Objektorientiertes Lösen von Gleichungssystemen mit MuPAD

von

Reinhard Oldenburg, Göttingen

Zusammenfassung: Die Theorie des Reifikationsprozesses wird in Bezug zum Paradigma der objektorientierten Modellierung gesetzt und in einem Unterrichtsversuch zur Lösung von linearen Gleichungssystemen umgesetzt.

Summary: The theory of reification is combined with the concept of object oriented modelling from computer science. An application in the teaching of linear systems of equations is described.

Einleitung

Das Operieren mit Objekten wird in der Folge von Piaget und anderer konstruktivistischer Schulen zu den Grundvoraussetzung des Aufbaus geeigneter kognitiver Strukturen gerechnet. Aus der Sicht der Mathematikdidaktik ist vor allem die individuelle Genese mathematischer Objekte wichtig, wie sie in der Reifikationstheorie untersucht wird.

Ausgangspunkt für die vorliegende Arbeit war die Beobachtung, dass es eine Parallelität dieser kognitiven Theorien mit der in der Softwaretechnik weithin eingesetzten Technik der objektorientierten Modellierung gibt (vgl. [Baumann 1996, S. 277ff]). Dies bietet einen praktischen wie theoretischen Rahmen, in dem die Existenz mathematischer Objekte auf den Computer ausgedehnt werden kann. Am Beispiel des Lösens linearer Gleichungen in Klasse 9 wird dieser neue Ansatz vorgestellt, und es wird über Erfahrungen im Unterricht berichtet.

Von der Reifikationstheorie zur Objektorientierung in der Computeralgebra

Die Objekte der Mathematik können den Schülern nicht gegeben werden, sondern sie müssen von ihnen erschaffen werden. Dieser Umstand nimmt eine zentrale Position in vielen aktuellen Erkenntnistheorien ein. Beispielsweise schreibt Steinbring im Kontext einer auf Saussures Semiotik aufbauenden Epistemologie [Steinbring 2000, S. 33], dass „das vom Mitteilenden intendierte Bezeichnete … nur vom Mitteilungsempfänger hergestellt werden [kann]“.

Ziel des Unterrichtenden muss es also sein, Lernumgebungen zu schaffen, in denen die Schüler die Konstruktionen effektiv durchführen können. Diese Konstruktionen können (in nicht klar trennbarer Weise) dem Erwerb von mathematischen Begriffen oder der Erzeugung mathematischer Objekte dienen. Die vorliegende Arbeit beschäftigt sich besonders mit dem letztgenannten Zweck.

Vollrath hat in seiner Didaktik der Algebra anhand des Beispiels der Funktionen den Weg des Lernens in vier Stufen eingeteilt, deren höchste darin besteht, dass die Schüler unter dem Begriff ein (abstraktes) Objekt verstehen, mit dem sie operieren können (die interessante Frage, inwieweit Vollraths Darstellung eine Konzentration auf bestimmte Begriffstypen zugrunde liegt, bleibt hier unbeantwortet). Hier bietet sich eine erste Parallele zur Softwaretechnik, in der denjenigen Programmiersprachen die höchste Ausdruckskraft zugesprochen wird, in denen alle Sprachobjekte first-class [siehe z.B. Abelson & Sussman 1998, S. 78] sind, also beliebig in Datenstrukturen gespeichert und als Argumente und Ergebnisse von Berechnungen anonym oder benannt erzeugt werden können. Dies entspricht auch der üblichen Arbeitshaltung der Hochschulmathematik.

Wie aber können Schüler dieses Niveau erreichen, auf dem sie problemlos mit abstrakten Objekten spielen können? Ich schließe mich der (in sich reichlich heterogenen) Position an, dass sich die Entwicklung mathematischer Konzepte auf die Ausführung von Operationen stützt [z.B. Piaget, Dörfler, Sfard]. Allerdings ist der Weg von der Handlung zum Konzept äußerst komplex, wie aus verschiedener Sicht dargelegt wurde. Beispielsweise ist das von Dörfler unter dem Schlagwort der „kognitiven Distanz zwischen Handlung und Konstrukt“ ins Bewußtsein gebrachte lernpsychologische Problem aus philosophischer Sicht ein Spezialfall der „Unterbestimmtheit der Theorie durch die Erfahrung“ [Quine 1992, S 96ff], wie sie in der naturalisierten Erkenntnistheorie schon länger diskutiert wurde. Die Handlungen legen also keinesfalls fest, welche kognitiven Konstrukte aufgebaut werden, aber sie geben einen Rahmen vor, gewissermaßen eine Schnittstelle, die die Menge der konkreten Handlungen absteckt.

Diese Bemerkungen machen deutlich, dass hier ausgiebige Untersuchungen nötig sind. Anna Sfard hat in einer Reihe von Arbeiten, die in den neunziger Jahren des letzten Jahrhunderts erschienen sind, hier Wesentliches geleistet. Bei ihrer Untersuchung zur Entstehung der Objekte, die sie mit dem Schlagwort „Reifikation“ bezeichnet, hat sie sowohl die individuelle als auch die kollektiv-geschichtliche Perspektive einbezogen. Dabei stellt sie den Prozess- und den Objektcharakter mathematischer Konzepte einander gegenüber und interpretiert ihre Dualität als zeitlich-logischen Ablauf: Kognitive Konzepte nehmen ihren Ausgang als Prozesse, die erst später eine klare Objektinterpretation bekommen (dabei verliert der Prozesscharakter aber nicht seinen Wert, beide bleiben in einer Art Komplementarität verbunden). In der historischen Dimension bedeutet das beispielsweise, dass komplexe Zahlen zunächst (in den Cardanoschen Formeln) als Hilfsmittel im Rechenprozess auftraten, bevor ihnen eine eigenständige Existenz zuerkannt wurde.

Diese wichtige Erkenntnis, dass nämlich die Reifikation erst am Ende eines notwendigen Prozesses stehen kann, wird auch von ganz anderer Seite theoretisch gestützt. Quine hat wiederholt auf die zentrale Stellung der Ontologie im „geistigen Betriebssystem“ (dieser Terminus ist von den Vertretern der Osnabrücker Schule [Sjuts 1999, S. 8] entliehen, er trifft aber das, was Quine meist mit der Metapher des Netzes belegt, recht gut) des Menschen hingewiesen (beispielsweise bestimmt sie die Bedeutung der Quantoren). Die Ontologie ist deshalb besonders immun gegen Änderungen. Der Mensch ist in diesem Bereich von Natur aus konservativ und reifiziert nur nach ausreichender Bewährung eines Konzeptes (in den emanzipierteren Bereichen des Denkens mag das nicht mehr so sein, doch auch dort ruft uns Oackhams Rasiermesser [Honderich 1995, S. 633] zu ontologischer Bescheidenheit).

Aus didaktischer Sicht ist es natürlich unbefriedigend, nur die Schwierigkeiten des Weges zu konstatieren. Quine hat in seiner naturalisierten Erkenntnistheorie aber auch verschiedene Prinzipien der Reorganisation des kognitiven Systems untersucht [Quine 1992, S 14ff]. Besonders wichtig ist das Prinzip der kleinsten Veränderung (kleinste Verstümmelung). Aus ihm kann man den Schluss ziehen, dass es zur Förderung des Reifikationsprozesses günstig ist, wenn die zur Reifikation führenden Operationen bereits in einer mit der aufzubauenden Objektstruktur konsistenten Art durchgeführt werden. Dann bleiben die Änderungen insgesamt gering, weil weniger vermittelnde Interpretation zwischen reifiziertem Objekt (interner Bezug) und Operationsgegenstand (externer Bezug) nötig ist. In Reinform gedacht ergibt sich freilich ein Dilemma: Idealerweise sollten die Schüler mit den Gegenständen operieren, die sie ja gerade erst erschaffen müssen. Es war wiederum Anna Sfard [Sfard 1998], die auf dieses konstruktivistischen Auffassungen inhärente Problem hingewiesen hat.

Als Lösung bietet sich eine externe, computer-basierte Realisierung der Objekte an, denn die Methoden der objektorientierten Modellierung sind auch auf algebraische Objekte anwendbar. Damit kann ein Operationsfeld bereit gestellt werden, in dem die Reifikation begünstigt wird. Dies ist die Grundthese der vorliegenden Arbeit.

Das Paradigma der Objektorientierung in der Informatik wurde in den letzten Jahrzehnten mit immer größerem Erfolg angewendet und stellt heute das Standardverfahren zur informatorischen Modellierung sowohl im Kleinen (Programmierung) als auch im Großen (Planung) dar [Meyer 1988]. Objekte im Sinne dieses Ansatzes besitzen einen inneren, gekapselten Zustand, der durch Nachrichten verändert werden kann. Jedes Objekt ist Instanz einer Klasse, die sein Protokoll, also die Menge der möglichen Nachrichten, bestimmt. In der Mathematik werden zwar gänzlich andere Begriffe verwendet, aber es gibt eine offensichtliche Analogie zwischen den beiden Bereichen:

Mathematik	Objektorientierung
Kategorie	abstrakte Klasse/ interface
algebraische Struktur (Menge)	Klasse
einzelnes Objekt	Objekt (Instanz einer Klasse)
Abbildung, Funktion, Relation, …	Methode

Gefüllt mit einem konkreten Beispiel sieht diese Tabelle so aus:

Mathematik	Objektorientierung (Java-Syntax)
Körper	interface field …
komplexe Zahlen	class Complex implements field …
3 – 4i	Complex z=new Complex(3,–4)
|z|	z.abs()

Natürlich soll keineswegs behauptet werden, dass diese Übertragung ein Schema zur Mechanisierung der Mathematik liefere. Fern von den theoretischen Grenzen eines solchen Vorhabens zeigen die Computeralgebrasysteme Axiom und MuPAD trotz erheblicher Erweiterung der informatorischen Modellierung (z.B. durch Kategorien und Axiome), dass hier nur Teile, wenn auch wichtige Teile, mathematischer Inhalte umgesetzt werden können.

Die hier aufgeführte Analogie beansprucht keineswegs Originalität, sie ist weit verbreitetet, ohne dass es eine klare Referenz darauf gibt. Es gibt allerdings auch abweichende Interpretationen der objektorientierten Modellierung, beispielsweise von [Leinbach&Fink 2002], bei denen lediglich der Modulaspekt von Klassen wichtig ist.

Wichtiger ist hier der Bezug zu den kognitiven Prozessen der Lernenden, die die folgende Tabelle umreißt:

Objektorientierung	Reifikationstheorie
Klasse	Begriff
Objekt	Objekt
Methoden	Operationen
Implementation verborgen, Methoden definieren Umgang	unzugängliche individuelle Konstruktion, durch Operationen definiert

Auch hier ist die Warnung angebracht, dass die Analogie nicht als Identität missverstanden werden darf. Gerade der Unterschied, dass die Objekte im Computer im Gegensatz zum Gehirn der Lernenden geplant und kontrolliert erzeugt werden können, ist ja für die hier beschriebene Anwendung entscheidend. In beiden Fällen aber wird die Bedeutung der Objekte durch ihre Objekte/Methoden definiert, mit Wittgenstein könnte man sagen, dass sie *ein* Sprachspiel definieren, an dem die Schüler teilnehmen können.

Es gibt mittlerweile auch eine Reihe von objektorientiert arbeitenden Computeralgebraprogrammen (Axiom, MuPAD und in Ansätzen Maple). Diese Systeme erlauben neben dem objektorientierten auch den funktionalen und prozeduralen Programmierstil.

Konkretisierung am Beispiel linearer Gleichungssysteme

Die theoretischen Überlegungen des letzten Abschnitts können naturgemäß auf eine ganze Reihe von Konstruktionen, die die Schüler im Mathematikunterricht leisten müssen, angewendet werden. Typische Beispiele könnten Wurzeln, Vektoren oder auch komplexe Zahlen als neuartige Objekte sein. Ich habe eine konkrete Umsetzung aber an einem anderen Thema untersucht, das möglicherweise nicht so naheliegend ist, nämlich den linearen Gleichungssystemen. Grundlage dieser Entscheidung war meine Beobachtung, dass die Schüler einer neunten Klasse beim Lösen von Gleichungssystemen sich vornehmlich auf die Umformung einzelner Gleichungen und gelegentliches, aber nicht sehr systematisches Einsetzen stützten. Dies resultierte sicher daraus, dass die Schüler das Lösen linearer Gleichungssysteme selbst erarbeiten mussten, eine Klassifizierung der verschieden Techniken in die üblichen Schemata (Einsetzungs- Gleichsetzungsverfahren) erfolgte erst im Nachhinein. Mich störte an diesem Zustand, dass die Schüler das Gleichungssystem nicht als ganzes Objekt manipulierten, sondern stets auflösten. Offensichtlich hatten sie die Systeme noch nicht als eigenständige Objekte reifiziert. Als Antwort habe ich gemäß den theoretischen Überlegungen eine spezialisierte Lernumgebung im CAS MuPAD realisiert. Sie wird im nächsten Abschnitt vorgestellt, bevor eine didaktische Bewertung der damit möglichen Interaktionsformen erfolgt.

Die Lernumgebung der Schüler

Die Bearbeitung von Gleichungen kann mittels eines kleinen Programmpakets (beim Autor via Email erhältlich) für das CAS MuPAD im Stil der objektorientierten Programmierung erfolgen. Dazu muss zunächst ein Gleichungssystem erzeugt werden:

```
g:=GSys([x+x+y=4,x*2=4*y]):
```

Über den Namen `g` kann das Gleichungssystem angesprochen werden (das ist wichtig, man kann nämlich mit mehreren Gleichungssystemen gleichzeitig arbeiten (z.B. mit `g2:=GSys([21*x=4-y,x=9*y])`).

Gemäß der Sichtweise der Objektorientierung können die so erzeugten Objekte manipuliert werden, indem ihnen Nachrichten geschickt werden. Diese Nachrichten stellen die Verbindung der autonomen Objekte zur Außenwelt dar, definieren also die Menge der möglichen Operationen. Beispielsweise schickt man zum Sehen dem Objekt `g` die `print`-Nachricht:

```
g::print()
```

$$[2x + y = 4, 2x = 4y]$$

Jetzt wird das System bearbeitet: Multiplikation der zweiten Gleichung mit ¼:

```
g::multipliziere(2,1/4)
```

$$\left[2x + y = 4, \frac{x}{2} = y\right]$$

Analog gestaltet sich die Addition eines Terms zu einer Gleichung:

```
g::addiere(1,-2*x)
```

$$\left[y = -2x + 4, \frac{x}{2} = y\right]$$

Sobald eine der Gleichungen nach einer Variablen aufgelöst ist, kann eingesetzt werden.

```
g::setzeein(1,2)
```

$$\left[y = -2x + 4, \frac{x}{2} = -2x + 4\right]$$

Bestimmte Schritte können beschleunigt werden, indem mit `g::loese(i,x)` eine der Gleichungen nach einer der Variablen aufgelöst wird. Da die Schüler diese Technik schon beherrschen (sollten), wird sie so gekapselt und an den Computer delegiert. Dies ist eine schöne Anwendung der Gerüstdidaktik nach Kutzler.

Recht nützlich ist die Eigenschaft der Gleichungssystem-Objekte, sich an ihre Vergangenheit zu erinnern: Man kann Aktionen rückgängig machen, und man kann sich den Verlauf der ganzen Rechnung anzeigen lassen. Nach den Beobachtungen im Unterricht machen allerdings nur wenige Schüler – dann aber intensiv – von dieser Möglichkeit Gebrauch.

Übersicht über die möglichen Nachrichten

Nachricht	Wirkung
`print()`	schreibe Gleichungssystem
`Rechnung()`	zeige die gesamte bisherige Rechnung an
`zurueck()`	mache den letzten Schritt rückgängig (Reversibilität!)
`addiere(i,a)`	addiere den Term a zur i-ten Gleichung
`multipliziere(i,a)`	multipliziere die i-te Gleichung mit a
`addiereGlg(i,j)`	addiert die i-te Gleichung zur j-ten
`setzeein(i,j)`	setze die i-te Gleichung (muss von der Form „Variable = Term“ sein) in die j-te ein
`vertausche(i)`	vertauscht in der i-ten Gleichung links und rechts
`vertausche(i,j)`	vertauscht die i-te mit der j-ten Gleichung
`loese()`	löse das System in einem Schritt
`loese(i,x)`	löse die i-te Gleichung nach x auf
`zeichne()`	zeichne die Lösungsmengen der einzelnen Gleichungen (nur möglich bei zwei oder drei Variablen)

Gründe für die Ausgestaltung der Lernumgebung

Im Sinne der im Theorieteil erläuterten Analogie zwischen Konzepten der Informatik und kognitiven Konzepten war zentrales Designziel der Lernumgebung, Gleichungssysteme als Ganzes als Objekte zu repräsentieren und über eine Menge von Nachrichten (Operationen) manipulierbar zu machen. Durch diese Grundidee sind bereits die meisten, aber nicht alle Implementationsentscheidungen gefallen. Eine aus didaktischer Sicht besonders diskussionswürdige Ausnahme ist der hier realisierte Identitätsbegriff. Ein Objekt der Klasse `GSys` erlaubt Operationen, bei denen sich das Gleichungssystem durchaus ändert, bei dem aber die Lösungsmenge invariant bleibt. Es repräsentiert also eigentlich eine Äquivalenzklasse von Gleichungssystemen. Aber dieses Repräsentation ist nicht 1:1, denn verschiedene `GSys`-Objekte können die gleiche Äquivalenzklasse realisieren, ohne dass dies durch den

Operator „=“ von MuPAD bemerkt würde (dies ließe sich allerdings mit vertretbarem Aufwand beheben, wenn es sich als sinnvoll erweisen sollte).

Gestaltung des Unterrichtsgangs

Die Erprobung des oben vorgestellten Konzeptes fand in einer neunten Klasse statt, die zunächst fünf Wochen lang Gleichungs- und Ungleichungssysteme in zwei Variablen von Hand gelöst hatte. Dabei lag der Schwerpunkt neben den Lösungsverfahren auf dem Aufstellen der Gleichungen (wobei auch Anwendungssituationen eingestreut wurden, die zu nichtlinearen Gleichungssystemen führen).

Nach der Klassenarbeit wurde eine knapp zweiwöchige CAS-Einheit durchgeführt:

1. Stunde: Kennenlernen von MuPAD mit den objektorientierten Erweiterungen. Nachvollziehen eines Beispiels zum schrittweisen Lösen von Gleichungssystemen
2. Stunde: Gleichungen am Computer schrittweise lösen, Graphen plotten
3. Stunde: Frage: Bei welchen Operationen ändert sich das Geradenbild?
4. Stunde: Drei Gleichungen in zwei Variablen, eine Gleichung in drei Variablen, Bedeutung, Plotten
5. Stunde: Gleichungssystem in drei Variablen: Graphische Interpretation, Erkennen von leicht lösbaren Fällen
6. Stunde: Lösungsstrategie für Systeme von drei Gleichungen in drei Variablen erarbeiten
7. Stunde (im Klassenraum): Lösungsstrategien anwenden

Beobachtungen und Folgerungen

Erwartungsgemäß standen in der ersten Stunde Probleme mit der Syntax im Vordergrund, die aber sehr schnell überwunden wurden (dies lag sicher daran, dass die Schüler vom TI-82 her bereits mit der computerüblichen Termschreibweise vertraut waren). Alle Schüler lernten die Anwendung der Nachrichten kennen und konnten sachgerecht mitarbeiten. Vor allem bei schwächeren Schülern gab es aber bis zum Schluss anhaltende Semantikprobleme. Besonders bei der Neudefinition des Gleichungssystem (also einer erneuten Zeile der Art `g:=GSys(...)`) kam es zu Fehlvorstellungen. So wurde erwartet, dass die alten Gleichungen, da sie ja noch am Bildschirm standen, noch „vorhanden“, also zusammen mit den neuen Gleichungen zu betrachten oder zumindest noch manipulierbar seien.

Der Umgang mit dem CAS half bei der Diagnose von Fehlvorstellungen. So zeigte sich eine mangelnde Integration der Lösungsverfahren in der Frage, warum das System einen Befehl zum Einsetzen, nicht aber zum Gleichsetzen habe.

Haben die Stunden nun dazu beigetragen, Gleichungssysteme als Objekte im „kognitiven Betriebsystem“ der Schüler zu verankern? Zunächst ein Gegenbeispiel: Eine Zweiergruppe wollte noch in der vierten Stunde die Gleichungen am liebsten aus dem System herausnehmen und getrennt bearbeiten. Mehrheitlich gingen die Schüler in ihrer Sprechweise nahtlos zur Objekt-Nachricht-Sprechweise über: Die von den Schülern gefundenen Verfahren zum Lösen von Gleichungssystemen mit drei Variablen wurden im Plenum anhand der Nachrichten-Folge erläutert.

Bei den abschließenden händisch zu bearbeitenden Aufgaben war zu bemerken, dass sich der Anteil derjenigen, die auch auf Papier stets mit vollständigen Gleichungssystemen arbeiteten, erhöht hatte.

Erwähnenswert ist weiter, dass die Fragestellung der dritten Stunde für die Schüler eine echte Herausforderung darstellte, die zu teils lautstarken Meinungsverschiedenheiten führte. Es spricht aber stark für den CAS-Einsatz (egal ob nun objektorientiert oder nicht), dass diese von den Schülern selbst (wohlgemerkt in der dritten Stunde des CAS-Einsatzes!) gelöst werden konnten.

Bewertung von Alternativen

Im Bereich der Computeralgebra stellt die Objektorientierung einen vergleichsweise neuen Ansatz dar. Traditionell wurde ein funktionaler Stil verwendet, in dem die Arbeit mit Gleichungssystemen folgenden typischen Schritt enthalten würde:

```
multGlg([x+4y=5,3x-y=7],2,4) → [x+4y=5,12x-4y=28]
```

Dem funktionalen Paradigma folgend, stellt dabei jedes Ergebnis einen neuen Wert dar, alte Objekte werden nicht verändert. In den alten Derive-Versionen manifestiert sich das besonders deutlich durch die unveränderliche Folge der Objekte #1, #2, #3 … (Derive 5.0 erlaubt hier etwas mehr destruktive Eingriffe, die aber nur auf das Aussehen des Dokumentes, nicht auf die interne Repräsentation wirken). Im funktionalen Ansatz wird also in jedem Schritt aus dem alten (unveränderten) Gleichungssystem ein neues gemacht. Im nächsten Schritt muss dieses in den neuen Befehl kopiert werden (oder jedes Zwischenergebnis hätte benannt werden müssen). Die parallele Arbeit an verschiedenen Gleichungssystemen erfordert deshalb vom Benutzer eine gewisse Disziplin. Das gilt auch dann, wenn die Anwendung der Funktionen durch das System (Beispiel: Derive) erleichtert wird, indem man die Argumente mit der Maus auswählen und die Funktionen in traditioneller Schreibweise angeben kann. Im Lichte der einleitenden Theorie ist hier vor allem die mangelnde Unterstützung der Vorstellung des Gleichungssystems als *ein* Objekt, das manipuliert (umgeformt) wird, zu bemängeln.

Der prozedurale Ansatz wäre durch die Verwendung destruktiver Operationen gekennzeichnet. Diese sind – wenn der Benutzer nicht durch Dokumentation seiner

Schritte und die Kenntnis der inversen Operationen dies umgeht – als solche irreversibel. Vom Standpunkt der Piagetsschen Theorie ist das problematisch.

Die Taschenrechner von Texas Instruments besitzen eine Reihe von Befehlen zur Manipulation von Matrizen, mit denen Gleichungssysteme ebenfalls schrittweise gelöst werden können. Der entscheidende Unterschied ist, dass die Verwendung eines CAS die Matrizenschreibweise vermeidet. Die Schüler können also unmittelbar mit den Gleichungen, die sie aus einem Modellbildungsprozess oder aus der Aufgabenstellung haben, arbeiten. Insbesondere können die Gleichungen nichtlinear sein. Da man m.E. immer wieder Anlass geben sollte, die Anwendbarkeit der Methoden kritisch zu prüfen, ist das ein wichtiger Punkt.

Denkbar – und wünschenswert – wäre es natürlich auch, die Objektorientierung an Mausereignisse zu koppeln, sprich: Man wählt mit der Maus die zu bearbeitende Gleichung innerhalb des Systems aus und sendet ihr eine aus einem Menü gewählte Nachricht. Bei Derive ist das möglich, sofern man nur einzelne Gleichungen (auch als Teil eines Systems) bearbeitet. Bei Gleichungssystemen wäre allerdings ein ständiges Aufschnüren und neues Zusammenpacken der Liste der Gleichungen des Systems nötig. Dies steht im Widerspruch zur zu fördernden Vorstellung, dass es sich um *ein* Gleichungssystem handelt.

Schlussbetrachtung

Die Beobachtungen in der Unterrichtsreihe deuten darauf hin, dass hier ein sinnvoller Ansatz vorliegt. Die empirische Basis ist aber zu dünn, um definitive Antworten geben zu können. Da auch andere CAS-gestützte Unterrichtsgänge zu diesem Thema positive Effekte zeigen, müsste eine empirische Vergleichsuntersuchung auf besonders große Trennschärfe ausgelegt werden. Es wäre aber sicher lohnend, die hier vorgestellte Theorie auch auf andere Themen des Mathematikunterrichts anzuwenden. Beispielsweise lässt sich der große Erfolg dynamischer Geometrieprogramme in Beziehung setzen zu ihrem intrinsisch objektorientierten Ansatz. Damit werden Designziele und Bewertungsmaßstäbe für weitere Softwareentwicklungen (beispielsweise [Oldenburg 2003]) geliefert.

Literatur

H. Abelson & G. J. Sussmann: Struktur und Interpretation von Computerprogrammen, Berlin 1998

R. Baumann: Didaktik der Informatik, Stuttgart 1996

B. Fuchssteiner et al.: MuPAD, http://www.mupad.de

T. Honderich: The Oxford Companion to Philosophy, Oxford 1995

B. Kutzler: Lineare Gleichungssysteme lösen mit dem TI-92, bk teachware, Hagenberg 1998

L. C. Leinbach, J. P. Fink: Growing ideas with Derive – An object-oriented approach to mathematical learning, Vortrag Visit-Me2002, Wien

B. Meyer: Object-oriented software construction, New York 1988

J. Piaget: Einführung in die genetische Erkenntnistheorie, Frankfurt 1973

W. v. Quine: Pursuit of Truth, Cambridge 1992

R. Oldenburg: Feli-X: Ein Prototyp zur Integration von CAS und DGS, Tagungsband des Arbeitskreises „Mathematikunterricht und Informatik“, erscheint 2003

J. Sjuts: Mathematik als Werkzeug zur Wissensrepräsentation, Osnabrück 1999

A. Sfard: On the dual nature of mathematical conceptions, Educational Studies in Mathematics 22 (1991), 1–36

A. Sfard: Symbolizing mathematical reality into beeing, in: P. Cobb et al.: Symbolizing and communication: perspectives on Mathematical discourse, Tools, and Instructional Design, Mahwah, 37–98

A. Sfard: Balancing the Unbalanceable: The NCTM Standards in the Light of Theories of Learning Mathematics, Conference on the Foundations to NCTM Standards (1998)

H. Steinbring: Mathematische Bedeutung als soziale Konstruktion, JMD 21 (2000) S. 28–49

H.-J. Vollrath: Algebra in der Sekundarstufe, Heidelberg 1994

Anschrift des Verfassers

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