\,x\,y^2\,xc[F] + 8\,x^2\,y^2\,xc[F] - \\
&4\,x^3\,y^2\,xc[F] - 16\,x\,y^3\,xc[F] - 2\,x\,y^4\,xc[F] + 8816\,xc[F]^2 + 416\,x\,xc[F]^2 - \\
&204\,x^2\,xc[F]^2 - 4\,x^3\,xc[F]^2 + x^4\,xc[F]^2 + 768\,y\,xc[F]^2 - 16\,x\,y\,xc[F]^2 + \\
&8\,x^2\,y\,xc[F]^2 + 208\,y^2\,xc[F]^2 - 4\,x\,y^2\,xc[F]^2 + 2\,x^2\,y^2\,xc[F]^2 + 8\,y^3\,xc[F]^2 + \\
&y^4\,xc[F]^2 - 1600\,x\,yc[F] + 1600\,x^2\,yc[F] - 15232\,y\,yc[F] - 32\,x\,y\,yc[F] + \\
&408\,x^2\,y\,yc[F] + 8\,x^3\,y\,yc[F] - 2\,x^4\,y\,yc[F] + 1664\,y^2\,yc[F] + 32\,x\,y^2\,yc[F] - \\
&16\,x^2\,y^2\,yc[F] + 384\,y^3\,yc[F] + 8\,x\,y^3\,yc[F] - 4\,x^2\,y^3\,yc[F] - 16\,y^4\,yc[F] - \\
&2\,y^5\,yc[F] + 1600\,xc[F]\,yc[F] - 1600\,x\,xc[F]\,yc[F] + 800\,y\,xc[F]\,yc[F] - \\
&800\,x\,y\,xc[F]\,yc[F] + 7616\,yc[F]^2 - 384\,x\,yc[F]^2 + 196\,x^2\,yc[F]^2 - \\
&4\,x^3\,yc[F]^2 + x^4\,yc[F]^2 - 832\,y\,yc[F]^2 - 16\,x\,y\,yc[F]^2 + 8\,x^2\,y\,yc[F]^2 - \\
&192\,y^2\,yc[F]^2 - 4\,x\,y^2\,yc[F]^2 + 2\,x^2\,y^2\,yc[F]^2 + 8\,y^3\,yc[F]^2 + y^4\,yc[F]^2
\end{aligned}
$$

Man beachte, dass die Gleichung hier die Koordinaten von F als Parameter enthält. Solche Ausdrücke sind intern nützlich für die Durchführung verschiedenster Operationen, wie oben aufgelistet. Für den Menschen sind sie natürlich wenig erquicklich. Allerdings wurde hier ein besonders komplexes Beispiel gezeigt. Die Gleichungen der klassischen algebraischen Kurven sind wesentlich kompakter. Aber auch mit solchen Monstern kann man sinnvolle Dinge tun, beispielsweise Faktorisieren, um herauszufinden, ob sich die Kurve aus mehreren Komponenten zusammensetzt. Durch Differenzieren und Lösen eines polynomiellen Gleichungssystems findet man die singulären Punkte der Kurve etc.

6.3 Knechtel-Kurven

Heiko Knechtel hat mir auf der Computeralgebra Tagung 2004 in Schönenberg die Frage gestellt, welche Kurven sich ergeben, wenn man in einem Dreieck ABC Punkt C längs einer Parallelen zur Grundseite c verschiebt und dabei den Schnitt *unterschiedlicher* Transversaler verfolgt. Im folgenden Bild wird beispielsweise die Höhe über b mit der Mittelsenkrechten von a geschnitten.

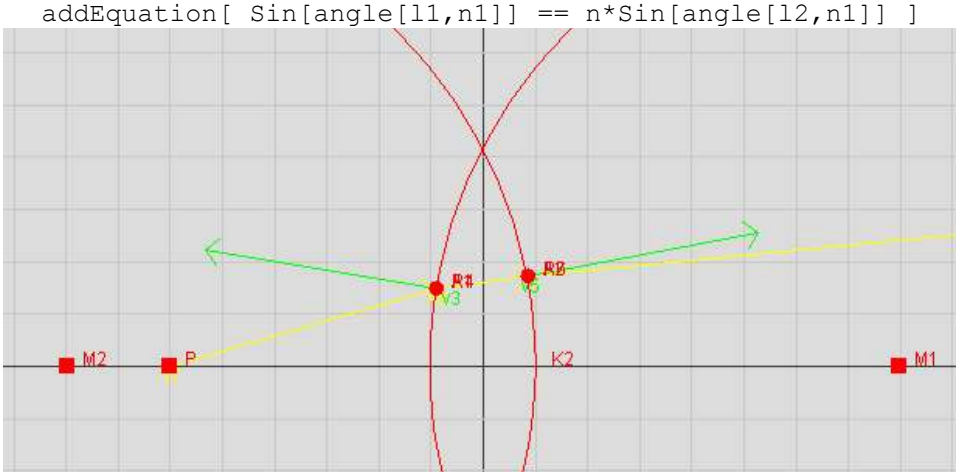


In dieser Konfiguration kann man an allen Punkten ziehen! Die Bahn von P bei festen Basispunkten PA,PB ist, wie FeliX in 1,4 Sekunden ausrechnet, [redacted] (=0), also ein Kegelschnitt.

6.4 Modellierung einer Linsenoberfläche

Mit ADGS können optische Geräte gut modelliert werden. Das folgende Bild zeigt beispielsweise den Strahldurchgang durch eine Linse, deren Oberfläche sich aus zwei Kreisen zusammensetzt. An den Grenzflächen Glas-Luft tritt Brechung auf. Diese wird durch das Snelliussche Brechungsgesetz gegeben $\sin(\alpha_1) = n \cdot \sin(\alpha_2)$ (dabei ist n der Brechungsindex und die Einfalls- und Ausfallswinkel werden gegen das Lot gemessen). Da ein ADGS auf beliebigen Kurven Normalen errichten kann (was hier, im Falle des Kreises, natürlich besonders einfach ist), kann das Brechungsgesetz in seiner mathematischen Form eingegeben werden. In Mathematica-Syntax lautet dies (n1 ist das Lot auf den Kreis mit Mittelpunkt M1):

```
addEquation[ Sin[angle[l1,n1]] == n*Sin[angle[l2,n1]] ]
```



Wieder ermöglicht das ADGS FeliX ein Ziehen an beliebigen Punkten. In einem ähnlichen Beispiel wird untersucht, wie eine sinusförmig gewellte Oberfläche sich auf einfallendes Licht auswirkt.

7 Didaktische Überlegungen

Ein ADGS ist mehr als die Summe seiner Teile CAS und DGS, sondern es ermöglicht grundsätzlich neue Arbeitsweisen. Deshalb ist auch die didaktische Bewertung mehr als eine Kombination von Überlegungen der CAS- und der DGS-Didaktik, obwohl die dort erzielten Erkenntnisse natürlich auch für ein ADGS relevant sind.

So wie schon herkömmliche DGS viele Anwendungen gefunden haben, die weit jenseits dessen lagen, was die Autoren ursprünglich intendierten, so ist auch bei ADGS zu erwarten, dass es noch viele originelle Einsatzmöglichkeiten geben wird, die heute noch nicht geahnt werden. Dies ist der Vorteil eines echten Werkzeugsystems (vgl. auch Biehler 1991), das den Nutzer in die Lage setzen will, möglichst viele Dinge zu tun, ein System, das gerade darauf verzichtet den Nutzer irgendwie einzuengen. Die Konsequenz ist natürlich, dass sich die Nutzer oft auf nicht vorhergesehen Wegen bewegen werden. Was das für die didaktische Nutzung bedeutet, liegt noch weitgehend im Dunkeln. Hier sollen einige Thesen aufgestellt werden, die meine Erwartungen umreisen.

7.1 Thesen zur didaktischen Relevanz von ADGS

These 1: ADGS ist kein System zum Erlernen geometrischer Konstruktionen
Die in der Folge von Cabri entstandenen Systeme sind vom Ansatz her zunächst der synthetischen Geometrie zuzurechnen gewesen, sie intendierten Konstruktionen, analog zu denen mit Zirkel und Lineal, zu ermöglichen (man vgl. dazu aber auch die kritische Diskussion in Hölzl 1999). Diese klassische geometrische Konstruktion ist ein funktionales Konzept: Zu bestimmten Eingabeparametern wird eine Menge von Ausgabeobjekten konstruiert. Dies kontrastiert zum stark relationalen Ansatz von ADGS. Man kann in einem ADGS Konstruktionen vornehmen, die mit Zirkel und Lineal unmöglich, oder wesentlich schwieriger wären. Beispielsweise findet man eine gemeinsame Tangente an zwei Kreise, indem man an jeden Kreis einen Tangentenvektor (in einem beliebigen Punkt) sowie eine beliebige Gerade zeichnet, und danach mittels "glue to" Fuß- und Endpunkte der Tangentialvektoren auf diese Gerade legt. Offensichtlich wird dadurch die intellektuelle Herausforderung des Problems zerstört, die Problemlösung reduziert sich auf eine Formulierung der geforderten Eigenschaften. Natürlich kann man auch mit ADGS die übliche Zirkel-und-Lineal-Konstruktion durchführen, aber das Werkzeug öffnet eben auch bequemere Wege.
Deshalb: Wenn das Unterrichtsziel im Erlernen des klassischen Konstruktionsbegriffs und der damit verbunden Fähigkeiten liegt, sollte ein herkömmliches DGS eingesetzt werden.

These 2: ADGS unterstützt verschiedene Grundvorstellungen von Variablen
Variablen werden in einem ADGS verwendet als Referenzen auf die aktuellen numerischen Werte und als Symbole in Gleichungen und Terme. Diese beiden Formen sind über die Einsetzungsfunktion miteinander verbunden.

These 3: ADGS unterstützt verschiedene Grundvorstellungen von Gleichungen
Gleichungen treten in ADGS in verschiedenen, aber eng verbundenen Formen auf:

- Gleichungen als Festlegungen, als "Gesetz" für die Bewegung von Objekten
- Gleichungen als Beschreibungen von geometrischen Örtern
- Gleichungen als Bestimmungsgleichungen

These 4: ADGS ermöglicht ein Erleben der Bedeutung von über- bzw. unterbestimmten Gleichungssystemen
Durch Ziehen mit der Maus kann die Dimension des Lösungsgebildes erfahren werden. Dies ist insbesondere interessant, wenn die Dimension 1 ist. In (Oldenburg 2003) wurde ausführlich die Erforschung von Parabeln in diesem Sinne dargestellt: Ein Punkt wird der Gleichung unterworfen, dass sein Abstand von einem festen (Brenn-)Punkt gleich seinem Abstand von einer (Leit-)Geraden sein soll. Die Maus macht die Bedeutung der Abstandsgleichung unmittelbar *erfahrbar*, noch bevor der Graph gezeichnet wird.
Auch bei anderen Gleichungssystemen lässt einen die Maus erfahren, wie viel Freiheit das Gleichungssystem den Variablen noch lässt. Da Objekte willkürlich als Basis- oder Nichtbasisobjekte deklariert werden können, eröffnet sich hier ein weiteres Interaktionsfeld mit der Konstruktion. Interessante Fragen sind z.B., wann eine bestimmte Konfiguration starr wird.

These 5: ADGS unterstützt physikalisches Modellbilden
Wir haben am Beispiel der Lichtbrechung gesehen, was diese These meint. Physikalische Gesetze stellen in der Regel Relationen zwischen Größen dar. Die Übersetzung in einen funktionalen Zusammenhang geht oft mit der Auszeichnung einer Ursache und einer Wirkung einher. Dieser Zusammenhang ist in der Physik aber prinzipiell umkehrbar, so dass die funktionale Sicht nur einen Aspekt wiedergibt. Ein ADGS dagegen erlaubt die Modellierung des physikalischen Gesetzes in seiner relationalen Allgemeinheit.

These 6: Horizontale Vernetzung: ADGS unterstützt realitätsnahes Operieren mit mechanischen Geräten
Dies ist eine Fortsetzung der letzen These, scheint mir aber von besonderer Wichtigkeit, so dass es hier hervor gehoben wird. Reale Gelenkmechanismen (aus Lego o.ä.) können an allen Punkten bewegt werden. In herkömmlichen DGS muss zur Modellierung ein Punkt als Basispunkt ausgezeichnet werden und nur der ist mit der Maus direkt beweglich. Die Realitätsnähe von ADGS fördert eine horizontale Form der Vernetzung, bei der gemeinsame intellektuelle Konstrukte auf reale Gelenke wie auf geometrische Konstruktionen angewendet werden können.

8 Fazit

Dieser Aufsatz hat die Vision eines ADGS präzisiert indem die Grundlagenfragen soweit beleuchtet wurden, dass die prinzipielle technische Realisierbarkeit klar wird. Mit dem Prototyp FeliX ist dieser Nachweis auch in der Praxis angetreten worden.
ADGS sind Lernumgebungen, die sehr reich an Mathematik sind. Es werden mehr mathematische Konzepte dynamisch computerisiert als bei herkömmlichen DGS oder CAS und es stellt eine interessante didaktische Herausforderung dar, die damit verbunden Chancen zu nutzen.

9 Literatur

Anderson, J. R. [1995]: Kognitive Psychologie. Heidelberg:Spektrum.

Biehler, R. [1991]: Fortschritte der Software und die Tradition der Schulmathematik. In: W. Dörfler: Computer – Mensch – Mathematik. Wien.

Borovcnik, M., Kautschisch, H. [2002]: Technology in Mathematics Teaching. Wien.

Botana, F. & Valcarce J. L [2004]: Lugares 1.1. http://rosalia.uvigo.es/sdge/lugares.

Cox, D. & Little, J. & O`Shea, D. [1997]: Ideals, Varieties and Algortihms. New York: Springer.

Cuoco, A. C. & Curcio, F. R. [2001]: The Roles of Representation in School Mathematics. Reston: NCTM.

Edelmann, W. [1996]: Lernpsychologie. Weinheim: Beltz.

Ehmann, M. [2004]: GEONExT: Integration algebraischer Objekte und Internationalisierung. Müchen: Verlag Dr. Huth.

Fernandes, H. [2004]: DrGeo. www.ofset.org/drgeo

Halverscheid, St. [2004]: Dynamic Geometry Software as a Simulation Tool for Algebra Problems. ICME10, Koppenhagen.

Hohenwater, M. [2004]: www.geogebra.at

Hölzl, R. [1994]: "Die konstruierten Punkte noch binden!" – Schülervorstellungen von der Cabri-Geometrie. In: H. Krautschitsch & W. Metzler: Anschauliche und Experimentelle Mathematik II. Wien: hpt.

Hölz, R. [1999]: Qualitative Unterrichtsstudien zur Verwendung dynamsicher Geometrie-Software. Augsburg: Wißner-Verlag.

Issing, L. J. & Klimsa, P. (Hrsg.) [2002]: Information und Lernen mit Multimedia und Internet. Weinheim: Beltz.

Kortenkamp, U. [1999]: Foundations of Dynamic Geometry. ETH Zürich.

Kortenkamp, U. & , Richter-Gebert, J. [2004]: Cinderella. www.cinderella.de

Malle, G. [2000]: Zwei Aspekte von Funktionen: Zuordnung und Kovariation. *mathematik lehren* 103, 8-11.

Mechling, R [2004]: Euklid Dynageo. www.dynageo.de

Oldenburg, R. [2003]: Feli-X: Ein Prototyp zur Integration von CAS und DGS. in: P.Bender et al.: Lehr- und Lernprogramme für den Mathematikunterricht, Hildesheim 2003, 123-132

Oldenburg, R. [2004a]: FeliX – ein Computeralgebra-gestütztes dynamisches Geometrieprogramm. *Computeralgebra-Rundbrief* , 34, 17-19.

Oldenburg, R. [2004b]: Ortslinien – eine Beziehungsreiche Idee mit Computerhilfe entdecken. *Der Mathematikunterricht*, 4/2004, 17-23.

Oldenburg, R. [2004c]: FeliX: www.oldenburg-goettingen.gmxhome.de

Oldenburg, R. [2004d]: Ortslinien – eine Beziehungsreiche Idee mit Computerhilfe entdecken. *Der Mathematikunterricht* 4/2004, 17-23.

Schumann, H. [1991]: Schulgeometrisches Konstruieren mit dem Computer. Stuttgart: Metzler.

Schumann, H. & Green, D. [2000]: New protocols for solving geoemtric calculation problems incorporating dynamic geometry and computer algebra software. *Int. J. Math. Educ. Sci. technology*, 31, 319-339

Sfard, A. [1991]: On the Dual Nature of Mathematical Conceptions. *Educational Studies in Mathematics*, 26, 191-228.

Wand, D. [1998]: Gröbner Bases Applied to Geometric Theorem Proving and Discovering. In: B. Buchberger & F. Winkler: Gröbner Bases and Applications. Cambridge: Cambridge University Press.

Zimmermann, B. [2003]: Darstellungswechsel als eine wichtige Methode zur Lösung von Problemen – schon in der Geschichte der Mathematik, MU 6/2003, 6-15

Adresse des Autors

Dr. Reinhard Oldenburg
Albrechtstr. 5
37085 Göttingen
roldenburg@gmx.de