Zur Didaktik der Algorithmik

Kerstin Strecker

Max-Planck Gymnasium
Theaterplatz 10
37073 Göttingen
kerstin.strecker@gmx.de

Abstract: In dieser Arbeit wird von der Tätigkeit des „Programmierens“ der Teil extrahiert, der sich auf die Algorithmik konzentriert und ein „roter Faden“ aufgezeigt, wie Schülerinnen und Schülern im Verlauf ihrer Schulzeit von der Grundschule bis zum Abitur das selbstständige Entwickeln von Algorithmen erlernen können. Dabei rückt neben dem klassischen systematisch-analytischen Zugang auch der experimentelle Zugang in den Blickpunkt der Betrachtungen, der nach Meinung der Autorin gerade bei jüngeren Schülerinnen und Schülern eine große Bedeutung im Lernverlauf hat.

1 Einleitung

Kinder in der dritten Klasse einer Grundschule lernen, große Zahlen schriftlich miteinander zu addieren. Dabei wird ihnen ein Algorithmus vorgesetzt, der in etwa folgende Struktur aufweist (siehe Abbildung 1):

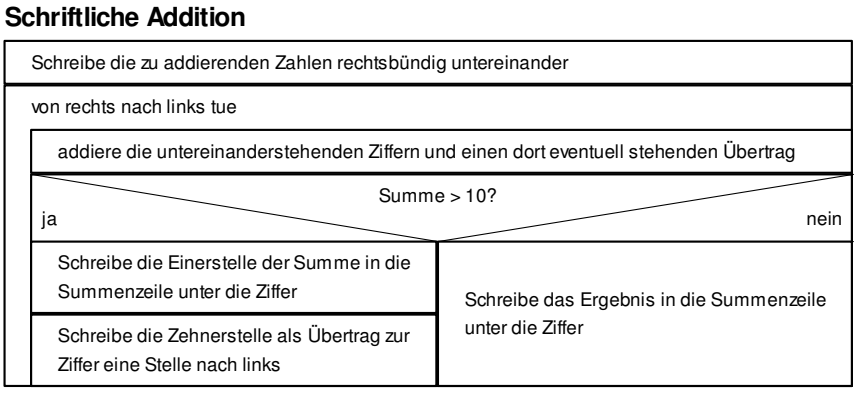

Abbildung 1 Algorithmus zur schriftlichen Addition

Wir finden als Grundstruktur eine Schleife mit einer Alternative im Schleifenrumpf. Die Kinder wenden diesen Algorithmus erfolgreich an. Obwohl das Stellenwertsystem im Unterricht behandelt wurde, können die Kinder aber kaum erklären, *warum* dieser Algorithmus funktioniert. Unbestritten ist aber, dass das Anwenden eines Algorithmus dieser Struktur für Kinder der dritten Klasse angemessen ist[1].

Ebenfalls angemessen erscheint der Zentralabiturkommission in Niedersachsen das Fordern der Lösung folgender Aufgabe im Abitur:

„Implementieren Sie in Java bzw. Pascal / Delphi eine Operation `verschluessele`, die eine Verschlüsselung nach dem Verfahren von Vigenere durchführt. Die Eingabeparameter sind eine Zeichenkette `klartext` und eine Zeichenkette `schluesselwort`. Es soll eine Zeichenkette zurückgegeben werden, die den verschlüsselten Text enthält.“ [ZA09]

Und die Struktur der Lösung? Ein Algorithmus bestehend aus einer Schleife, mit innerer Alternative (wie in Abbildung 2 ersichtlich):

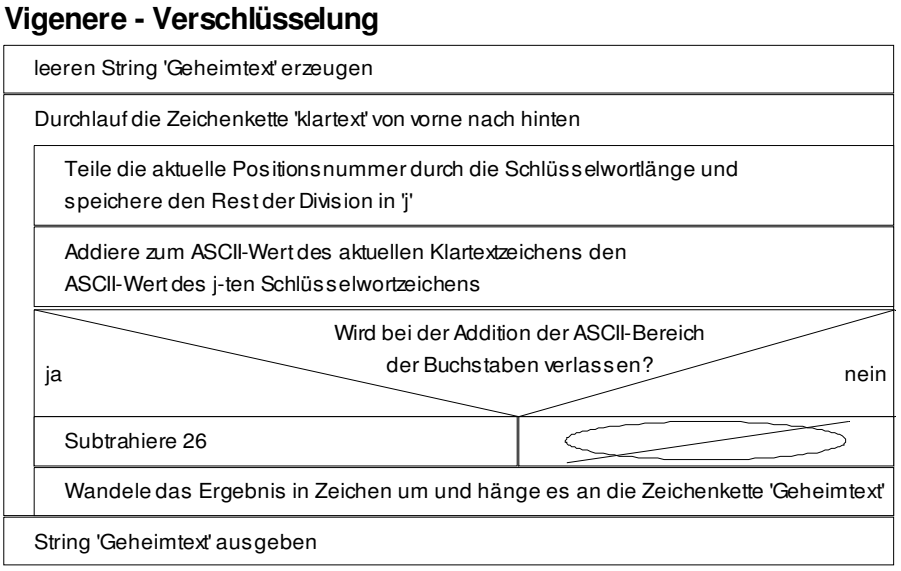

Abbildung 2 Vigenere-Verschlüsselung

[1] Der Algorithmus der schriftlichen Multiplikation (Jg. 4) beinhaltet sogar doppelt geschachtelte Schleifen.

Von der Struktur her sind die beiden vorgestellten Algorithmen, der aus der Grundschule und der aus der Abituraufgabe, durchaus *vergleichbar*. Ihre Struktur liegt in ein und derselben **Komplexitätsklasse**.

Trotzdem würde natürlich niemand behaupten, dass sich zwischen der Grundschule und dem Abitur nichts verändert. Nur müssen im Mittelpunkt der Überlegungen nicht die Algorithmen, sondern der *Umgang* mit ihnen stehen.

Betrachten wir im Folgenden einige der hier aufgelisteten Kompetenzen, die man im Umgang mit Algorithmen identifizieren kann:

- ➔ Teilalgorithmen anwenden (ohne Rechner)
- ➔ Teilalgorithmen modifizieren
- ➔ Teilalgorithmen erklären
- ➔ Teilalgorithmen zusammensetzen / vernetzen
- ➔ Teilalgorithmen zielgerichtet entwickeln

Dann finden wir den Bereich „Teilalgorithmen anwenden“ in Klasse 3, aber auch im Abitur, wenn die Schülerinnen und Schüler in Teilaufgabe a) der oben gezeigten Abituraufgabe eine Vigenere-Verschlüsselung durchführen müssen; selbst im Studium finden sich in Klausuren Aufgaben, bestimmte Algorithmen auf Beispiele anzuwenden. Aufgaben im Bereich „Teilalgorithmen anwenden“ verschwinden also nicht völlig, nur ihre Gewichtung verändert sich mit dem Alter und der Erfahrung der Schülerinnen und Schüler. Je älter die Lernenden werden, desto mehr ist zusätzlich die Kompetenz des eigenständigen Algorithmenentwurfs gefordert.

Wie schaffen wir aber den Übergang vom stark Angeleitetsein zur Selbständigkeit bei gleichbleibender Komplexität der Algorithmen?

2 Modifikation vorgegebener Programme in realen Miniwelten

2.1 Jahrgang 3/4

Im Rahmen eines aktuellen Schulversuchs[2], verzeichnen wir Erfolge mit folgendem Konzept für die Klassenstufe 3/4 im Bereich Technik: Die Schülerinnen und Schüler bekommen eine reale Miniwelt präsentiert, eine kleine Stadt aus LEGO, mit Straßen, einer Eisenbahn, Häusern, Spielplatz, Bahnhof,... Zunächst bauen sich die Kinder auch ihre eigenen Häuser in die Stadt oder erweitern sie um Blumenbeete, Reitplatz, Feuerwehr,... da sich gezeigt hat, wie groß (gerade bei jungen Schülerinnen und Schülern) das Interesse hieran ist und auf diesem Weg ein Hereindenken in die Miniwelt

[2]Schulversuch „Naturwissenschaftliche und Technische Bildung am Übergang von der Primar- zur Sekundarstufe“

und eine Identifizierung mit dieser erfolgt. Jetzt wird WeDo [WD11] in Kombination mit „scratch“ [Sc11][SW11] verwendet, um einige Dinge in dieser Stadt zu automatisieren. Die Schranke geht hoch, wenn sich ein Auto nähert (Entfernungssensor, Motor,...), die Häuser werden beleuchtet, wenn jemand in kurzem Abstand vorbei geht, mit dem Kippsensor als Schalter wird die Luke für das Silo mit Streusalz geöffnet,..... Alle Programme in dieser Welt haben den Aufbau:

Wert von Sensor überschritten/ unterschritten → Motor an/aus für ... Sekunden, …

Diese Programme sind fertig **vorgegeben**, die Automatisierung der LegoWelt funktioniert, nur eben nicht schön. Die Schranke öffnet sich nur zu 2/3, weil der Motor nicht lang genug läuft. In dem roten Haus wohnt eine Oma, die sehr langsam ist, weshalb die Beleuchtung länger eingeschaltet sein muss als anderswo oder der Entfernungssensor muss so eingestellt werden, dass nicht immer die Katze die Beleuchtung einschaltet, wenn sie aus ihrem Katzenhaus läuft,... Durch eine Modifikation der gegebenen Programme verbessern die Schülerinnen und Schüler jetzt ihre Welt. Sie verändern dabei aber nur folgende **Parameter**: Einmal die Schwellwerte in der Bedingung, die zum Sensor gehört und einmal die Zeitangaben oder Motorrichtungen der Aktionen, die ausgelöst werden.

Durch experimentelles Vorgehen in der Parametermodifikation, werden die Lernenden mit diesen immer strukturgleichen Programmen vertraut.

2.2 Jahrgang 5/6

In Klasse 5/6 wurden ebenfalls gute Erfahrungen damit gemacht, in realen Miniwelten WeDo einzusetzen. Hier mussten die Kinder aber anhand der Aufgabenstellung selbst entscheiden, welche Sensoren und Aktoren wo in der Welt platziert werden müssen. Die Programm**struktur** wurde **fest vorgegeben,** wie in Abbildung 3 zu sehen.

Abbildung 3 vorgegebene Programmstruktur

Abbildung 4 Werkzeugkasten

Die Schülerinnen und Schüler mussten aus einem Werkzeugkasten (Abbildung 4) die richtigen Elemente entnehmen und aus den Aufgabenstellungen Sinn entnehmend erlesen, welche Bedingungen und Aktionen man an welcher Stelle einsetzt. Dabei waren alle Programme ebenfalls strukturgleich.

Durch die Eigenheiten der verwendeten Programmiersprache „scratch“, Syntaxfehler im Vorfeld zu vermeiden, erkennen die Kindern bei der Ergänzung der Bedingung beispielsweise, dass sie nur eine Wahl zwischen den „grünen“ Komponenten haben. Es muss nicht vollständig durchdrungen sein, dass im Kopf einer Alternative ein boolescher Ausdruck stehen muss; die Bedingung kann auch ergänzt werden, ohne dass die Systematik dahinter transparent gemacht wird.

Durch die vorgegebene Auswahl von Bausteinen, können Schülerinnen und Schüler das richtige Konstrukt nicht nur entnehmen, sondern die möglichen Konstrukte auch gegeneinander abwägen und aus *diesen* Überlegungen ein Konstrukt auswählen. Verdeutlicht werden soll das an einem Beispiel:

Ein Quiz, bei der der Kandidat eine Frage gestellt bekommt, die er ohne Hilfe beantworten muss, ist schwerer als ein Quiz, bei dem der Kandidat die Antwort auf eine Frage aus vier möglichen vorgegebenen Antworten heraussuchen muss, weil man die Möglichkeiten gegeneinander abwägen oder die falschen Antworten ausschließen kann.

Eine Ergänzung von Bedingungen und Anweisungen in Algorithmen einer vorgegebenen Struktur erfordert noch keine Durchdringung der Problemlösestrategie, wir sprechen eher von einer

Anpassung einer gegebenen Strategie an die konkrete Aufgabe.

3 Experimentelle Lösungszugänge in virtuellen Miniwelten

Folgende Ergebnisse zeigen ein Schulversuch[3] und [St09]: In der Mittelstufe (Jahrgang 7/8/9) können den Kindern bereits Aufgaben gestellt werden, bei denen die Algorithmenstruktur *nicht* mehr vorgegeben ist, sondern von den Schülerinnen und Schülern aus vorgegebenen Algorithmenbausteinen zusammengesetzt werden muss. Ebenso müssen Anweisungen und Bedingungen ergänzt und eventuell ebenfalls zusammengesetzt werden. Richtig „offen“ sind die Strukturen dennoch nicht, weil die Schülerinnen und Schüler Alternativen und Schleifen nur ineinander verschachteln oder die bekannte Struktur um Alternativen und Schleifen nacheinander erweitern.

[3]Schulversuch „InTech – Informatik mit technischen Aspekten“ in Niedersachsen

Eine eigenständig entwickelte Schülerlösung in der Programmierumgebung „scratch“ zeigt Abbildung 5:

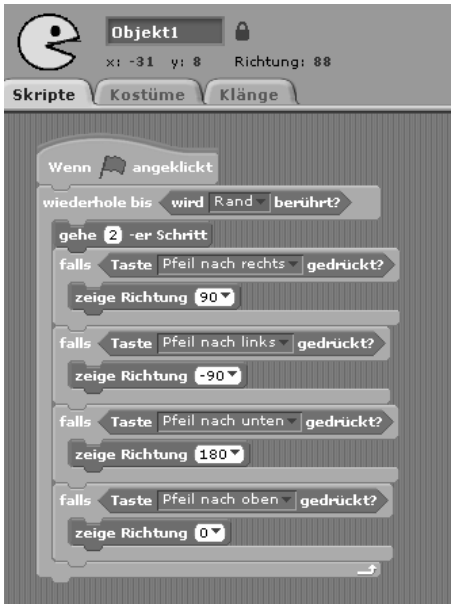

Abbildung 5 Programm eines Schülers einer 8. Klasse mit „scratch“

Wieder ein strukturgleicher Algorithmus, der hier zum ersten Mal von den Schülerinnen und Schülern selbst entwickelt wurde, ganz ohne vorgegebene Programmstruktur. Werkzeuge wie „scratch“ oder „Robot Karol“ [Rk11], „AutomatenKara“ [Ka11] und andere, die virtuelle kleine Miniwelten vorsehen, sind in diesen Jahrgängen weit verbreitet. **Diese Sprachen bieten alle Werkzeugkästen, integriert in ihre Programmierumgebungen**.

Lassen wir die Schülerinnen und Schüler selbst entscheiden welche Algorithmen sie implementieren wollen, dann werden fast in allen Jahrgangsstufen in „scratch“ gerne kleine „Spiele“ programmiert, wie „Pacman“, „Pong“ o.a. In der Mittelstufe können Aufgaben gelöst werden, bei denen die Algorithmenstruktur von den Schülerinnen und Schülern aus vorgegebenen Algorithmenbausteinen nach dem Baukastenprinzip zusammengesetzt werden muss. Durch Beobachtungen der Schülerinnen und Schüler müssen wir davon ausgehen, dass ein eigenständiges Finden von geeigneten Algorithmen zur Lösung einer Aufgabe mit den obigen Werkzeugen, zum Großteil durch **experimentelle** Herangehensweisen bestimmt ist.

Das Werkzeug „scratch“, wie auch andere, ermöglicht durch vielfältiges Kombinieren der einzelnen Bausteine eines Werkzeugkastens und dadurch, dass *keine Syntaxfehler* gemacht werden können und *immer ein ausführbares Programm* entsteht, einen experimentellen Zugang bei der Entwicklung eines Algorithmus. Die Kinder können Bausteine kombinieren, die entstehenden Algorithmen ausprobieren und zumindest Teile ihrer späteren Lösung durch Versuch und Irrtum entwickeln. Ihre korrekte Lösung entsteht u.a. durch ein Verwerfen von falschen Lösungen. Da die Bausteine und Konstrukte in ihrer Anzahl beschränkt sind, insbesondere Datenstrukturen keine Rolle spielen, ist dieser Zugang möglich.

Wenn erste Algorithmen so entwickelt werden, dann machen die Kinder Erfahrungen, die sie in späteren Algorithmen wieder verwenden. Dann systematisieren sie bei einer häufigen Suche verschiedenster Algorithmen ihre Erfahrungen und abstrahieren sie im optimalen Fall. Diese Kompetenz kann im Unterricht unterstützt werden. Indem man die Schülerinnen und Schüler in verschiedenen gleichartigen Lernumgebungen wie „AutomatenKara“, „Lego-Mindstorms“[Le11], „scratch“, „Robot Karol“ u.a. Algorithmen entwickeln lässt, und anschließend im Unterrichtsgespräch systematisiert, dass alle Sprachen gleichartige Bausteine, wie „Schleife“, “Alternative“,... enthalten, machen für die Lernenden auch Konstrukte wie Struktogramme oder UML-Aktivitätsdiagramme als Sprachen übergreifende Verbindung Sinn. In den Jahrgangsstufen 7, 8 und 9 steht also der experimentelle Zugang zur selbstständigen Algorithmensuche im Mittelpunkt. Genauer:

Verwendung von Lösungsstrategien, die durch ein neues Zusammensetzen bekannter Strategien entwickelt werden, mit teilweise experimentellem Probieren, bis der gewünschte Erfolg erreicht wird.

An dieser Stelle wird sich Widerspruch regen und der Wunsch laut werden, doch nicht noch zu fördern, dass die Lernenden „planlos“ irgendetwas zusammenklicken. Gewünscht ist doch der **analytische** Zugang. Das stimmt. Doch trial-and-error gibt es überall, zumindest ein Stück weit. Beobachtungen zeigen, dass der experimentelle Lösungszugang auch bei Informatik-Studenten nicht verschwunden ist und sich der Siegeszug der Sprachen, die diesen Ansatz unterstützen, vielleicht zum Teil auch mit diesem Argument erklären lässt.

Außerdem wird noch einmal ausdrücklich darauf verwiesen, dass die Sprachen, die einen experimentellen Ansatz unterstützen (eingeschränkte) *Werkzeugkästen* mit Algorithmenbausteinen enthalten müssen, die die Schülerinnen und Schüler syntaxfehlerfrei kombinieren können. **Jede beliebige Kombination muss dabei ein ausführbares** (jedoch nicht unbedingt semantisch korrektes) **Programm erzeugen**, damit die Schülerinnen und Schüler eine Reflexion aus dem Programm und der Beobachtung bei der Ausführung anschließen können. (Dies ist bei universellen Programmiersprachen nicht möglich. In Java z.B. ist eine Klassendeklaration im Schleifenrumpf nicht zulässig, nicht jede Kombination der Bausteine erzeugt also ein lauffähiges Programm) [St09].

4 Algorithmik mit universellen Sprachen

Die Algorithmen der Oberstufe sind nicht komplexer als in der Grundschule, wie bereits gezeigt wurde. Da sie aber auf beliebigen Datenstrukturen (Zeichenketten, Pixeln, Bäumen oder Datensätzen einer Datenbank beispielsweise) operieren, gelingt das eigenständige Entwickeln eines Algorithmus zur Lösung eines Problems nur dann, wenn die Schülerinnen und Schüler beim Entwickeln ihres Algorithmus genau durchdrungen haben, welche Semantik sich hinter welchem Konstrukt verbirgt. Durch die **Komplexität der Kombination von Möglichkeiten** wie z.B. in Java kann ein ausschließlicher experimenteller Zugang nicht mehr erfolgreich sein. Die Schülerinnen und Schüler müssen die Systematik der Algorithmik verstanden haben, die Struktur der Algorithmenbausteine muss durchdrungen sein, Probleme müssen analysiert werden können, um mittels eines analytischen Zugangs gezielt die Bausteine eines Algorithmus zusammensetzen zu können. Es reicht bei dem Schreiben einer Bedingung nicht nur, sich bewusst zu sein, dass hier ein boolescher Ausdruck folgen muss, man muss auch den Datenstrukturen entnehmen können, welche Methoden boolesche Ausdrücke liefern oder wie deren sonstige Rückgabewerte zu booleschen Ausdrücken verknüpft werden können. Durch das im obigen Kapitel skizzierte Verfahren der Systematisierung im Anschluss an eine Phase der konkreten Erfahrungen mit Algorithmen in den unterschiedlichsten Sprachen, kann der analytische Zugang gefördert werden [St09] und folgendes Ziel zumindest für die stärkeren Schülerinnen und Schüler auch erreicht werden:

Oberstufenschüler im Abitur entwickeln Lösungsstrategien m.E. auch gezielt.

5 Fazit

Zusammenfassend lassen sich also folgende Zugänge zur Algorithmik in den einzelnen Jahrgangsstufen herausfiltern:

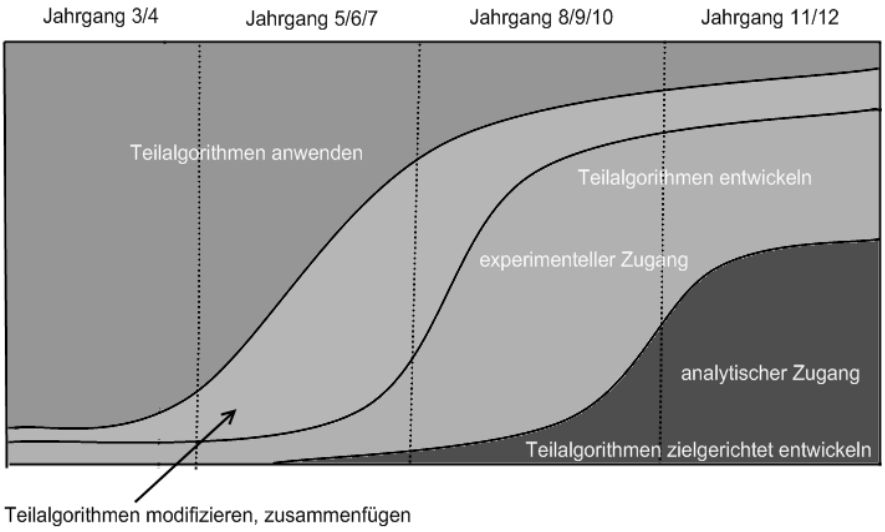

Abbildung 6 Algorithmenzugänge in den einzelnen Jahrgangsstufen

Die Kompetenzen:

- ➔ Teilalgorithmen anwenden
- ➔ Teilalgorithmen modifizieren, zusammenfügen
- ➔ Teilalgorithmen entwickeln mit dem experimentellen Zugang
- ➔ Teilalgorithmen (zielgerichtet) entwickeln mit dem klassischen systematisch-analytischen Zugang

wie in den einzelnen Kapiteln beschrieben, finden sich dabei unterschiedlich gewichtet in den verschiedenen Jahrgangsstufen. Dabei werden am Ende der Schulzeit immer noch **alle** Kompetenzen abgeprüft und verwendet. Und auch Schülerinnen und Schülern in Jahrgang 4 muss ermöglicht werden, bereits kleine eigene Algorithmen zu entwickeln.

Ein experimenteller Zugang zur Algorithmik verbietet nicht den analytischen Zugang, kann aber schwächeren Schülerinnen und Schülern helfen, überhaupt Algorithmen selbstständig entwickeln und nicht nur nachvollziehen zu können. Und dies ist für ein kreatives Fach Informatik (überlebens-)wichtig.

Literaturverzeichnis

[ZA09] Niedersächsisches Kultusministerium: Aufgaben des Zentralabiturs Informatik 2009

[Ka11] Raimond Reichert, Jürg Nievergelt, Werner Hartmann: „Programmieren mit Kara“, Springer-Verlag, 2005

[Le11] URL: http://education.lego.com/en-gb/preschool-and-school/secondary-11-18/11plus-lego-mindstorms-education/, Zugriff: 9.1.2011

[Rk11] URL: http://www.schule.bayern.de/karol/download.htm, Zugriff: 9.1.2011

[Sc11] URL: http://scratch.mit.edu/, Zugriff: 9.1.2011

[St09] Kerstin Strecker: „Informatik für Alle – wie viel Programmierung braucht der Mensch?“ Dissertation, Universität Göttingen, 2009, URL: http://webdoc.sub.gwdg.de/diss/2009/strecker/, Zugriff: 9.1.2011

[SW11] URL: http://info.scratch.mit.edu/WeDo, Zugriff: 9.1.2011

[WD11] URL: http://www.lego-in-der-schule.de/lego-learning-blog-1/lego-education-wedo-robotics-ein-neues-konzept-fuer-robotik-in-der-grundschule, Zugriff: 9.1.2011