

Die Nutzung von Programmierwerkzeugen für das Lösen offener Aufgaben im Mathematikunterricht der Sekundarstufe I

Bernd Hafenbrak, Weingarten (Germany)

Heinz Schumann, Weingarten (Germany)

Abstract: *The use of programming tools for the solution of open problems in middle secondary mathematics teaching. Due to the fact that the computer tools available at present in mathematics teaching offer only inferior capabilities for solving (open) problems, their capabilities being restricted to sequential algorithms, and also because the of mathematics - informatics paradigm is inadequately formulated, standardized programming tools are needed in order to handle certain types of problems. The article presents examples of arithmetic problems and their solution using standardized programming tools, so as to illustrate their applications in: developing programs for the derivation of multi-element solution sets (as well as for strengthening the inductive basis required in order to find propositions); modifying available programs in order to be able to deal with diversified or extended open problems; using standardized programs as a „black-box“ software for checking or elaborating strategies for open problem solving.*

Kurzreferat: Sowohl die Beschränktheit der heute im Mathematikunterricht verwendeten Computerwerkzeuge, mit denen beim Lösen von (offenen) Aufgaben nur sequentielle Algorithmen abgearbeitet werden können, als auch das mathematisch-informatische Arbeitsparadigma machen den Einsatz von standardisierten Programmier-Werkzeugen notwendig, um gewisse offene Aufgaben lösen zu können. An vorwiegend arithmetischen Aufgabenbeispielen werden folgende Einsatzmöglichkeiten verdeutlicht: Die Entwicklung von Programmen zur Bestimmung von mehrelementigen Lösungsmengen (auch zur Stärkung der induktiven Basis für das Finden von Aussagen) sowie die Modifikation bereitgestellter Programme, um variierte oder erweiterte offene Aufgabenstellungen zu bearbeiten, und die Benutzung fertiger Programme als „Black-Box-Programme“ zur Kontrolle oder zur Bestimmung der Lösungen offener Aufgaben

ZDM-Classification: F10, D50, P50, U50

1. Einleitung

Beim Lösen offener Aufgaben mit Unterstützung von Computerwerkzeugen im Sinne von Anwenderprogrammen kann man im allgemeinen nur sequentielle Algorithmen abarbeiten (vgl. u.a. Schumann 2000), – wenn von der Verwendung der spezifischen Benutzersprachen in solchen Tools abgesehen werden soll. Es fehlen Kontrollstrukturen – wie sie in den standardisierten Programmiersprachen z.B. in BASIC und in LOGO verfügbar sind – für die Wiederholung gleichförmiger Arbeitsschritte, z.B. FOR ... TO ... NEXT..., REPEAT..., und für die Auswahl von Arbeitsschritten z.B. IF ... THEN Das Fehlen kann in herkömmlichen Lernumgebungen oder in Anwenderprogramm-Lernumgebungen zu zeitaufwendigen oder sogar zu überhaupt nicht ausführbaren Lösungen gewisser offener Aufgaben führen.

Obwohl das Programmieren, vor allem wegen der

„Codierungsprobleme“, in der Sekundarstufe I heute keinen curricularen Stellenwert besitzt, kann es die Lösung geeigneter offener Aufgaben effektiv unterstützen, wenn man sich auf einfache, überschaubare und sich selbst dokumentierende Programme beschränkt. Das Entwerfen eines Programms ist selbst wieder das Lösen einer offenen Aufgabe, denn für einen zu erzielenden Output gibt es meist unterschiedliche Programme. – Einen Überblick über die allgemeine Problematik des Lernens zu Programmieren geben u.a. Rogalski u. Samurçay (1993).

Wir unterscheiden folgende idealtypischen Einsatzmöglichkeiten des Programmierens für das Lösen offener Aufgaben alphanumerischer und grafischer Art:

- Die Entwicklung von Programmen zur Bestimmung von mehrelementigen Lösungsmengen (auch zur Stärkung der induktiven Basis für das Finden von Aussagen).
- Die Modifikation bereitgestellter Programm, um variierte oder erweiterte offene Aufgabenstellungen zu bearbeiten.
- Die Benutzung fertiger Programme zur Kontrolle oder zur Bestimmung der Lösungen offener Aufgaben in Form von „Black-Box-Programmen“.

Im Folgenden konkretisieren wir solche Nutzungsmöglichkeiten an ausgewählten Beispielen repräsentativer Art.

2. Beispiele für die Einsatzmöglichkeiten von Programmierwerkzeugen für das Lösen offener Aufgaben

2.1 Vorbemerkungen

Bei der Formulierung der Programme kommt es nicht auf Eleganz oder Laufzeiteffizienz an; im Vordergrund steht eine Programmierung nach der Methode „by brute force“. In heuristischer Sicht entspricht das Abarbeiten von vielen Arbeitsschritten und Fallunterscheidungen mit Hilfe eines Programms der Methode des „systematischen Probierens“.

Da die Domäne für das Lösen offener Aufgaben durch Programmieren in der Sekundarstufe I die Arithmetik ist, erfährt der Anteil solcher Aufgaben eine angemessene Gewichtung.

Wir verwenden Q-BASIC und MSW-LOGO: Q-BASIC ist ein Zubehör des Windows-Betriebssystems, für das es auch eine Igel-Grafik-Erweiterung gibt; MSW-LOGO steht im Internet als Freeware zur Verfügung.

Die Lösungsverfahren für offene Aufgaben können auch im Assistenzprogramm DERIVE, implementiert z.B. im TI-89 oder T-92, programmiert und ausgeführt werden; in diesem Zusammenhang kann es aber Schwierigkeiten bei der Dokumentation des Ergebnis-Outputs kommen.

Gegebenenfalls kann die Bereitstellung entsprechender Programm-Bausteine den jeweiligen Programmentwurf erleichtern.

2.2 Aus der Arithmetik

Aufgabe 1:

Finde heraus, welche natürlichen Zahlen als Summe von drei aufeinander folgenden natürlichen Zahlen geschrieben werden können.

Hier genügt eine einfache FOR-Schleife, um einen Überblick über solche Zahlen zu gewinnen:

```
z = 1000
FOR n = 1 TO z
  IF z = n + (n + 1) + (n + 2) THEN PRINT z
NEXT n
```

Für 1000 gibt es keine Lösung usw. – Termumformung zu $3(n+1)$ lässt die Gesetzmäßigkeit erkennen. Die Aufgabe bietet sich natürlich für eine weitere Öffnung an: vier aufeinanderfolgende Summanden, fünf Summanden usw.

Aufgabe 2:

Bestimme alle Teiler einer Zahl, z.B. von 6666205. Findest du Zahlen mit sehr wenigen Teilern und solche mit sehr vielen Teilern? Kannst du Gesetzmäßigkeiten erkennen?

```
z = 6666205
FOR n = 1 TO z
  IF z MOD n = 0 THEN PRINT n,
NEXT n
```

Kommentar zum Programm (Aufgabe 2):

U.a. folgende Aussagen für zusammengesetzte Zahlen können gefunden werden: Der kleinste von 1 verschiedene Teiler ist immer eine Primzahl. Ist der Teiler kleiner (gleich) der Wurzel aus der Zahl, so ist sein Ergänzungsteiler größer (gleich) dieser Wurzel.

Aufgabe 3:

Bestimme alle Primzahlen kleiner als 1000. Welche Besonderheit findet man bei Primzahlen, deren Differenz 2 ist (sog. Primzahlzwillinge)?

Für einen Überblick ist die Funktion $\text{prim}(x)$ hilfreich, die den Wert 1 ergibt, falls x eine Primzahl ist, sonst den Wert 0.

```
FUNCTION prim(z)
  ungeteilt=1
  FOR i = 2 TO z-1
    IF z MOD i = 0 THEN ungeteilt = 0
  NEXT i
  prim = ungeteilt
END FUNCTION
```

Diese Funktion kann den Schülern als Black-Box mitgeteilt werden; sie kann aber auch aus Aufgabe 2 entwickelt, zumindest kann der Zusammenhang aufgezeigt werden. Mit dieser Funktion wird die Auflistung der Primzahlen leicht:

```
FOR z = 2 TO 100
  IF prim(z) THEN PRINT z
NEXT z
```

Die Primzahlzwillinge können aus der entstehenden Liste herausgesucht werden, es kann aber auch folgendes Programm zum Einsatz kommen:

```
FOR z = 2 TO 100
  IF prim(z) AND prim(z+2) THEN PRINT z,
  z+2
NEXT z
```

Aufgabe 4:

Wann teilt eine ganze Zahl t das Produkt $x \cdot y$, wann teilt t die Summe $x + y$, wann teilt t die Summe $x^2 + y^2$, usw.

Einen guten Überblick über die Lösungen erhält man durch grafische Darstellung der Paare (x, y) im Koordinatensystem. Ist die Bedingung erfüllt, so wird der Punkt markiert (in Q-BASIC mit PSET).

```
t=7
FOR x = 0 TO 640
  FOR y = 0 TO 480
    IF x*y MOD t = 0 THEN PSET (x, y)
  NEXT y
NEXT x
```

Für $t = 7$ erhält man ein sehr schlichtes Bild (Abb. 1, links), das auch leicht erklärt werden kann. Für $t = 48$ ist das Bild schon reichhaltiger (Abb. 1, rechts).

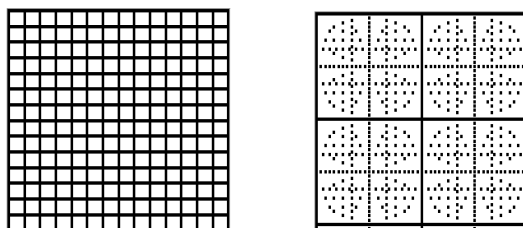


Abb.1

Wird der Ausdruck $x^2 + y^2$ durch t geteilt, sind die Zusammenhänge nicht mehr so einfach. Die Abbildung 2 zeigt die Bilder für $t = 7$ (links) und für $t = 25$ (rechts).

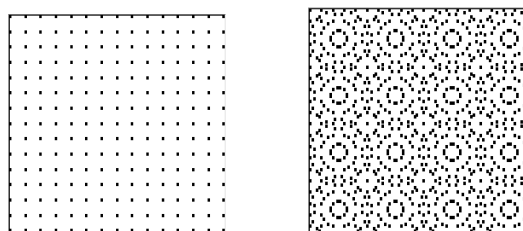


Abb. 2

Es bietet sich an, hier mit Farben zu arbeiten, die in Abhängigkeit von den Divisionsresten vergeben werden. Bei der Bearbeitung dieser Aufgabe ist viel Raum zum selbständigen Experimentieren; die grafisch dargestellten Ergebnisse beeindrucken auch Nichtmathematiker.

Aufgabe 5:

Wann ist ein Bruch (kleiner 1) durch eine Summe von Stammbrüchen darstellbar? Gibt es u.U. mehrere

Darstellungen? Welche ist die beste?

Diese Aufgabe, die an die Bruchdarstellung der alten Ägypter anknüpft (Padberg, 1995), ist schon recht anspruchsvoll, wenn man die allgemeine Lösung anstrebt. Mit Hilfe von kleinen Programmen kann man aber leicht einige Teillösungen finden. So kann man etwa untersuchen, welche Brüche z/n sich als Summe von zwei Stammbrüchen darstellen lassen: $z/n = 1/i + 1/j$. Das betreffende Programm kann so aussehen:

```
z = 2
n = 13
FOR i = 2 TO n
  FOR j = n+1 TO 1000
    IF z*i*j = n*i + n*j THEN PRINT i, j
  NEXT j
NEXT i
```

Dabei haben wir den Zähler größer als 1 vorausgesetzt, dann ist einer der Nenner der beiden Stammbrüche sicher kleiner als n . Für den anderen Nenner haben wir sehr grob die Grenze 1000 gewählt. Eine Verallgemeinerung auf drei Stammbrüche ergibt sich sofort.

Überprüft man nun viele Brüche mit diesen beiden Programmen, so sieht man, dass alle Brüche mit Zähler 2 sich als Summe von zwei Stammbrüchen darstellen lassen. Ist der Zähler 3, so gibt es zwei oder drei Stammbrüche als Summe. Bei kleinen Nennern gibt es jeweils nur eine Lösung, bei größeren Nennern können auch mehrere Lösungen auftauchen. Die Beispiele der Brüche mit den Zählern 2 oder 3 führen in naheliegender Weise auf einen Algorithmus, der jeden Bruch z/n in höchstens z verschiedene Stammbrüche zerlegt (Ziegenbalg, 1996).

Aufgabe 6:

Bestimme alle ganzzahligen Variablenbelegungen für die der Term $((abc)/(a+b+c) \mid a-b-c) / c$ einen ganzzahligen Wert annimmt.

Kommentar zum Programm (Aufgabe 6):

Ein Überblick über die Lösungstripel für gewisse Bereiche gelingt mit drei geschachtelten FOR-Schleifen. An den Lösungstripeln liest man die trivialen Lösungen ($a = b = c$) und die Parameterdarstellung für die nichttrivialen Lösungen ($a; b = a+12; c = a+6$) ab usw.

Aufgabe 7:

$\frac{16}{64} = \frac{1}{4}$ erhält man auch durch Streichung der 6 im Zähler und Nenner von $\frac{16}{64}$. Gibt es noch andere solche Brüche mit dieser Eigenschaft, auch solche, deren Zähler und Nenner dreistellig sind?

Auch hier genügen geschachtelte FOR-Schleifen, um einen Überblick zu erhalten.

Bei dreistelligem Nenner und Zähler gibt es mehrere Möglichkeiten Ziffern zu streichen, um einen wertgleichen Bruch zu erhalten. Eine der Möglichkeiten ist ausgeführt.

```
FOR a = 1 TO 9
  FOR b = 1 TO 9
    FOR c = 1 TO 9
      FOR d = 1 TO 9
        IF (a*100+b*10+c)*d = a*(b*100+c*10+d)
        THEN
          PRINT a*100+b*10+c, b*100+c*10+d
        END IF
      NEXT d
    NEXT c
  NEXT b
NEXT a
```

Man erhält als Ergebnisausdruck, der auch die trivialen Lösungen aufweist:

```
111 / 111 = 1 / 1
166 / 664 = 1 / 4
199 / 995 = 1 / 5
222 / 222 = 2 / 2
266 / 665 = 2 / 5
333 / 333 = 3 / 3
444 / 444 = 4 / 4
484 / 847 = 4 / 7
499 / 998 = 4 / 8
555 / 555 = 5 / 5
654 / 545 = 6 / 5
666 / 666 = 6 / 6
742 / 424 = 7 / 4
777 / 777 = 7 / 7
888 / 888 = 8 / 8
999 / 999 = 9 / 9
```

Aufgabe 8:

Welche natürlichen Zahlen sind gleich der dritten Potenz ihrer Quersumme?

Hier können wir die Funktion "quersumme(z)" benutzen, die eventuell den Schülern als Black-Box in die Hand gegeben wird:

```
FUNCTION quersumme(z)
  s = 0
  x = z
  WHILE z > 0
    s = s + x MOD 10
    x = x/10
  WEND
  Quersumme = s
END FUNCTION
```

Mit dieser Funktion gestaltet sich die Suche einfach:

```
FOR n = 1 TO 99
  z = n*n*n
  IF n = quersumme(z) THEN PRINT z
NEXT n
```

Kommentar zum Programm (Aufgabe 8):

Da aus $z = z_m 10^m + z_{m-1} 10^{m-1} + \dots + z_1 10^1 + z_0 10^0 = (z_m + z_{m-1} + \dots + z_1 + z_0)^3$ mit $z_m \neq 0$ durch grobe Abschätzung $10^m < (9+9+\dots+9+9)^3 = 9^3(m+1)^3$ bzw. vereinfacht $m < 2,87+3 \cdot \lg(m+1)$ folgt, was nur für $m < 6$ richtig ist, genügt es Zahlen z kleiner als 10^6 , also n kleiner als 10^2 , zu untersuchen.

2.3 Aus der Kombinatorik

Aufgabe 9:

Bestimme die Teilmengen einer 7-elementigen Menge, zum Beispiel von $\{A, B, C, D, E, F, G\}$.

Mit den Befehlen der String- oder der Listenverarbeitung kann der folgende Algorithmus, als ein Beispiel für einen nichtnumerischen symbolverarbeitenden Algorithmus, codiert werden.

Ein Algorithmus zur Bestimmung der Teilmengen einer endlichen Menge in muttersprachlicher Darstellung:

- 1) Schreibe die leere Menge als Teilmenge auf.
- 2) Gibt es ein Element der Menge, das du noch nicht verwendet hast, um Teilmengen zu bilden? Wenn ja, dann weiter bei 3), sonst weiter bei 5).
- 3) Füge nur eines der noch nicht verwendeten Elemente zu jeder schon gebildeten Teilmenge hinzu.
- 4) Schreibe die neuen Teilmengen unter die schon aufgeschriebenen. Weiter bei 2).
- 5) Fertig!

Das entsprechende BASIC- oder LOGO-Programm wird den Schülern zur Verfügung gestellt und der unterliegende Algorithmus erklärt.

2.4 Aus der Geometrie

Aufgabe 10:

Zeichne möglichst viele Rechtecke in horizontaler Lage, die beliebig dimensioniert und die in- oder teilweise übereinander liegen können und die ein schönes "Zufallsbild" geben sollen.

Folgendes LOGO-Programm im Bottom-Up-Entwurf führt solche Zufallsbilder aus:

```
PR Rechteck :Seite1 :Seite2
Wiederhole 2 [Vorwaerts :Seite1 Rechts 90 Vorwaerts :Seite2 Rechts 90]
Ende
PR Zufallsrechtecke :N
Loeschebild Versteckigel
Wiederhole :N [Rechts Zufallszahl 360 Vorwaerts Zufallszahl 300 Stiftab
Aufkurs 90 Rechteck Zufallszahl 50 Zufallszahl 50 Stifthoch Aufxy 0 0]
Ende
```

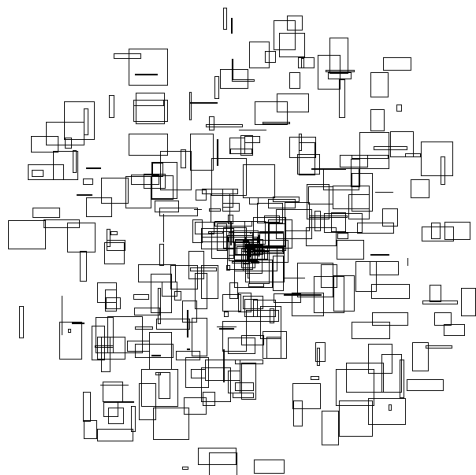


Abb. 3

Kommentar zum Programm (Aufgabe 10):

Das sich selbst dokumentierende LOGO-Programm liefert ästhetisch ansprechende Zufallsgrafiken. Mit der Spur-Option z.B. mit der von Cabri-Géomètre ließen sich solche Bilder nur für einander ähnliche Figuren (z.B. für Quadrate oder Kreise) erzeugen.

3. Abschließende Bemerkungen

Bemerkung 1:

Das Lösen offener Mathematikaufgaben und das Entwerfen von Programmen stehen in einem engen Zusammenhang.

Einerseits ist jede Programmieraufgabe offen: Sehr oft ist das Ziel nicht genau festgelegt; in der Regel gibt es verschiedene Wege; immer kann das Programm noch verbessert und weiterentwickelt werden; jedes fertige Programm weckt neue Wünsche und gibt Anregungen zu weiteren Programmen. Andererseits können bei mathematischen Aufgaben, die offen hinsichtlich der Lösungsmenge sind, Programme eine effektive Lösungshilfe sein, wie die obigen Beispiele zeigen. Insofern können sich beide Aspekte wechselseitig befruchten. Auf diese Möglichkeit hat schon früh Papert hingewiesen (Papert, 1982). Sein Konzept des Bereitstellens einer Programmierumgebung und des Bottom-Up-Entwurfes baut auf offene Aufgabenstellungen, offen sowohl in informatischer als auch in mathematischer Hinsicht. Die Argumente, die Papert für diese Art des Vorgehens angeführt hat, sind im Wesentlichen dieselben, die heute für die Behandlung offener Aufgaben vorgebracht werden.

Bemerkung 2:

In der Mathematik als Wissenschaft ist das Lösen offener Probleme mit Hilfe von Programmierwerkzeugen eine mit Erfolg angewendete Arbeitsmethode. So ist diese bei der Lösung komplexer Fallunterscheidungs- und Anzahlbestimmungsprobleme nicht mehr wegzudenken; Beispiele dafür sind: Beweis des Vierfarbensatzes (Appel u. Haken, 1977), Bestimmung der Anzahl konvexer Polyeder mit vorgegebener Anzahl von Flächen und Ecken (vgl. u.a. Bender 1987) oder – als Beispiel für das "zellulare Wachstum" – die (Anzahl-) Bestimmung von Polyominos (vgl. u.a. Lunnon 1971).

Bemerkung 3:

Sowohl die Beschränktheit der heute im Mathematikunterricht verwendeten Computerwerkzeuge, beim Lösen von (offenen) Aufgaben nur sequentielle Algorithmen abarbeiten zu können, als auch das mathematisch-informatische Arbeitsparadigma begründen die Notwendigkeit des Einsatzes von Programmierwerkzeugen in der Sekundarstufe I auf einem propädeutischen Niveau.

4. Literatur

- Appel, K.; Haken, W. (1977): Every plane map is four colorable. – In: Illinois J. Math. Jg. 21, S. 429 – 567
 Bender, E.A. (1987): The number of 3-dimensional convex polyhedra. – In: Amer. Math. Monthly H.9, S. 7- 21
 Göhner.H.; Hafenbrak, B. (1995): Arbeitsbuch Q-BASIC. – Bonn: Dümmler

- Hafenbrak, B. (2000): Programmieren im Mathematikunterricht der Sekundarstufe I. – In: Herget, W. u.a. (Hrsg.): Standardthemen des Mathematikunterrichts in moderner Sicht. Hildesheim: Franzbecker
- Lunnon, W.F. (1971): Counting polyominoes. – In: Computers in Number Theory. London: Academic press, S. 347 – 372
- Menzel, K. (1981): BASIC in 100 Beispielen. – Stuttgart: B.G. Teubner
- Padberg, F. (1995): Didaktik der Bruchrechnung. – Heidelberg: Spektrum
- Papert, S.(1982): Mindstorms – Kinder, Computer und neues Lernen. – Basel: Birkhäuser
- Rogalski, J.; Samurçay, R. (1993): Task Analysis and Cognitive Model as a Framework to Analyse Environments for Learning Programming. – In: Cognitive Models and Intelligent Environments for Learning Programming. Berlin: Springer, S. 6-19
- Schumann, H. (2000): Computerunterstütztes Lösen offener raumgeometrischer Aufgaben. – In: ZDM Zentralblatt für Didaktik der Mathematik 32(6), S. 175-185 (<http://www.fiz-karlsruhe.de/restricted/zdm/articles/zdm006a2.pdf> until Dec 2001, then <http://www.fiz-karlsruhe.de/fiz/publications/zdm/zdm006a2.pdf>)
- Ziegenbalg, J. (1996): Algorithmen. – Heidelberg: Spektrum

Autoren

- Hafenbrak, Bernd, Prof. Dr., Institut für Bildungsinformatik, Fakultät III, Mathematik/Informatik, PH Weingarten, D-88250 Weingarten.
E-mail: hafenbrak@ph-weingarten.de
- Schumann, Heinz, Prof. Dr., Institut für Bildungsinformatik, Fakultät III, Mathematik/Informatik, PH Weingarten, D-88250 Weingarten.
E-mail: schumann@ph-weingarten.de
Homepage: <http://www.mathe-schumann.de>