Rekursive Strukturen in Einführungsvorlesungen der Informatik

Markus Schneider
Institut für Informatik, Technische Universität München
Boltzmannstr. 3, 85748 Garching
E-mail: markus.schneider@informatik.tu-muenchen.de

Abstract: Rekursive Funktionsaufrufe, reguläre Ausdrücke in BNF oder rekursive Datenstrukturen sind Themen, die in der Anfängervorlesung zur Informatik von großer Bedeutung sind. Das aus der Sicht der Informatik diesen Themen gemeinsame zentrale Konzept ist die Rekursion. In dieser Arbeit wird zunächst der Begriff der rekursiven Struktur syntaktisch definiert, die verschiedenen Formen kategorisiert und die Semantik mit Hilfe der Fixpunkttheorie mathematisch festgelegt. Aus dieser Analyse werden methodische und didaktische Schlussfolgerungen für die Vermittlung rekursiver Strukturen im Hochschulbereich gezogen. Andererseits wird diskutiert, wie rekursive Strukturen in Grundvorlesungen vermittelt werden und untersucht, inwieweit die hier beschriebenen Schlussfolgerungen mit einem typischen Vorlesungsentwurf verträglich sind.

1 Einleitung

Didaktische Betrachtungen zum Thema Rekursion sind Gegenstand zahlreicher Veröffentlichungen. Hierbei wird in der Regel untersucht, wie Rekursion, – insbesondere für rekursive Funktionen –, didaktisch sinnvoll vermittelt werden kann, welche Schwierigkeiten die Studenten dabei haben und wie diese Probleme gelöst werden können; entsprechende Beiträge finden sich beispielsweise bei Ginat und Shifroni [GS99], C.G. George [Geo00] oder Velazquez-Iturbe [VI00].

Die Intention der vorliegenden Arbeit ist anders. Die zentrale Frage, die hier betrachtet wird, ist, welche gemeinsamen Charakteristika die in BNF-Produktionen, in Sorten- und Funktionsdeklarationen auftretenden rekursiven Strukturen aufweisen und was allgemein das Konzept **Rekursion** aus der Sicht der Informatik kennzeichnet. Eine derartige Untersuchung ist zunächst von grundsätzlicher Bedeutung; letztendlich ergeben sich daraus aber auch Konsequenzen didaktischer und methodischer Art.

2 Syntaktische Beschreibung und Kategorisierung rekursiver Strukturen

Es ist zunächst zu klären, wie sich die Rekursion bei den hier betrachteten Strukturen syntaktisch äußert und allgemein beschreiben lässt. Anschließend wird versucht, die Rekursionstypen „lineare“, „nichtlineare“ und „verschränkte“ Rekursion allgemein zu definieren.

2.1 Typische rekursive Strukturen einer Grundvorlesung

Reguläre Ausdrücke in Backus Naur Form
Zunächst ist man geneigt, die durch Produktionen in BNF deklarierten regulären Ausdrücke dann als rekursiv zu bezeichnen, wenn die rechte Seite einer Produktion das zu deklarierende Nonterminal enthält. Dass diese Definition nicht ausreicht, zeigt folgendes Beispiel, das arithmetische Ausdrücke mittels eines verschränkt rekursiven Systems von Produktionen beschreibt:

```
<Ausdruck>  ::=  ['-'] <Term> (('+' | '-') <Term> )*
<Term>      ::=  <Faktor> ('*' <Faktor>)*
<Faktor>    ::=  '(' <Ausdruck> ')' | <Zahl>
```

(auf die Deklaration des Nonterminals `<Zahl>` wurde hier verzichtet). Abbildung 1 zeigt ein Diagramm, das ähnlich dem bekannten Stützgraphen für Systeme von rekursiven Funktionen, die rekursive Struktur der durch das Nonterminal `<Ausdruck>` deklarierten formalen Sprache veranschaulicht. Das Zeichen „+“ am Ende eines Pfeiles, bedeutet dabei, wie in der BNF-Notation, dass das entsprechende Nonterminal mindestens einmal auftritt. Das Dreieck symbolisiert die in der letzten Produktion auftretenden „Oder“-Verknüpfung. Man erkennt, dass der Graph in Abbildung 1 mehr als einen Zyklus enthalten kann; man

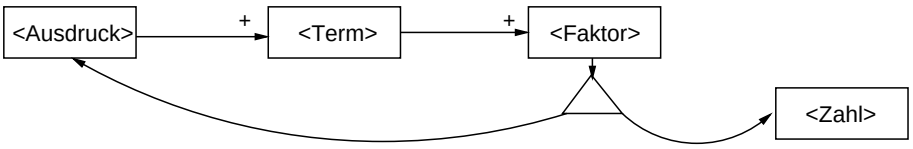

Abbildung 1: Syntaxgraph der durch `<Ausdruck>` deklarierten Sprache

kann somit die untersuchte Struktur als nichtlinear rekursiv bezeichnen.

Rekursive Sorten
Rekursive Sortendeklarationen sind, was ihre syntaktische Struktur betrifft, Produktionen in BNF-Notation sehr ähnlich. Auch hier genügt im Allgemeinen nicht die Festlegung, wonach eine Sorte rekursiv ist, wenn die rechte Seite der Deklaration die Sorte selbst enthält. Die rekursive Struktur komplexer Sortendeklarationen, wie etwa die folgende

Deklaration von Bäumen mit einer endlichen Anzahl von „Kindbäumen“ (vgl. [Bro98])

$$\begin{array}{lcl} \textbf{sort tree} & = & \text{emptyTree} \mid \text{treeConstructor}(\textbf{m} \text{ root}, \textbf{forest} \text{ children}) \\ \textbf{sort forest} & = & \text{emptyForest} \mid \text{forestConstructor}(\textbf{tree} \text{ first}, \textbf{forest} \text{ rest}) \end{array} \quad (1)$$

wird deutlich, wenn man die Zyklus im zugehörigen (vereinfachten) Signaturdiagramm (Abbildung 2) betrachtet: Es treten zwei Zyklus auf, wovon sich einer über zwei Sorten erstreckt; man kann die Struktur deshalb als nichtlineare, verschränkte Rekursion bezeichnen. (Das Syntaxdiagramm ist insofern vereinfacht, als nur die Beziehungen zwischen den

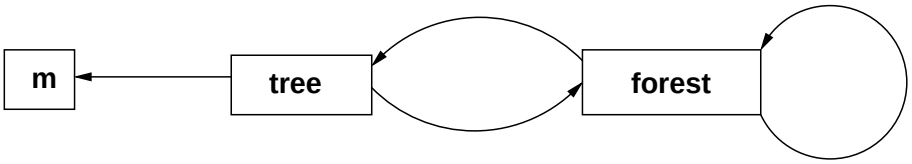

Abbildung 2: Vereinfachtes Signaturdiagramm der durch (1) deklarierten Sorte

Sorten dargestellt sind. Ein Pfeil von einer Sorte **A** zu einer Sorte **B** bedeutet dabei: Sorte **B** tritt in der Deklaration von Sorte **A** auf.)

Rekursive Funktionen
Solange man auf die Betrachtung von Funktionssystemen verzichtet, kann eine Funktion dann als rekursiv bezeichnet werden, wenn die „rechte“ Seite der Funktionsdeklaration das Funktionssymbol selbst enthält. Erweitert man jedoch die Betrachtungen auf verschränkt rekursive Funktionen, so ist es sinnvoll, die Rekursion mit Hilfe von Zyklen im Stützgraph des Systems von Funktionen zu definieren.

Auf Beispiele rekursiver Funktionen soll an dieser Stelle verzichtet werden. Bei Bedarf werden typische, allgemein bekannte Beispiele eingesetzt.

Weitere rekursive Strukturen
Auch Termersetzungssysteme weisen häufig rekursionsähnliche Strukturen auf. Die jeweils anzuwendende Regel ist hierbei im Allgemeinen nicht eindeutig vorgegeben; bei Termersetzungssystemen kann es sich also um nichtdeterministische, rekursive Strukturen handeln. Daneben treten auch bei Klassendiagrammen (Composite-Pattern), bei Schaltwerken und Textersetzungssystemen rekursionsähnliche Strukturen auf. Im Rahmen dieser Arbeit sollen jedoch nur solche rekursive Strukturen betrachtet werden, deren Semantik mittels der Fixpunkttheorie präzise angegeben werden kann.

2.2 Syntaktische Kategorisierung rekursiver Strukturen

Im vorhergehenden Abschnitt wurde exemplarisch gezeigt, wie das Auftreten von Rekursion mit dem Auftreten von Zyklen im Stützgraph oder im (vereinfachten) Signaturdiagramm der Struktur assoziiert werden kann. Im Weiteren wird zunächst präzisiert, wie die

besprochenen rekursiven Strukturen einheitlich durch einen Syntaxgraphen beschrieben werden. Anschließend wird gezeigt, dass auch die Kategorien wie lineare, nichtlineare oder verschränkte Rekursion mit Hilfe von Zyklen rein syntaktisch festgelegt werden können.

Präzisierung der graphischen Beschreibung:
Damit die verschiedenenen rekursiven Strukturen durch Syntaxgraphen einheitlich beschrieben werden, sollen folgende Vereinbarungen gelten:

- Grundsätzlich gibt der Syntaxgraph nur die Aufrufstruktur der rekursiven Struktur wieder. Beispielsweise bleiben Sortenkonstruktoren unberücksichtigt.
- Tritt im **Aufrufverhalten** einer rekursiven Struktur eine Verzweigung auf, d.h. es werden **Aufrufalternativen** durch das Symbol „|“ bei BNF-Produktionen und Sorten, bzw. durch einen bedingten Ausdruck bei Funktionsdeklarationen deklariert, so kennzeichne man dies im Syntaxgraphen durch einen speziellen Knoten (vgl. obiges Beispiel zu arithmetischen Ausdrücken).

Beispiele: Gegeben seien die Fibonacci-funktion `fib(n)`:

fct fib = (**nat** n) **nat**: **if** n $\stackrel{?}{=}$1 or n $\stackrel{?}{=}$2 **then** 1
else fib(n-1) + fib(n-2)

und die Funktion `ggT(a, b)` zur Berechnung des größten gemeinsamen Teilers zweier Zahlen a und b:

fct ggT = (**nat** a, **nat** b) **nat**: **if** a $\stackrel{?}{=}$b **then** a
else if a $>$ b **then** ggT(a-b, b)
else ggT(a, b-a)

Erstellt man für die beiden Funktionen den Syntaxgraphen, so ergibt sich Abbildung 3. Daran wird deutlich, weshalb es sinnvoll ist, in einem Syntaxgraphen die Verzweigung

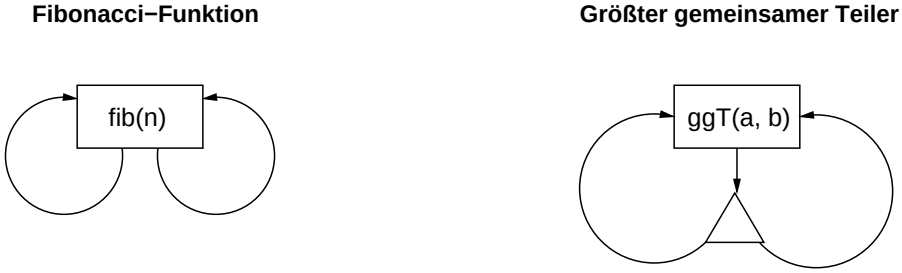

Abbildung 3: Syntaxgraphen der Funktionen `fib(n)` und `ggT(a, b)`

des Aufrufverhalten zu dokumentieren: Die Fibonacci-Funktion ist nichtlinear rekursiv,

da sie zweimal rekursiv aufgerufen wird; dies äußert sich im Syntaxgraphen in den beiden Zyklen. Die Funktion `ggT(a, b)` dagegen ist linear rekursiv, da die beiden rekursiven Aufrufe jeweils in der Alternative eines bedingten Ausdruck auftreten; von den beiden Zyklen im Syntaxgraphen wird nur einer durchlaufen.

Allgemeine Beschreibung von linearer, nichtlinearer und verschränkter Rekursion Es ergibt sich nun die Möglichkeit, anhand des Syntaxgraphen zwischen Linearität und Nichtlinearität zu unterscheiden: Durch eine Verzweigung (im oben beschriebenen Sinne) zerfällt ein Syntaxgraph in zwei Teilgraphen, von denen bei einem Aufruf nur einer durchlaufen wird. Im Allgemeinen zerlegt man also den Syntaxgraphen in die Menge aller verzweigunsfreien Teilgraphen. Von diesen verzweigungsfreien Teilgraphen wird bei einem Aufruf nur einer durchlaufen. Dann lässt sich sagen: Die betrachtete Struktur ist

- rekursiv, wenn sie mindestens einen verzweigunsfreien Teilgraphen mit Zyklen enthält,
- linear rekursiv, wenn sie rekursiv ist und in jedem verzweigungsfreien Teilgraphen höchstens ein Zyklus auftritt,
- nichtlinear rekursiv, wenn ein verzweigungsfreier Teilgraph mit mindestens zwei Zyklus existiert,
- verschränkt rekursiv, wenn ein verzweigunsfreier Teilgraph mit einem Zyklus der Länge > 1 existiert.

3 Semantik rekursiver Strukturen

Nachdem im vorherigen Abschnitt die gemeinsamen syntaktischen Strukturen von BNF-Produktionen, rekursiven Sortendeklarationen und rekursiven Funktionen analysiert wurden, soll nun skizziert werden, wie die Fixpunkttheorie die Semantik der verschiedenen rekursiven Strukturen auf einheitliche Weise beschreibt.

Die Details der Fixpunkttheorie werden hier nicht erläutert, sondern nur deren Anwendung exemplarisch für die hier betrachteten rekursiven Strukturen. Eine ausführliche mathematische Darstellung der Fixpunkttheorie findet man bei [LS84].

Die Anwendung der Fixpunkttheorie auf Systeme von BNF-Produktionen

Gegeben sei über dem Zeichensatz `C = {a, b}` das System:

```
<A>  ::=  {a <A>} | {b <B>}
<B>  ::=  {b <B>}
```

Es stellt sich die Frage, welche formale Sprache dadurch beschrieben wird. Die Interpretation der Produktionen liefert Fixpunktgleichungen für die Sprachen **A** und **B**:

$$\begin{aligned} \mathbf{A} &= \{\varepsilon\} \cup \{a \circ x,\ x \in \mathbf{A}\} \cup \{b \circ y,\ y \in \mathbf{B}\} \\ \mathbf{B} &= \{\varepsilon\} \cup \{b \circ y,\ y \in \mathbf{B}\} \end{aligned} \tag{2}$$

Unter Verwendung eines geeigneten Fixpunktoperators τ erhalten wir also Gleichung:

$$\begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix} = \tau \left[\begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix} \right] \tag{3}$$

Die Bestimmung der formalen Sprachen **A** und **B** reduziert sich somit auf die Lösung der Fixpunktgleichung 3, also die Bestimmung des Tupels $(\mathbf{A}, \mathbf{B})$.

Die Fixpunktheorie stellt nun ein anschauliches iteratives Verfahren zur Verfügung, um die Lösung schrittweise zu charakterisieren: Man berechnet Näherungslösungen $(\mathbf{A}, \mathbf{B})_i := (\mathbf{A}_i, \mathbf{B}_i),\ i \in \mathbb{N}$ mit:

$$\begin{array}{lcl} (\mathbf{A}, \mathbf{B})_0 & := & (\emptyset,\ \emptyset) \\ (\mathbf{A}, \mathbf{B})_{i+1} & := & \tau[(\mathbf{A}, \mathbf{B})_i],\ \ i \in \mathbb{N} \end{array} \tag{4}$$

Nach dem Satz von Kleene ([LS84]) ist dann $(\mathbf{A}, \mathbf{B})_\infty := \sup\{(\mathbf{A}, \mathbf{B})_i,\ i \in \mathbb{N}\}$ die gesuchte Lösung. (Das Supremum wird hier gemäß einer partiellen Ordnung bestimmt, die sich durch Verallgemeinerung der üblichen Teilmengenrelation auf Tupel ergibt.)

Führt man einige Schritte dieser Iteration durch, so ergibt sich:

i	$\mathbf{A}_i$	$\mathbf{B}_i$
0	$\emptyset$	$\emptyset$
1	$\{\varepsilon\}$	$\{\varepsilon\}$
2	$\{\varepsilon,\ a,\ b\}$	$\{\varepsilon,\ b\}$
3	$\{\varepsilon,\ a,\ b,\ a^2,\ ab,\ b^2\}$	$\{\varepsilon,\ b,\ b^2\}$
$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$

Man erkennt, dass nach der i-ten Iteration alle Sprachelemente der Länge (i-1) erfasst sind. Mit jedem Iterationsschritt nähert man sich somit der Lösung der Fixpunktgleichung.

Fixpunkttheorie für rekursive Sortendeklarationen

Wie für Systeme von BNF-Produktionen, so lassen sich auch für Sortendeklarationen nach Wahl einer geeigneten Trägermenge rekursive Gleichungen über Mengen formulieren. So liefert etwa die Interpretation der Sortendeklaration eines Binärbaumes über den natürlichen Zahlen

sort btree = empty | construct(**btree** left, **nat** element, **btree** right) (5)

die Gleichung:

$$\mathbf{BT} = \{\varepsilon\} \cup \{(\mathrm{l},\ n,\ \mathrm{r}),\ n \in \mathbb{N} \wedge \mathrm{l} \in \mathbf{BT} \setminus \{\bot\} \wedge \mathrm{r} \in \mathbf{BT} \setminus \{\bot\}\} \cup \{\bot\} \tag{6}$$

Auch die Menge **BT** läßt sich wie in (4) leicht induktiv charakterisieren.

Fixpunkttheorie für rekursive Funktionen

Da in der Literatur (etwa [BR98]) zahlreiche Beispiele für die Anwendung der Fixpunkttheorie auf rekursive Funktionen zu finden sind, soll hier auf die Angabe eines Beispiels verzichtet werden. Die grundsätzliche Vorgehensweise ist dieselbe, wie bei Sorten und Produktionen. Jedoch benötigt man hier, insbesondere für eine präzise mathematische Diskussion der Sätze von Knaster-Tarski und Kleene anspruchsvollere mathematische Methoden: etwa besondere partielle Ordnungen, um für die verschiedenen iterativen Lösungen der Fixpunktgleichung eine Approximationshierarchie zu formulieren.

4 Folgerungen aus der snytaktischen und semantischen Analyse

4.1 Einheitliche Beschreibung rekursiver Strukturen

Die Zusammenfassung der bisherigen Ergebnisse liefert:

Syntax: Die hier betrachteten rekursiven Strukturen weisen in ihren Syntaxgraphen Zyklen auf. Über diese Zyklen lassen sich die Begriffe lineare, nichtlineare und verschränkte Rekursion definieren.

Semantik: Die Semantik der hier betrachteten rekursiven Strukturen wird jeweils über eine Fixpunktgleichung beschrieben, deren Lösung sich iterativ charakterisieren lässt.

Es wäre nun wünschenswert, auch für Strukturen, wie rekursive Klassendiagramme oder Termersetzungssysteme mit rekursiven Elementen eine derartige einheitliche Beschreibung angeben zu können. Für diese Strukturen ist jedoch eine formale Beschreibung der Semantik nicht bekannt. Allenfalls die syntaktische Struktur von rekursiven Objektstrukturen läßt sich mit Hilfe des Composite-Pattern beschreiben.

Die hier angegebene einheitliche Beschreibung rekursiver Strukturen beschränkt sich also vorerst auf BNF-Produktionen, rekursive Sorten und rekursive Funktionen.

4.2 Folgerungen für den Ablauf einer Vorlesung

Diskussion der syntaktischen Gemeinsamkeiten: Im Hinblick auf die Betonung der Rekursion als themenübergreifendes Grundkonzept der Informatik scheint es sinnvoll, in der Grundlagenvorlesung auch die syntaktischen Gemeinsamkeiten auf geeignete Weise darzustellen; dabei sollte zumindest deutlich werden, dass die syntaktischen Kategorien: „Linearität“, „Nichlinearität“ und „Verschränkheit“ nicht auf rekursive Funktionen beschränkt sind, sondern ein allgemeines Kennzeichen rekursiver Strukturen sind.

Diskussion der Semantik von BNF-Produktionen oder rekursiven Sorten: Diese Thematisierung ist aus mehreren Gründen notwendig: Bei diesen rekursiven Strukturen steht

im Gegensatz zu den rekursiven Funktionen keine einfache Implementierungsmöglichkeit zur Verfügung. Die Studenten können also nicht empirisch testen, ob ihre Deklaration der Spezifikation genügt. Die Frage nach dem „Warum“ der Untersuchung der Semantik stellt sich hier nicht; im Gegensatz zu rekursiven Funktionen, wo man aus praktischen Gründen nur relativ einfache Funktionen untersuchen kann, und dann bei den Studenten das Gefühl aufkommt, man verifiziere nur das, was man sowieso bereits weiß.

Des weiteren würden sich durch eine Diskussion der Semantik von rekursiven BNF-Produktionen später die erfahrungsgemäß großen Probleme der Studenten mit der Semantik rekursiver Funktionen reduzieren. Denn die wesentlichen Elemente sind dieselben; die einzige Hürde bei rekursiven Funktionen würden dann die erwähnten partiellen Ordnungen für Funktionsapproximationen darstellen.

Reihenfolge der rekursiven Strukturen in der Vorlesung: Wie in Abschnitt 5 näher ausgeführt, werden in den Einführungsvorlesungen rekursive Funktionen im Allgemeinen vor rekursiven Sorten thematisiert. Nach dem bisher Dargestellten scheint das nicht zwingend erforderlich, da die Semantik rekursiver Sortendeklarationen zum einen mit einfachen mathematischen Mitteln hinreichend erläutert werden kann; zum anderen würde durch Diskussion der Semantik von Sorten und BNF-Produktionen das Problembewusstsein der Studenten für dieselbe Problematik bei rekursiven Funktionen geschärft werden. Und schließlich sind Bäume als graphisches Hilfsmittel zur Problemlösung den Studenten aus der Schulzeit bekannt und wohl auch intuitiv verständlich. Man würde die Studenten, um mit Wagenschein zu sprechen, dort abholen, wo sie sich, was ihre Wissenstruktur betrifft, befinden.

Diese Vorgehensweise hätte darüber hinaus den Vorteil, dass anschließend bei der Betrachtung rekursiver Funktionen mit komplexen Datenstrukturen gearbeitet und somit praktisch relevante Beispiele, – anstatt der sonst üblichen, mathematischen –, eingesetzt werden könnten. Derartiges wurde auch bereits von Hubwieser [Hub00] vorgeschlagen.

Folgerungen: Es sprechen also nicht nur systematische Gründe dafür, die in der Grundlagenvorlesung auftretenden rekursiven Strukturen einheitlich, d.h. mit ihrer gemeinsamen syntaktischen und semantischen Wurzel darzustellen, sondern auch didaktische: Prinzipiell ist die Betonung derartiger Gemeinsamkeiten natürlich wichtig, damit die Studierenden im Laufe der Grundlagenvorlesung ein konzeptuelles Bild der Informatik gewinnen. Zum anderen erleichtert das Aufzeigen der gemeinsamen Wurzeln aber auch den Lernprozess, da Themen, die zunächst disjunkt erscheinen, auf gleiche Methoden zurückgreifen.

5 Vergleich verschiedener Vorlesungskonzepte

Bisher wurde erläutert, wie die in BNF-Produktionen, Sorten- und Funktionsdeklarationen enthaltenen rekursiven Strukturen sowohl syntaktisch als auch semantisch einheitlich beschrieben werden können. Im Folgenden soll anhand zweier Vorlesungen, die an der TU-München gehalten wurden, exemplarisch untersucht werden, wie die verschiedenen rekursiven Strukturen und deren gemeinsame Wurzeln vermittelt werden.

- **Broy [Bro98]:** Die rekursiven Strukturen werden in folgender Reihenfolge besprochen: BNF-Notation, rekursive Funktionen, rekursive Datenstrukturen. Für BNF-Notationen und Sortendeklarationen wird die Fixpunkttheorie informell und für rekursive Datenstrukturen mathematisch exakt diskutiert.
- **Brügge [Brü01]** Auch dieser Vorlesungsentwurf thematisiert die hier diskutierten rekursiven Strukturen in der Reihenfolge: BNF-Notation, rekursive Funktionen, rekursive Datenstrukturen. Mittels Fixpunkttheorie werden semantische Analysen für rekursive Funktionen durchgeführt.

Die syntaktischen Kategorien „lineare“, „nichtlineare“, und „verschränkte“ Rekursion werden bei beiden Autoren primär für rekursive Funktionen diskutiert.

Vergleicht man diese Vorlesungsentwürfe mit den Ausführungen der vorhergehenden Abschnitte zur Syntax und Semantik rekursiver Strukturen, so fällt auf:

- Die syntaktischen Gemeinsamkeiten der angesprochenen rekursiven Strukturen werden in keiner der Vorlesungen allgemein thematisiert.
- Es werden zuerst rekursive Funktionen und dann rekursive Sorten behandelt.
- Das Hauptgewicht der Diskussion der Semantik liegt bei rekursiven Funktionen.
- Eine systematische, parallele Betrachtung der wichtigsten rekursiven Strukturen und somit die Diskussion der Rekursion als themenübergreifendes Grundkonzept der Informatik wird in keiner der untersuchten Vorlesungsentwürfe durchgeführt.

6 Analyse und Fazit

Der vorhergehende Abschnitt verdeutlicht, dass zwischen den hier beschriebenen didaktischen Schlussfolgerungen (Abschnitt 4.2) und den Vorlesungsentwürfen (Abschnitt 5) einige Diskrepanzen auftreten; mögliche Gründe dafür sollen im Folgenden skizziert werden:

Die Anordnung der rekursiven Strukturen: Ein Grund für die heutzutage übliche Reihenfolge könnte sein, dass rekursive Funktionen häufig anhand mathematischer Beispiele eingeführt werden und für diese eine rekursive Sorte nicht benötigt wird. (Andererseits verwenden viele Lehrbücher für anspruchsvollere Anwendungen informell rekursive Sorten (etwa Listen) ohne die dahinterstehende Theorie im Detail zu behandeln.) Historische Gegebenheiten könnten für diese klassische Reihenfolge ebenfalls verantwortlich sein, da etwa die Systematik, die dem baumartigen, rekursiven Aufbau vieler Datenstrukturen innewohnt, erst 1975 von C.A. Hoare [Hoa75] erkannt wurde.

Einer Änderung der Reihenfolge stände also nichts Prinzipielles im Wege. Weder aus methodisch–didaktischer Sicht noch aus informatischer Sicht. Auch die konkrete Implementierung rekursiver Sorten in einer funktionalen Sprache ist problemlos, erfordert sie

doch zunächst keine rekursiven Funktionen, sondern lediglich die entsprechenden Sprachelemente.

Die Beschränkung auf die Semantik rekursiver Funktionen: Die Beschränkung auf die Semantik rekursiver Funktionen dürfte in erster Linie praktische Gründe haben, da häufig aufgrund der Fülle der zu behandelnden Themengebiete für eine Diskussion der Semantik von Sorten und BNF-Produktionen keine Zeit vorhanden ist. Es sollte allerdings bedacht werden, dass die vorhergehende Untersuchung der Semantik rekursiver Sorten die Diskussion der Semantik rekursiver Funktionen unter Umständen erheblich verkürzen würde.

Fazit: Durch das Vorziehen der rekursiven Sorten und eine geringe Verlagerung der Gewichtung der verschiedenen Themen, d.h. eine Behandlung der Fixpunkttheorie bereits für BNF-Produktionen und Sorten, ließen sich die hier diskutierten Elemente also nahezu zeitneutral in eine Grundvorlesung einbauen.

Literaturverzeichnis

[BR98] Broy M.; Rumpe B.: Übungen zur Einführung in die Informatik. Springer-Verlag, 1998.

[Bro98] Broy M.: Informatik, Eine grundlegende Einführung, Band 1. Springer-Verlag, 1998.

[Brü01] Brügge B.: Einführung in die Informatik I und II. Vorlesung, 2000/2001. http://wwwbruegge.in.tum.de/teaching/ss01/Info2/index.html.

[Geo00] George C.G.: EROSI – Visualising Recursion and Discovering New Errors. The proceedings of the 31^{st} SIGCSE Technical Symposium on Computer Science Education. Austin, Texas. SIGCSE Bulletin 1, March, 2000, S. 305 - 309.

[GS99] Ginat D.; Shifroni E.: Teaching Recursion in a Procedural Environment – How much should we emphasize the Computing Model ? SIGCSE Bulletin 1, March, 1999, S. 127 - 131.

[Hoa75] Hoare C.A.: Recursive Data Structures. Int. Journal of Computer an Information Science, 4(2), 1975, S. 105-132.

[Hub00] Hubwieser P.: Rekursion im didaktischen Querschnitt. Habilitationsvortrag, 2000.

[LS84] Loeckx J.; Sieber K.: The Foundations of Program Verification. Teubner, 1984.

[VI00] Velazquez-Iturbe J.A.: Recursion in Gradual Steps (Is Recursion Really that Difficult ?). The proceedings of the 31^{st} SIGCSE Technical Symposium on Computer Science Education. Austin, Texas. SIGCSE Bulletin 1, March, 2000, 310 - 314.