Verifikation: Zu schwierig für die Schule? Drei Gegenbeispiele!

Die beiden fundamentalen Ideen der Informatik im Zusammenhang mit der *Bewertung von Algorithmen* sind *Verifikation* (d.h. Überprüfung der Korrektheit des Algorithmus bzgl. einer vorgegebenen Spezifikation) und *Komplexität* (d.h. Analyse des Laufzeit- und Speicherplatzverhaltens des Algorithmus). Mit beiden Ideen sind eine Reihe weiterer fundamentaler Ideen verbunden (Abb. 1). Hierzu vgl. Schwill 1993. Während Fragen der Komplexität in der Schule im angemessenen Umfang und mit formalen Mitteln behandelt werden, führen Verifikationen und Korrektheitsanalysen noch ein Schattendasein (s. etwa die Lehrplanentwürfe von Rheinland-Pfalz oder Niedersachsen aus dem Jahre 1992). Sie gelten als zu formal und zu schwierig, um sie in der Schule in einer Weise zu behandeln, die ihrem Stellenwert in der Informatik entsprechen. Häufig wird dieses Gebiet nur im Zusammenhang mit dem Testen von Programmen thematisiert.
Akzeptiert man jedoch, daß es sich bei der Verifikation um eine fundamentale Idee der Informatik handelt, so muß man sie zwangsläufig auf (fast) allen kognitiven Niveaus - auf einem niedrigen Primarstufenniveau ebenso wie auf Hochschulniveau - in geeignet aufbereiteter Form vermitteln können (sog. *Vertikalkriterium* einer fundamentalen Idee, s. Schwill 1993).

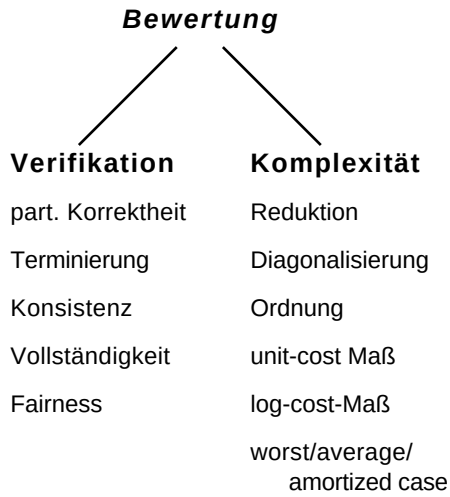

Abb. 1: Fundamentale Ideen Im Bereich der Bewertung von Algorithmen

In den folgenden drei Unterrichtsbeispielen wollen wir diese Behauptung im Ansatz nachweisen und zeigen, wie zwei zentrale Aspekte der Verifikation, namentlich die partielle Korrektheit und die Terminierung eines Algorithmus, auf den jeweiligen kognitiven Niveaus der Primarstufe und den Sekundarstufen I und II vermittelt bzw.

spiralig vertieft werden können. Zugleich untermauern diese Beispiele die Leistungsfähigkeit eines an fundamentalen Ideen orientierten Unterrichts.

Beispiel 1: Das Bohnenproblem

Thema der Unterrichtseinheit: Gegeben sind zwei Urnen, eine *Spielurne* gefüllt mit weißen und schwarzen Bohnen und eine *Vorratsurne* gefüllt mit einer (theoretisch) unbegrenzten Menge von schwarzen Bohnen (Abb. 2). Ein Spieler verändert den Inhalt der Urnen durch eine Folge von Spielzügen. Jeder Zug verläuft wie folgt:

> Ziehe blind zwei Bohnen aus der Spielurne.
> Falls sie die gleiche Farbe besitzen, wirf beide weg und lege eine Bohne aus der Vorratsurne in die Spielurne.
> Anderenfalls wirf die schwarze Bohne weg und gib die weiße zurück in die Spielurne.

Das Verfahren endet, wenn sich nur noch eine Bohne in der Spielurne befindet.
Dieses Problem stammt aus D. Gries 1989 und wird dort "Coffee Can Problem" genannt.

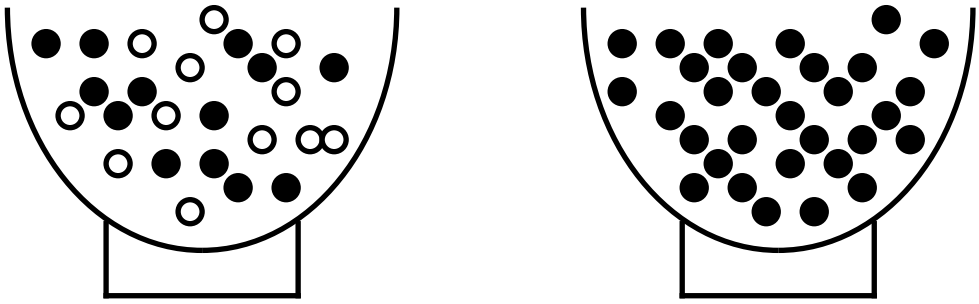

Abb. 2: Bohnenspiel

Problem 1: Endet das Spiel für jede Anfangsbelegung der Spielurne?

Problem 2: Wenn das Spiel endet, welche Farbe besitzt dann die letzte Bohne in der Spielurne?

Lösung zu 1: Mit jedem Spielzug werden zunächst zwei Bohnen aus der Spielurne entfernt und anschließend eine wieder zurückgelegt. Die Zahl der Bohnen in der Spielurne vermindert sich also je Zug um Eins. Befinden sich also anfangs n Bohnen in der Spielurne, so ist das Spiel nach n-1 Zügen beendet.

Lösung zu 2: Über die Zahl der schwarzen Bohnen im Verlaufe eines Spiels ist keine zweckdienliche Aussage möglich, da das Verfahren nichtdeterministisch ist. Zwischenzustände hängen also bei gleichen Startbedingungen vom Zufall ab, da blind gezogen wird. Für die Zahl der weißen Bohnen gilt jedoch: Entweder vermindert sie sich in einem Zug um 2, wenn zwei weiße Bohnen gezogen wurden, oder aber sie

bleibt konstant in allen übrigen Fällen. Also wird die *Parität* der weißen Bohnen durch einen Spielzug *nicht* beeinflußt. Folglich ist die letzte Bohne weiß bzw. schwarz, wenn die Zahl der weißen Bohnen anfangs ungerade bzw. gerade war.

Man beachte, daß das Verfahren zwar nichtdeterministisch, aber determiniert ist, also bei gleichen Startbedingungen unterschiedliche Zustandsfolgen durchlaufen kann, die aber immer zum gleichen Ergebnis führen.

Sachanalyse: Aus informatischer Sicht beschreibt die Aufgabenstellung einen nichtdeterministischen Algorithmus, dessen berechnete Funktion zu ermitteln ist. Zum Beweis der totalen Korrektheit des Algorithmus bzgl. des vermuteten funktionalen Zusammenhangs zwischen der Zahl der weißen Bohnen zu Beginn und der Farbe der letzten Bohne ist der Nachweis der Terminierung des Verfahrens sowie der partiellen Korrektheit erforderlich. Beides ist oben schon gezeigt worden. Der Nachweis der partiellen Korrektheit beruht dabei auf der Konstruktion einer Schleifeninvariante, also einer Eigenschaft, die durch die Ausführung eines Spielzugs nicht verändert wird. Schleifeninvariante ist hier die Parität der weißen Bohnen (gerade oder ungerade).

Zielgruppe: Schüler der Primarstufe oder der Sekundarstufe I.

Dauer der Unterrichtseinheit: ca. zwei bis drei Doppelstunden.

Lernziele:

- Einfache Algorithmen nachvollziehen können.
- Die Ideen von Alternative und Iteration entwickeln.
- Einführung in die Verifikation, speziell die Terminierung.
- Präfiguration von Schleifeninvarianten.
- Präfiguration von Nichtdeterminismus und Determiniertheit durch handlungsorientiertes (*enaktives*) vorwegnehmendes, dem kognitiven Niveau der Schüler entsprechendes Lernen.
- Vorbereitung des Übergangs vom empirisch-induktiven zum hypothetisch-deduktiven Problemlösen.

Voraussetzungen: Vier Grundrechenarten, Begriff der geraden und ungeraden Zahl.

Materialien: Ein Urnensatz für je zwei Schüler, je Schüler ca. 6 Protokollblätter (s.u.).

Möglicher Unterrichtsverlauf:

Der Lehrer schreibt die Spielregeln ohne die Abbruchbedingung an die Tafel und führt anschließend einige Spielzüge durch. Dann werden die Schüler in Zweiergruppen

aufgeteilt. Jede Gruppe erhält einen Urnensatz sowie mehrere Protokollblätter der Form aus Abb. 3, auf denen die Spielzüge aufgezeichnet werden können.

Name: _____ Nr. des Spiels: _____

Wieviele weiße Bohnen waren zu Beginn des Spiels in der Spielurne: _____

Wieviele schwarze Bohnen waren zu Beginn des Spiels in der Spielurne: _____

Nr. des Zuges	Welches Bohnenpaar wurde gezogen? Kreuze an.		
	zwei weiße	zwei schwarze	eine weiße und eine schwarze

Welche Farbe hat die letzte Bohne in der Spielurne? _____

Abb. 3: Protokollblätter

Anschließend führen die beiden Schüler jeder Gruppe abwechselnd je zwei bis drei Spiele durch und protokollieren Anfangsbedingungen und Verläufe der Spiele. Jeder Spieler kann sich pro Spiel aus einem zentralen Behälter eine beliebige Kombination von weißen und schwarzen Kugeln zusammenstellen.

Nach diesen Experimenten beginnt die Analyse des Spiels. Zunächst ist zu klären, wann das Spiel beendet ist. Hierzu können etwa die Schüler befragt werden, die mit dem Ausruf "fertig" auf sich aufmerksam gemacht haben: "Warum meint ihr, euer Spiel sei beendet?" Möglicherweise haben die Schüler bereits während der Experimente Überlegungen angestellt, was denn zu tun sein, wenn sich nur noch eine Bohne in der Spielurne befindet, wo man doch immer zwei herausnehmen muß. In diesem Fall problematisiert man die Fragestellung an Ort und Stelle. Die Schüler sollen bei den Überlegungen erkennen, daß Verfahren vollständig definiert werden müssen und keine Ungenauigkeiten enthalten dürfen. Nun wird die Bedingung "Spielende, falls weniger als zwei Bohnen in der Spielurne enthalten sind" zu den an die Tafel geschriebenen Spielregeln hinzugefügt.

Es folgen weitergehende Analysen unter Verwendung der Spielprotokolle:

Wird das Spielende immer erreicht?

Da alle Zweiergruppen ihre Spiele beendet haben, werden die Schüler schnell die Vermutung äußern, daß jedes Spiel irgendwann einmal endet. Zur Begründung werden sie die Beobachtung heranziehen, daß die Bohnenzahl der Spielurne fortwährend kleiner wird. Diese Vermutung ist nun genauer zu begründen, indem die Zugregel daraufhin überprüft wird, wie sich ein Spielzug auf die Zahl der Bohnen in der Spielurne auswirkt. Diese Untersuchungen legen den Grundstein zur Beantwortung der Frage:

Nach wievielen Zügen endet das Spiel?

Möglicherweise führen die Vorüberlegungen schon direkt zum richtigen Ergebnis:

"Zugzahl eines Spiels"="Anzahl der weißen Bohnen zu Beginn" +
"Anzahl der schwarzen Bohnen zu Beginn" - 1.

Anderenfalls fordert man die Schüler auf, ihre Protokollzettel tabellarisch an der Tafel zusammenstellen, etwa in der Form aus Abb. 4.

Name	Nr. des Spiels	Anzahl der weißen Bohnen in der Spielurne zu Beginn des Spiels	Anzahl der schwarzen Bohnen in der Spielurne zu Beginn des Spiels	Anzahl der Züge bis zum Spielende

Abb. 4: Tafelprotokoll

Sodann regt man die Schüler an, über Beziehungen zwischen den Zahlenwerten nachzudenken. Die erwartete Vermutung ist wiederum anhand der Spielregeln zu verifizieren. Hier ist Wert auf eine exakte Begründung zu legen, z.B.:

> "In beiden Alternativen eines Spielzugs entfernt man immer erst zwei Bohnen aus der Spielurne und legt dann eine wieder hinein, so daß sich die Zahl der Bohnen jeweils um Eins vermindert. Eine Bohne verbleibt am Schluß des Spiels in der Spielurne."

Die letzte Frage erfordert umfangreichere Überlegungen:

Welche Farbe hat die Bohne, die am Schluß des Spiels in der Spielurne verbleibt?

Zunächst läßt man die Schüler die ggf. zuvor an der Tafel erstellte Tabelle aller Spiele um eine Spalte für die Farbe der jeweils letzten Bohne ergänzen. Wiederum sollen die Schüler über funktionale Beziehungen zwischen den Einträgen nachdenken. Anregende Fragen, die die Schüler zur Lösung geleiten können, sind in diesem Zusammenhang:

> Welche Elemente des Spiels bestimmen, welche Bohne zuletzt übrigbleibt?
> Ist es die Zugfolge?

Ist es die Zahl der weißen Bohnen?
Ist es die Zahl der schwarzen Bohnen?
Ist es die Zahl aller Bohnen zu Beginn des Spiels?

Durch Analyse der Protokolle, der Tabelle an der Tafel und ggf. durch Probespiele in Eigenarbeit und vor der ganzen Klasse lassen sich die jeweiligen Vermutungen verwerfen oder erhärten. Darüberhinaus kann durch Variation einzelner Parameter und Festhalten der übrigen ein funktionaler Zusammenhang zwischen Startsituation und Endsituation hergestellt und Parameter ausgesondert werden, die auf die Farbe der letzten Bohne keinen Einfluß haben. Schließlich sollten die Schüler die Anzahl der weißen Bohnen zu Beginn eines Spiels als maßgebende Größe identifizieren. In weiteren Versuchsspielen kann dann eine Wertetabelle durch Gegenüberstellung der anfänglichen Zahl weißer Kugeln zur Farbe der letzten Kugel aufgestellt werden. Hieraus läßt sich leicht die gewünschte Vermutung entwickeln, daß die Farbe der letzten Kugel genau dann weiß ist, wenn die Zahl der weißen Kugeln anfangs ungerade war. Diese letzten Schritte können von den Schülern (in Gruppenarbeit) weitgehend selbständig durchgeführt werden. Geringfügige Hilfen des Lehrers sollten genügen. Die Vermutung ist schließlich wiederum anhand der Zugregel genau zu überprüfen: Ein Spielzug ändert die Parität der weißen Kugeln nicht.

Nun sollte man die gesamte Unterrichtseinheit noch einmal Revue passieren lassen, wobei die Schüler den Ablauf erläutern. Hierbei kann man überprüfen, ob sie schon ein Gespür für die Vorgehensweise bei der Verifikation entwickelt haben (Abb. 5).

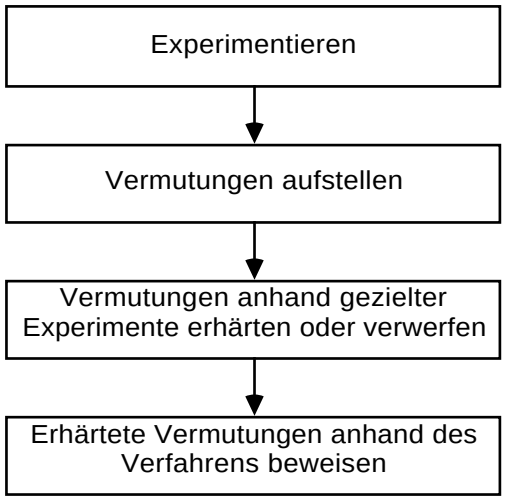

Abb. 5: Unterrichtliche Vorgehensweise

Eine Verinnerlichung dieser Strategien ist jedoch erst im Laufe der Zeit zu erwarten, wenn man die Thematik spiralig vertiefend mehrfach wieder aufgegriffen hat. Überdies befinden sich die Schüler erst im Übergang vom empirisch-induktiven zum hypothetisch-

deduktiven Problemlösen. Diese generelle Vorgehensweise beim Problemlösen ist ihnen also noch nicht geläufig und auch nur eingeschränkt vermittelbar.

Im weiteren Verlauf kann der Lehrer den Begriff des Nichtdeterminismus problematisieren: Bei gleichen Anfangsbedingungen können zwei Spieler ganz unterschiedliche Spielzüge durchführen, weil die Bohnen blind gezogen werden. Beide Spieler werden aber am Schluß jeweils genau eine Bohne in der Spielurne behalten, und diese beiden Bohnen besitzen die gleiche Farbe. Die Schüler können nun weitere Beispiele für Verfahren oder Situationen aus ihrer Lebenswelt aufzählen, in denen nichtdeterministische Effekte auftreten, die jedoch auf das Gesamtergebnis keinen Einfluß besitzen.

Beispiel 2: Die Collatz-Funktion

Thema der Unterrichtseinheit: Die Collatz-Funktion ist definiert durch

$$C: \mathbb{N} \to \mathbb{N} \text{ mit } C(n) = \begin{cases} n/2, & \text{falls } n \text{ gerade} \\ 3n+1, & \text{sonst.} \end{cases}$$

Es wird vermutet, daß es zu jedem $n \in \mathbb{N}$ ein $k \in \mathbb{N}$ gibt, so daß $C^k(n)=1$ ist. Bis heute ist kein formaler Beweis für diese Aussage bekannt. Die Aussage ist allerdings für alle natürlichen Zahlen bis etwa 10^{40} verifiziert.

Sachanalyse: Hintergrund der Collatz-Funktion bilden die sog. *Conway-Spiele* (benannt nach J.H. Conway). Idee dieser Spiele ist das wiederholte Anwenden einer einfachen arithmetischen Funktion, die durch endlich viele Alternativen definiert ist. Formal:

Definition:
Ein Zweitupel (R,g) heißt *Conway-Spiel* ($k \in \mathbb{N}$), wenn gilt:

(1) $R=((a_1,b_1),...,(a_k,b_k)) \in (\mathbb{Q}_+ \times \mathbb{Q}_+)^k$ ist ein k-Tupel von Paaren positiver rationaler Zahlen;

(2) $g: \mathbb{N} \to \mathbb{N}$ ist eine Funktion mit

$$g(n) = \begin{cases} a_j \cdot n + b_j \text{ mit } j=\min\{i \mid a_j \cdot n + b_j \in \mathbb{N}\}, & \text{falls } j \text{ existiert,} \\ \text{undefiniert,} & \text{sonst.} \end{cases}$$

Man wählt also bei einem Funktionsaufruf von g das Paar (a_j,b_j) mit kleinstem Index j, für das $a_j \cdot n + b_j$ eine natürliche Zahl ist, sofern solch ein j existiert.

Beispiel: Sei k=2 und R=(($^1/_2$,0),(3,1)). Dann beschreibt (R,g) die Collatz-Funktion C.

Interessant werden Conway-Spiele, wenn man ihre sog. *Kontraktionseigenschaft* untersucht, also ihr Verhalten bei wiederholter Anwendung.

Definition:
Ein Conway-Spiel (R,g) heißt *kontrahierbar* auf 1, falls für jedes n IN eine der beiden folgenden Bedingungen gilt:
(1) g(n)=n,
(2) Es gibt ein k IN mit $g^k(n)=1$.

Die Kontraktionseigenschaft ist algorithmisch nicht entscheidbar.

Satz (J.H. Conway 1972):
Es gibt keinen Algorithmus, der zu jedem Conway-Spiel (R,g) entscheidet, ob (R,g) kontrahierbar ist.

Die Kontraktionseigenschaft von C ist auch noch nicht genau bekannt. Der folgende Satz läßt jedoch eine gewisse Kontraktion vermuten.

Satz (C.J. Everett 1977):
Für fast alle n IN gibt es ein k IN mit $C^k(n)<n$.

Zielgruppe: Schüler der Sekundarstufe I. Bezieht man sich nicht auf eine konkrete Programmiersprache, sondern beläßt es bei einem Algorithmus in umgangssprachlicher Notation, so eignet sich diese Einheit auch für die Primarstufe.

Dauer der Unterrichtseinheit: etwa zwei Doppelstunden.

Lernziele:
- Weiterentwicklung der fundamentalen informatischen Ideen von *Alternative* und *Iteration*.
- Erstmalige Verwendung der PASCAL-Sprachelemente für bedingte Anweisungen und bedingte Schleifen.
- Vertiefung der Idee der *Terminierung* im Zusammenhang mit bedingten Schleifen.

Voraussetzungen: Kenntnis der Zuweisung, der Sequenz und des Datentyps integer mit seinen wichtigsten Operationen.

Möglicher Unterrichtsverlauf:

Der Lehrer gibt folgende Zahlenfolge vor

(A) 17 52 26 13 40 20 10 5 ...

und bittet die Schüler herauszufinden, welches Bildungsgesetz sich hinter der Folge verbirgt und die Folge korrekt fortzusetzen. Erfahrungsgemäß wird es hierbei noch Schwierigkeiten geben, da die Folge zur Erkennung eines Bildungsprinzips noch zu kurz ist. Daher setzt der Lehrer fort:

... 16 8 4 2 1 4 2 1 4 2 1 ...

Bei der zweiten Folge mit gleichem Bildungsgesetz wird es bereits einige Vorschläge geben, wie man fortzusetzen hat:

(B) 37 112 56 28 14 7 22 11 34 17 52 ...

Die meisten Schüler werden wie in Folge (A) fortsetzen, da die Zahlen 17 und 52 dem Beginn der Folge (A) entsprechen. Hierbei setzen sie stillschweigend voraus, daß zur Berechnung eines Elements der Folge nur das unmittelbar vorhergehende heranzuziehen ist. Auch wenn diese Annahme im allgemeinen bei solchen Folgenspielen nicht zutreffen mag, empfiehlt es sich hier auf eine diesbezügliche Bemerkung zu verzichten, um den Verlauf der Diskussion nicht zu unterbrechen.

Die dritte Folge schließlich offenbart den meisten Schülern das Bildungsgesetz:

(C) 30 15 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1 4 2 1 ...

Die vierte Folge dient zur Erhärtung der Vermutung:

(D) 75 226 113 340 170 85 256 128 64 32 16 8 4 2 1 4 2 1 ...

Man läßt das Bildungsgesetz nun für alle von einem Schüler an die Tafel schreiben. Das Ergebnis könnte etwa sein:

a) umgangssprachlich:

Wenn die letzte Zahl eine gerade Zahl ist, dann halbiere sie
und erhalte die nächste;
Wenn sie ungerade ist, dann multipliziere sie mit Drei und addiere Eins.

b) formal: Sei x_1, x_2, x_3, ... die gesuchte Folge. x_1 ist gegeben. Dann berechnet sich x_i für $i \geq 2$ durch

$$x_i = \begin{cases} x_{i-1}/2, & \text{falls } x_{i-1} \text{ gerade,} \\ 3x_{i-1}+1, & \text{sonst.} \end{cases}$$

Einige weitere Übungen mit unterschiedlichen Anfangswerten x_1 werden die Schüler recht schnell zur Vermutung führen, daß alle Folgen schließlich in ... 4 2 1 4 2 1 ... übergehen. Diese Vermutung kann der Lehrer im allgemeinen scheinbar entkräften, indem er z.B. den Anfangswert $x_1=27$ vorgibt:

27 82 41 124 62 31 94 47 142 71 214 107
322 161 484 242 121 364 182 91 274 137 412
206 103 310 155 466 233 700 350 175 526 263
790 395 1186 593 1780 890 445 1336 668 334 167

502 251 754 377 1132 566 283 850 425 1276 638
319 958 479 1438 719 2158 1079 **3238 1619 4858**
2429 7288 3644 1822 911 2734 1367 4102 2051 6154
3077 9232 4616 2308 1154 577 1732 866 433 1300
650 325 976 488 244 122 61 184 92 46 23 70
35 106 53 160 80 40 20 10 5 16 8 4 2 1...

Wenn die Schüler beim Rechnen in den fett markierten Zahlenbereich gelangen, werden sie zunächst den Glauben an ihre Vermutung verlieren. Da die Berechnung der Folgenglieder nun schon recht aufwendig geworden ist, wird leicht der Wunsch nach Programmierung der Berechnung auf einem Computer geweckt.

Zunächst wird die Aufgabe des Algorithmus genau spezifiziert: Er soll eine natürliche Zahl einlesen und nacheinander alle Zahlen der Folge nach obiger Formel berechnen und ausgeben.

Man beginnt zunächst mit einem umgangssprachlichen Algorithmus, den die Schüler selbst entwickeln können (man sollte sich hierbei nach der Ausdrucksweise der Schüler richten, meist hat sich allerdings im vorangegangenen Unterricht schon ein Sprachgebrauch etabliert), z.B.:

Eingabe einer Zahl x;
Ausgabe von x;
Wiederhole:
    Wenn x gerade ist, dann setze x auf x/2;
    Wenn x ungerade ist, dann setze x auf 3x+1;
    Ausgabe von x.

Als nächstes ist herauszuarbeiten, welche bereits bekannten und welche neuen Elemente in diesem Algorithmus vorkommen. Welche Teile können in PASCAL bereits dargestellt werden? Offenbar:

Eingabe einer Zahl x --> read(x)
Ausgabe von x --> write(x)
setze x auf x/2 --> x:=x div 2
setze x auf 3x+1 --> x:=3*x+1

Die beiden neuen Kontrollstrukturen "Wenn ... dann ..." und "Wiederhole ..." werden als *bedingte Anweisung* und als *Schleife* identifiziert. Mit bedingten Anweisungen kann man (hier zunächst einseitige) *Alternativen*, mit Schleifen *Iterationen* formulieren.
Wo treten Alternativen im täglichen Leben auf? Z.B. bei der Wahl einer Busverbindung, bei der Kurswahl zu Beginn eines Schuljahres, bei der Wahl eines Mopeds, der Ausbildungsstelle.

Wo findet man Iterationen? Z.B. als Reim, Rhythmus, Refrain, Ornament in Sprache, Musik und Kunst, bei der Autowaschanlage, die zyklisch dasselbe Waschprogramm abfährt, bei der Unterrichtswoche mit ständig wiederkehrenden Fächern, beim Tagesrhythmus der Schüler und Eltern.

An dieser Stelle empfiehlt es sich, auf die fundamentale informatische Idee der *Hierarchisierung* durch *Schachtelung* und ihre Darstellung (hier: durch Einrückung) hinzuweisen. Als Aufhänger kann etwa die Frage dienen: Welche Anweisungen gehören zur Schleife, welche nicht?

Der Lehrer kann nun einen Schüler auffordern, den Algorithmus etwa für die Zahl 5 nachzuvollziehen: 5 16 8 4 2 1 4 2 1 4 2 1; spätestens an dieser Stelle wird den Schülern die Problematik klar: es wurde eine *Endlosschleife* definiert. Das Programm *terminiert* (zumindest für die gewählte Eingabe) nicht. Eine sprachliche Präzisierung des Begriffs Termination für einzelne/alle Eingaben kann sich anschließen.
Der Lehrer rät nun dazu, den obigen Algorithmus so abzuändern, daß die Erzeugung von neuen Folgengliedern nur *solange* fortgesetzt wird, wie x *noch ungleich* 1 ist, da man anderenfalls in den bekannten Zyklus 4 2 1 4 2 1 ... hineinläuft. Die Schüler werden auf diese Anregung hin vermutlich genau auf eine umgangssprachliche Fassung der while-Schleife kommen:

Solange x≠1 ist, wiederhole: ... .

Der Begriff der *bedingten Schleife* wird nun präzisiert und ggf. an einigen kleineren ad hoc-Beispielen (wie Summe der ersten n Zahlen) weiter erläutert.

Eine nochmalige Überprüfung des Algorithmus für den Eingabewert 5 liefert jetzt ein zufriedenstellendes Ergebnis. Terminiert das Programm nun für *alle* Eingaben? Eine Antwort hierauf wird vorerst bis zum Schluß der Unterrichtseinheit zurückgestellt.

Es folgt eine Beschreibung der Syntax von (einseitiger) bedingter Anweisung und Schleife in PASCAL. Hierbei ist ggf. zu erklären, daß der Anweisungsteil der beiden Strukturen aus einzelnen Elementaranweisungen oder einer durch begin/end geklammerten Sequenz bestehen kann. Auf die exakte Bedeutung dieser beiden Kontrollstrukturen und die möglichen Ausprägungen der Bedingungen bei Alternative und Schleife braucht zu diesem Zeitpunkt noch nicht eingegangen zu werden, da die Schüler den Sinn intuitiv begreifen können. Einzelne Details können zu passender Gelegenheit (s. unten) ergänzt werden (genetisches Unterrichtsprinzip).

Nun entwickeln die Schüler leicht folgendes Programm:

```
program folge1(input,output);
var x:integer;
begin
```

```
        read(x); write(x);
        while x≠1 do
        begin
            if x mod 2=0 then x:=x div 2;
            if x mod 2≠0 then x:=3x+1;
            write(x)
        end
    end.
```

Das Programm ist in dieser Form mit den Vorbemerkungen selbsterklärend. Der Lehrer kann auf eine Abkürzung hinweisen: Die beiden if-Anweisungen, deren Bedingungen sich gegenseitig ausschließen, können in einer Anweisung (der zweiseitigen Alternative) zusammengefaßt werden:

```
if x mod 2=0 then x:=x div 2 else x:=3x+1.
```

Auch die Bedeutung dieser Kontrollstruktur können die Schüler intuitiv erfassen.

Nach einigen Experimenten am Rechner schreibt der Lehrer einen Wettbewerb aus: Wer findet die längste Folge bei einer Anfangszahl zwischen 1 und 1000? Die Schüler werden nun nach kurzer Zeit, statt die Elemente jeder Folge abzuzählen, selbständig zusätzlich eine Zählervariable einführen, in der für den eingegebenen Anfangswert die Länge der jeweiligen Folge festgehalten wird:

```
program folge1(input,output);
var x,laenge:integer;
begin
    read(x); write(x);
    laenge:=1;
    while x≠1 do
    begin
        if x mod 2=0 then x:=x div 2 else x:=3x+1;
        write(x);
        laenge:=laenge+1
    end;
    writeln('Länge der Folge',laenge)
end.
```

Nach weiteren Testläufen wird sich der Wunsch einstellen, alle Anfangswerte zwischen 1 und 1000 systematisch durchzuprobieren, um denjenigen zu finden, der die längste Folge liefert. Wiederum wird zunächst ein umgangssprachlicher Algorithmus angefertigt:

Für jeden Anfangswert a zwischen 1 und 1000 tue folgendes:

    Ausgabe von a;

    Setze x auf a;

    Setze laenge auf 1;

    Solange x≠1 ist, wiederhole:

        Wenn x gerade ist, dann setze x auf x/2, anderenfalls setze x auf 3x+1;

        Erhöhe laenge um 1;

    Ausgabe von laenge.

Nun kann man entweder diesen Ansatz aufgreifen und die Zählschleife einführen oder aber die Schüler anregen, den Algorithmus so umzuschreiben, daß nur bereits bekannte programmiersprachliche Elemente verwendet werden können.
Wiederum ist auf die Idee der textuellen Hierarchisierung/Schachtelung durch Einrückung hinzuweisen, aus der sich jeweils genau die Reichweite beider Schleifen ergibt.

Hausaufgabe:
Setze den Algorithmus in ein PASCAL-Programm folge2 um und bestimme den Anfangswert zwischen 1 und 1000, der die längste Folge liefert.

Die obigen Überlegungen zur *Terminierung* des Programms werden nun in der nächsten Stunde wieder aufgegriffen. Geeignete Fragen zur Motivierung sind:
- Wie ist die Anfangsvermutung "Alle Folgen enden bei Eins bzw. mit 1 4 2 1 4 2 ..." nach diesen Experimenten zu entscheiden?
- Wie reagiert das Programm folge1, wenn man eine Zahl eingibt, deren Folge nicht bei 1 endet?
- Wie ist zu bewerten, wenn der Computer bei der Ausführung des Programms folge1 ungewöhnlich lange rechnet, also z.B. nach zwei Stunden immer noch keine Eins ausgegeben hat?
- Terminiert das Programm folge1 für alle Eingaben?

Zum Schluß der Diskussion kann auf die wissenschaftliche Attraktivität dieser Fragestellung eingegangen werden:
- Hinter der Vermutung verbirgt sich ein schwieriges zahlentheoretisches Problem.
- Es ist nicht bekannt, ob alle Folgen bei Eins enden.
- Die Vermutung ist für alle Zahlen bis etwa 10^{40} nachgewiesen.

Was bedeuten diese Aussagen für die Terminierung des Programms folge1?

Beispiel 3: Formaler Terminationsnachweis

Thema der Unterrichtseinheit: Untersuchung der Terminierung des folgenden Programmstücks:

```
var x,y: integer;
while y≠0 do
    if x=0 then y:=y-1 else
    begin
        y:=x+1;
        x:=x-1
    end
end.
```

Es ist zu beweisen, daß das Programmstück für beliebige Anfangswerte x,y $\mathbb{N}_0$ terminiert. Hierzu wählt man eine Funktion

$$\tau: \mathbb{Z}\times\mathbb{Z} \to \mathbb{Z}$$

und zeigt, daß sich der Wert von τ angewendet auf die Variablen x und y mit jedem Schleifendurchlauf verkleinert, gleichzeitig aber für x,y $\mathbb{N}_0$ nach unten durch 0 beschränkt ist. Eine geeignete Wahl für τ ist z.B.

$$\tau(x,y) = \begin{cases} y, & \text{falls } x=0 \\ x+2, & \text{falls } x\neq 0. \end{cases}$$

Offenbar gilt dann $\tau(x,y)\geq 0$ für alle x,y $\mathbb{N}_0$. Nun ist zu zeigen, daß sich der Wert von τ mit jedem Schleifendurchlauf vermindert. Seien zu Beginn des Schleifendurchlaufs

$x=\alpha$ und $y=\beta$ mit α,β $\mathbb{N}_0$.

Da die Abbruchbedingung dann nicht erfüllt ist, folgt $\beta\neq 0$, also $\beta>0$.

Fall 1: $\alpha=0$.

Dann ist $\tau(x,y)=\beta$ vor dem Schleifendurchlauf und $\tau(x,y)=\beta-1$ nach dem Schleifendurchlauf.

Fall 2: $\alpha\neq 0$.

Fall 2.1: $\alpha>1$.

Dann ist $\tau(x,y)=\alpha+2$ vor dem Schleifendurchlauf und $\tau(x,y)=\alpha-1$ nach dem Schleifendurchlauf.

Fall 2.2: $\alpha=1$.

Dann ist $\tau(x,y)=3$ vor dem Schleifendurchlauf und $\tau(x,y)=2$ nach dem Schleifendurchlauf.

Damit ist gezeigt, daß τ streng monoton fällt. Folglich terminiert die Schleife für x,y $\mathbb{N}_0$.

Sachanalyse: Den Nachweis der Terminierung von Programmen kann man nicht maschinell durchführen, genauer: Es gibt keinen Algorithmus, der für beliebige Programme nachprüft, ob sie für alle Eingaben terminieren oder nicht (*Halteproblem*). Zum Nachweis der Terminierung eines Programms ist also stets eine gewisse Handarbeit und die Intuition eines Menschen erforderlich.

Offenbar kann ein Programm nur terminieren, wenn zumindest alle Schleifen terminieren. Ein recht praktikables und häufig (aber nicht immer) zum Ziel führendes Verfahren, die Termination einer Schleife nachzuweisen, besteht darin, eine Funktion τ von den Wertemengen der in der Schleife vorkommenden Variablen in die Menge der ganzen Zahlen zu ermitteln, für die gilt: τ ist nach unten beschränkt, und τ verkleinert sich mit jedem Schleifendurchlauf. Findet man τ, so terminiert die Schleife.

Genauer: Seien $x_1,...,x_r$ die in einer Schleife vorkommenden Variablen mit den Wertebereichen $W_1,...,W_r$. Man sucht eine Funktion

$$\tau: W_1 \times ... \times W_r \to \mathbb{Z}$$

mit

(1) $\tau(x_1,...,x_r) \geq 0$ für alle $x_i \in W_i$, $i=1,...,r$.

(2) Gilt vor Ausführung des Schleifenrumpfes $\tau(x_1,...,x_r)=t>0$, so gilt nach Ausführung des Rumpfes $\tau(x_1,...,x_r)<t$.

Den Nachweis von (2) führt man im allgemeinen Fall mit Methoden der axiomatischen Semantik. Für die Schule sind solch hohe Anforderungen an die formale Strenge entbehrlich. Hier genügt es die Wirkung des Schleifenrumpfes auf die Variablen und den Wert von τ mithilfe von Ablaufprotokollen zu bestimmen.

Zielgruppe: Schüler der Sekundarstufe II.

Dauer der Unterrichtseinheit: ca. zwei bis drei Doppelstunden.

Lernziele: Vertiefung der fundamentalen Idee der Terminierung durch Formalisierungsansätze.

Voraussetzungen: Kenntnis von Ablaufprotokollen, des Begriffs der Terminierung sowie verschiedener mathematischer Grundbegriffe wie Monotonie und Beschränktheit von Funktionen.

Materialien: Um Denkfähigkeit und Abstraktionsvermögen zu trainieren sowie Ablenkungen möglichst gering zu halten, wird der Rechner *nicht* genutzt.

Möglicher Unterrichtsverlauf:

Der Lehrer schreibt das Programmstück an die Tafel und fordert die Schüler auf, Anfangswerte(-mengen) von x und y zu nennen, für die das Programm terminiert. Die Zuhilfenahme des Rechners ist nicht erlaubt. Für jeden genannten Anfangswert bzw. jede genannte Menge von Anfangswerten verlangt man vom Schüler eine Begründung, warum das Programm für diese Werte terminiert. Als Vorstufe zu einer späteren Formalisierung sollen die Schüler hier zu einer möglichst präzisen Erläuterung angehalten werden. Wertebereich und Begründung werden tabellarisch an der Tafel festgehalten, z.B. wie in Abb. 6.

Anfangswert oder Anfangswertemenge für x und y	Begründung
y=0	Die Schleife wird gar nicht betreten.
x=0 und y>0	Die Schleife wird betreten, wobei nur der then-Zweig durchlaufen wird. Hier wird y fortlaufend um 1 vermindert, bis y=0 ist und das Abbruchkriterium erreicht ist.
x>0 und y>0	Die Schleife wird betreten, wobei zuerst im else-Zweig x um 1 heruntergezählt wird, bis x=0 ist (zu diesem Zeitpunkt ist y=2). Dann weiter wie im Fall x=0, y>0.
x=-1 und y≠0	…
x>0 und y≠0	…

Abb. 6: Terminationsbereiche (mögliches Tafelbild)

Zur Veranschaulichung und um eine bessere Übersicht zu behalten, welche Eingabepaare (x,y) bereits untersucht worden sind, kann man die genannten Anfangswerte(-mengen) zusätzlich in einem Koordinatensystem festhalten (Abb. 7).

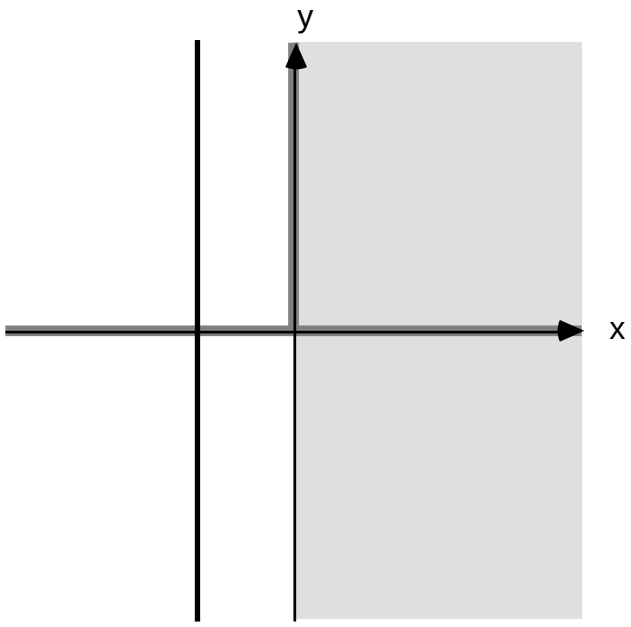

Abb. 7: Graphische Darstellung der Terminationsmenge

Schließlich wird durch Zusammenfassung aller Anfangswertemengen der mutmaßliche Terminationsbereich T des Programmstücks ermittelt:

$$T=\{(x,y) \mid x>0 \text{ oder } x=-1 \text{ oder } y=0 \text{ oder } (x=0 \text{ und } y>0)\}.$$

Nun sind die Überlegungen zu präzisieren und zu formalisieren. Der Einfachheit halber beschränkt man sich zunächst auf Anfangswerte x,y≥0. Mittels folgender motivierender Fragen kann man die Schüler zu einer Lösung geleiten:

- Was ist die notwendige Bedingung für die Termination der Schleife?
  Erwartete Antwort: Die Abbruchbedingung y=0 muß erreicht werden.
- Auf welche Weise erreicht man von den Anfangswerten die Abbruchbedingung?
  Erwartete Antwort: y muß fortlaufend vermindert werden.
- Ist dies denn der Fall, wird y also in der Schleife immer vermindert, z.B. für (x,y)=(17,1)?
  Erwartete Antwort: Nein, y kann sich im else-Zweig auch erhöhen, aber dafür wird ja dann x vermindert. Zwischenzeitliche Erhöhungen von y schaden also nicht. y muß sich nur auf lange Sicht verringern.
- Wie kann man formulieren, daß man sich trotz zwischenzeitlicher Erhöhungen von y dennoch der Abbruchbedingung y=0 nähert?
  Erwartete Antwort: Es ist nicht nur der Werteverlauf von y maßgebend, sondern man muß auch berücksichtigen, daß sich x vermindert, was dann im else-Zweig dafür sorgt, daß sich später trotz zwischenzeitlicher Erhöhungen auch y wieder vermindert.
- Wie kann man das mathematisch formulieren?

An dieser Stelle wird der geführte Dialog vermutlich abbrechen, da die Schüler nicht in der Lage sind, die beiden Beobachtungen zur gewünschten Funktion τ zu kombinieren. Als weitere Hilfe kann man evtl. für gewisse Anfangswerte (x,y) den Werteverlauf im Koordinatensystem darstellen, z.B. für (x,y)=(4,2) und (x,y)=(3,5) wie in Abb. 8.

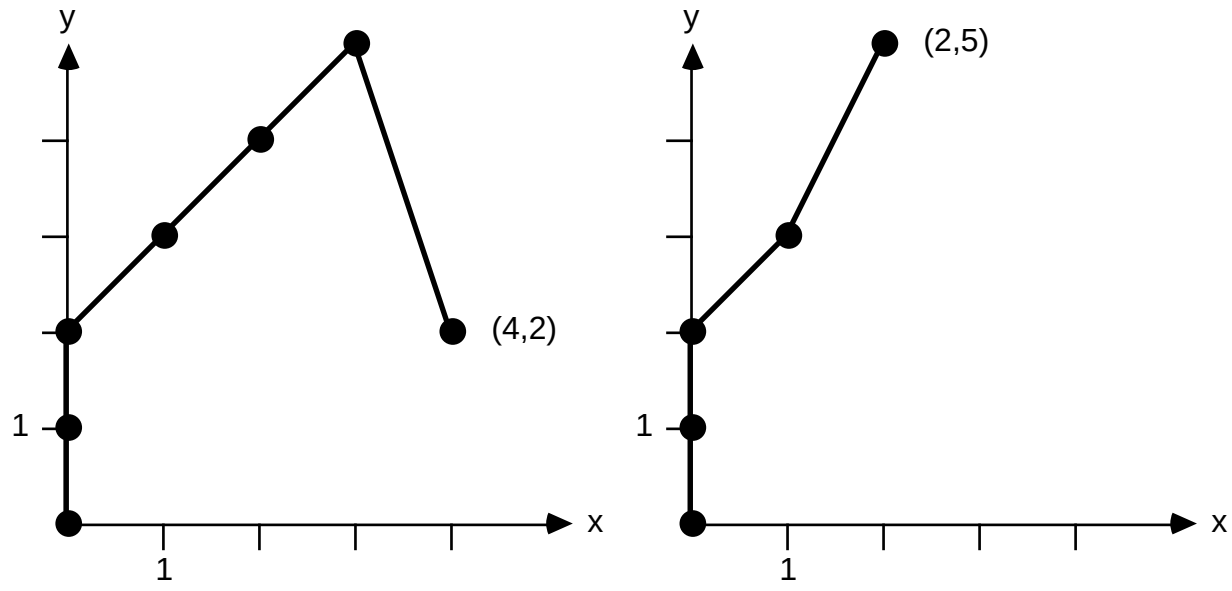

Abb. 8: Werteverläufe der Variablen x und y

Der Lehrer kann nun weiterhelfen und die Funktion

$$\tau(x,y) = \begin{cases} y, & \text{falls } x=0 \\ x+2, & \text{falls } x\neq 0 \end{cases}$$

anschreiben. Danach befragt man die Schüler nach den Eigenschaften von τ für x,y≥0 und ihrem Nutzen für den Nachweis der Terminierung der Schleife. Zur Unterstützung können die Schüler das Programmstück für einzelne Anfangswerte x,y≥0 nachvollziehen, Ablaufprotokolle hierzu anfertigen und den Verlauf von τ beobachten sowie alles im Koordinatensystem festhalten. Diese Aufgabe eignet sich auch als Hausaufgabe zwischen den beiden Doppelstunden.

Erwartete Antworten:

(1) τ ist nie negativ, falls die Argumente nicht negativ sind.

(2) τ verringert sich mit jedem Schleifendurchlauf um Eins.

(3) τ hat den Wert 0, wenn die Abbruchbedingung erreicht ist.

τ beschreibt also die Beobachtung, daß die Dekrementierung sowohl von x als auch von y zum Erreichen der Abbruchbedingung y=0 beiträgt.

Diese drei Aussagen sind jetzt allgemein zu verifizieren. Aussagen (1) und (3) sind einfach. Für Aussage (2) verwendet man wiederum Ablaufprotokolle: Sei zu Beginn des Schleifenrumpfes x=α und y=β, α,β≥0:

Fall 1:

x	y	τ	
α=0	β	β	
α=0	β-1	β-1	√

Fall 2:

x	y	τ	
α≥2	β	α+2	
α-1	α+1	α+1	√

Fall 3:

x	y	τ	
α=1	β	3	
0	2	2	√

Zwischenfrage zur Vertiefung des Verständnisses: Die obige Funktion τ sieht recht gekünstelt aus. Wie wäre es also, wenn man folgende einfachere Funktion τ' gewählt hätte:

$$\tau'(x,y) = \begin{cases} y, & \text{falls } x=0 \\ x, & \text{falls } x\neq 0 \ ? \end{cases}$$

Erwartete Antwort ggf. unter Verwendung von Ablaufprotokollen: Dieses τ' eignet sich nicht, da beim Übergang vom else- zum then-Zweig für x=1, y≥0 der Wert von τ' von 1 auf 2 springt, sich also nicht vermindert.
Im letzten Schritt sind nun die bisherigen Überlegungen zu verallgemeinern. Wie kann man im allgemeinen Fall nachweisen, ob eine Schleife terminiert? Erwartet werden mit Unterstützung des Lehrers die im Abschnitt "Sachanalyse" genannten Kriterien.
In einer Hausaufgabe können die Schüler dann versuchen, eine entsprechende Funktion τ für den gesamten mutmaßlichen Terminationsbereich T aufzustellen. Abschließend kann ggf. in einer weiteren Doppelstunde auf das Halteproblem eingegangen werden.

Dr. Andreas Schwill
Fachbereich Mathematik/Informatik
Universität - Gesamthochschule Paderborn
D-33095 Paderborn

Literatur

Conway, J.H.: Unpredictable iterations. Proceedings of the Number Theory Conference (1972) 45-52

Everett, C.J.: Iterations of the number-theoretic function f(2n)=n, f(2n+1)=3n+2. Advances in Mathematics 25 (1977) 42-45

Gries, D.: The Science of Programming. Springer Verlag 1989

Schwill, A.: Fundamentale Ideen der Informatik. Zentralblatt für Didaktik der Mathematik (1993) H.1, 20-31