HANDLUNGSORIENTIERTE LERNUMGEBUNGEN UND KOGNITIVE STRATEGIEN BEIM PROGRAMMIEREN

E. Cohors-Fresenborg, C. Kaune, Osnabrück

Inhalt:

1. Einleitung

Lange Zeit standen bei Fragen des Einsatzes von Computern in Ausbildung und Beruf Entscheidungen über die Gerätekonfiguration und die für Problemlösungen geeignete *Sprache* im Vordergrund des Interesses, dagegen traten Überlegungen zur kognitiven und affektiven Disposition des Benutzers beim Umgang mit den Maschinen oder der Software in den Hintergrund. So konnte KLING [6] feststellen, daß es keine Untersuchungen über kognitive Aspekte des Programmierens gibt. Auch VAN MUYLWIJK,VAN DER VEER [10] stellen noch ein erhebliches Defizit fest.
Erst in den letzten Jahren ist dieser Problemkreis auf dann rasch zunehmendes Interesse gestoßen, wie z.B. das Buch von CARD, MORAN, NEWELL [1] zeigt. Während in diesem mehr kognitive Strategien bei der Software-Benutzung untersucht werden, behandeln die Forschungen von MOLZBERGER [7] den Problembereich unterschiedlicher Vorstellungswelten, in denen Spitzenprogrammierer ihre Leistungen vollbringen.

Im Gegensatz zu diesen von ihrem Interesse her auf Problemlösungen in der Berufswelt zielenden Untersuchungen sind uns im Bereich Schule bis heute im deutschsprachigen Raum keine Untersuchungen bekannt, die sich mit ähnlichen Fragen bezüglich der kognitiven Organisation und der bevorzugten Denkstrategien von Schülern beim Umgang mit Algorithmen beschäftigen.

Die von uns durchgeführten Untersuchungen mit Schülern (an Gymnasien der 7. Klasse) haben zwei Schwerpunkte:

(1) Bei der Entwicklung und Erprobung geeigneter Lernumgebungen hat sich gezeigt, daß es Möglichkeiten einer nonverbalen (d.h. auch nicht programmiersprachlich) gebundenen Form der Repräsentation von Algorithmen gibt, welche frühzeitig ohne jegliche Vorkenntnisse die Auseinandersetzung mit der Konstruktion und Analyse von Algorithmen ermöglicht. Solche nonverbalen Formen lassen sich als Vorstufen für die programmiersprachliche Realisierung von Algorithmen zu einem grundlegenden Verständnis nutzen.

(2) Bei unseren mit diesem Material durchgeführten kognitionspsychologischen Untersuchungen hat sich gezeigt, daß es individuelle Unterschiede in der Leistungsdimension bei Anforderungen des Konstruierens von neuen bzw. Analysierens von vorgegebenen Algorithmen gibt. Außerdem haben wir eine Alternative zu der vielerorts propagierten begrifflich-strukturellen Vorgehensweise beim Konstruieren von Algorithmen gefunden, mit der auch hohe Leistungen erzielt werden können.

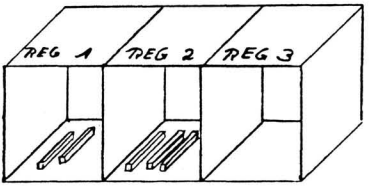
Die von uns im Laufe unserer Untersuchungen gemachten Beobachtungen und theoretischen Erklärungen greifen weit über den Bereich von Schule hinaus und betreffen grundlegende Fragen der Begriffsrepräsentation und Problemlösestrategie.

2. Verschiedene Repräsentationen von Algorithmen

Unsere grundlegende These ist, daß das Kernproblem beim Programmieren in der *Organisation* von Handlungssequenzen liegt, die ein Computer auszuführen hat. Folgt man dieser These, so stellt sich die Frage, welche Bedeutung die *Sprache* -einmal als Umgangssprache zur Strukturierung von Problemen und einmal als formale Programmiersprache zur Darstellung eines gefundenen Algorithmus- beim Zustandekommen und Arbeiten mit Algorithmen hat. Für unsere Untersuchungen suchten wir deshalb nach einer Möglichkeit, Algorithmen als Sequenzen einfacher Handlungen oder in ikonischer Form zu repräsentieren. Wir wählten dafür folgenden Ansatz (vgl. [2]):
Wir beschränken uns in der Datenstruktur auf natürliche Zahlen, die Elementaroperationen des Prozessors sollen Vorwärts- und Rückwärtszählen sein. Die Kontrollstrukturen bestehen aus der Hintereinanderausführung von Teilalgorithmen und der Iteration (solange ein Zahlspeicher nicht leer ist) von Teilalgorithmen.

2.1 Hantieren mit Stäbchen

Die Elementaroperationen des Vorwärts- und Rückwärtszählens mit natürlichen Zahlen lassen sich in besonders anschaulicher Weise realisieren, wenn man die natürlichen Zahlen durch einen Haufen von farbigen Stäbchen (Streichhölzer) darstellt, die man in Fächer entsprechender Farbe einer sog. Registerkiste (s. Abb.) legt. Dann bestehen die Elementaroperationen aus dem Hinzufügen bzw. Wegnehmen eines einzelnen Stäbchens in einem entsprechenden Fach.



Ein Algorithmus für die *Addition* zweier natürlicher Zahlen, die in den Registern R_1 und R_2 gegeben sind, lautet dann folgendermaßen:

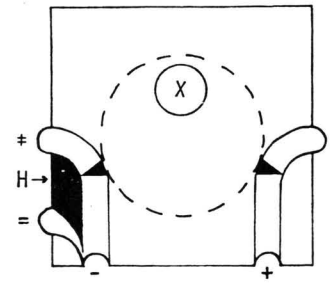
Wiederhole solange, bis in R_2 kein Stäbchen mehr liegt:
Nimm aus Register 2 ein Stäbchen weg;
füge in Register 1 ein Stäbchen hinzu.

Diese Realisierung von Datenspeicher und Prozessor hat folgende Vorteile:
Die Vorstellung von Variablen wird dadurch unterstrichen, daß der Versuchsleiter vor den Augen der Versuchsperson mit der Hand einen Haufen Stäbchen ergreift und diesen jeweils in die Kiste wirft. Diese Handvoll Stäbchen entspricht der mathematischen Redeweise "sei x eine beliebig, aber fest gewählte Zahl". Die allgemeine Lösung muß deshalb nicht aus mehreren Beispielen *abstrahiert* werden, sondern kann schon aus einem Beispiel *verallgemeinert* werden. Auch die Steuerung des Algorithmus durch die Null-Abfrage wird beim Hantieren aus der Registerkiste handgreiflich erfahren: Vor jedem Herausnehmen eines Stäbchens muß die Vp gewissenhaft nachfühlen, ob sich in dem Fach noch ein Stäbchen befindet.
Unsere Beobachtungen haben gezeigt, daß dieses Vorgehen der Vp ein lokales Nachdenken beim Erfinden einer geeigneten Lösungsstrategie für die Konstruktion von Algorithmen erlaubt. Es hat aber den Nachteil, daß am Ende der Problemlösung zwar das Ergebnis des Algorithmus sichtbar ist, aber nicht der Algorithmus selbst. Das Hantieren mit Stäbchen ist aber empfindlich gegen "Störungen" des Prozesses durch das Aufkommen neuer Ideen oder Änderungen der Strategie. Solche Unterbrechungen des Handlungsflusses oder seine Änderungen erfordern meistens einen Neubeginn. Das Hantieren mit Stäbchen muß aber nicht unbedingt dazu führen, daß die Struktur der gefundenen Problemlösung der Vp auch bewußt ist. (Es muß z.B. nicht dazu führen, daß der Vp bewußt wird, daß ihr gefundener Algorithmus aus zwei nacheinander auszuführenden Wiederholungen besteht.) Andererseits kann man beobachten, daß Vpn die Handlungen gruppieren, d.h. daß sie entsprechend der Struktur des Algorithmus Pausen einlegen. Es wird so in die Abfolge des Tuns ein Stück "Bedeutung" eingebracht. Aus dem Tun wird so ein zielorientiertes Handeln.
Andererseits kann das Hantieren mit Stäbchen eine Strukturierung auf der Ebene des Problems erleichtern. Bei der Strukturierung der anzustrebenden Lösung kann die Vp sich z.B. vorstellen, daß der Haufen Stäbchen von einem in ein anderes Fach verschoben werden muß oder daß als Zwischenlösung in einem Fach doppelt so viele liegen müssen wie in einem anderen. Die Darstellung der

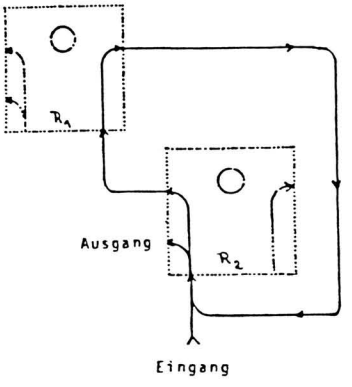
Zahlen in den Registern durch einen Haufen von Stäbchen kann also einen Weg der Problemlösung erleichtern, der darin besteht, daß sich die Vp vorstellt, welche Teilergebnisse erreicht werden müssen. Das Erarbeiten eines Algorithmus auf der Ebene der Stäbchen kann also durchaus mit einer begrifflichen Analyse des Problems verbunden sein.

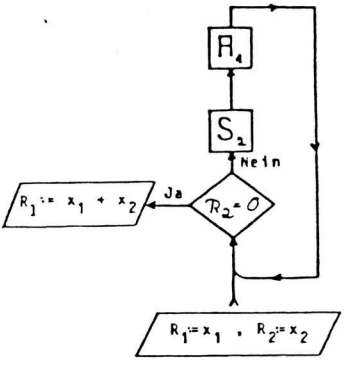
2.2 Rechennetze

Die vorgestellte Darstellungsebene der Stäbchen erleichtert zwar das lokale Erfinden eines Algorithmus, hat aber den Nachteil, daß nach Abschluß der Ausführung keine Dokumentation der Tätigkeit, sondern nur das Ergebnis vorliegt. Wir haben deshalb als zweite Ebene sogenannte Rechennetze entwickelt, auf die im folgenden näher eingegangen werden soll. Die Fächer der Registerkiste werden ersetzt durch die Zählbausteine aus dem Unterrichtsmittel "Dynamische Labyrinthe" ([3], [11]). In den Zählbausteinen befindet sich ein Zahnrad, auf dem die Zahlen Null bis Zehn notiert sind. Jeweils eine Zahl X ist im Fenster sichtbar. Der Zählbaustein hat zwei Eingänge, "+" und "-". Fährt man mit einem Stift durch den Eingang "+", so erreicht man den Ausgang "a" und das Zahnrad zählt vorwärts. Fährt man in den Eingang "-", so sind zwei Fälle zu unterscheiden: $X > 0$, dann zählt das Zahnrad rückwärts und man erreicht den Ausgang "≠"; $X = 0$, dann ist durch einen umgelegten Hebel H der Weg zum Zahnrad versperrt, so daß man ohne Rückwärtszählen den Ausgang "=" sofort erreicht. Fährt man im Zustand 0 durch den Eingang "+", so wird der Hebel H wieder in die Normalstellung geschoben. Zusammenfassend kann man also sagen, daß man mit dem Zählbaustein die Speicherung natürlicher Zahlen und durch passendes Befahren seiner Eingänge die Operationen Vorwärtszählen, Rückwärtszählen und Nullabfrage realisieren kann. Die Realisierung eines Algorithmus in der Ebene der Rechennetze besteht darin, die Eingänge und Ausgänge der Zählbausteine so mit Bausteinen (Geraden, Kurven, Kreuzungen, Einmündungen) zu einem



Netz zu verbinden, daß folgendes erreicht wird: Stellt man in den im Netz vorhandenen Zählbausteinen die natürlichen Zahlen ein und beginnt man die Durchfahrt durch das Netz beim Eingang E, so erreicht man automatisch den Ausgang A, und in dem vorher angegebenen Zählbaustein steht das Ergebnis der Anwendung des Algorithmus. Im folgenden ist ein Rechennetz für die Addition dargestellt, welches in R_1 die Summe der beiden Zahlen liefert, die zu Beginn in R_1 und R_2 gestanden haben.
Ein Blick auf die Abbildungen zeigt, daß die zeichnerische Darstellung eines Rechennetzes große Verwandtschaft mit Flußdiagrammen hat:





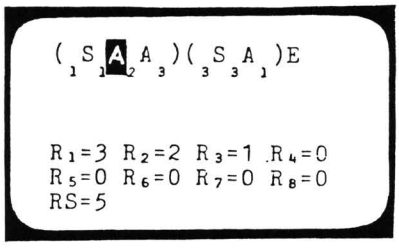
Dabei bezeichnen A_1 das Vorwärtszählen in Register 1 und S_2 das Rückwärtszählen in Register 2. Man kann die mit Bausteinen gesteckten Rechennetze als funktionsfähige Flußdiagramme ansehen.

Die Darstellung eines Algorithmus in der Repräsentationsform der Rechennetze hat gegenüber dem Hantieren mit Stäbchen einen großen Vorteil: Am Ende der Problemlösungsphase ist auf dem Brett eine Repräsentation des Algorithmus vorhanden und nicht nur eine Repräsentation seines Ergebnisses. Außerdem bewirkt der Zwang, die Zähler nacheinander verbinden zu müssen, eine deutliche Hervorhebung des Gedankens, daß ein Algorithmus in der Organisation einer geeigneten Abfolge von Handlungen besteht. Besonders soll noch darauf hingewiesen werden, daß die Wiederholung von Programmteilen in der Repräsentationsform der Rechennetze eine besonders einprägsame Darstellung hat: Das wiederholte Herumfahren in der gebauten Schleife wird von Vpn mit

den Worten beschrieben: "Ich fahre ... Runden". Unsere Beobachtungen haben gezeigt, daß auch folgende Mischform in der Phase des Erfindens eines Algorithmus nützlich ist: Auf dem Steckbrett sind nur die Zähler gesteckt; die Vpn fahren nur mit dem Finger entlang der in Gedanken "gesteckten" Bahnen bzw. zeichnen die verbindenden Linien mit einem Stift auf Papier, verändern aber beim Durchfahren den Zählerinhalt. Diese Technik erhält den Vorteil der Rechennetze gegenüber dem Flußdiagramm: Die Veränderung in den Zählern wird handgreiflich erfahren; der Vorteil gegenüber der vollständigen Konstruktion der Rechennetze besteht darin, daß die Zeit zwischen dem Aufkommen einer spontanen Idee und der Fertigstellung des gebauten Netzes nicht so lang ist und daß man sich eher auf vorläufige Ideen einläßt, wenn ihre Realisierung nicht mit so viel konstruktivem Aufwand verbunden ist.

2.3 Programmworte für Registermaschinen

Für die Repräsentation der hier betrachteten Algorithmen in der symbolischen Ebene einer Programmiersprache wurde von uns der Modell-Computer "Registermaschine" [12] benutzt, der auf einem Apple IIe durch ein Softwaresystem simuliert wird. Die Registermaschine besitzt eine einfache Programmiersprache und gestattet es, den Ablauf des Programms sowie die Veränderung der Registerinhalte auf dem Monitor schrittweise zu verfolgen.



Die Sprache der Registermaschine ist folgendermaßen definiert ([4] Seite 45/46):

Syntax

(1) A_i, S_i sind Programmworte ($1 \leq i \leq 9$)

(2) Sind P und Q Programmworte, so ist auch PQ ein Programmwort.

(3) Ist P ein Programmwort, so ist auch $(_iP)$ ein Programmwort ($1 \leq i \leq 9$).

Semantik

(1) A_i, S_i bedeutet Vorwärtszählen um 1 im Register i bzw. Rückwärtszählen um 1 im Register i ($1 \leq i \leq 9$).

(2) PQ bedeutet: Führe erst P aus und danach Q ("Sequenz")

(3) $({}_iP)$ bedeutet: Wiederhole P so lange, bis Register i den Inhalt 0 hat ($1 \leq i \leq 9$). ("Iteration")

Die Sprache für die Registermaschine ist also rekursiv definiert aus den Elementaroperationen Vorwärts- und Rückwärtszählen mit Hilfe der Kontrollstrukturen Sequentialisierung und Iteration. Die Sprache sieht außerdem die Benutzung von Unterprogrammen (einschließlich rekursiver Prozeduren) vor.

Die Beobachtungen haben gezeigt, daß die Darstellung eines Algorithmus als ein Programmwort für die Registermaschine keine Schwierigkeit bereitet, wenn dieser vorher auf der Ebene von Stäbchen oder Rechennetzen erfunden worden ist. Es handelt sich dann nur noch um eine Frage der Übersetzung von einer Sprache -die Vokabeln sind "Zähler", "Verbindungsstücke"- in eine andere -die Vokabeln sind "A_i", "S_i", "$({}_i$", ")".
Auch die Bedienung der Registermaschine wird in wenigen Minuten beherrscht.
Der in Abschnitt 2.1 entwickelte Algorithmus für Addieren wird in der Ebene der Programmworte folgendermaßen dargestellt:
$({}_2S_2A_1)$.

3. Vorstellung eines Untersuchungsdesigns

3.1 Versuchspersonen

Beobachtet wurden bisher im Zeitraum von 1981 bis zum September 1984 von den Mitgliedern unserer Arbeitsgruppe 130 Schüler im Alter von 12 bis 15 Jahren. In der hier referierten Hauptuntersuchung beziehen wir uns auf 23 Schüler aus einer Klasse 7 in einem Gymnasium im südlichen Landkreis von Osnabrück.

3.2 Ablauf der Untersuchungen

Bevor die eigentliche Hauptuntersuchung begann, wurden die Schüler zum einen einem sprachfreien Intelligenztest, dem Advanced Progressive Matrices-Test von RAVEN, unterzogen.

Die eigentliche Untersuchung bestand aus 4 Stunden Einzelunterricht für jede Vp,die mit einer Videoanlage aufgezeichnet wurden. Zum Abschluß mußten sich alle Schüler dem Spionageringtest ([8],[9]) unterziehen. Dieser Test ermittelt die individuellen Präferenzen der Vpn für bestimmte Lernstile.

3.3 Aufgaben

Die den Schülern während der Hauptuntersuchung vorgelegten Aufgaben lassen sich 3 Typen zuordnen:

- Einmal handelt es sich um die Aufgabe, für vorgegebene Probleme einen Algorithmus zu konstruieren (sog. konstruktive Aufgaben).
- Zum anderen handelt es sich um das Problem, vorgegebene Algorithmen in der Form von Programmen auf ihre Wirkungsweise zu analysieren (sog. analytische Aufgaben).
- Einen dritten Aufgabentyp, der sowohl analytische als auch konstruktive Komponenten aufweist, bilden die sog. Debugging-Aufgaben. Dabei wurden den Schülern fehlerhafte Algorithmen vorgelegt, mit dem Hinweis, für welches Problem sie eine Lösung darstellen sollen. Die Aufgabe der Schüler bestand darin, den Fehler zu analysieren und ihn schließlich zu beheben.

Die eingesetzten Aufgaben sind in Anlehnung an [4] entwickelt worden. Bei den konstruktiven Aufgaben konnten die Schüler die Ebene, in der sie den Einstieg zum Erfinden des Algorithmus suchen wollten (Hantieren mit Stäbchen, Rechennetze, Programmworte) wählen. Das Ziel der Aufgabe war immer die Angabe eines Programmwortes. In den analytischen Aufgaben wurden Programmworte vorgegeben, bei denen einmal nach ihrer Wirkungsweise und einmal nach ihrer Schrittzahlfunktion (Komplexität) gefragt wurde.
Die vorgelegten Aufgaben waren zwar ohne Vorkenntnisse im Programmieren zu lösen, erreichten aber schon zu Ende der ersten Untersuchungsstunde einen solchen Schwierigkeitsgrad, der es ausschloß, daß alle Schüler diese Aufgaben ohne Hilfestellung eines Vls vollkommen selbständig lösen konnten. Deshalb wurde zur Unterstützung der Schüler beim Problemlösen ein Katalog von Hilfen vorbereitet, mit denen die Vl eingreifen konnten. Dadurch sollte gewährleistet werden, daß jeder Schüler zur Lösung einer

jeden Aufgabe gelangen konnte.

3.4 Auswertung unter dem Aspekt der Leistungsdimension

Die Auswertungsphase begann nach Abschluß der Einzeluntersuchungen und war Ende Dezember 1984 abgeschlossen. Aus den Videobändern wurde für jeden Schüler und für jede Aufgabe erfaßt:

- in welchem Ausmaß er die gestellten Aufgaben selbständig gelöst hat,
- mit welchem Material er die Aufgabe bearbeitet hat.

Zusätzlich wurde die Vorgehensweise und an einigen, vorher festgelegten Stellen im Unterrichtsablauf, auch die verwendete Sprache analysiert und mit einem vorher entwickelten Kategoriensystem kategorisiert.
Nimmt man die Anzahl der vom Versuchsleiter für den Lösungsweg eines Schülers für notwendig gehaltenen Hilfen als Maßstab für die Leistung dieser Vpn, so kann man die Leistung in den verschiedenen Anforderungsbereichen vergleichen. Ein Befund der Untersuchungen ist, daß es sowohl Schüler gibt, die erheblich besser beim Konstruieren von Algorithmen als bei deren Analyse sind. Aber es existieren auch Schüler mit umgekehrter Leistungsdimension. Für diese beiden Typen wurden die Begriffe "konstruktiver" und "analytischer" Typ geprägt. Durch die Bezeichnungen "konstruktiver" oder "analytischer" Typ wird *keine* Aussage über die Gesamtleistung gemacht.
Vergleicht man die Rangskala der Leistungen der Vpn im Umgang mit Algorithmen mit der Rangskala ihrer Leistungen beim Bearbeiten des RAVEN, so ergibt sich kein signifikanter Zusammenhang ($\varrho = 0,17$). Außerdem beschreibt die Einteilung in konstruktive und analytische Typen ein anderes Phänomen als die intellektuelle Leistungsfähigkeit.
Vergleicht man die erzielten Leistungen von Mädchen und Jungen, so fällt auf, daß im oberen Quartil drei (von sechs) Jungen, aber nur zwei (von sechzehn) Mädchen liegen. Im letzten Quartil liegt kein Junge, jedoch vier Mädchen. Da unsere Klasse aber so ungleichmäßig mit Jungen und Mädchen besetzt war, können aus diesen Befunden keine signifikanten Ergebnisse abgeleitet werden.

3.5 Auswertung unter dem Aspekt kognitiver Strategien

Bei den umfangreichen Voruntersuchungen ergab sich die Frage, ob es sinnvoll ist, das offenkundig sehr unterschiedliche Verhalten der Vpn bei den Problemlösungen nur als Ausprägung einer Leistungsdimension zu interpretieren. Es wurde schon früh die Hypothese formuliert, daß es nützlich sei, die Existenz unterschiedlicher geistiger Mechanismen zu postulieren und das unterschiedliche Verhalten der Schüler als die jeweils spezifische Mischung in der Anwendung dieser kognitiven Werkzeuge zu interpretieren. Es stellte sich deshalb die Frage, von welcher Art diese verschiedenen Mechanismen sein könnten und aus welchen Verhaltensweisen der Vpn sich auf die Werkzeuganwendung schliessen ließe. Als eine Richtung zur Analyse der Befunde wurde der Weg eingeschlagen, die Art und den Inhalt der verbalen Schüleräußerungen näher zu untersuchen. Wir gingen dabei von der Vorstellung aus, daß unterschiedliche kognitive Strukturen ihre Spur im lauten Denken der Schüler hinterlassen müßten. Weiterhin stand in Form der Videoprotokolle auch eine Aufzeichnung der Gesten der Schüler und ihrer Handlungen mit dem Material zur Verfügung.

Unabhängig von den Prädikaten konstruktiv / analytisch lassen sich die Schüler hinsichtlich dieser Merkmale in zwei Gruppen einteilen:
Für die Art, wie ein Vertreter der einen Gruppe versucht, die Aufgaben zu lösen, halten wir folgendes für typisch: Wenn er einen Algorithmus konstruieren soll, stellt er sich vor, was der Rechner im einzelnen nacheinander zu tun hat. Soll er einen gegebenen Algorithmus kommentieren, so schildert er Stück für Stück, welche Schritte der Rechner nacheinander macht. Wichtig für ihn scheint zu sein, das Arbeiten der Maschine in seiner Sequentialität zu erfinden bzw. zu analysieren. Wir haben deshalb für ihn den Begriff *sequentiell* gewählt.
Für einen Vertreter der zweiten Gruppe ist es wichtig, beim Erfinden von Algorithmen zunächst das neu gestellte Problem an schon bekannte Konzepte anzubinden. Es fällt auf, daß er versucht, seine Ideen mit Begriffen zu beschreiben. Beim Analysieren äußert sich dies dahingehend, daß er versucht, vorgegebene Programme in Teile zu zerlegen und begrifflich über diese zu

verfügen. Wir haben deshalb das Wort *begrifflich* für ihn gewählt.

Die beiden unterschiedlichen kognitiven Strategien äussern sich bei den beiden Aufgabentypen wie folgt: Ist ein Algorithmus in symbolischer Form als Programmwort vorgegeben, sieht man bei einer begrifflichen Strategie die Aufgabe darin, die Bedeutung des durch das Programmwort gegebenen Algorithmus zu verstehen und begrifflich zu beschreiben; bei einer sequentiellen Strategie sieht man die Aufgabe darin, das Programmwort als Handlungssequenz für die Maschine zu interpretieren, um dann durch mehrere Beispiele zur abstrakten Beschreibung zu gelangen. Bei einer konstruktiven Aufgabe sieht man mit einer begrifflichen Strategie das Problem darin, die Aufgabe begrifflich zu zergliedern und sie mit bekannten Teilalgorithmen zu lösen; mit einer sequentiellen Strategie versucht man, eine Handlungssequenz mit Stäbchen oder mit Netz zu erfinden und diese dann "nur noch" in ein Programm zu übersetzen.

Eine Charakterisierung der Verhaltensweisen begrifflicher bzw. sequentieller Typen sowie eine ausführliche Dokumentation der Lösungsprozesse zweier Schüler anhand von Transkripten einer konstruktiven Aufgabe ist bei KAUNE [5] nachzulesen.

Wir haben also zwei Begriffspaare: konstruktiv/analytisch und sequentiell/begrifflich. Diese beiden Begriffspaare unterscheiden sich zunächst einmal durch ihre unterschiedliche Definition: Konstruktiv/analytisch benutzt eine Leistungsdimension bei verschiedenen Aufgabentypen, sequentiell/begrifflich bezieht sich auf beobachtbares Schülerverhalten bei der Auseinandersetzung mit diesen Aufgaben.
Untersucht man den Zusammenhang zwischen den gewählten Denkstrategien und der Leistungsfähigkeit, so fällt auf, daß die begrifflichen Schüler die besten Leistungen insgesamt bei den algorithmischen Aufgaben zeigen, gefolgt von denen, die die Strategie wechseln. Es läßt sich aber nicht nachweisen, daß sich eine der beiden Strategien besonders dazu eignet, die Aufgaben eines bestimmten Aufgabentyps zu lösen. Außerdem scheinen die ausgeprägt sequentiell vorgehenden Schüler den geringsten Erfolg im RAVEN zu haben. Dieses ist nicht verwunderlich, da die uns aus der Literatur bekannten Lösungsstrategien für diesen Test in

unserem Sinne als begriffliche zu bezeichnen sind.
Vergleicht man unsere Beschreibung der sequentiellen/begrifflichen Typen mit den serialistischen/holistischen Typen von PASK [8], so könnte man eine gewisse Ähnlichkeit vermuten. Wir sind dieser Frage nachgegangen und haben in den Arbeiten von VAN DER VEER eine von ihm überarbeitete Fassung der Theorie von PASK gefunden. Es ist uns möglich gewesen, kurzfristig eine Rohfassung der deutschen Version seines sog. Spionageringtestes [9] noch bei unserer Untersuchung einzusetzen.

Die Auswertung dieses Tests, die ebenfalls in enger Zusammenarbeit mit VAN DER VEER durchgeführt worden ist, ergab folgendes: Da wir sowohl begrifflich/serielle, begrifflich/holistische als auch sequentielle/serielle und sequentielle/holistische Schüler fanden, sahen wir uns in unseren Vermutungen bestätigt, daß unsere Begriffe sequentiell und begrifflich nicht das gleiche Phänomen beschreiben wie das Begriffspaar seriell/holistisch von PASK.

Zusammenfassend zeigen unsere Untersuchungen, daß für die Beschreibung und Beurteilung der geistigen Auseinandersetzung mit Algorithmen verschiedene Aspekte von Bedeutung sind:

- Die Auseinandersetzung mit Algorithmen geschieht in den meisten Fällen in einer handlungsorientierten Weise unterhalb der Ebene einer Programmiersprache. Überläßt man den Vpn die Wahl, so bevorzugen über 60 % von ihnen die Rechennetze.
- Es gibt einen Unterschied in der Leistungsfähigkeit beim Konstruieren und Analysieren von Algorithmen.
- Es gibt unterschiedliche kognitive Strategien, sich mit diesen Problemen auseinanderzusetzen. Die von uns beschriebene sequentielle Denkweise entspricht nicht weitverbreiteten Urteilen über das zweckmäßige Vorgehen beim Programmieren. Die Fähigkeit zum Wechseln der Strategien kann hohe Leistungen erklären; auch die Untersuchungen von MOLZBERGER [7] deuten in diese Richtung.

4. Literaturverzeichnis

[1] Card, S.; Moran, T.; Newell, A.: The Psychology of Human-Computer Interaction, Hillsdale 1983

[2] Cohors-Fresenborg, E.: On the Representation of Algorithmic Concepts, in: F. Lowenthal et al (Eds.): Pragmatics and Education, Plenum Press New York 1985

[3] Cohors-Fresenborg, E.; Finke, D.; Schütte, S.: Dynamische Labyrinthe, Osnabrücker Schriften zur Mathematik, Reihe U, Hefte 1-9, 1A-9A, 21

[4] Cohors-Fresenborg, E.; Griep, M.; Schwank, I.: Registermaschinen und Funktionen - Ein Schulbuch zur Einführung des Funktionsbegriffs auf der Grundlage von Algorithmen, Osnabrücker Schriften zur Mathematik, Reihe U, Heft 22, 3. Aufl. 1982, Heft 25 (Lehrerhandbuch) und Heft 22L (Lösungsheft)

[5] Kaune, C.: Kognitive Strategien beim Umgang mit Algorithmen, in: Schriftenreihe des Forschungsinstituts für Mathematikdidaktik Nr. 1, Osnabrück 1985

[6] Kling, U.: Kognitive Aspekte bei Mensch/Maschine - Interaktionsformen im Bereich des Lernens und Problemlösens, in: Ueckert, H.; Rhenius, D.: Komplexe menschliche Informationsverarbeitung, Bern 1979

[7] Molzberger, P.: Und Programmieren ist doch eine Kunst, in: Schelle; Molzberger: Psychologische Aspekte der Software-Entwicklung, München 1983

[8] Pask, G.: Spy Ring History Test Form IV, System Research Ltd., Richmond 1976

[9] Van der Veer, G.: Comments on the Problems with the German Version of the Spy Ring and Smugglertest, Arbeitspapier, Universität Amsterdam 1984

[10] Van Muylwijk, B.; Van der Veer, G.; Warren, Y.: On the implications of user variability in open systems, in: Behaviour and Information Technology, Vol. 2, mo. 4, 1983

[11] Unterrichtsmaterial "Dynamische Labyrinthe": Beschützende Werkstatt, Industriestr. 7, D-4500 Osnabrück

[12] "Registermaschinen-System": Forschungsinstitut für Mathematikdidaktik e.V., Postfach 1847, D-4500 Osnabrück

Prof. Dr. Elmar Cohors-Fresenborg
Dr. Christa Kaune
Fachbereich Mathematik/Informatik
Universität Osnabrück
Postfach 4469

D-4500 Osnabrück