Funktionale Modellierung und Rekursion

Praxis – Didaktik – Theorie

von
Gernot Lorenz

Oldenbourg Verlag München

Gernot Lorenz ist Studiendirektor an einem Gymnasium. Schwerpunkte seiner vielfältigen didaktischen Tätigkeiten im Bereich Mathematik und Informatik sind funktionale Modellierung und Programmieren-Lernen. Im vorliegenden Werk hat er seine langjährigen Erfahrungen, insbesondere zum Thema Rekursion eingebracht.

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
www.oldenbourg-verlag.de

Lektorat: Dr. Gerhard Pappert
Herstellung: Dr. Rolf Jäger
Titelbild: Entwurf des Autors; Bearbeitung: Irina Apetrei
Einbandgestaltung: hauser lacour
Gesamtherstellung: Grafik & Druck GmbH, München

Dieses Papier ist alterungsbeständig nach DIN/ISO 9706.

ISBN 978-3-486-71788-4
eISBN 978-3-486-72108-9

Vorwort

Funktionales Modellieren ist schon längere Zeit im didaktischen Fokus der Mathematik. Für die unterrichtliche Praxis heißt das, daß eine *explizit* definierte Zielfunktion als Modell des untersuchten Sachverhaltes oder Phänomens angestrebt wird.
Dagegen bleibt das mächtigere Prinzip der *Rekursion* nur eine Randerscheinung, obwohl es didaktisch ergiebiger ist, und die Idee der Selbstbezüglichkeit weit über die Mathematik hinausreicht. Dieser bedauerliche Umstand ist z. T. in den Lehrplänen begründet, vor allem aber auch durch den Mangel an geeignetem didaktischem und fachlichem Material.

Vor diesem Hintergrund sieht der Autor einen Bedarf an Literatur, den dieses Buch decken helfen soll, und zwar als ein Lehrbuch im doppelten Sinn:

- ein Handbuch als Nachschlagewerk und Fundgrube für alle, die auf der Suche sind ist nach Anregungen, didaktischen Tipps, theoretischen Hintergründen, zahlreichen Beispielen, Aufgaben für Klausuren, Themen für Referate oder Jahresarbeiten usw.
- eine Monographie, d.h. man kann es auch von vorn bis hinten lesen.

Im didaktischen Zentrum stehen dabei rekursive Modellierungsprinzipien, insbesondere die deskriptive Modellierung in mehreren Varianten für Anfänger und Einsteiger, je nach Alter, unterrichtlichem Kontext und Softwarewerkzeug.

Für die praktische Umsetzung werden zwei Softwarewerkzeuge vorgestellt: zum einen die Tabellenkalkulation, ein funktionales Werkzeug, das in seiner Universalität häufig nicht die notwendige Beachtung im schulischen Bereich findet, zum anderen die funktionale Programmierung, die, wenn den Schülern ein geeignetes Programmiersystem an die Hand gegeben wird – und nur dann – bereits ab der 8. Klasse ein erfolgreiches rekursives Modellieren und Denken ermöglicht.

Darüber hinaus wird für die Sek.-Stufe II der theoretische Hintergrund bereitgestellt: der Begriff der Berechenbarkeit wird passenderweise anhand der *Theorie der rekursiven Funktionen* beleuchtet, und nicht – wie häufig im Informatikunterricht – über TURING-Berechenbarkeit. Auf diese Weise sind Praxis und Theorie eng verzahnt und wirken wie aus einem Guß.

Adressaten sind naturgemäß in erster Linie Mathematiklehrer, aber auch Informatiklehrer, die neben einem altersgemäßen Einstieg in die theoretische Informatik auch komplexere Beispiele für das praktische Problemlösen suchen oder eine alternative Einführung in Rekursion mittels Grafikprogrammierung anstreben.

Mein besonderer Dank gilt Prof. OStD JOSEF LEISEN (Koblenz), MinDg' BARBARA MATHEA (Mainz), StD' NINA PFEIL (Höhr-Grenzhausen), Dr. MICHAEL SPERBER (Tübingen) und Prof. CHRISTIAN WAGENKNECHT (Görlitz) für ihre Unterstützung auf mannigfache Art.

Bendorf am Rhein GERNOT LORENZ

Inhalt

1 Einstieg

> *Von all den Ideen, die ich Kindern vorgestellt habe, zeichnet sich die Rekursion als die eine Idee aus, die eine besonders aufgeregte Reaktion hervorrufen konnte.*
> *Ich glaube, das kommt zum Teil daher, daß der Gedanke einer endlosen Fortsetzung die Phantasie jedes Kindes anspricht, und zum Teil daher, daß die Rekursion selbst Wurzeln in der Alltagskultur hat.*
>
> SEYMOUR PAPERT in *Gedankenblitze*

Die gute...

Eine Glücksfee spricht zum rekursionsgeübten Schüler:

Du hast zwei Wünsche frei. Ich werde sie beide erfüllen.

Nach der Erfüllung des ersten Wunsches, den der Autor leider vergessen hat, formuliert der Schüler seinen zweiten Wunsch: „Ich möchte zwei Wünsche erfüllt bekommen". Wie lautet wohl der (neue) zweite Wunsch?

Offensichtlich hat er die Glücksfee überlistet, denn wer er es richtig anstellt, kann er jeden Wunsch erfüllt bekommen, oder?
Die gute Nachricht: Diesen „Trick" kann man in abgewandelter Form auch in der Mathematik und Informatik – und nicht nur da – anwenden!

...und die schlechte Nachricht

Ebenso bekannt wie die Glücksfee dürfte die folgende Redewendung sein:

Keine Antwort ist auch eine Antwort

Wir alle wissen, wie dieser anscheinend widersinnige Satz zu verstehen ist.
Allerdings hat man dabei die Zeit außer Acht gelassen und meint eigentlich

(Bis jetzt) Keine Antwort ist auch eine Antwort

Wir wollen ja nicht ewig warten! Aber könnte es sein, daß wir – wenn wir keine Frist gesetzt haben – nicht lange genug gewartet haben?

Auf unseren Computer übertragen müssen wir den Satz abermals modifizieren zu

Keine Antwort ist auch keine Antwort

Man ahnt, welche Situation gemeint ist: Wir sitzen davor und das Gerät – oder zumindest das benutzte Programm – „antwortet" einfach nicht mehr…
Die schlechte Nachricht: Wer programmiert, insbesondere wenn er den oben erwähnten „Trick" anwendet, dem kann das öfter passieren.

Aber es gibt Hoffnung, daß wir diese Wartesituation vermeiden können, wie wir in den Kapiteln 7 und 8 sehen werden.

1.1 Pünktchen als verkappte Rekursion

Ohne es zu merken, begegnet uns Rekursion in der Mathematik – auch schon früh im Unterricht – immer da, wo durch „Pünktchen" fehlende Glieder einer Aufzählung oder Reihe ersetzt werden:

- Wie berechnen wir 2^7?
  Nun ja, viele wissen, daß $2^4 = 16$, weiter rechnet man dann $2 \cdot (2 \cdot (2 \cdot 16)) = 2 \cdot (2 \cdot 32) = 2 \cdot 64 = 128$, d.h. durch *Zurücklaufen* oder *Rückgriff* auf einen „früheren" oder „einfacheren", d.h. bereits bekannten oder berechneten Wert.

  Im Schulunterricht werden Potenzen a^n mit $n \in \mathbb{N}$ mittels

  $$a^n = \underbrace{a \cdot a \cdot \ldots \cdot a}_{n-mal}$$

  definiert. Aber die Rückführung der Potenzrechnung auf wiederholte Multiplikation mit dem gleichen Faktor ist ohne Pünktchenschreibweise formal nur durch

  $$a^n = \begin{cases} a & \text{falls } n = 1 \\ a \cdot a^{n-1} & \text{falls } n > 1 \end{cases}$$

  möglich. (Der Fall $n = 0$ wird im Unterricht erst später behandelt)

- Wer die Multiplikation als wiederholte Addition mit dem gleichen Summanden erklären will, kommt auch nicht um das Rekursionsschema herum:

  $$a \cdot n = \underbrace{a + a + \ldots + a}_{n-mal} = \begin{cases} 0 & \text{falls } n = 0 \\ a + a \cdot (n-1) & \text{falls } n > 0 \end{cases}$$

  Genau genommen gilt das auch für die Addition von natürlichen Zahlen als wiederholte Nachfolgerbildung, siehe PEANO-Axiome, s. Abschnitt 2.2.

- Bei der Aufsummierung von Folgengliedern hat man die Pünktchenschreibweise durch die Einführung des Summenzeichens umgangen. Aber auch dessen

Definition ist ohne Pünktchenschreibweise nur rekursiv möglich:

$$\sum_{i=1}^{n} a_i \;= a_1 + a_2 + a_3 + \ldots + a_n \;= \begin{cases} a_1 & \text{falls } n = 1 \\ a_n + \sum\limits_{i=1}^{n-1} a_i & \text{falls } n > 1 \end{cases}$$

Gleiches gilt für das Produkt $\prod$.

- Ein populäres Beispiel für „Pünktchenvermeidung" ist die bekannte Fakultäts-Schreibweise mittels Aufrufezeichen:

$$n! \;=\; 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n \;= \begin{cases} 1 & \text{falls } n = 1 \\ n \cdot (n-1)! & \text{falls } n > 1 \end{cases}$$

(Der Fall $n = 0$, d.h. $0! = 1$ wird speziell für $\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$ ergänzt)

1.2 Ein paar Abmachungen

Zur Darstellung

Bestimme Passagen werden unterschiedlich hervorgehoben:

■ **So werden wesentliche Ergebnisse hervorgehoben, die sich aus dem Kontext oder dem jeweiligen Abschnitt ergeben**

■ So werden Ergebnisse hervorgehoben, die eher von lokaler Bedeutung sind

Ausgearbeitete Beispiele werden mit

E06	**Gitterwege**	*R-Var:*	ÜNN
Anmerkung:	vgl. $\binom{n}{k}$	*Schema:*	DO

eingeleitet und mit ♦ abgeschlossen. Dabei geben z. B. E06 die Nummer des Beispiels an und ÜNN sowie DO die Klassifizierung nach Struktur der Rekursion an wie in Abschnitt 7.1 beschrieben.

Am Ende mancher Abschnitte oder Kapitel finden sich

ANREGUNGEN:

1. ...
2. ...

Diese enthalten Übungsmaterial, Aufgaben und Anregungen unterschiedlichster Art.

Schreibweisen

$\mathbb{N}$	Menge der (modernen) natürlichen Zahlen $\{0, 1, 2, 3, \ldots\}$
$\mathbb{N}^+$	Menge der (traditionellen) natürlichen Zahlen $\{1, 2, 3, \ldots\}$
$\widehat{\mathbb{Q}} \subset \mathbb{Q}$	(Teil-)Menge aller rationalen Zahlen, die bei Computeranwendungen die jeweilige Zahlenimplementation verarbeiten oder ausgeben kann.
$[x]$	größte ganze Zahl $\leq x$ (wie in der Zahlentheorie üblich)
$\left[\frac{x}{y}\right]$	Quotient bei der Ganzzahldivision (Division mit Rest)

Begriffe

Für die Klärung der wichtigsten Begriffe, die im Folgenden bei der mathematischen Formulierung (= Formalisierung) der Rekursion benutzt werden, beziehen wir uns auf das *Standard-Rekursionsschema*, das in Abschnitt 2.1 ausführlich hergeleitet wird:

$$f(p,x) = \begin{cases} c(p,x) & \text{falls } P(x) \quad \textit{Basisfall} \\ h(p,x,f(p,\varphi(x))) & \text{sonst} \quad \textit{Allgemeiner Fall} \end{cases}$$

Standard-Rekursionsschema mit Parameter

mit $x, p \in \mathbb{Q}$ und den Begriffen

x	*Rekursionsvariable*
p	*Parameter*
φ	*Vorgängerfunktion*
h	*Schrittfunktion*
$f(p, \varphi(x))$	*Selbstaufruf*
$P(x)$	*Terminierungs-* bzw. *Abbruchsbedingung* (*Prädikat* von x)

Den *Allgemeinen Fall* werden gelegentlich auch als *selbstaufrufenden Zweig* bezeichnen. Bei der Modellierung mittels Rekursion im Anfängerunterricht, Kapitel 5, wird der Basisfall als *Verankerung*, der allgemeine Fall, d.h. der Aufruf der Schrittfunktion h als *Vererbung* bezeichnet.

Auf Erweiterungen und Varianten dieses Schemas wird in Abschnitt 7.1 ausführlich eingegangen.

2 Was ist und woher kommt Rekursion?

Wir beginnen mit einer sehr allgemeinen Formulierung, wie man sie häufig in der Literatur findet, so z.B. in [GEB], S. 164:

„...man definiert etwas nicht explizit,
sondern durch einfachere Versionen seiner selbst...." (1)

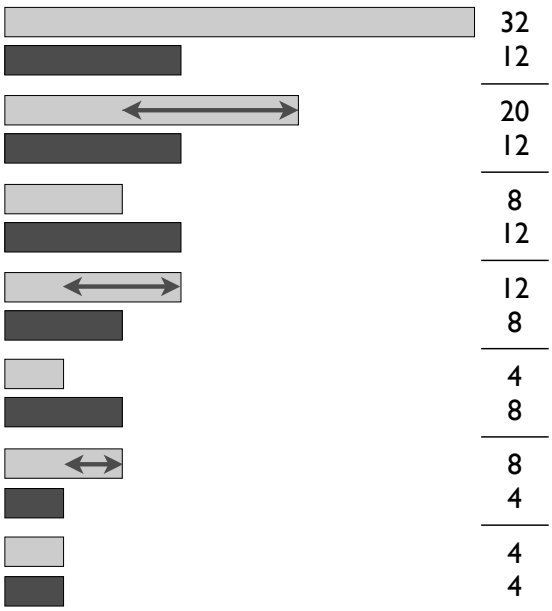
Aber was heißt das?

Zur Beantwortung dieser Frage wollen wir zwei historische Spuren verfolgen: die eine ist über zweitausend Jahre alt und rührt von einem alltagspraktischen Meß- und Vergleichsverfahren her, die andere wurde vor gut hundert Jahren innermathematisch aus den Eigenschaften der natürlichen Zahlen entwickelt, wobei auch erstmalig der Begriff „rekursiv" verwendet wurde.

2.1 Historischer Ansatz: Rekursion als Verfahren

Im Allgemeinen wird EUKLID der erste rekursive Ansatz zugesprochen: Er soll sich die Frage *„Haben zwei Stäbe der Längen a und b ein gemeinsames Maß?"* gestellt und dazu das Verfahren der *wechselnden Wegnahme* entwickelt haben.

Dabei wird der jeweils längere Stab schrittweise um die Länge des anderen verkürzt, bis beide Stäbe die gleiche Länge haben:



Offensichtlich liefert dieses Verfahren den *größten gemeinsamen Teiler (ggT)* zweier Zahlen $a,\ b \in \mathbb{N}$ und lautet in heutiger Notation

$$ggT(a,b) = \begin{cases} a & \text{falls } a = b \\ ggT(a-b,b) & \text{falls } a > b \\ ggT(a,b-a) & \text{sonst} \end{cases} \tag{2}$$

Das Bemerkenswerte an EUKLIDs einfach zu verstehender Methode ist, daß ein praktisches, alltagstaugliches *Verfahren* beschrieben wird, das einen Wert liefert, während (2) eher als mathematische Definition einer Funktion aufzufassen ist (s. 9.2).

Diese Interpretation von Rekursion als Verfahren wollen wir allgemein am Beispiel der Berechnung eines Funktionswertes $f(x)$ entwickeln: Die Formulierung „einfachere Version" in obigem Zitat (1) kann dabei nur bedeuten, daß es sich um den Wert eines anderen Arguments x' handeln muß, das nicht zufällig, sondern durch eine Vorschrift oder Funktion φ bestimmt sein kann, also etwa $x' = \varphi(x)$.

$\Rightarrow$ Wir nennen φ von jetzt ab die *Vorgängerfunktion*.

Im Sonderfall $x = n \in \mathbb{N}$ kann z.B. $\varphi(n) = V(n) = n - 1$ sein, s. Abschnitt 2.2.

Wie geschieht nun der „Rückgriff" auf die „einfachere Version" $f(x') = f(\varphi(x))$, die wir als *Selbstaufruf* bezeichnen, aus ?
Die Art des Selbstaufrufs soll durch eine weitere Funktion, etwa h, beschrieben werden, so daß der Term $h(\ldots f(\varphi(x))\ldots)$ den neuen Funktionswert $f(x)$ liefert. Zusätzlich erlauben wir noch, daß auch auf das aktuelle Argument x „zugegriffen" werden kann, und erhalten so eine 2-stellige Funktion $h(x, f(\varphi(x)))$

$$f(x) = h(x, f(\varphi(x))) \tag{3}$$

$\Rightarrow$ Wir nennen h von jetzt ab die *Schrittfunktion*.

Dabei kann (3) als

1. Funktionalgleichung aufgefaßt werden, die, wenn sie lösbar ist, als Lösungsmenge eine Schar expliziter Funktionen liefert. Zusätzlich wäre eine Anfangswert $f(x_0) = c$ nötig für eine konkrete Lösung. Diesen Weg werden wir nicht weiter verfolgen, da er uns aus dem Thema Rekursion hinausführt.
2. Rechenvorschrift aufgefaßt werden, was bei fortlaufendem Selbstaufruf durch Substitution zu

$$\begin{aligned} f(x) &= h(x, f(\varphi(x))) \\ &= h(x, h(\varphi(x), f(\varphi(\varphi(x))))) \\ &= h(x, h(\varphi(x), h(\varphi(\varphi(x)), f(\varphi(\varphi(\varphi(x))))))) \\ &= \ldots \end{aligned}$$

führt und offensichtlich einer Beendigung oder *Terminierung* (s. 6.2) des Rückgreifens bedarf.

Dieser Fall soll durch die Erfüllung einer Bedingung $P(x)$ eintreten, und zwar mit dem Funktionswert $c(x)$.

$\Rightarrow$ Wir nennen

$$f(x) = c(x) \quad \text{falls } P(x) \tag{4}$$

den *Basisfall*.

Basisfall (4) und Aufruf der Schrittfunktion (3) ergeben ein Gleichungssystem

$$\left| \begin{array}{ll} f(x) = c(x) & \text{falls } P(x) \\ f(x) = h(x, f(\varphi(x))) & \text{sonst} \end{array} \right|$$

das man üblicherweise als Fallunterscheidung schreibt:

$$f(x) = \begin{cases} c(x) & \text{falls } P(x) \quad \textit{Basisfall} \\ h(x, f(\varphi(x))) & \text{sonst} \quad \textit{Allgemeiner Fall} \end{cases}$$

Standard-Rekursionsschema

Dieses Schema bezeichnen wir von jetzt ab als *Standard-Rekursionsschema*. Es kann noch erheblich erweitert bzw. verallgemeinert werden (s. 7.1).
Z. B. ergibt sich mit Einführung eines Parameters das eingangs erwähnte

$$f(p,x) = \begin{cases} c(p,x) & \text{falls } P(x) \quad \textit{Basisfall} \\ h(p,x, f(p, \varphi(x))) & \text{sonst} \quad \textit{Allgemeiner Fall} \end{cases}$$

Standard-Rekursionsschema mit Parameter

Insgesamt haben wir jetzt EUKLIDs Idee verallgemeinert und als mathematische Funktion formalisiert; ein ebenso wichtiger Aspekt ist aber folgender:

■ **Rekursion kann, ausgehend z.B. vom historischen Beispiel EUKLIDs, als Rechen-*Verfahren* betrachtet werden.**

Allerdings fällt bei diesen Schema auf, daß weder über die Funktionen φ und h noch über das Prädikat P irgendwelche Einschränkungen gemacht werden, d.h. die Frage, unter welchen Voraussetzungen das Verfahren „funktioniert", ob also das Prädikat $P(x)$ des Basisfalles für jedes x entscheidbar ist und nach endlich vielen Selbstaufrufen der Basisfall eintritt, bleibt offen. Dieser Problematik gehen wir im Abschnitt *Berechenbarkeit* 7.2 nach.

2.2 Innermathematischer Ansatz: Induktion & Rekursion

Wir gehen von einer konstruktiven Definition der natürlichen Zahlen aus (PEANO):

Definition: *Die Menge* $\mathbb{N}$ *der natürlichen Zahlen ist gegeben durch*

1. $0 \in \mathbb{N}$
2. *Zu jedem* $n \in \mathbb{N}$ *gibt es ein* $S(n) \in \mathbb{N}$, *den Nachfolger von* n
3. *Für alle* $n \in \mathbb{N}$ *gilt* $S(n) \neq 0$
4. *Aus* $S(n) = S(m)$ *folgt* $n = m$
5. *Sei* $M \subseteq \mathbb{N}$ *mit* $0 \in M$ *und mit* $n \in M$ *ist auch* $S(n) \in M$.
   Dann gilt $M = \mathbb{N}$

Welche Konsequenzen ergeben sich daraus?

Zunächst geben die Axiome *2.* bis *4.* an, wie man die natürlichen Zahlen aus der 0 konstruiert: Schreiben wir für die Nachfolgerbildung $S(n) = n + 1$, ergibt sich

$$\text{Eine } \textbf{\textit{natürliche Zahl}} \text{ ist} \quad \begin{cases} 0 \\ \quad \text{oder} \\ \text{Nachfolger einer } \textbf{\textit{natürlichen Zahl}} \end{cases} \qquad n = \begin{cases} 0 \\ \quad \text{oder} \\ S(n-1) \end{cases}$$

Induktive Definition

Eine solche konstruktive Definition auch *induktiv* genannt und spielt bei funktionalen Programmiersprachen eine wichtige Rolle, s. 8.1.

Aus Punkt *5.* der Definition – auch *Induktionsaxiom* genannt – lassen sich zwei wichtige Prinzipien herleiten:

Das Beweis-Prinzip der vollständigen Induktion

Wir nehmen an, daß M diejenigen natürlichen Zahlen umfassen soll, die eine bestimmte Eigenschaft haben bzw. für die ein bestimmtes Prädikat P wahr ist, und erhalten durch Umformulierung

(I)	$0 \in M$	0 hat die Eigenschaft P	*Verankerung*
(II)	$n \in M \Rightarrow S(n) \in M$	n hat P$\Rightarrow$ $S(n)$ hat P	*Vererbung*
(I),(II)$\Rightarrow$	$\forall n : n \in \mathbb{N}$	$\forall n : n$ hat P	

oder in der üblichen Formulierung

Wenn

$$\begin{array}{lll} P(0) & & \textit{Verankerung} \\ P(n) \Rightarrow P(n+1) & \text{für alle } n > 0 & \textit{Vererbung} \end{array} \tag{5}$$

dann $P(n)$ für alle n

Die zweite, für unser Thema grundlegende Konsequenz ist

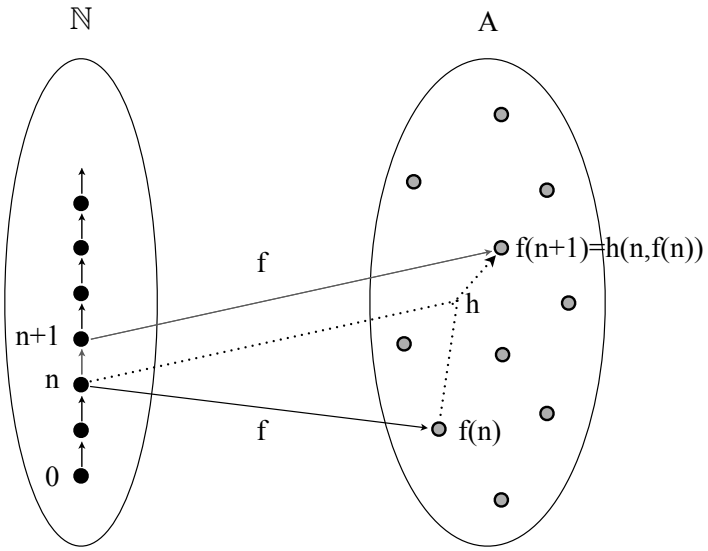
Das Rekursions-Prinzip

Satz 1 (Rekursionssatz, DEDEKIND)**.** *Ist $c \in A$ und $h : \mathbb{N} \times A \to A$, dann gibt es genau eine Funktion $f : \mathbb{N} \to A$ mit*

(1) $f(0) = c$
(2) $f(n+1) = h(n, f(n))$ *für alle* $n \in \mathbb{N}$

Zum Beweis ⇝[Eb].

Was bedeutet das?



Die Bedingungen (1) und (2) bewirken gleichermaßen eine Zählung in der Menge A:

(1) gibt an, welches das „erste“ Objekt in A ist.

(2) besagt, wie man von einem Objekt aus A zum nächsten kommt, während man in der Zahlenreihe von n zu dessen Nachfolger übergeht. Der Funktionswert von h kann u. U. außer von $f(n)$ auch von n abhängen, d.h. wie weit schon „gezählt“ worden ist.

Es ist also gleichbedeutend, ob man ausgehend von n den Nachfolger $n+1$ bildet und darauf die Funktion f anwendet, oder man f auf n anwendet und darauf zusammen mit n die 2-stellige Funktion h anwendet.

Damit steht uns ein effektives Instrument zur Konstruktion von Funktionen zur Verfügung: Da h nach Voraussetzung existiert, existieren auch die Funktionswerte $f(0), f(1), f(2), \ldots$

Für die praktische Anwendung nehmen noch eine Modifikation vor: Da wir üblicherweise Funktionswerte von n und nicht von $n+1$ berechnen, transformieren wir (2) zu:

$$\textit{(2')}: \qquad f(n) = h(n-1, f(n-1))$$

mit dem korrigierten Prädikat „für alle $n > 0$" bzw. „falls $n > 0$". Außerdem ersetzen wir der Einfachheit halber (2') durch $f(n) = h(n, f(n-1))$ (wenn h eine Funktion von $n-1$ ist, dann auch von n) und erhalten das *primitive Rekursionsschema*

$$f(n) = \begin{cases} c & \text{falls } n = 0 \qquad \textit{Basisfall} \\ h(n, f(n-1)) & \text{falls } n > 0 \qquad \textit{Allgemeiner Fall} \end{cases} \tag{6}$$

primitives Rekursionschema

Wir vergleichen (5) mit (6):

- Das (Beweis-)Prinzip *vollständige Induktion* und das Konstruktionsprinzip *primitives Rekursionsschema* folgen aus den Eigenschaften der natürlichen Zahlen und sind strukturell gleich.

Berücksichtigt man noch, daß der oben erwähnte Rekursionssatz auch für Funktionen mit Parametern gilt und daß – wie bei einer Induktionsaussage – der Basisfall nicht notwendigerweise bei $n = 0$ beginnen muß, sondern bei $n = n_0$ (Näheres s. *Transformationen* 7.3), erhält man ein erweitertes primitives Rekursionsschema:

$$f(p, n) = \begin{cases} c_p & \text{falls } n = n_0 \qquad \textit{Basisfall} \\ h(p, n, f(p, n-1)) & \text{falls } n > n_0 \qquad \textit{Allgemeiner Fall} \end{cases} \tag{7}$$

primitives Rekursionschema ab n_0 mit Parameter

Somit ist eine mit dem primitiven Rekursionschema konstruierte Funktion f durch c_p, n_0 und h eindeutig bestimmt. Das bedeutet, daß für vorgegebene $p, c_p \in A$ und $n > n_0 \in \mathbb{N}$ der Funktionswert $f(n) \in A$ existiert und nach dem Rekursionsschema berechnet werden kann.

Wie in wir in 7.2 sehen werden, müssen wir uns auf $A = \mathbb{Q}$ beschränken, wobei uns schon jetzt klar ist, daß in der Praxis nur $A = \widehat{\mathbb{Q}} \subset \mathbb{Q}$ zur Verfügung steht. Dabei soll nicht unerwähnt bleiben, daß das primitive Rekursionschema in (6) und (7) in der Literatur häufig nur für $A = \mathbb{N}$ definiert wird.

An dieser Stelle liegt ein Vergleich nahe: Das primitive Rekursionsschema ist ein Sonderfall des im vorherigen Abschnitt 2.1 behandelten Standard-Rekursionsschemas, und zwar insofern, daß bei ersterem die Rekursionsvariable x eine natürliche Zahl ist und für die Vorgängerfunktion gilt: $\varphi(x) = \varphi(n) = n - 1$. Diese Einschränkung stellt offensichtlich die „Berechenbarkeit" von f sicher.
Ohne den Begriff *berechenbar*, auf den wir in 7.2 ausführlich eingehen, zu präzisieren, können wir zusammenfassend feststellen:

■ **Das *primitive Rekursionschema***

- **ist ein Spezialfall des Standard-Rekursionsschemas**
- **erlaubt die Konstruktion von *berechenbaren* Funktionen**

Darüber hinaus wir mit dem primitiven Rekursionschema eine fundamentale Methode zur Funktions-Konstruktion ohne die eingangs erwähnte „Pünktchenschreibweise" an der Hand, wie z.B.

– Sei $h = +$ die Addition in $\mathbb{N}$, dann liefert (7) mit

$$f(a,n) = \begin{cases} 0 & \text{falls } n = 0 \\ a + f(a, n-1)) & \text{falls } n > 0 \end{cases}$$

die Multiplikation $a \cdot n$ in $\mathbb{N}$

– Sei $h = \cdot$ die Multiplikation und $a \in \mathbb{Q}$, dann liefert (7) mit

$$f(a,n) = \begin{cases} 1 & \text{falls } n = 0 \\ a \cdot f(a, n-1)) & \text{falls } n > 0 \end{cases}$$

die Potenzierung a^n

– Sei $h = \cdot$ die Multiplikation in $\mathbb{N}$, dann liefert (6) mit

$$f(n) = \begin{cases} 1 & \text{falls } n = 0 \\ n \cdot f(n-1)) & \text{falls } n > 0 \end{cases}$$

die Fakultät $n!$

3 Didaktik: Rekursive Modellierung

Die Eleganz von EUKLIDs *Verfahren zur ggT-Bestimmungen (s. 2.1) legt es nahe, auch für andere Problemstellungen eine rekursive Funktions-Konstruktion als Lösung anzustreben.*
Damit betreten wir den Bereich der funktionalen Modellierung, einem Kernstück der angewandten Mathematik zur Problemlösung, dessen Bedeutung im Bildungsgang – um nicht von den aktuell gehandelten „Kompetenzen" zu reden – nicht überschätzt werden kann.
Den Schwerpunkt bildet dabei die Gewinnung eines rekursiven Modells im Gegensatz zur üblichen Lösung in Form einer explizit definierten Funktion. Insbesondere wird die Entwicklung eines „rekursiven Gedankens" als didaktisches Prinzip vorgestellt.

3.1 Funktionale Modellierung

Stand der Dinge in der Bildungslandschaft

Obwohl die *funktionale Modellierung* in den letzten Jahren ein häufig erwähntes Konzept der Mathematik-Didaktiker geworden ist und mittlerweile sich in den meisten Lehrplänen der Bundesländer findet, taucht die Rekursion nur punktuell und unsystematisch auf, und zwar üblicherweise erstmalig in Kl. 10, etwa

- bei der Definiton von Folgen (linear und geometrisch) als alternative Beschreibung zur „expliziten" (geschlossenen, d.h. durch einen Term)
- bei Näherungsverfahren beim Wurzelziehen (HERON, s. 9.7) und der Kreisberechung, s. 9.7

und in Kl. 11/12

- im Rahmen der Analysis: wie Kl. 10, allerdings weitergehend, gelegentlich auch Näherungsverfahren wie z.B. NEWTON-Verfahren oder CAUCHY-EULER (s. 11.4)
- in der Stochastik: Permutationen, Wahrscheinlichkeitsverteilungen u.ä. (s. 9.2)

Der Begriff „Rekursion" wird dabei weitgehend vermieden, stattdessen wird z.B. bei Näherungsverfahren von „Iterationsvorschrift" gesprochen. Ansonsten wird – auch wenn die Problemstellung einen rekursiven Ansatz nahelegt – beim funktionalen Modellieren grundsätzlich eine *explizit* definierte Funktion

$$f: \; x_1, x_2, \ldots, x_n \; \rightarrow T(x_1, x_2, \ldots, x_n)$$

angestrebt und erarbeitet. Man kann sagen

- Rekursion wird (wenn überhaupt) zu spät eingeführt
- Rekursion wird nicht systematisch erlernt
- Rekursion wird zu selten benutzt

Symptomatisch für diese Situation ist ein Auszug aus einem Internetforum zur Mathematik im Anhang, s. Abschnitt 12.

Wir fassen zusammen:

■ **In den allgemeinbildenden Schulen (bis hin zum Abitur) führen Rekursion und rekursives Modellieren ein Schattendasein.**

Dieser Zustand erscheint umso bedauerlicher, weil Rekursion ein algorithmisches Grundparadigma für funktionales und prozedurales Modellieren ist:

■ **Rekursion bzw. rekursives Modellieren ist etwa zeitgleich mit der *expliziten* Funktionsmodellierung und gleicher Intensität einzuführen und ebenso in den folgenden Schuljahren fortzuführen.**
Damit stehen den Schülern zwei gleichwertige Varianten für *funktionale* Modellierung zur Verfügung.

In der gymnasialen Oberstufe und vor allem bei den Hochschulanfängern fällt häufig der Satz: *Rekursion ist schwer.*
Wie kann man das erklären? Machen wir uns die Situation beim Programmierkurs klar: Die Erstsemester kommen mit mehr oder weniger vorhandenen Kenntnissen einer bestimmten Programmiersprache von Schule und sollen jetzt – vermutlich in einer anderen Programmiersprache als in der Schule – systematisch programmmieren lernen; handelt es sich z.B. um eine für Anfänger ungeeignete Sprache wie JAVA, müssen sie sich aufgrund der überfrachteten Syntax mehr mit dem Lernen einer Programmiersprache herumschlagen als mit dem Programmieren-Lernen an sich im Sinne von Modellieren und „Denken": In dieser Situation werden sie mit der Rekursion als Novum konfrontiert und kämpfen somit an zwei Fronten.
Diese Zwickmühle entstünde für die Lernenden höchstwahrscheinlich nicht, wären sie von der Schule her mit Rekursion vertraut.

■ **Rekursion ist ein wichtiger Bestandteil der mathematischen Grundkenntnisse, die für ein Informatikstudium Voraussetzung sind**

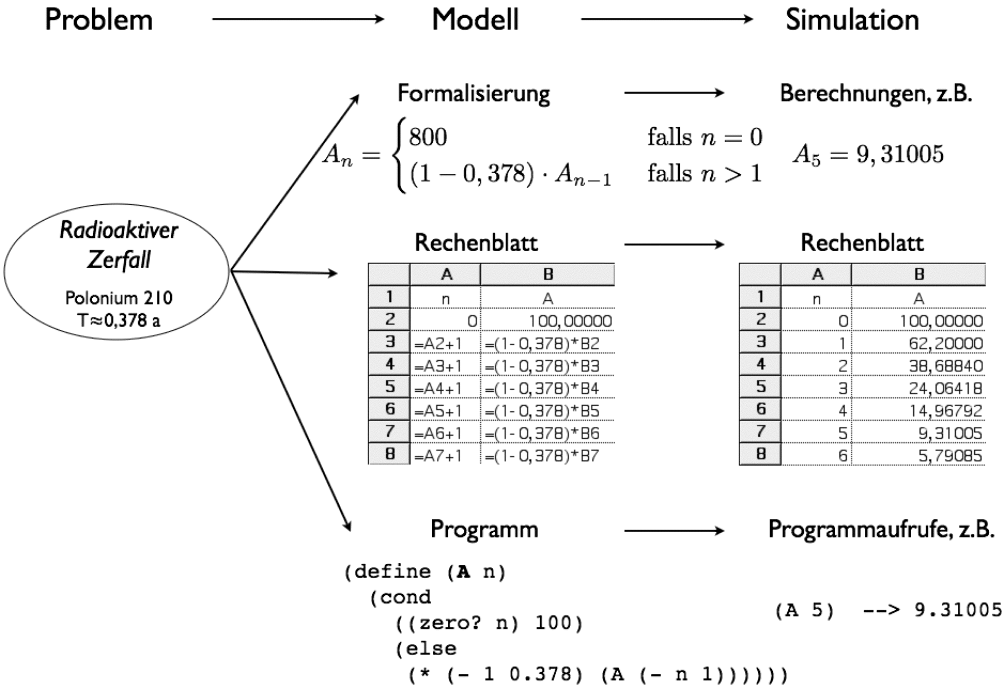
Modell vs. Modellierung

Unter *mathematischer Modellierung* oder *mathematischer Modellbildung* versteht man einen offenen Prozeß,

- dem ein reales Problem bzw. Phänomen aus Natur, Technik, Wirtschaftswissenschaft u.a. zugrunde liegt, zu dem reale (Mess-)Daten vorliegen
- der zu einem mathematischen Modell führt, nämlich einer *Funktion*, die durch mathematische Beschreibung („Deskription") einen Zusammenhang zwischen Ein- und Ausgabedaten herstellt

Die *Simulation* ist die Anwendung des Modells auf Eingabedaten, zu denen gemessene Ausgabedaten vorliegen, aber auch und insbesondere zu denen keine Messungen

vorliegen. Dabei sollen gemessene und simulierte Ausgabedaten möglichst gut übereinstimmen. Entsprechend den Simulationsergebnissen kann das Modell nach dem Prinzip eines Regelkreises verbessert werden.



	A	B
1	n	A
2	0	100,00000
3	=A2+1	=(1-0,378)*B2
4	=A3+1	=(1-0,378)*B3
5	=A4+1	=(1-0,378)*B4
6	=A5+1	=(1-0,378)*B5
7	=A6+1	=(1-0,378)*B6
8	=A7+1	=(1-0,378)*B7

	A	B
1	n	A
2	0	100,00000
3	1	62,20000
4	2	38,68840
5	3	24,06418
6	4	14,96792
7	5	9,31005
8	6	5,79085

Abb. 1: Modellierung: deskriptives Modell

Bei dem bisher Beschriebenen spricht man von einem *deskriptiven Modell*[1], da es als Abbild bestimmte quantitative Eigenschaften eines Phänomens beschreibt.

Demgegenüber gibt es auch das *normative Modell*: es wird kein Bezug auf ein vorhandenes beobachtbares Phänomen genommen, vielmehr werden mehr oder weniger willkürliche Vorgaben gemacht im Sinne einer Normierung oder Regel wie z. B. die Steuerformel der Einkommensteuer oder der Normierung von Papiergrößen, S. 146 oder die Tilgung eines Darlehens, S. 179.

Schließlich gibt es auch Rekursionen, die weder aus der Modellierung irgendeines Phänomens noch zur Normierung eines Sachverhaltes entstanden sind, sondern künstlich konstruiert oder einfach nur zufällig gefunden wurden, wie z.B. die ACKERMANN-Funktion und die COLLATZ-Rekursion, s. Abschnitt „Klassiker", S. 107 oder auch einfach nur innermathematischer Natur sind, s. Abschnitt 9.2, wie etwa die Anzahl der Primzahlen p mit $p \leq n$. Solche Rekursionen kann man als *modellfrei* bezeichnen.

[1] Nicht zu verwechseln mit *deskriptiver Modellierung*, s. 3.2

■ **Eine Rekursion kann**

- **ein *deskriptives* Modell**
- **ein *normatives* Modell**

sein oder

- **„modellfrei"**

Dabei können viele Phänomene zum gleichen (deskriptiven) Modell führen:

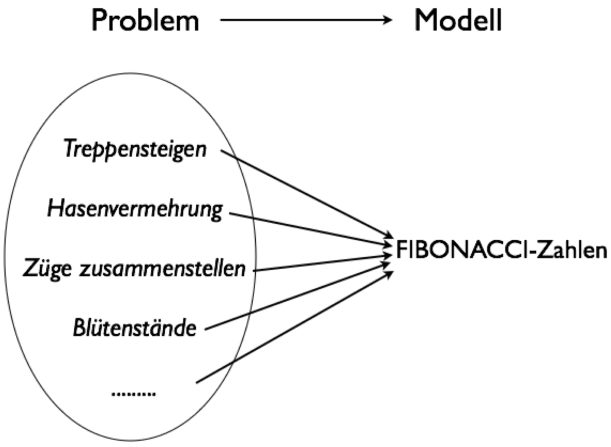


Abb. 2: Viele Phänomene, ein Modell

Umgekehrt gibt es auch zum gleichen Phänomen verschiedene Modelle, im einfachsten Fall die Entscheidung zwischen diskret oder stetig, s. S. 17.

Darüber hinaus gehört zu einer offenen funktionalen Modellierung aber auch Wahl zwischen einer *explizit* (geschlossen) definierten oder *rekursiv* definierten Funktion, und zwar in Abhängigkeit

- von der Natur des Sachproblems, z.B. diskret oder stetig bei dynamischen Modellen
- von der Schwierigkeit, einen geeigneten Funktionsterm zu finden
- vom rechnerischen Zeitaufwand
- von der Eleganz und der Einfachheit der Lösung
- von der Verfügbarkeit geeigneter Modellierungswerkzeuge

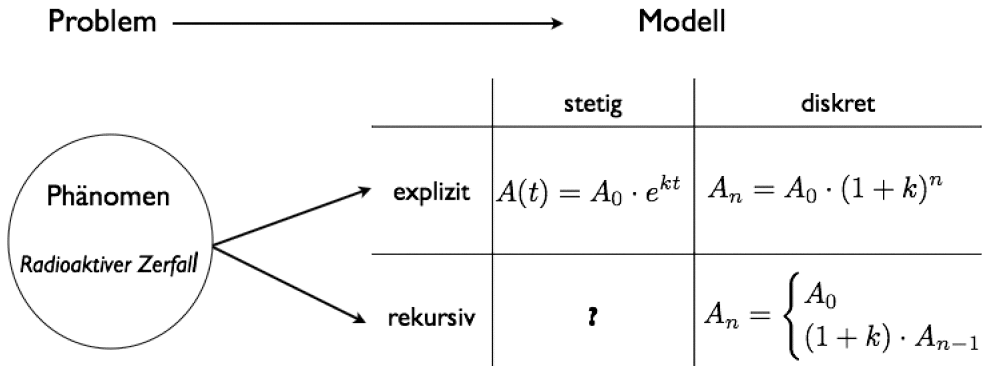


Abb. 3: Funktionale Modellierung: Varianten

Diskretes vs. stetiges Modell

Ein wichtige Klasse von Modellierungsaufgaben liefern die *dynamischen Prozesse*: es sind in der Regel *zeitabhängige* Prozesse oder, seltener, *ortsabhängige* Prozesse, bei denen sich eine Größe A in Abhängigkeit von x ändert, und zwar mit $x \mathrel{\widehat{=}}$ Zeit oder $x \mathrel{\widehat{=}}$ Abstand/Entfernung. Häufig handelt es sich um *Wachstumsprozesse*, die mit einer monotonen Funktion $A(x)$ modelliert werden (bei monoton fallenden Funktionen spricht man von *„negativem" Wachstum* oder *Abnahme*).
Die meisten realen Anwendungen stammen aus

- Natur und Technik (Anzahl von Lebewesen, Fläche, Gewicht, Länge etc.)
- dem Finanzwesen (Kapital, Rente, Hypothekenlast etc.)

Dabei müssen wir zwischen *diskreten* Prozessen und *stetig* (konitinuierlich) verlaufenden Prozessen unterscheiden:

Im ersten Fall setzt sich der aktuelle Wert sich $A(x)$ einer zeitabhängigen Größe A aus dem Wert $A(x - \Delta x)$ zu einem früheren Zeitpunkt und einem Zuwachs ΔA zusammen:

$$A(x) = A(x - \Delta x) + \Delta A$$

Ist der Zuwachs ΔA eine Funktion des früheren Wertes, also $\Delta A = H(A(x - \Delta x))$, liegt eine *Differenzengleichung 1. Ordnung* vor:

$$A(x) = A(x - \Delta x) + H(A(x - \Delta x))$$

Ändert sich dagegen die Größte A stetig, ergibt sich entsprechend eine *Differentialgleichung 1. Ordnung*

$$A'(x) = F(A(x))$$

Grundsätzlich können sowohl die Differenzengleichung als auch die Differentialgleichung als Beschreibungs-Modell für ein Phänomen angesehen werden.

Für eine vollständige Lösung muss in beiden Fällennoch ein Basisfall bzw. ein Anfangswert hinzugenommen werden.

	diskretes Modell	stetiges Modell
Phänomen	$A(x)$ ändert sich zu festen Zeitpunkten $x = n \cdot \Delta x$ mit $n \in \mathbb{N}$ um $\Delta A = H(A(x - \Delta x)$	$A(x)$ ändert sich zu jedem Zeitpunkt Δx beliebig klein
	$\downarrow$	$\downarrow$
(Modell)	$A(x) = A(x - \Delta x) + H(A(x - \Delta x))$ *Differenzengleichung 1. Ordn.* + Basisfall	$A'(x) = F(A(x))$ *Differentialgleichung 1. Ordn.* +Anfangswert
	$\downarrow$	$\downarrow$
Modell	rekursive Funktion	explizite Funktion

Abb. 4: diskrete vs. stetige Modellierung

Die bekanntesten Wachstumsprozesse führen zu einem Sonderfall, bei dem der Zuwachs bzw. die Differenz $\Delta A = H(A(x - \Delta x))$ eine lineare Funktion von $x - \Delta x$ ist, also

$$A(x) = A(x - \Delta x) + a \cdot A(x - \Delta x) + b$$

Man spricht man von einer *linearen Differenzengleichung 1. Ordnung* (ähnlich wie bei einer Differentialgleichung). Dazu gehören z.B. lineares, beschränktes und exponentielles Wachstum.
In diesen Fällen können die zugehörige Rekursion

$$A(x) = \begin{cases} A_0 & \text{falls } x = x_0 \\ A(x - \Delta x) + a \cdot A(x - \Delta x) + b & \text{sonst} \end{cases} \tag{8}$$

aufgelöst werden zu einer *expliziten diskreten* Lösung, wie man am Beispiel in Abb. 3 sieht. Allerdings beschreibt die stetige Lösung der entsprechenden Differentialgleichung 1. Ordnung i. A. nicht das gleiche Phänomen, wie in Abschnitt 9.4 gezeigt wird.

Fassen wir zusammen:

■
- **Eine *rekursive* Funktion stellt ein *diskretes* Modell dar**
- **Eine differenzierbare, *explizit* definierte Funktion stellt als Lösung einer Differentialgleichung ein *stetiges* Modell dar**

Typische Beispiele werden in 9.4 ausführlich behandelt, wobei man feststellen wird, daß es nicht immer notwendig oder üblich ist, diskrete Phänomen auch diskret zu modellieren oder stetige Phänomene auch stetig zu modellieren:

Phänomen	Modell	
	diskret	stetig
diskret	Kapitalentwicklung bei jährlicher Verzinsung ↓ z.B. Tabellierung mit Rechenblatt	Atomarer Zerfall (Idealisierung) ↓ z.B. Zerfallskurve
stetig	Tägliches Wachstum eines Fußnagels (Vereinfachung) ↓ z.B. Tabellierung mit Rechenblatt	Höhe eines Baumes als Funktion der Zeit ↓ z.B. Funktion $h : t \to h(t)$

■ Allerdings ist es auch möglich, bestimmte Rekursionen so zu programmieren, daß sie für $\widehat{\mathbb{Q}}$ ein stetiges Modell liefern! (s. *Variable Schrittweite* im Abschnitt 8.4, S. 95). Dagegen stellt die Tabellierung einer explizit definierten Funktion im Rechenblatt immer eine diskrete Modellierung dar (s. Abschnitt 4).

Was ist eigentlich gesucht?

Damit ist gemeint: Soll die Simulation einen einzelnen Funktionswert oder eine Liste bzw. Tabellierung von Werten liefern? Diese Frage taucht z. B. dann auf, wenn man die Wahl zwischen den Werkzeugen Tabellenkalkulation oder Programmierung hat: z.B. liefert bei einer Ratentilgung eine passende Funktion $R(n)$ die Restschuld nach n Jahren, also eine Zahl. Dagegen erwarten viele Kunden einen Tilgungsplan, in dem auch Zeit und Restschuld, ggf. auch der jährliche Zins tabelliert werden. Genau das liefert ein passendes Rechenblatt, während das Programm für $R(n)$ so modifiziert werden muß, daß es eine Tabellierung liefert. Wir unterscheiden daher:

1. Der **Funktionswert** $f(x)$ zu einem $x \in D_f$
   d.h. nur das Ergebnis im Sinne **eines** Wertes, also der Funktionswertes.

   Beispiele:

   - `(n! 50)`
     `-> 81591528324789773434561126959611589427200000000`
   - $\sqrt{5} \approx$ `(wurzel-a 1.5 5 10)`
     `-> 2.23606797749978969640917 36...`

2. Die **Wertetabelle**, also eine *endliche Folge* von *Paaren* aus Argument und Funktionswert, bestehend aus Basisfall und den Selbstaufrufen. Bei Rekursionen, die nicht über $\mathbb{N}$ laufen, müssen wir unterscheiden zwischen

   a) $((x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n)))$
   – Rekursionsvariable als Argument