Jorge Bogarin

# SEMIOTISCHE ANSÄTZE ZUR ANALYSE DER REKURSIVEN FUNKTIONEN

## 1. Vorbemerkung

Ein deutscher Wissenschaftler hat vor einigen Jahren geschrieben:

"Weder der Begriff der Ideologie oder der Konfession, noch der Begriff des Systems oder des Stils kann sich mit dem der Theorie messen, wenn es darum geht, die wesentliche Form unseres geistigen Lebens, das entscheidende Instrument unseres Erkennens und Machens zu nennen." (Bense 1956)

Der Autor betont also die Wichtigkeit des Begriffs der Theorie, ich aber möchte hier die Aufmerksamkeit auf einen anderen Begriff richten, der in der letzten Zeit immer mehr in den Vordergrund getreten und mindestens genauso wichtig wie der der Theorie geworden ist: der Begriff der Sprache (im Sinne der Zeichensysteme im allgemeinen. Beide Begriffe sind außerdem eng miteinander verbunden und beeinflussen sich gegenseitig. Bei manchen modernen Autoren (wie Quine) wird zwischen Theorie und Sprache keine deutliche Grenze mehr gezogen. Andere behaupten sogar, daß Sprachen "have a greater generative power than theories" (Sloman 1978). Die Sprache, in der eine Theorie formuliert wird, enthält auch die Mittel, um andere sogar miteinander inkompatible Theorien zu formulieren. Sie stellt eine der formalen Bedingungen der Möglichkeit von Theorien überhaupt dar, und in diesem Sinne, wenn ich so sagen darf, ein *a priori* der Theorie.

Im Rahmen einiger Überlegungen über Sprache bzw. formale Sprachen möchte ich hier über das Problem der Rekursion und die entsprechenden semiotischen Bestimmungen berichten. Als Motivation bringe ich noch ein Zitat, diesmal über das semiotische Universum:

"Die autoreproduktive 'Selbst-Entwicklung' der Zeichen und das zeichenthematisierende Kreativitätsprinzip determinieren ... das semiotische Universum, das von der ... Transformationsmatrix durchaus algorithmisch beherrscht wird." (Bense 1983)

Zwei Begriffe dieses Zitats interessieren mich zunächst: "autoreproduktive Selbst-Entwicklung" und "Algorithmus". Fangen wir mit dem letzten an.

2. Algorithmus

Intuitiv kann man unter Algorithmus verstehen: "eine allgemeine Methode zur Lösung einer Klasse von Problemen" (Brauer), "ein Spiel mit Ziffern und Zeichen nach festen Regeln", "der Inbegriff jeder Rechevorschrift schlechthin" (Behnke), "ein spezifischer Satz von Regeln, d.h. ein Rezept, das bei genauer Befolgung Erfolg verspricht", usw. (Bauer/Wössner 1984).

Der erste Vorschlag, den zunächst intuitiv gegebenen Begriff des Algorithmus mit einem bestimmten, exakt definierten Begriff gleichzusetzen, stammt von Church. Später führte Turing den Begriff der Turing-Maschine ein, um den Algorithmusbegriff präzisieren zu können. Neben diesen Vorschlägen gibt es eine Anzahl weiterer, wie die $\mu$-rekursiven Funktionen (Kleene), die $\lambda$-K-Definierbarkeit (Church), die normalen Markov-Algorithmen usw.

Obwohl die angeführten Definitionsvorschläge oft von ganz verschiedenen Ausgangspunkten herrühren, haben sie sich alle als äquivalent erwiesen, was die These unterstützt, man habe mit diesen formalen Beschreibungen alle Möglichkeiten des (intuitiven) Begriffs der Berechenbarkeit ausgeschöpft. (S. Maurer 1969 und Hermes 1971).

3. Die Programmiersprachen

Speziell zur Niederschrift von Algorithmen, die auf Rechenanlagen ablaufen sollen, dienen Programmiersprachen. Eine Programmiersprache über einem Alphabet $\Sigma$ besteht aus einer formalen Sprache $L \subseteq \Sigma^{*}$(1), der Menge der *syntaktisch richtigen Programme* und einer Vorschrift, die syntaktisch richtigen Programmen eine *Bedeutung* zuordnet. Das geschieht z.B. dadurch, daß man angibt, welches *Resultat* (Output) ein gegebenes *Programm* für eine gegebene *Eingabe* (Input) liefert. (Maurer 1969)

4. Das Programm

Ein Rechenprogramm besteht aus zwei wesentlichen Teilen, einer Beschreibung der Aktionen, die auszuführen sind, und einer Beschreibung der Daten, die in diesen Aktionen benutzt werden.

---

(1) $\Sigma^{*}$ ist die Menge aller endlichen Zeichenfolgen (Worte) über $\Sigma$, einschließlich des leeren Wortes $\varepsilon$

Aktionen werden durch sogenannte *Anweisungen* beschrieben. Die Daten werden durch Werte von *Variablen* repräsentiert, (S. Wirth 1972.) Wir haben damit, meine ich, die drei wesentlichen Begriffe genannt: Variable, Anweisung und Programm.
In einer anderen Arbeit (Bogarin 1985) habe ich die entsprechenden semiotischen Bestimmungen eingeführt und ausführlich beschrieben. Ich gebe diese Charakterisierung wieder, mit dem Vorbehalt, daß die Expertengemeinschaft sich noch nicht darüber einig ist, ob sie stimmt oder nicht.

*Variable*

$DS_{(Va)}$ (Zkl: 3.1 2.3 1.3 x Rth: 3.1 3.2 1.3)
I-thematisiertes Mittel Rpw=13

*Anweisung*

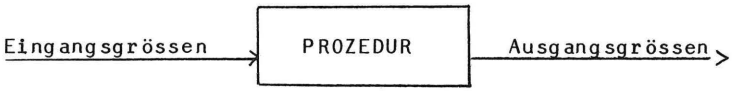
$DS_{(An)}$ (Zkl: 3.2 2.2 1.3 x Rth: 3.1 2.2 2.3)
O-thematisierter Interpretant Rpw=13

*Programm* (terminierendes, korrektes)

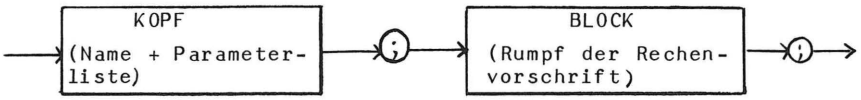
$DS_{(Pr)}$ (Zkl: 3.3 2.3 1.3 x Rth: 3.1 3.2 3.3)
I-thematisierter Interpretant Rpw=15

5. Prozeduren und Funktionen

Prozeduren und Funktionen haben die Aufgabe, Teile eines Programms unter einen eigenen Namen zusammenzufassen. Eine Prozedur kann man sich als ein Gebilde wie in folgendem Diagramm vorstellen: (Herschel/Pieper 1981)




Hat eine Prozedur die spezielle Eigenschaft, nur eine Ausgangsgröße zu produzieren, dann handelt es sich um eine Funktion. Die Funktionsvereinbarung hat folgende Form (Herschel/Pieper 1981):




Der Gebrauch von Prozeduren und Funktionen hat folgende Vorteile:
1) Das Programm wird strukturiert und übersichtlicher;
2) das Programm wird kürzer, die betreffenden Programmteile sind nur einmal aufzuschreiben;
3) Es ist möglich, vorgefertigte Prozeduren zu übernehmen;
4) Prozeduren und Funktionen können mit wechselnden Parametern aufgerufen werden;
5) Das wichtigste hier: die rekursive Benutzung von Prozeduren und Funktionen.

## 6. Die Rekursion

Zuerst werde ich einige intuitive Definitionen des Begriffs der Rekursion angeben, um danach speziell auf die rekursiven Funktionen einzugehen.
"Eine Situation, in der eine Definition (ein Begriff, ein Vorgang) auf dieselbe Definition (denselben Begriff, denselben Vorgang) als Bestandteil zurückgreift, heißt rekursiv." (Bauer/Wössner 1984)
"Ein Objekt heißt rekursiv, wenn es sich teilweise selbst enthält. Eine Definition ist rekursiv, wenn das zu Definierende teilweise durch sich selbst definiert wird." (Herschel/Pieper 1981) Sprachen z.B. sind extrem rekursiv, Musikstücke oder Bilder können ebenfalls rekursiv sein. Auch Datenstrukturen sind häufig rekursiv: "Eine Liste ist entweder leer oder hat ein Element, dessen Nachfolger eine Liste ist. Ein Baum ist entweder leer oder hat einen Knoten, an dessen Ästen Bäume hängen." (Herschel/Pieper 1981)

## 7. Rekursive Funktionen

"Wir nennen eine Rechenvorschrift *rekursiv*, wenn sie sich direkt oder indirekt auf sich selbst stützt" (Bauer/Wössner 1984). Eine Rechenvorschrift A stützt sich (direkt) auf eine Rechenvorschrift B, wenn in der Aufschreibung des Rumpfs von A ein Aufruf von B vorkommt. A stützt sich (indirekt) auf B, wenn A sich auf eine Rechenvorschrift C direkt stützt, die sich direkt oder wieder indirekt auf B stützt. Wie bei einem Fernsehbild, wo die Kamera auf den eigenen Monitor gerichtet ist, bewirkt die Rekursion so eine unendlich tiefe Verschachtelung. Jede Rekursion muß daher einmal

abgebrochen werden. Der *circulus vitiosus* des nicht endenden Rückgriffs auf sich selbst kann mit dem Hilfsmittel der Fallunterscheidung (bzw. Abbruchskriterium) vermieden werden. Das Rekursionsschema muß so beschaffen sein, daß der Einsetzungsmechanismus zu dieser nicht rekursiven Definition (der Rückkehrstelle) hinstrebt (s. Herschel/Pieper 1981 und Bauer/Wössner 1984). Die für Algorithmen naiv zu fordernde Eigenschaft der Terminierung ist also bei rekursiven Rechenvorschriften nicht mehr selbstverständlich gegeben.

8. Die Fakultät

Ein oft angeführtes Beispiel für rekursive Algorithmen ist die Berechnung der Fakultät.

Definition: Die für alle natürlichen Zahlen $n$ definierte Funktion $f(n)=n!$, für die gilt: $f(0)=1$ und für alle $n$: $f(n+1)=(n+1)\cdot f(n)$, heißt *n-Fakultät*.

Um das Beispiel einfacher zu machen, werde ich eine äquivalente Definition mit Hilfe der Multiplikation und der Vorgängerfunktion angeben:

$f(0) = 1$
$F(n) = n \cdot f(n-1)$

Wir wollen die entsprechende Funktion *fac* nennen.
Das dynamische Diagramm (Bild 1) zeigt uns die Aufrufpfeile sowie die Rückkehrverpflichtungen, die beim Funktionaufruf entstehen. Dabei bedeutet $fac^i$ die i-te Replica (Inkarnation!) von *fac*.

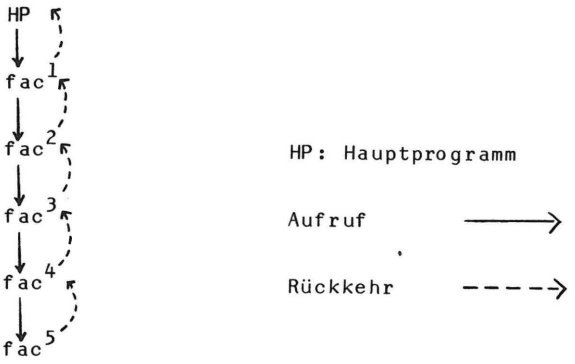



BILD 1

Das Bild 2 zeigt die Verschachtellung der sukzessiven Inkarnationen von *fac* und spiegelt die Gültigkeitsbereiche der Vereinbarungen wieder.

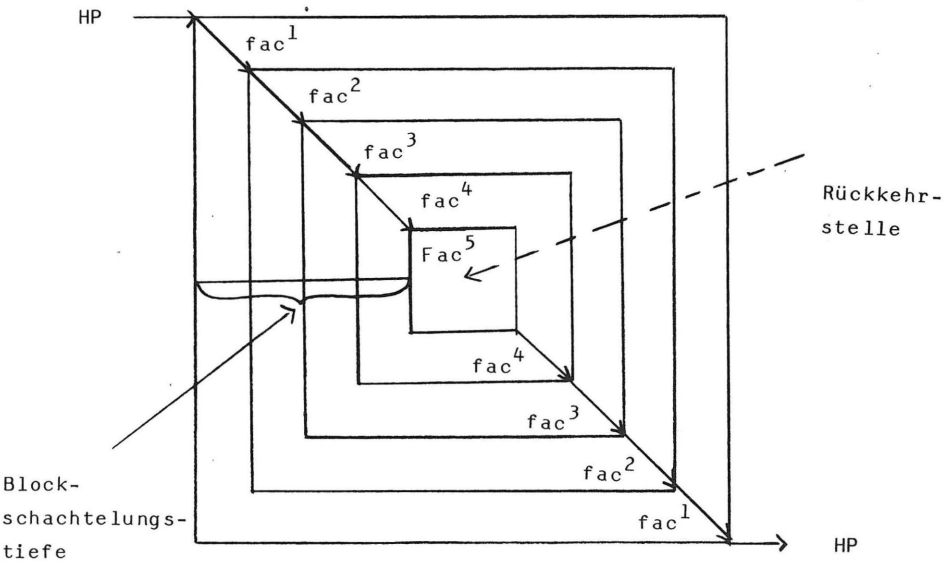



BILD 2

9. Beispiel

Nehmen wir das konkrete Beispiel 4! und verfolgen wir die Berechnung Schritt für Schritt. Die wichtigsten "Momentaufnahmen" sind im folgenden Diagramm zu sehen:

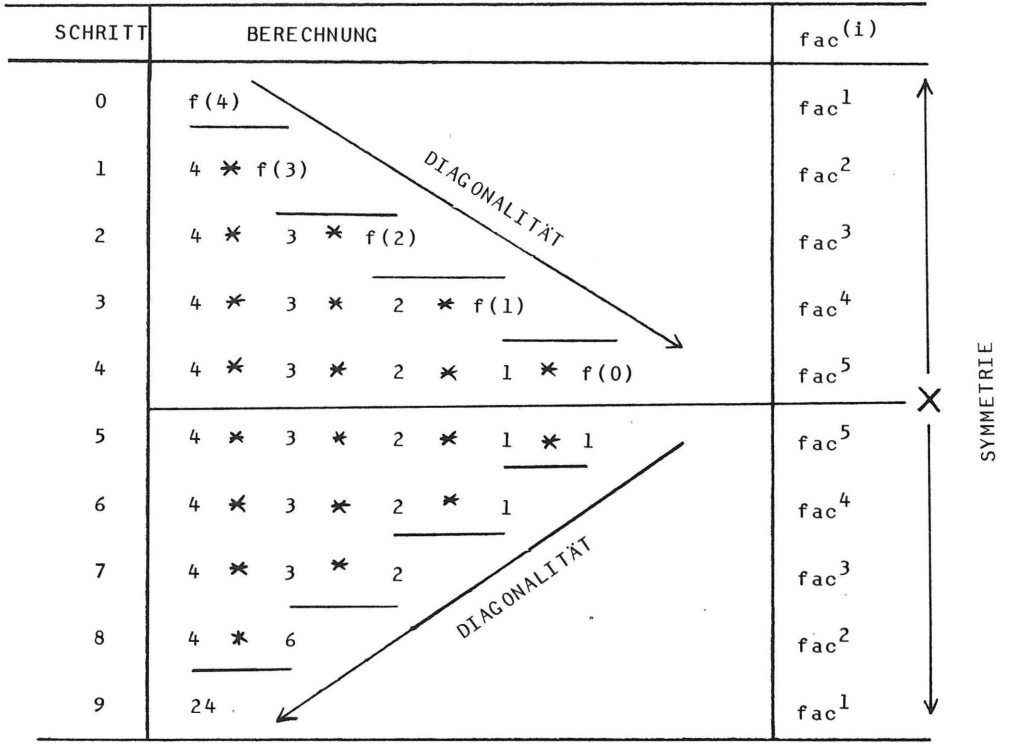

SCHRITT
BERECHNUNG
fac(i)
0
f(4)
fac1
1
4 * f(3)
fac2
2
4 * 3 * f(2)
fac3
3
4 * 3 * 2 * f(1)
fac4
4
4 * 3 * 2 * 1 * f(0)
fac5
DIAGONALITÄT
5
4 * 3 * 2 * 1 * 1
fac5
6
4 * 3 * 2 * 1
fac4
7
4 * 3 * 2
fac3
8
4 * 6
fac2
9
24
fac1
DIAGONALITÄT
SYMMETRIE

BILD 3

10. Semiotische Überlegungen

Es ist schwierig, die Zeichenklasse für die Rekursion bzw. Rekursivität zu bestimmen, ohne gleichzeitig die entsprechende Realitätsthematik zu berücksichtigen. Handelt es sich um eine Eigenschaft (die Rekursivität?) oder um ein vollständiges Objekt im Sinne des vollständigen Objektbezugs (Ic, Ind, Sy) oder um einen Zustand, einen Prozeß oder ein unvollständiges Objekt?

Ich ziehe die letzte Möglichkeit vor und charakterisiere die Rekursion mit der dual-invarianten Zeichenklasse, die auch dem Zeichen selbst, der Zahl und dem ästhetischen Zustand entspricht. Also das Dualitätssystem:

$DS_{(Rek)}$ (Zkl: 3.1 2.2 1.3 x Rth: 3.1 2.2 1.3) Rpw=12

Statt die einzelnen Subzeichen getrennt zu bestimmen, werde ich das Dualitätssystem als Ganzes beobachten und die zu erwartenden Eigenschaften nachzuweisen versuchen (nach Bense 1986). Diese werden mit Hilfe von Bild 3 anschaulich gemacht.

*Dreifache Realitätsthematik:* Wenn die aufgestellte Hypothese stimmt, dann werden wir eine dreifache Realitätsthematik unterscheiden können.

Betrachten wir den Zeichenkontext als Ganzes, dann wird die Rekursions-Realität ein *M+O-thematisierter Interpretant* als offener Zusammenhang der in einer Richtung potentiell unendlichen Reihe von ineinander verschachtelten, auf sich selbst bezogenen Entitäten. Eine Ähnlichkeit mit der Reihe der natürlichen Zahlen ist nicht zu übersehen.

Wenn wir das konfigurative Zeichenobjekt studieren, so haben wir es mit einem *M+I-thematisierten Objekt* zu tun im Sinne des vom dynamischen Vorgänger und dynamischen Nachfolger indexikalisch bestimmten Objektbezugs. Auch hier ist an die Zahl bzw. an die Ordnungszahl (Peano/von Neumann) zu denken.

Beobachten wir dagegen das Zeichenmittel, dann handelt es sich um ein *O+I-thematisiertes Mittel* im Sinne des konventionalisierten repertoiriellen Zeichenmittels bzw. als intelligible allgemeine Realität, deren repertoirieller kardinaler Wert durch die sogenannte "Blockschachtelungstiefe" wiedergegeben wird.

*Diagonalität:* Auch die fundamentalkategoriale Diagonalität der betreffenden Zeichenklasse spiegelt sich in der dynamischen Verkettung

wieder. Sowohl beim Aufruf der Funktionen wie auch bei der Rückkehr ins Hauptprogramm wird ein diagonaler Durchlauf deutlich. Die Funktionen werden sozusagen "quer" ausgeführt. (siehe Bild 3).
*Symmetrie:* Um der dual-invarianten Zeichenklasse gerecht zu werden, sollen wir auch die Symmetrie-Eigenschaft nachweisen können. Wenn wir das Bild 3 noch einmal anschauen und diesmal unsere Aufmerksamkeit den Replicas der Funktion widmen, werden wir merken, daß der sukzessive Aufruf der Rechevorschriften bei der Rückkehr bis zur Rückkehrstelle dualisiert wiederholt wird, wodurch ein symmetrisches Gebilde entsteht.

11. Schluß

Ich möchte diese Überlegungen damit schließen, daß ich eine Vermutung äußere, die mir ziemlich plausibel erscheint:
Die Rekursion stellt einen wesentlichen semiotischen Prozeß dar und ist mit dem Begriff des Zeichens als solchem, der Zahl und des ästhetischen Zustands eng verbunden. Eine genaue Untersuchung der Rekursivität wird uns auch ein besseres Verständnis dieser drei Quasi-Objekte ermöglichen.

*BIBLIOGRAPHIE*


*Max Bense. Rationalismus und Sensibilität. 1956*
*" Das Universum der Zeichen. 19883*
*" Vorlesungs-Manuskript, Januar 1986*
*J. Bogarin. Semiotische Analyse der Programmiersprache "Pascal". Magisterarbeit, Stuttgart 1985*
*F. Bauer und H. Wössner. Algorithmische Sprache und Programmentwicklung. 1984*
*H. Hermes. Aufzählbarkeit, Entscheidbarkeit, Berechenbarkeit. 1971*
*Herschel und F. Pieper. Pascal. 1981*
*H. Maurer. Theoretische Grundlagen der Programmiersprachen. 1969*
*A. Sloman. The computer revolution in Philosophie. 1978*
*N. Wirth. The programming language Pascal: Revised Report. 1972*


SUMMARY


Without doubt, recursion is an interesting phenomenon in the development of languages and theories. Its characteristic peculiarities are investigated in this essay on the basis of an example of recursive function. The sign-class 3.1 2.2 1.3 was determined as semiotic definition of this process.

# SEMIOSIS 42

Internationale Zeitschrift
für Semiotik und Ästhetik
11. Jahrgang, Heft 2, 1986

*INHALT*