Ersch. in C. Kaune, I. Schwank, J. Sjuts (Hg.) Mathematikdidaktik im Wissenschaftsgefüge – Festschrift für Elmar Cohors-Fresenborg (Band 2), pp. 179–193. Osnabrück: FMD (2005)

Computersimulationen in der mathematikdidaktischen Grundlagenforschung*

Ipke Wachsmuth

Kurzfassung: Der Beitrag beschreibt die Technik der Computersimulation auf der Basis von Produktionssystemen als Modellierungsmittel. Am Beispiel einfacher mathematischer Textaufgaben wird gezeigt, wie mit einem Produktionssystem Problemlöseprozesse im Computer modelliert werden können. Der mögliche Beitrag von Computermodellen zur mathematikdidaktischen Grundlagenforschung wird diskutiert, insbesondere, wie sie helfen könnten, durch detailliertes Verständnis fehlerhafter Strategien gezielt auf eine Verbesserung der Unterrichtsmethodik hinzuarbeiten.

Abstract: This article describes the technique of computer simulations that are based on production systems as a means of modeling. Simple mathematical word problems serve to explain how a production system can be used to model problem-solving processes with the computer. The possible contribution of computer models in instructional research is discussed, in particular, how they could help to improve instruction by a detailled understanding of error strategies.

---

* Dieser Artikel, vorveröffentlicht in der Preprint-Reihe der Osnabrücker Schriften zur Mathematik (OSM Heft 82, 1985), war ursprünglich für ein Sonderheft von DER MATHEMATIKUNTERRICHT vorgesehen, wo er dann durch eine Verkettung verschiedener Umstände nicht erscheinen konnte. Er entstand in einer Aufbruchsphase des verstärkten Einbezugs von Methoden der Künstlichen Intelligenz in der Kognitionsforschung, die in die Zeit meiner früheren Tätigkeit in der AG Cohors-Fresenborg an der Universität Osnabrück fiel und wo Themen wie diese gut aufgehoben waren. Mit einem Spektrum diverser Ansätze sind Computersimulationen heute als Instrument der kognitiven Grundlagenforschung in vielen Disziplinen etabliert und auch ein integraler Bestandteil einer empirisch-simulativen Forschungsmethodik des seit vielen Jahren von mir in Bielefeld mitgetragenen DFG-Sonderforschungsbereiches. Wenn der Artikel jetzt, 20 Jahre später, in einer Festschrift aus Anlass des 60. Geburtstages von Elmar Cohors-Fresenborg erscheint, so sind auch in der mathematikdidaktischen Grundlagenforschung zwischenzeitlich vielfältige Fortschritte des technischen Instrumentariums zur Untersuchung und Modellierung kognitiver Prozesse zu verzeichnen. Ich habe deshalb darauf verzichtet, den Artikel auf einen aktuellen Stand zu bringen, und lege ihn, von einigen Druckfehlern bereinigt, so vor, wie er damals geschrieben wurde. – *IW*

1. Einleitung

In den neueren Ansätzen der kognitiven Psychologie wird, ausgehend von der weithin beachteten Arbeit von Newell und Simon (1972), Problemlösen unter dem Gesichtspunkt der Informationsverarbeitung analysiert ("information processing approach"). Im Gegensatz zu früheren Ansätzen der psychologischen Erklärung von Problemlöseprozessen liegt der wesentliche Vorteil dieses Ansatzes in der detaillierten Analyse von Problemlöseprozessen. Er ist aus der Künstliche-Intelligenz-Forschung hervorgegangen, deren theoretisches Ziel es ist, die Prinzipien intelligenten Verhaltens so genau zu verstehen, dass dieses durch Simulation im Computer nachvollzogen werden kann. Dabei wird im psychologisch ausgerichteten Ansatz gefordert, dass nicht nur das *Produkt*, also das gezeigte "intelligente" Verhalten, sondern auch die *Prozesse*, auf denen es beruht, mit denen der beim Menschen beobachtbaren in Übereinstimmung gebracht werden können. Soweit menschliche Intelligenz auf formaler Grundlage erklärt werden kann, könnte "künstliche Intelligenz" die präziseste Form eines psychologischen Modells liefern.

Für die mathematikdidaktische Grundlagenforschung ist dieser Ansatz insofern von Interesse, als das Lösen von Problemen ein wichtiges Thema des Mathematikunterrichts ist. Aufschlüsse über die vom Schüler bei der Bewältigung von Sachaufgaben und Problemen zu erbringenden geistigen Prozesse können die Entwicklung von Problemlösefertigkeiten und die Ursache von fehlerhaften Strategien erklären helfen.

Der erste Schritt zur Lösung eines Problems ist, es zu verstehen. Newell und Simon haben "Verstehen" dabei definiert als das Erstellen eines geistigen Modells ("Repräsentation") der Problemsituation, das der Problemlöser konstruiert und manipuliert, um eine Lösung zu finden. Aus der Forschung der letzten Jahre ergeben sich deutliche Hinweise darauf, dass die Problemrepräsentation eine kritische Komponente für Problemlösefertigkeiten ist. Auf der Grundlage von Protokollanalysen aus Beobachtungen an menschlichen Problemlösern haben erstmalig Newell und Simon (1972) Problemlöseprozesse im Computer simuliert.

In einer Reihe von Ansätzen der Modellierung kognitiver Prozesse in Computern werden so genannte Produktionssysteme verwendet. Dort werden die Handlungsmöglichkeiten des Individuums in Gestalt bestimmter Regeln beschrieben, die ausdrücken, was das Individuum unter welchen Bedingungen tun kann. Die Prämisse solchen Vorgehens ist es, dass jede solche Regel ("Produktion") einem "Stück" Wissen entspricht und dass die kognitiven Fähigkeiten des Menschen auf der Interaktion einer großen Anzahl solcher "Stücke" beruhen.

Die Verwendung von Produktionssystemen als Beschreibungsmittel ist hierbei nicht zwingend (letztlich werden die Produktionen im Computer auf einfachere Maschinenbefehle übertragen). Sie erfreut sich jedoch einer Reihe von Vorzügen, die vermutlich in ihrer strukturellen Ähnlichkeit mit einer Vorstellung vom Handeln als Abfolge bedingter Einzelprozesse begründet sind. Als Programme sind Produktionssysteme leicht modifizierbar; das Hinzufügen oder Wegnehmen von Produktionen beeinträchtigt i.a. nicht die Lauffähigkeit des Programms, kann jedoch sein Verhalten drastisch ändern. Es ist technisch möglich, *selbst-*

modifizierende Produktionssysteme zu erstellen, die bei der Modellierung von Lernprozessen eine wichtige Rolle spielen.

Um dem interessierten Leser einen Einblick in das Gebiet der Computersimulation zu geben, soll in den folgenden Abschnitten diese Technik und ihre Anwendung in der mathematikdidaktischen Grundlagenforschung an einem Beispiel verdeutlicht werden. Dazu wird zunächst eine Programmiersprache für Produktionssysteme in Grundzügen vorgestellt, die in der Künstlichen Intelligenz und der kognitiven Psychologie sowie bei der Erstellung von sog. Expertensystemen zum Einsatz gelangt. Im Anschluss wird beschrieben, wie mit dieser Sprache Problemlöseprozesse im Computer modelliert werden können, wobei wegen der besseren Überschaubarkeit einfache Textaufgaben zu Addition und Subtraktion gewählt wurden. Schließlich wird ein Ausblick auf weitere Entwicklungen und den möglichen Beitrag dieses Gebiets zur Mathematikdidaktik gegeben.

2. Produktionssysteme als Mittel der Modellierung

Ein *Produktionssystem* ist ein Programm, das ausschließlich aus bedingten Aussagen, den *Produktionen*, besteht. Diese operieren auf Ausdrücken, die in einem Arbeitsspeicher (working memory) verfügbar sind. Die Produktionen selbst sind als ungeordnete Liste in einem separaten Speicher, dem Produktionenspeicher (production memory) abgelegt. Eine *Produktion* ist ein Bedingung-Handlungs-(condition-action)Paar der Form

$$C_1\ C_2\ \ldots\ C_n \rightarrow A_1\ A_2\ \ldots\ A_m$$

und bedeutet: Wenn die Bedingungen $C_1\ C_2\ \ldots\ C_n$ durch Ausdrücke im Arbeitsspeicher erfüllt sind, dann können die Aktionen $A_1\ A_2\ \ldots\ A_m$ ausgeführt werden. Üblich ist die schematische Darstellung einer Produktion als LHS → RHS, wobei mit LHS die linke Seite (left hand side), also die Gesamtheit der Bedingungen, und mit RHS die rechte (right hand side), also die Gesamtheit der Handlungen zusammengefasst wird.

Stellen wir uns jetzt vor, dass ein Computer ein derartiges Produktionssystem (im Produktionenspeicher) gespeichert hat und dass auch der Arbeitsspeicher Elemente enthält, die sich als Gegenstand der Anwendung von Produktionen eignen. "Sich eignen" heißt dabei, dass sie von der Form her einzelne Bedingungen von Produktionen erfüllen können. Dann können bei Erfülltsein aller Bedingungen einer Produktion (also ihrer LHS) die Handlungen (ihrer RHS) ausgeführt werden. Die Handlungen wirken beispielsweise auf Inhalte des Arbeitsspeichers, wodurch sich für den nächsten Schritt veränderte Bedingungen ergeben können; andere Produktionen werden möglicherweise anwendbar usf. Falls zu einem Zeitpunkt mehrere Produktionen mit erfüllter LHS vorliegen, so muss zunächst der Konflikt geeignet gelöst werden, welche Produktion dann angewendet werden soll (Einzelheiten siehe bei Forgy, 1981). Dieser

gesamte Prozess wird durch den sog. *Interpreter* abgewickelt, der in Zyklen arbeitet. Ein Zyklus ("recognize-act cycle") besteht aus den folgenden Schritten:

1. Prüfen, welche Bedingungsseiten (LHS) durch den Inhalt des Arbeitsspeichers erfüllt sind (das sog. *pattern matching*);
2. Auswählen einer Produktion mit erfüllter LHS (*conflict resolution*); falls keine LHS erfüllt ist: STOP;
3. Ausführen aller Aktionen (RHS) der gewählten Produktion (*act*);
4. Zurück zu Schritt 1.

Der Produktionenspeicher, auch "Produktionengedächtnis", wird mit dem Langzeitgedächtnis verglichen und enthält eine unlimitierte Anzahl von Produktionen in beliebiger und unstrukturierter Reihenfolge (es gibt keine Unterprozedurtechnik). Bis auf ggfs. zunächst aufzulösende Konflikte kann eine Produktion angewendet werden ("feuern"), sowie alle ihre Bedingungen simultan durch Arbeitsspeicherelemente erfüllt sind.

Im Vergleich zum Produktionenspeicher ist der Arbeitsspeicher i.a. relativ klein; er enthält eine unstrukturierte Sammlung von Elementen, die man als Objekte der Informationsverarbeitung durch das Produktionssystem auffassen kann. Häufig wird hierin eine Entsprechung zum menschlichen Kurzzeitgedächtnis gesehen; man spricht auch von "Arbeitsgedächtnis".

Zunächst soll jetzt konkretisiert werden, in welcher Weise sich Objekte (genauer: Modelle von Objekten der Welt) mit einer Programmiersprache für Produktionssysteme im Arbeitsspeicher eines Computers repräsentieren und manipulieren lassen. Diese Sprache heißt OPS5 (Forgy, 1981) und ist als "Komfortebene" verschiedener LISP-Systeme verfügbar. Die Beispiele hier stammen von den historisch bedeutsamen sog. Block-Welten, in denen man erstmals versucht hat, Lagebeziehungen und Bewegungen einfacher Objekte ("Blöcke") mit dem Computer zu simulieren.

Zur Beschreibung von Objekten und Relationen zwischen Objekten wird in OPS5 eine sog. *Attribut-Wert*-Repräsentation benutzt: Jedes Arbeitsspeicherelement ist als Objekt durch eine Beschreibung zugeordneter Attribut-Wert-Paare definiert. Zum Beispiel bezeichnet das folgende Element ein Objekt der Klasse "Block", das "Block1" heißt, rot ist, eine Masse von 500g hat und an jeder Seite 100mm misst:

```
(block
      ↑name      block1
      ↑color     red
      ↑mass      500
      ↑length    100
      ↑width     100
      ↑height    100)
```

Die Attribute sind hierbei mit ↑ gekennzeichnet, daneben stehen jeweils ihre Werte.

Die im Arbeitsspeicher repräsentierten Elemente müssen nicht notwendig gegenständlicher Natur sein wie der Block im obigen Beispiel. Es kann sich dabei auch um so etwas wie eine *Intention* handeln, etwa die Absicht, einen roten Block zu besorgen. (Man nimmt an, dass auch eine Intention im Kurzzeitgedächtnis eines Menschen gespeichert sein muss, und zwar als ein zu erreichendes Ziel, das solange "aktiv" bleibt, bis es erreicht ist). In OPS5 kann man solch ein Ziel wie folgt im Arbeitsspeicher repräsentieren:

```
(goal
      ↑status      active
      ↑type        find
      ↑object      block
      ↑color       red)
```

Zu Aufbau und Wirkungsweise der Produktionen: Die *Bedingungsseite* (LHS) einer Produktion LHS → RHS besteht aus einer Kollektion von Bedingungselementen. Beim "pattern matching" (s.o.) prüft der Interpreter jede Produktion, ob sie anwendbar ist, d.h. ob zu allen ihren Bedingungselementen passende Elemente im Arbeitsspeicher vorhanden sind. Hierbei bedeutet "Passen" nicht ein völliges Übereinstimmen: Die in der LHS auftretenden Pattern können auch *abstrakte* Repräsentationen von Arbeitsspeicherelementen sein. Man unterscheidet dabei zwei Arten der Abstraktion. Im einen Fall kann die LHS schlicht weniger Attribut-Wert-Paare enthalten als ein Element im Arbeitsspeicher, z.B. passt die Bedingung

```
(block
      ↑color       red)
```

(u.a.) auf den individuellen Block

```
(block
      ↑name        block1
      ↑color       red
      ↑mass        500
      ↑length      100
      ↑width       100
      ↑height      100)
```

D.h. wird in der LHS einer Produktion ein roter Block erwartet, so wäre diese Bedingung durch den "konkreten" Block1 im Arbeitsspeicher erfüllt; in diesem Sinne wird in der Produktion "abstrakt" nach (irgend)einem roten Block gefragt. Im anderen Fall beruhen abstrakte Repräsentationen darauf, dass Werte von Attributen unvollständig spezifiziert werden, indem Variable (Schreibweise: <x>) verwendet werden; so passt das Bedingungselement

```
(block
      ↑length      <x>
      ↑width       <x>
      ↑height      <x>)
```

ausschließlich auf Blöcke, die Würfel sind (z.B. den obigen Block1).

Die *Handlungsseite* (RHS) einer Produktion LHS → RHS enthält Anweisungen, die bei ihrem "Feuern" vom Interpreter ausgeführt werden. Die Handlungen können Objekte im Arbeitsspeicher manipulieren, die Ein- und Ausgabe zwischen Computer und Benutzer bewerkstelligen oder auch neue Produktionen in den Produktionenspeicher einordnen ("Lernen"). Mit der Handlung *make* können beispielsweise neue Elemente im Arbeitsspeicher erzeugt werden; so fügt

```
(make block
      ↑name      block1
      ↑color     blue
      ↑mass      1000
      ↑length    120
      ↑width     120
      ↑height    300)
```

im Arbeitsspeicher die Repräsentation einer blauen quadratischen Säule von 1000g Masse hinzu. Entsprechend können mit der Handlung *remove* Elemente aus dem Arbeitsspeicher entfernt und mit der Handlung *modify* Attributwerte von Arbeitsspeicherelementen geändert werden. Beispiel:

```
(modify 1
      ↑status    satisfied)
```

ändert das Attribut "Status" eines im 1. Bedingungselement der LHS beschriebenen Arbeitsspeicherelements auf "satisfied" ("erfüllt").

Es folgt ein einfaches Beispiel für eine Produktion in OPS5. Den Kommentaren (jeweils rechts neben dem Semikolon) entnimmt man folgendes: Diese Produktion hat das Ziel, im Arbeitsspeicher einen farbigen Block zu finden. Wenn sie einen solchen findet, markiert sie ihn und kennzeichnet dann das auf sie passende Ziel im Arbeitsspeicher als erfüllt (wodurch die Produktion im nächsten Schritt nicht mehr anwendbar ist, sofern kein weiteres solches Ziel aktiv ist):

```
(p  find-colored-block                  ; (Name der Produktion)
      (goal                             ; Falls es ein Ziel gibt
            ↑status     active          ; das aktiviert ist
            ↑type       find            ; zum Finden
            ↑object     block           ; eines Blocks
            ↑color      <z>)            ; einer bestimmten Farbe
      (block                            ; und es gibt einen Block
            ↑color      <z>             ; dieser Farbe
            ↑name       <block>         ; eines bestimmten Namens
→                                       ; dann
```

```
(make result                        ; erzeuge ein Element als Resultat
      ↑pointer      <block>)        ; das auf benannten Block weist
(modify 1                           ; und ändere das Ziel
      ↑status       satisfied))     ; auf "erfüllt"
```

Im nächsten Abschnitt wird ein Computermodell vorgestellt, das die oben erläuterten Techniken verwendet, um die kognitiven Prozesse von Kindern beim Lösen einfacher mathematischer Textaufgaben zu simulieren.

3. Die Technik der Computersimulation von Problemlöseprozessen am Beispiel des CHIPS-Modells von Briars und Larkin

In einem Ansatz für eine "vereinheitlichte Theorie darüber, wie Leute Mathematik benutzen", unternimmt Larkin (1983) den Versuch, theoretisch zu fassen, wie der Mensch mathematisches Wissen in anfangs nicht-mathematischen Situationen (z.B. in Textaufgaben oder in naturwissenschaftlichen Problemen) anwendet. Ein von ihr entworfenes Problemlösemodell schreibt erfolgreichen Problemlösern das folgende schrittweise Vorgehen zu: Zunächst wird eine *Basisrepräsentation* des Problems im Gedächtnis des Problemlösers erstellt; sie besteht aus geistigen Objekten und Beziehungen dazwischen, die direkt zur im Text beschriebenen Situation korrespondieren. In der nächsten Phase wird diese Basisrepräsentation durch Hinzufügen weiterer Objekte und Beziehungen vornehmlich mathematischer Natur angereichert, um eine *mathematische Repräsentation* des Problems zu erhalten. Aus dieser ergibt sich schließlich eine *Berechnungsrepräsentation*, die die Lösung des Problems gestattet. Nach Larkins Ansicht liegt eine gemeinsame Ursache von Misserfolgen beim mathematischen Problemlösen in den verschiedensten Bereichen darin, dass eine Berechnung "zu früh", nämlich direkt von der Basisrepräsentation aus versucht wird.

Für verschiedene Problembereiche sind an der Carnegie-Mellon University Computer-Modelle entwickelt worden, die durch die unterschiedliche Qualität der erzeugten Repräsentationen verschiedene Grade von Problemlösefähigkeit simulieren. Solche Computer-Modelle ermöglichen Vorhersagen über eine Staffelung in der Problemschwierigkeit und über zu erwartende Fehlermuster. In einigen Fällen wurden empirische Studien herangezogen, um die Vorhersageleistung der Modelle zu erproben.

Ein Computer-Modell, das Textaufgaben zu Addition und Subtraktion lösen kann, heißt CHIPS ("Concrete Human-like Inferential Problem Solver", Briars & Larkin, 1984). Als "Minimalmodell" enthält CHIPS die kleinste Menge Wissen, die als ausreichend zum Lösen elementarer Textaufgaben angesehen werden kann. Für eine große Vielfalt solcher Aufgaben produziert CHIPS Antworten (richtige und falsche) und eine Folge von Schritten, die die "Lösungsstrategie" angibt. Eine wesentliche Annahme des Modells ist es, dass Kinder derartige Problemsituationen konkret oder im Geiste mit Hilfe von Zählobjekten ("Chips") repräsentieren. Das "Problemverständnis" äußert sich in der Qualität der Repräsentation. Ver-

schiedene Auslegungen des CHIPS-Computermodells bilden unterschiedlich weit entwickelte Fähigkeiten zum Verständnis von Problemsachverhalten ab. Die Arbeitsweise des Modells soll an einem Beispiel verdeutlicht werden.

Ein Problem, das mit der einfachsten Auslegung des CHIPS-Modells gelöst werden kann, ist die folgende Aufgabe: "Joe hat 5 Murmeln. Tom gibt ihm 3 weitere Murmeln dazu. Wie viele Murmeln hat Joe jetzt?" Nachstehend ist das bei der Lösung dieser Aufgabe ausgedruckte Protokoll wiedergegeben (hinzugefügt wurde eine Zeilennummerierung):

```
1.  pl(Joe had 5 marbles Then Tom gave-him 3 more marbles
2.  How-many marbles does Joe have now)
3.      Joe
4.      had
5.  There is a set
6.      5
7.  move until the total set is 5
8.  1 chips have been moved
9.  2 chips have been moved
10. 3 chips have been moved
11. 4 chips have been moved
12. 5 chips have been moved
13. enough chips have been moved
14. now 5 chips in set
15.     marbles
16. set of marbles
17.     Tom
18.     gave-him
19. move total amount into set
20.     3
21. move until the total set is 3
22. 1 chips have been moved
23. 2 chips have been moved
24. 3 chips have been moved
25. enough chips have been moved
26. now 8 chips in set
27.     more
28. more marbles
29.     marbles
30. set of marbles
31.     How-many
32.     marbles
33. count set of marbles
```

34. 1
35. 2
36. 3
37. 4
38. 5
39. 6
40. 7
41. 8
42. set counted 8

Zur Erläuterung: Das dem Computer übergebene Problem (Zeilen 1 und 2) wird Wort für Wort eingelesen (im Protokoll eingerückt; Mitteilungen des Systems sind nicht eingerückt). Das Wort "hatte" (Zeile 4) ist signifikant für eine Produktion, die auf das Vorhandensein einer Menge zählbarer Objekte schließt: Im Protokoll wird mitgeteilt, dass eine Menge im Arbeitsspeicher repräsentiert und ein "move-Schema" angelegt wurde, welches eine bislang unspezifizierte Anzahl von Zählobjekten in diese Menge bewegen kann (Zeile 5). Das jetzt eingelesene Zeichen "5" ist signifikant für eine andere Produktion, die im move-Schema die Objektanzahl spezifiziert (Zeile 7). Darauf kann eine weitere Produktion fünfmal angewendet werden (Zeilen 8-12), bis die Stopp-Bedingung des move-Schemas erfüllt ist und die Menge 5 Zählobjekte enthält (Zeilen 13-14).

Das Einlesen des Problemtextes wird fortgesetzt (Zeile 15), worauf eine Produktion die Repräsentation der Menge durch eine Typ-Beschreibung der Zählobjekte ("Murmeln") ergänzt und dies mitteilt (Zeile 16). Das nächste signifikante Wort "gab-ihm" (Zeile 18) führt dazu, dass ähnlich wie vorher 3 weitere Zählobjekte *in dieselbe Menge* bewegt werden, wodurch die Menge nun 8 Zählobjekte enthält (Zeilen 21-26). Wichtig ist hier, dass dies schon geschieht, *bevor* überhaupt bekannt ist, dass es sich wieder um Murmeln handelt. Erst jetzt wird "more" eingelesen (Zeile 27), was von einer Produktion wiederum als Hinweis auf weitere Murmeln (Zeile 28) aufgefasst wird; beim Lesen des Worts "Murmeln" wird die Typ-Beschreibung der Zählobjekte nur wiederholt.

Das Wort "Wie-viele", in Verbindung mit dem wiederum als Typ-Beschreibung aufgefassten Wort "Murmeln" (Zeilen 31-32), führt schließlich dazu, dass passende Produktionen ein "Zählschema" anlegen und die Objekte der Menge des Typs "Menge von Murmeln" abzählen (Zeilen 33-42). Auch hierbei wird diese Handlung bereits durchgeführt, bevor die letzten Worte des Problemtextes überhaupt eingelesen werden.

Es verwirrt hier vielleicht die Tatsache, dass die Anzahl der Objekte schon in Zeile 26 genannt, die Menge, der Zählobjekte jedoch später noch abgezählt wird. Bei dieser einfachsten Version des CHIPS-Modells ist aber überhaupt nur *eine* Menge im Spiel, und diese wird schließlich abgezählt. In den komplexeren Versionen kann es durchaus der Fall sein, dass mehrere Mengen vorkommen und eine bestimmte davon (z.B. eine Menge, die Teilmenge einer anderen ist) am Schluss zur Beantwortung der Frage "Wie-viele" abgezählt werden muss.

Briars und Larkin haben drei Modellversionen gestaffelter Komplexität entwickelt, die wie folgt konzipiert sind:

I. Das Modell CHIPSsr ("single-role counters")

CHIPSsr kann den Zählobjekten (counters) einer Menge nur eine einzige "Rolle" zuschreiben. Damit können Aufgaben des Typs "A + B = ?" gelöst werden wie die oben behandelte:

Joe hat 5 Murmeln. Tom gibt ihm 3 weitere Murmeln dazu. Wie viele Murmeln hat Joe jetzt?

CHIPSsr löst jedoch nicht Aufgaben des Typs "A + ? = B" , wie:

Jerry hat 3 Bonbons. Er bekommt einige dazu. Nun hat er 8. Wie viele hat er dazubekommen?

In diesem Beispiel würde CHIPSsr "mangelndes Verständnis" insofern zeigen, als die besitzanzeigenden Relationen "gehört-jetzt" und "gehörte-ursprünglich" in der Problemrepräsentation nicht unterschieden werden. Dadurch wird eine auf dem Abzählen der Differenzmenge beruhende Lösung unmöglich. Genau dieses kann von der nächsten Version des Modells zusätzlich geleistet werden.

II. Das Modell CHIPSdr ("double-role counters")

CHIPSdr kann den Zählobjekten einer Menge zwei verschiedene Rollen attribuieren. Damit gelingt die Lösung von Aufgaben der beiden Typen "A + B = ?" und "A + ? = B", also insbesondere des zweiten obigen Beispiels. Aber auch CHIPSdr kann einen Aufgabentyp, nämlich "? + A = B", nicht lösen:

Sue hat einige Karussellkarten. Sie bekommt noch 5 dazu. Nun hat sie 8. Wie viele hatte sie anfangs?

Hier scheitert die Lösung daran, dass das Wort "einige" nicht repräsentiert werden kann (genauer: die Objektanzahl im zuerst angelegten move-Schema kann nicht spezifiziert werden). Eine weitere Version des Modells kann jedoch auch diesen Problemtyp bewältigen:

III. Das Modell CHIPSrr ("re-representation")

Diese Version enthält leistungsfähige Produktionen, die einen Wechsel der Problemrepräsentation zulassen dergestalt, dass seine konkrete Abarbeitung (mittels counters) möglich wird. Dazu gibt es zwei Verfahren: das *Zeitumkehr-Schema* und das *Menge/Teilmenge-Schema*. Das Zeitumkehr-Schema verfährt in folgender Weise: Es sammelt Informationen über eine unbekannte erste Menge, eine dieser Menge zugefügte Veränderung und eine Zielmenge. Unter Umkehr der Zeitrichtung wird das Problem dann quasi "rückwärts" gelöst dadurch, dass die (bekannte) Zielmenge die inverse Änderung erfährt. Das Menge/Teilmenge-Schema sammelt Information über eine Gesamtmenge (von 8) mit einer Teilmenge (von 5). Das Problem wird dann gelöst durch Abtrennen einer Menge mit 5 Elementen von der Gesamtmenge.

Auf Vorhersagen des CHIPS-Modells kann hier nur andeutungsweise eingegangen werden. Briars und Larkin thematisieren zwei unabhängige Quellen von Problemschwierigkeiten: Die mathematischen Anforderungen und die im Problemtext verwendete Sprache. Betreffs der mathematischen Anforderungen lassen sich drei den verschiedenen Modellversionen entsprechende Problemklassen unterscheiden. In Bezug auf die verwendete Sprache unterscheiden Briars und Larkin mehrere Typen, z.B. danach, ob direkte Handlungshinweise (wie "gave-him") vorhanden sind oder nicht; fehlen sie, so würde eine erfolgreiche Problemrepräsentation zunächst zusätzliches Wissen über die Zusammenhänge erfordern. (Nicht für alle Sprachtypen wurden tatsächlich Modellversionen programmiert.) Auch anhand dieser Unterscheidungen werden, bezogen auf das Modell, bestimmte Staffelungen von Problemklassen vorhergesagt.

Außerdem haben Briars und Larkin von den verschiedenen Versionen des Computer-Modells auch damit nicht lösbare Probleme bearbeiten lassen und dabei Protokolle und Fehler registriert. Auf diese Weise ermöglicht das CHIPS-Modell Vorhersagen über *Fehlerstrategien*. Ferner kann die *Entwicklung von Zählstrategien* beim Rechnen, die beim Addieren z.B. vom Abzählen beider Mengen zum Weiterzählen von der ersten Menge aus fortschreitet usw., im Rahmen der Modellhierarchie diskutiert werden.

Um ihr Modell zu erproben, haben Briars und Larkin eine ganze Reihe von empirischen Studien fremden und eigenen Ursprungs zum Lösen von Textaufgaben vergleichend interpretiert. Dabei bezogen sich die untersuchten Variablen auf mathematische Anforderungen, die verwendete Sprache, das Alter der Schüler, das Vorhandensein oder Nichtvorhandensein von Zählchips und die Zahlengröße; betreffs Einzelheiten muss hier auf die Originalarbeit verwiesen werden. Beim Vergleich von Modellvorhersagen und Beobachtungen zeigte sich die Brauchbarkeit des CHIPS-Modells in vielen Fällen. Briars und Larkin behaupten nicht, dass ihr Modell in der vorliegenden Fassung die Daten vollständig erklären kann.

4. Ausblick

Wie aus dem Vorangegangenen deutlich geworden sein mag, richtet sich die Technik der Computersimulation auf eine präzise Beschreibung von Denkmechanismen. Damit könnte sie im Erklärungswert einer statistischen Feststellung von Zusammenhängen zwischen Untersuchungsvariablen weit überlegen sein; zudem ergeben sich Hinweise auf kritische Abschnitte in der Problembearbeitung. Das Computermodell CHIPS kann dem Lehrer eine höhere Sensibilität verschaffen für die im Denken des Kindes bei der Bearbeitung von Textaufgaben ablaufenden (oder möglicherweise nicht ablaufenden) Vorgänge. Die Staffelung verschiedener Modellversionen verweist dabei auf Differenzierungen im Denken, die eine notwendige Grundlage für die Bewältigung komplexer Textaufgaben zu bilden scheinen. Insbesondere das Modell des Problemlösens durch Repräsentationswechsel – präzisiert durch die Simulationen im Computer – scheint ein hilfreicher Ansatzpunkt, um gezielt auf eine Verbesserung

der Leistungen bei Textproblemen hinzuarbeiten. Als wesentliches Element der Problembearbeitung schält sich dabei der Übergang vom anfänglichen Problemverständnis zu einer mathematischen Repräsentation heraus, an der sich die Berechnung der Lösung orientiert.

Eigene Arbeiten des Verfassers richten sich auf eine andere Problematik des Mathematikunterrichts: Die Instabilität von Schülerleistungen in mathematischen Anwendungssituationen. Häufig wird festgestellt, dass die Standardaufgaben eines Stoffgebiets zwar beherrscht werden, dass der Schüler aber in Kontexten versagt, in denen das erworbene Wissen bisher nicht eingesetzt wurde, obwohl das notwendige Wissen eigentlich zur Verfügung stehen sollte. Die Erforschung der Ursachen solchen Fehlverhaltens soll Aufschluss über Lehraktivitäten erbringen, die zu einer breiten, situationsunabhängigen Anwendbarkeit des Schülerwissens führen. Gestützt auf Daten aus empirischen Untersuchungen werden zu diesem Zweck Computersimulationen durchgeführt, die eine Erklärung der beobachteten Fehlverhalten erlauben und damit eine Grundlage zur Verbesserung von Lehrstrategien bilden könnten. Eine einführende Darstellung dazu liegt vor (Wachsmuth, 1985). Die Zukunft wird zeigen, inwieweit Computersimulationen in der mathematikdidaktischen Grundlagenforschung einen nützlichen Beitrag zur Verbesserung des Mathematikunterrichts leisten können.

Literatur

Briars, D. J. & Larkin, J. H. (1984). An integrated model of skill in solving elementary word problems. *Cognition and Instruction 1*(3), 245-296.

Forgy, C. L. (1981). *OPS5 user's manual* (Tech. Report). Computer Science Department, Carnegie-Mellon University, Pittsburgh.

Larkin, J. (1983, June). *Working towards a unified theory of how people use mathematics* (A.P.C.12). Department of Psychology, Carnegie-Mellon University. Eingeladener Vortrag auf der Jahrestagung der American Educational Research Association, Montreal.

Newell, A. & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, New Jersey: Prentice Hall.

Wachsmuth, I. (1985). LAKOS – ein Modell der Wissensrepräsentation zur Erklärung kognitiven Verhaltens. In H. Mandl & P. M. Fischer (Hrsg.): *Lernen im Dialog mit dem Computer (pp. 24-39)*. München: Urban und Schwarzenberg.