Die Rekursivität der Programmierungsprache „Lisp 1.5“[1] in Spezialfällen der angeordneten freien holomorphen Mengen

Von R. PÉTER

1. In 1959 habe ich[2] den allgemeinen Begriff der angeordneten freien holomorphen Mengen (Mengen mit „zahlenartig aufbaubaren“ Elementen) eingeführt, und die Theorie der rekursiven Funktionen für solche abstrakte Mengen als Definitionsbereiche verallgemeinert; ferner als wichtigsten Spezialfall den Fall der „Wortemengen“ (Mengen der endlichen Ketten, „Worte“ genannt, aus Elementen je einer gegebenen Menge, „Alphabet“ genannt) über Alphabete beliebiger Mächtigkeiten ausgearbeitet. Auch von anderen Verfassern wurde besonders dieser Spezialfall des allgemeinen Begriffes vielseitig untersucht und angewandt. Als einen davon abweichenden Spezialfall habe ich[3] den Begriff der PAIRschen „freien Binoiden“ in dieser Hinsicht untersucht. Als ein anderer Spezialfall bietet sich die Menge der „symbolischen Ausdrücke“ in der Programmierungsprache „Lisp 1.5“, worum es sich in dieser Arbeit handelt.

2. Zuerst gebe ich die notwendigen Kenntnisse über die angeordneten freien holomorphen Mengen an.

Sei H eine beliebige nicht leere Menge, H_0 eine nicht leere Teilmenge von H (alle Elemente von H_0 werden die Rolle von 0 spielen) und F eine nicht leere Teilmenge der auf H definierten und nur Werte aus H annehmenden Funktionen beliebig vieler Variablen (alle Elemente von F werden die Rolle der Nachfolgerfunktion spielen). Ferner sei H_n für $n=1, 2, \ldots$ folgendermaßen angegeben: sind die Mengen $H_0, H_1, \ldots, H_n$ bereits definiert, dann sei H_{n+1} die Menge jener Elemente von H, welche als Werte der Funktionen aus F angenommen werden, falls für ihre Argumente Elemente aus $H_0 \cup H_1 \cup \ldots \cup H_n$ gesetzt werden, und dabei mindestens für ein Argument ein Element aus H_n. Wird nun H durch die Union sämtlicher Teilmengen $H_0, H_1, H_2, \ldots$ erschöpft, dann wird H eine *holomorphe Menge* genannt.

[1]Siehe: *LISP 1.5 Programmer's Manual*, The Computation Center and Research Laboratory of Electronics, Massachusetts Institute of Technology (1962). Zitiert als [1].

[2]Siehe: R. PÉTER: *Über die Varellgemeinerung der Theorie der rekursiven Funktionen für abstrakte Mengen geeigneter Stuktur als Deflnitionsbereiche*, Acta Math. Ac. Sci. Hung., Teil I in **12** (1961), S. 271-314 (mit Angabe in der ersten Fußnote der Geschichte dieses Themenkreises); Teil II in **13** (1962), S. 1-24 (mit einigen Berichtigungen zum Teil I; betrefs weiterer Berichtigungen berufe ich mich auf [3]). Zitiert als [2].

[3]Siehe: R. PÉTER: *Die* PAIR*schen freien Binoiden als Spezialfälle der angeordneten freien holomorphen Mengen*, Acta Math. Ac. Sci. Hung. **21** (1970), S. 297-313. Zitiert als [3].

Und zwar eine *freie holomorphe Menge,* falls die Mengen H_n paarweise disjunkt sind, und für jedes nicht zu H_0 gehörige Element von H eindeutig bestimmt ist, aus welcher zu F gehörigen Funktion und durch Einsetzen welcher Elemente für deren Argumente es entsteht. Im weiteren wird H immer eine freie holomorphe Menge bezeichnen, und auch die Bezeichnungen $F, H_0, H_1, H_2, \ldots$ werden beibehalten.

$h \in H_i$ wird auch so ausgedrückt, daß die Ordnung $o(h)$ von h gleich i ist.

Es ist auch die Einführung des Begriffes der *Vorgänger* je eines Elementes von H notwendig; diese kann aber in verschiedenen Anwendungen verschiedenartig angegeben werden; jedoch (mit der Bezeichnung „$\prec$" für echte Vorgänger) so, daß die folgenden Forderungen erfüllt seien:

(1) $x \leqq x$.

(2) Ist x der Form $f(y_1, \ldots, y_n)$, wo $f \in F$ ist, so gehören die „unmittelbaren Konstituenten" $y_1, \ldots, y_n$ von x zu den echten Vorgängern von x.

(3) Aus $x \prec y$ und $y \prec z$ folgt $x \prec z$.

(4) Ein echter Vorgänger von x ist immer kleinerer Ordnung als x.

Der ebenfalls notwendige Begriff der *unmittelbaren Vorgänger* der Elemente von X kann ebenfalls verschiedenartig angegeben werden, allein mit den Forderungen:

(5) Es gebe für jedes festgewählte $f \in F$ eine von den Argumenten unabhängige obere Schranke für die Anzahl der unmittelbaren Vorgänger von Elementen der Struktur

$$x = f(\ldots).$$

(6) Jeder echte Vorgänger eines x sei Vorgänger eines unmittelbaren Vorgängers von x.

Eine freie holomorphe Menge H mit einem den Forderungen (1)—(6) genügenden Vorgängerbegriff wird eine (partiell) *angeordnete freie holomorphe Menge* genannt. Von nun an sei durch H immer eine solche Menge bezeichnet.

Auf Grund einer beliebigen solchen Menge H als Definitionsbereich kann eine rekursive Theorie aufgebaut werden.

3. In der Programmierungsprache „Lisp 1.5" werden „Listen" (endliche geordnete Mengen) untersucht, die auf Grund gewisser „Atome" aufgebaut werden. Die Atome sind Elemente einer Wortemenge über ein endliches Alphabet A_0, das Buchstaben, Ziffern, und einige spezielle Charaktere enthält. Die Länge der verwendbaren Worte wird aber eingeschränkt; so gibt es nur endlich viele Atome. Die Glieder der Listen sind Atome oder Listen. Die beliebigvielgliedrigen Listen können auf Paare abgebaut werden: als Gegenstück des ersten Gliedes der Liste kann die Liste der übrigen Glieder gewählt werden; dieses letztere kann ebenfalls auf diese Weise als ein Paar aufgezeichnet werden, usw.; als Gegenstück des letzten Gliedes kann die durch „NIL" bezeichnete leere Liste aufgenommen werden, die auch als ein Atom betrachtet wird. So werden statt Listen sog. „S-Ausdrücke" („symbolische Ausdrücke") behandelt, wobei erstens die Atome als S-Ausdrücke gelten, dann aus je zwei beliebigen S-Ausdrücken s_1 und s_2 als ein Paar der folgenderweise bezeichnete S-Ausdruck s gebildet wird:

$$s = (s_1 \cdot s_2).$$

Gehört s zu einer Liste, so gehört s_1 zum ersten Glied („head") dieser Liste, und s_2 zu jener (eventuell leeren) Liste („tail"), die aus der ursrünglichen Liste durch Weglassen des ersten Gliedes entsteht.

s_1 bzw. s_2 als Funktionen von s werden durch $s_1 = \text{car}(s)$ bzw. $s_2 = \text{cdr}(s)$ bezeichnet, und s als Funktion von s_1 und s_2 durch $s = \text{cons}(s_1, s_2)$.

Man zeigt leicht,[4] daß diese Funktionen, und viele mittels diesen definierten weiteren Begriffe des „Lisp 1.5" in der Wortemenge sowohl über das Alphabet A_0 als auch über das aus den zugelassenen Atomen bestehenden Alphabet A (beidenfalls mit Hinzunahme der Klammern und der Trennzeichen) primitiv-rekursiv sind.

4. Doch die rekursive Theorie der S-Ausdrücke kann nicht nur in eine Wortemenge eingebettet behandelt werden, sondern sie bietet sehr natürlicher Weise ein Beispiel für einen anderen Fall der angeordneten freien holomorphen Mengen. Darin spielt jedes Atom (von nun an — NIL inbegriffen — nicht als Zeichenkette, sondern als ein einziges Zeichen betrachtet) die Rolle von 0, und $\text{cons}(x, y)$ die Rolle der „Nachfolgerfunktion"; ferner besitzt jedes nicht-Atom Element x zwei „unmittelbare Vorgänger":

$$\text{car}(x) \quad \text{und} \quad \text{cdr}(x).$$

Der Aufbau der Theorie der rekursiven Funktionen für eine derartige Menge H wird dadurch erleichtert, daß die (einzige) Nachfolgerfunktion hier zweistellig ist. Für Fälle, wobei es unter den Nachfolgerfunktionen auch eine mindestens zweistellige gibt, habe ich in der Bemerkung (I) auf S. 299 von [2] eine Zuordnung geeigneter Elemente zu je einer Elementenfolge angegeben, und diese auch in [3] benutzt. Hier bietet sich nun eine ähnliche Zuordnung eines S-Ausdrucks zu je einer S-Ausdruck-Folge, wie je einer Liste ein S-Ausdruck zugeordnet wurde; und eine derartige Zuordnung kann auch auf beliebige solche H-Mengen verallgemeinert werden, bei welchen die Menge F auch eine mindestens zweistellige Nachfolgerfunktion enthält.

5. Nun sei die H-Menge der S-Ausdrücke näher betrachtet.

Dabei ist H_0 die (endliche) Menge der Atome, und davon ausgehend entstehen für $n = 1, 2, \ldots$ die Elemente von H_n wie im allgemeinen Fall, durch Anwendungen der einzigen Nachfolgerfunktion $\text{cons}(x, y)$. Jedes nicht-Atom Element x von H hat die Form

$$(x_1 \cdot x_2),$$

wobei x_1 und x_2 eindeutig bestimmte Zeichenketten sind:

$$x_1 (= \text{car}(x))$$

ist jene Zeichenkette, die man aus der Zeichenkette x erhält, wenn man nach Weglassen der ersten Anfangsklammer nach rechts gehend die Zeichen bis zum ersten solchen Punkt kopiert, bis welchem die Anzahl der aufgetretenen Anfangs- und Endklammern übereinstimmt; dann ist

$$x_2 (= \text{cdr}(x))$$

[4] Siehe das Buch: R. PÉTER: *Rekursive Funktionen in der Komputer-Theorie* (im Erscheinen).

jene Zeichenkette, die aus dem übriggebliebenen Teil der Zeichenkette x durch Weglassen des ersten Punktes und der letzten Endklammer entsteht.

Als festgesetzte Elemente je einer der Mengen

$$H_0, H_1, H_2, \ldots$$

können

$$h_0, h_1, h_2, \ldots$$

gewählt werden, wobei

$$h_0 = \text{NIL}$$

und für jedes n

$$h_{n+1} = \text{cons}\,(h_n, h_n)$$

ist.

Jedes Atom besitzt als einzigen (unechten) Vorgänger sich selber. Als unmittelbare Vorgänger eines Elementes $x = \text{cons}\,(x_1, x_2)$ gelten seine unmittelbare Konstituenten x_1 und x_2, und als echte Vorgänger von x gelten die Vorgänger dieser Konstituenten. Offenbar werden für diese Begriffe die Forderungen (1)—(6) erfüllt.

Nach alldem ist unser H eine angeordnete freie holomorphe Menge.

6. Für dieses H, wobei die Menge der Atome

$$A = \{a_1, a_2, \ldots, a_t\}$$

sei, lautet das Schema der primitiven Rekursion:

$$\begin{cases} f(a, u_1, \ldots, u_n) = g_a(u_1, \ldots, u_n), \text{ falls } a \in H_0 (= A) \\ f(\text{cons}\,(x_1, x_2), u_1, \ldots, u_n) = g(x_1, x_2, u_1, \ldots, u_n, f(x_1, u_1, \cdots, u_n), f(x_2, u_1, \ldots, u_n)), \end{cases}$$

wobei

$$g_{a_1}, \cdots, g_{a_t}, g$$

bereits definierte Funktionen bezeichnen.

Als Ausgangsfunktionen gelten unbedingt die Atome, d. h. die Elemente von H_0 (als Konstanten), das einzige Element $(\text{cons}\,(x_1, x_2))$ von F und eventuell noch weitere hinzuzunehmende Funktionen. Die in H primitiv-rekursiven Funktionen sind jene, die aus den Ausgangsfunktionen ausgehend durch endlich viele Substitutionen und primitive Rekursionen entstehen. Dabei darf jede Funktion so betrachtet werden, als eine Funktion ihrer Argumente und beliebig (endlich) vieler „hinzugenommenen" Argumente, von denen sie nicht tatsächlich abhängt.

Eine Relation $R(x_1, \ldots, x_n)$ ist primitiv-rekursiv, falls ihre „charakteristische Funktion"

$$r(x_1, \ldots, x_n) = \begin{cases} h_0, \text{ falls } R(x_1, \ldots, x_n) \\ h_1 \text{ sonst} \end{cases}$$

primitiv-rekursiv ist.

Zum Beispiel sind car (x) und cdr (x), die nur dann eine Rolle spielen, wenn x kein Atom ist, primitiv-rekursiv, da sie durch die folgenden primitiven Rekursionen definiert werden können:

$$\begin{cases} \text{car}\,(a) = \text{NIL} = h_0, \text{ falls } a \in H_0 \\ \text{car}\,(\text{cons}\,(x_1, x_2)) = x_1 \end{cases}$$

bzw.

$$\begin{cases}\mathrm{cdr}\,(a) = h_0, \text{ falls } a \in H_0\\ \mathrm{cdr}\,(\mathrm{cons}\,(x_1, x_2)) = x_2.\end{cases}$$

Bereits in diesen Definitionen wurde die Primitiv-Rekursivität der Identitätsfunktion $f(x)=x$ benutzt. Diese kann durch

$$\begin{cases}\mathrm{id}\,(a) = a, \text{ falls } a \in H_0\\ \mathrm{id}\,(\mathrm{cons}\,(x_1, x_2)) = \mathrm{cons}\,(x_1, x_2)\end{cases}$$

definiert werden, wobei jede Konstante $a \in H_0$ und die Nachfolgerfunktion „cons" zu den Ausgangsfunktionen gehört.

Die festgelegten Elemente

$$h_0, h_1, h_2, \ldots$$

sind primitiv-rekursiv, da dies für das Atom h_0 gilt, und gilt es bereits für h_n, so vererbt es sich auch auf das durch die Substitution cons (h_n, h_n) entstehende h_{n+1}.

Die charakteristische Funktion

$$\mathrm{atom}\,(x) = \mathrm{sig}\,(x)$$

der Eigenschaft „ein 0-Element", d. h. „ein Element von H_0", d. h. „ein Atom zu sein", und das entgegengesetzte $\overline{\mathrm{sig}}\,(x)$ können durch folgende primitive Rekursionen definiert werden:

$$\begin{cases}\mathrm{sig}(a) = h_0, \text{ falls } a \in H_0\\ \mathrm{sig}(\mathrm{cons}(x_1, x_2)) = h_1,\end{cases} \qquad \begin{cases}\overline{\mathrm{sig}}\,(a) = h_1, \text{ falls } a \in H_0\\ \overline{\mathrm{sig}}\,(\mathrm{cons}(x_1, x_2)) = h_0.\end{cases}$$

Wird im letzteren für x die charakteristische Funktion $r(u_1, \ldots, u_n)$ einer Relation $R(u_1, \ldots, u_n)$ eingesetzt, so erhält man in $\overline{\mathrm{sig}}\,(r(u_1, \ldots, u_n))$ die charakteristische Funktion der Negation $\bar{R}(u_1, \ldots, u_n)$ dieser Relation.

Auch die zweistelligen logischen Operationen (die Disjunktion $R_1 \vee R_2$, die Konjunktion $R_1 \& R_2$, die Implikation $R_1 \rightarrow R_2$) sind primitiv-rekursiv, wie das in [2] allgemein bewiesen wurde.

Ebenfalls wurde dort allgemein bewiesen, daß samt den Funktionen $g_1, g_2, \ldots, g_{m+1}$ und den Relationen $R_1, R_2, \ldots, R_m$, wobei an jeder Stelle höchstens eine der genannten Relationen gilt, auch die aus ihnen folgenderweise „zusammengeflickte" Funktion f in H primitiv-rekursiv ist:

$$f(x_1, \ldots, x_n) = \begin{cases}g_1(x_1, \ldots, x_n), \text{ falls } R_1(x_1, \ldots, x_n)\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ g_m(x_1, \ldots, x_n), \text{ falls } R_m(x_1, \ldots, x_n)\\ g_{m+1}(x_1, \ldots, x_n) \text{ sonst}\end{cases}$$

(wobei „sonst" auch mit dem Bestehen einer primitiv-rekursiven Relation

$$R_{m+1}(x_1, \ldots, x_n) = \bar{R}_1(x_1, \ldots, x_n) \& \ldots \& \bar{R}_m(x_1, \ldots, x_n)$$

gleichbedeutend ist).

Hier bemerke ich, daß die Fallunterscheidungsform (die „konditionelle" Definition) von McCarthy, die in [1] benutzt wird, der folgende Spezialfall dieser Definition ist:

$$f(x_1, \ldots, x_n) = \begin{cases} g_1(x_1, \ldots, x_n), \text{falls } R_1(x_1, \ldots, x_n) \\ g_2(x_1, \ldots, x_n), \text{ falls } \bar{R}_1(x_1, \ldots, x_n) \,\&\, R_2(x_1, \ldots, x_n) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ g_m(x_1, \ldots, x_n), \text{falls } \bar{R}_1(x_1, \ldots, x_n) \,\&\, \ldots \,\&\, \bar{R}_{m-1}(x_1, \ldots, x_n) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \&\, R_m(x_1, \ldots, x_n). \end{cases}$$

Sind hier nicht alle Möglichkeiten erschöpft, so ist f im „Sonstfall" undefiniert; könnte aber beliebig, z. B. als h_0 definiert werden, da dies in den Anwendungen immer belanglos ist.

7. Zu den weiteren wird auch die primitiv-rekursive Definition der charakteristischen Funktion der Gleichheit:

$$\text{equal}(x, y) = \begin{cases} h_0, \text{falls } x = y \\ h_1 \text{ sonst} \end{cases}$$

gebraucht.

Zuerst können die charakteristischen Funktionen der Eigenschaften: „mit je einem Atom a_i gleich zu sein" durch die primitiven Rekursionen

$$\begin{cases} f(a_i) = h_0 \\ f(a_j) = h_1 \text{ für jedes von } a_i \text{ verschiedene } a_j \in H_0 \\ f(\text{cons}(x_1, x_2)) = h_1 \end{cases}$$

definiert werden.

Dann kann die charakteristische Funktion $\text{eq}(x, y)$ der Relation, daß x ein mit y übereinstimmendes Atom ist, durch die „Zusammenflickung"

$$\text{eq}(x, y) = \begin{cases} h_0, \text{falls } (x = a_1 \,\&\, y = a_1) \vee \ldots \vee (x = a_t \,\&\, y = a_t) \\ h_1 \text{sonst} \end{cases}$$

definiert werden.

Damit läßt sich $\text{equal}(x, y)$ wie folgt definieren:

Sei

$$g(x, y) = \begin{cases} h_0, \text{falls } x = h_0 \,\&\, y = h_0 \\ h_1 \text{ sonst} \end{cases}$$

(da h_0 ein Atom ist, ist g eine primitiv-rekursive Funktion), dann gilt

$$\begin{cases} \text{equal}(a, y) = \text{eq}(a, y), \text{falls } a \in H_0 \\ \text{equal}(\text{cons}(x_1, x_2), y) = g\big(\text{equal}(x_1, \text{car}(y)), \text{equal}(x_2, \text{cdr}(y))\big). \end{cases}$$

Das ist aber keine primitive Rekursion, da dabei der Parameter y nicht unverändert bleibt, sondern dafür einmal $\text{car}(y)$ einmal $\text{cdr}(y)$ eingesetzt wird.

Ich habe zwar in [2] allgemein gezeigt, daß — bei eventueller Hinzunahme gewisser neuen Ausgangsfunktionen — eine derartige Definition auf die sog.

„Wertverlaufsrekursion", und diese auf primitive Rekursion (nebst Substitution) zurückgeführt werden kann; aber dabei wurde auch die charakteristische Funktion der Gleichheit benutzt. Diese (also equal (x, y)) wird daher auch hier zu den Ausgangsfunktionen hinzugenommen. Die Aufnahme weiterer Ausgangsfunktionen ist aber in unserem Spezialfall überflüssig; dies zeige ich in den folgenden.

8. Ich schicke voraus, daß es überflüssig ist in den einzelnen Fällen der „Zusammenflickung" immer wieder derartige Hilfsfunktionen einzuführen, wie g in der Definition von equal (x, y) war. Man kann sich z. B. ein für allemal klarlegen, daß auch die folgende „erweiterte Zusammenflickung" (wobei auch Parameter auftreten können), wo sich die verwendeten Relationen gegenseitig ausschließen, nicht von der Klasse der primitiv-rekursiven Funktionen hinausführt:

$$\begin{cases} f(a) = g_a, \text{falls } a \in H_0 \\ f(\text{cons}(x_1, x_2)) = \begin{cases} g_1'(x_1, x_2, f(x_1), f(x_2)), \text{falls } R_1(x_1, x_2, f(x_1), f(x_2)) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ g_m'(x_1, x_2, f(x_1), f(x_2)), \text{falls } R_m(x_1, x_2, f(x_1), f(x_2)) \\ g_{m+1}'(x_1, x_2, f(x_1), f(x_2)) \text{ sonst;} \end{cases} \end{cases}$$

denn mit

$$g(x_1, x_2, v_1, v_2) = \begin{cases} g_1'(x_1, x_2, v_1, v_2), \text{falls } R_1(x_1, x_2, v_1, v_2) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ g_m'(x_1, x_2, v_1, v_2), \text{falls } R_m(x_1, x_2, v_1, v_2) \\ g_{m+1}'(x_1, x_2, v_1, v_2) \text{ sonst} \end{cases}$$

kann $f(x)$ durch die folgende primitive Rekursion definiert werden:

$$\begin{cases} f(a) = g_a, \text{falls } a \in H_0 \\ f(\text{cons}(x_1, x_2)) = g(x_1, x_2, f(x_1), f(x_2)). \end{cases}$$

Als Spezialfälle der „erweiterten Zusammenflickung" wurden in [2] bei primitivrekursivem $R(x, u_1, \dots, u_n)$ auch die Relationen: „Es gibt ein Vorgänger von x, für den R besteht":

$$(\exists y)[y \leqq x \,\&\, R(y, u_1, \dots, u_n)],$$

und „für alle Vorgänger von x gilt R":

$$(\forall y)[y \leqq x \to R(y, u_1, \dots, u_n)],$$

ferner die Funktion:

$$\mu_y[y \leqq x \,\&\, R(y, u_1, \dots, u_n)]$$

als primitiv-rekursiv definiert, wobei das letzte den ersten in einer bestimmten Reihenfolge der Vorgänger von x bedeutet, für den R besteht, falls ein solcher Vorgänger von x existiert; und sonst als h_0 definiert wird. (Ist x eine natürliche Zahl, dann bedeutet so y die kleinste natürliche Zahl gewünschter Art bis x.)

Als jene bestimmte Folge (mit erlaubten Wiederholungen) der Vorgänger von x ist es in unserem Spezialfall zweckmässig die Folge

$$(*) \qquad \bar{x}_0, \bar{x}_1, \dots, \bar{x}_b$$

zu wählen, worin für $x \in H_0$ natürlich

$$b = 0 \quad \text{und} \quad \bar{x}_0 = x$$

gilt, und für ein $x = \text{cons}(x_1, x_2)$ erst die Vorgänger von x_2 (in der betreffenden Reihenfolge), dann die Vorgänger von x_1 (in der betreffenden Reihenfolge) auftreten, und endlich

$$x = \bar{x}_b.$$

9. Mit Benutzung von equal (x, y) kann nun die charakteristische Funktion vorg (x, y) der Relation $y \leqq x$ wie folgt als eine primitiv-rekursive Funktion definiert werden:

$$\begin{cases} \text{vorg}(a, y) = \text{eq}(y, a), \text{ falls } a \in H_0 \\ \text{vorg}(\text{cons}(x_1, x_2), y) = \begin{cases} h_0, \text{ falls } \text{cons}(x_1, x_2) = y \vee \text{vorg}(x_1, y) = h_0 \vee \\ \qquad\qquad \vee \text{vorg}(x_2, y) = h_0 \\ h_1 \text{ sonst.} \end{cases} \end{cases}$$

Damit ist auch die Relation

$$y \prec x \equiv y \leqq x \,\&\, y \neq x$$

primitiv-rekursiv.

Werden nun die natürlichen Zahlen

$$0, 1, 2, \ldots$$

der Reihe nach mit

$$h_0, h_1, h_2, \ldots$$

identifiziert, so gilt für jede natürliche Zahl i:

$$o(h_i) = h_i = i = o(i),$$

ferner ist

$$o(x) \prec o(y)$$

mit

$$o(x) \prec o(y)$$

gleichbedeutend, und aus

$$x \leqq o(y)$$

folgt, daß x eine natürliche Zahl, also

$$x = o(x)$$

ist; und $o(x)$ kann wie folgt als eine primitiv-rekursive Funktion definiert werden:

$$\begin{cases} o(a) = h_0, \text{ falls } a \in H_0 \\ o(\text{cons}(x_1, x_2)) = \begin{cases} \text{cons}(o(x_1), o(x_1)), \text{ falls } o(x_2) \leqq o(x_1) \\ \text{cons}(o(x_2), o(x_2)) \text{ sonst.} \end{cases} \end{cases}$$

Für $x = \text{cons}(x_1, x_2)$ wurde $o(x)$ eigentlich mit Verwendung des früheren Wertes $o(x^{-1})$ definiert, wobei durch x^{-1} unter x_1, x_2 ein bestimmtes von nicht-kleinerer Ordnung bezeichnet wird (dessen Ordnung um 1 kleiner als $o(x)$ ist).

In den weiteren wird auch benutzt, daß das Schema (wo auch Parameter auftreten können)

$$\begin{cases} f(a) = g_a, \text{ falls } a \in H_0 \\ f(\text{cons}(x_1, x_2)) = g\big(x_1, x_2, f(x_1), f(x_2), f(\text{cons}^{-1}(x_1, x_2))\big) \end{cases}$$

nicht von der Klasse der in H primitiv-rekursiven Funktionen hinausführt. Denn mit Verwendung der primitiv-rekursiven Hilfsfunktion

$$g'(x_1, x_2, v_1, v_2,) = \begin{cases} g(x_1, x_2, v_1, v_2, v_1), \text{ falls } o(x_2) \leqq o(x_1) \\ g(x_1, x_2, v_1, v_2, v_2) \text{ sonst} \end{cases}$$

kann $f(x)$ durch die primitive Rekursion

$$\begin{cases} f(a) = g_a, \text{ falls } a \in H_0 \\ f(\text{cons}(x_1, x_2)) = g'(x_1, x_2, f(x_1), f(x_2)) \end{cases}$$

definiert werden.

Im Spezialfall $g_a = a$ und $g(x_1, x_2, v_1, v_2, v_3) = v_3$ ergibt sich selbst x^{-1} als primitiv-rekursive Funktion.

10. Mit Verwendung von $o(x)$ wurde in [2] der — im folgenden als *Satz in* [2] zitierte — Satz erhalten, daß jede zahlentheoretische primitiv-rekursive Funktion zu einer primitiv-rekursiven Funktion in H erweitert werden kann. Genauer: Es gibt zu jeder primitiv-rekursiven zahlentheoretischen Funktion $\varphi(m_1, \ldots, m_n)$ eine primitiv-rekursive Funktion $f(x_1, \ldots, x_n)$ in H, für welche

$$o(f(x_1, \ldots, x_n)) = \varphi(o(x_1), \ldots, o(x_n))$$

gilt. Daraus folgt, daß die in H primitiv-rekursive Funktion

$$g(x_1, \ldots, x_n) = o(f(x_1, \ldots, x_n))$$

und die zahlentheoretische primitiv-rekursive Funktion φ für natürliche Zahlen als Argumente übereinstimmen. Derartige „Vertreter" der zahlentheoretischen primitiv-rekursiven Funktionen in H werden hier ebenso bezeichnet, wie die Funktionen die sie vertreten.

Statt einer beliebigen natürlichen Zahl n ist in H passender $o(x)$ zu gebrauchen. Z. B. wird die $o(x)$-te Iterierte einer primitiv-rekursiven Funktion $f(y)$ folgenderweise als primitiv-rekursive Funktion definiert:

$$\begin{cases} f^{(o(a))}(y) = y, \text{ falls } a \in H_0 \\ f^{(o(\text{cons}(x_1, x_2)))}(y) = f(f^{(o(\text{cons}^{-1}(x_1, x_2)))}(y)). \end{cases}$$

Ich bemerke hier, daß die nicht 0-ten Iterierten von $o(y)$ alle gleich $o(y)$ sind, da $o(y)$ immer eine natürliche Zahl ist, und so

$$o(o(y)) = o(y)$$

gilt.

11. Zur Behandlung der sog. Wertverlaufsrekursion hat man den endlichen Folgen der Elemente von H je ein Element derart zuzuordnen, daß daraus die Glieder der Folge wiedererkannt werden können. Für den Fall, wobei unter den Nach-

folgerfunktionen auch ein mindestens zweistelliger vorkommt, habe ich in [2] eine verhältnismäßig einfache derartige Zuordnung angegeben. In unserem Spezialfall liegt es aber an der Hand ein anderes Verfahren (das auch verallgemeinert werden kann) anzuwenden: die endlichen Elementenfolgen als Listen (von S-Ausdrücken) zu betrachten, welchen in den Früheren bereits je ein S-Ausdruck zugeordnet wurde:

$$\begin{aligned}
&\text{der aus } s_0 \quad \text{bestehenden Liste } (s_0 \cdot \mathrm{NIL}),\\
&\text{der aus } s_0, s_1 \quad \text{bertehenden Liste } (s_0 \cdot (s_1 \cdot \mathrm{NIL})),\\
&\text{der aus } s_0, s_1, s_2 \text{ bestehenden Liste } \big(s_0 \cdot (s_1 \cdot (s_2 \cdot \mathrm{NIL}))\big),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&\text{und der leeren Liste etwa} \qquad \mathrm{NIL} = h_0.
\end{aligned}$$

So wird einer Elementenliste

$$s_0, s_1, \ldots, s_n$$

ein Element

$$s = c_n(s_0, s_1, \ldots, s_n) = \mathrm{cons}\,(s_0, \mathrm{cons}\,(s_1, \ldots, \mathrm{cons}\,(s_n, h_0)\ldots))$$

zugeordnet. Man sieht, daß dabei

$$n = o(n) \leqq o(s)$$

gilt.

Aus s können die Glieder der Liste als primitiv-rekursive Funktionen zurückerhalten werden:

$$s_0 = \mathrm{car}\,(s),\ s_1 = \mathrm{car}\,(\mathrm{cdr}\,(s)),\ s_2 = \mathrm{car}\,(\mathrm{cdr}\,(\mathrm{cdr}\,(s))),\ \ldots,\ s_n = \mathrm{car}\,(\mathrm{cdr}^{(n)}\,(s)),$$

und es gilt

$$\mathrm{cdr}^{(n+1)}(s) = h_0.$$

Nach den Bisherigen kann die charakteristische Funktion der Eigenschaft „x entspricht einer Elementenliste“ wie folgt primitiv-rekursiv definiert werden:

$$\mathrm{list}(x) = \begin{cases} h_0, \text{ falls } x = h_0 \vee (\exists y)[y \leqq o(x) \;\&\; \mathrm{cdr}^{(o(y))}(x) \notin H_0 \;\&\; \mathrm{cdr}^{(o(y)+1)}(x) = h_0] \\ h_1 \text{ sonst} \end{cases}$$

und

$$\mathrm{long}(x) = \mu_y[y \leqq o(x) \;\&\; \mathrm{cdr}^{(o(y))}\,(x) \notin H_0 \;\&\; \mathrm{cdr}^{(o(y)+1)}(x) = h_0]$$

ergibt für ein S-Ausdruck x, der einer Liste zugeordnet ist, die „Länge“ dieser Liste (eigentlich um 1 weniger als die Gliederanzahl einer nicht leeren Liste, da das erste Glied der Liste den Index 0 erhalten hat). Für andere x gilt

$$\mathrm{long}\,(x) = h_0.$$

Jedenfalls ist long (x) eine natürliche Zahl, so daß

$$\mathrm{long}\,(x) = o\,(\mathrm{long}\,(x))$$

besteht.

Ist die Abhängigkeit einer primitiv-rekursiven Funktion f von x derartig, daß ihre Werte für jedes x dergleichen Ordnung übereinstimmen, so wird f (das auch

von anderen Variablen abhängen kann) auch als $f_{o(x)}$ geschrieben, und eine primitiv-rekursive Folge genannt.

Nun kann nach den Vorherigen eine primitiv-rekursive — und nur Vorgänger von x enthaltende — Folge, welche für solches x, das einer Liste entspricht, bei $o(y) \leqq \text{long}(x)$ das $o(y)$-te Glied dieser Liste liefert, z. B. wie folgt definiert werden:

$$k_{o(y)}(x) = \begin{cases} \text{car}\left(\text{cdr}^{(o(y))}(x)\right), \text{ falls } \text{list}(x) = h_0 \;\&\; o(y) \leqq \text{long}(x) \\ x \text{ sonst.} \end{cases}$$

Es gilt auch der folgende (zur Ausschaltung der Wertverlaufsrekursionen entscheidende) *Satz:* nicht nur die einzelnen Funktionen

$$c_n(s_0, s_1, \ldots, s_n) \quad \text{für} \quad n = 0, 1, 2, \ldots$$

sind primitiv-rekursiv, sondern für jede primitiv-rekursive Folge $s_{o(x)}$ (die auch von Parametern abhängen kann) ist

$${}^{(s)}c(x) = c_{o(x)}(s_0, s_1, \ldots, s_{o(x)})$$

eine primitiv-rekursive Funktion. Diese bezeichnet ja den S-Ausdruck

$$\text{cons}\,(s_0, \text{cons}\,(s_1, \ldots, \text{cons}\,(s_{o(x)-1}, \text{cons}\,(s_{o(x)}, h_0))\ldots)).$$

Deren Werte können von innen nach aussagen gehend berechnet werden. Dabei erhält man den sich sukzessiv verändernden Zwischenwert g zuerst (im „0-ten Schritt") als

$$h_0,$$

dann mit diesem g als

$$\text{cons}\,(s_{o(x)}, g),$$

dann mit diesem neuen g als

$$\text{cons}\,(s_{o(x)-1}, g),$$

usw.; im $i+1$-ten Schritt mit dem im i-ten Schritt erhaltenen g als

$$\text{cons}\,(s_{o(x)-i}, g),$$

für $i=0, 1, 2, \ldots, o(x)$.

Daher kann der Zwischenwert im $o(y)$-ten Schritt wie folgt als eine primitiv-rekursive Funktion definiert werden:

$$\begin{cases} g_{o(a)}(x) = h_0, \text{ falls } a \in H_0 \\ g_{o(\text{cons}(y_1, y_2))}(x) = \text{cons}\left(s_{o(x) \dot{-} o(\text{cons}^{-1}(y_1, y_2))}, g_{o(\text{cons}^{-1}(y_1, y_2))}(x)\right), \end{cases}$$

und damit auch

$${}^{(s)}c(x) = g_{\text{cons}\,(o(x),\, o(x))}(x)\,(= g_{o(x)+1}(x)).$$

Der Index

$$o(x) \dot{-} o\left(\text{cons}^{-1}(y_1, y_2)\right)$$

in der Definition von $g_{o(y)}(x)$ ist exakt folgenderweise zu verstehen: Die zahlentheoretische Funktion $m \dot{-} n$ bedeutet die Differenz $m-n$, falls $m \geqq n$ ist, und 0 sonst (wo der letztere Fall hier ohne Belang ist). Nach *Satz in* [2] gibt es dazu eine in H primitiv-rekursive Funktion $f(x, y)$, so daß

$$o\left(f(x, y)\right) = o(x) \dot{-} o(y)$$

gilt. So kann der betreffende Index von s durch

$$o(f(x, \mathrm{cons}^{-1}(y_1, y_2)))$$

vertreten werden. (In ähnlichem Sinn werden auch in den weiteren zahlentheoretische Funktionen verwendet.)

12. Die allgemeine Form des in unserem Spezialfall eben bewiesenen *Satzes* wurde in [2] zur Zurückführung der „Wertverlaufsrekursion" auf primitive Rekursion benutzt. Da handelt es sich um ein Rekursionschema, wobei zur Definition des Funktionswertes an einer Stelle x Funktionswerte an beliebigen echten, nicht nur an unmittelbaren Vorgängern von x verwendet werden.

Hierzu wird eine bestimmte Liste aus sämtlichen Vorgängern je eines Elementen x in Betracht genommen; in unserem Spezialfall die in Nr. **8** angegebene Folge

$$(*) \qquad \bar{x}_0, \bar{x}_1, \ldots, \bar{x}_b = x.$$

Das der aus den Gliedern

$$f(\bar{x}_0, u_1, \ldots, u_n),\ f(\bar{x}_1, u_1, \ldots, u_n),\ \ldots,\ f(\bar{x}_b, u_1, \ldots, u_n)$$

bestehenden Liste zugeordnete

$$f^*(x, u_1, \ldots, u_n) = c_b(f(\bar{x}_1, u_1, \ldots, u_n), \ldots, f(\bar{x}_b, u_1, \ldots, u_n))$$

kann die „Wertverlaufsfunktion" der Funktion $f(x, u_1, \ldots, u_n)$ genannt werden; daraus ergibt sich für jedes $i \leqq b$

$$f(\bar{x}_i, u_1, \ldots, u_n) = k_i(f^*(x, u_1, \ldots, u_n)),$$

und das Schema der Wertverlaufsrekursion lautet hier:

$$\begin{cases} f(a, u_1, \ldots, u_n) = g_a(u_1, \ldots, u_n), \text{ falls } a \in H_0 \\ f(\mathrm{cons}(x_1, x_2), u_1, \ldots, u_n) = g(x_1, x_2, u_1, \ldots, u_n, f^*(x_1, u_1, \ldots, u_n), f^*(x_2, u_1, \ldots, u_n)), \end{cases}$$

wobei die Funktionen g_a, g primitiv-rekursiv sind.

Wird bewiesen, daß

$$f^*(x, u_1, \ldots, u_n) \quad \text{und} \quad b = b(x)(=o(b(x)))$$

primitiv-rekursiv sind, so ist auch

$$f(x, u_1, \ldots, u_n) = k_{b(x)}(f^*(x, u_1, \ldots, u_n))$$

primitiv-rekursiv.

Überblicke man noch einmal die folgenden: Für ein $x = \mathrm{cons}(x_1, x_2)$ folgen in $(*)$ erst die Vorgänger von x_2 nacheinander; die letzte unter diesen erhält den Index $b(x_2)$. Danach folgen die Vorgänger von x_1, der Reihe nach mit Indizes

$$b(x_2)+1, \ldots, b(x_2)+b(x_1)+1.$$

Dann folgt noch x, das den Index

$$b(x) = b(x_2)+b(x_1)+2$$

erhält. So kann $b(x)$ (mit Anwendung des *Satzes in* [2]) durch folgende primitive Rekursion definiert werden:

$$\begin{cases} b(a) = 0, \text{ falls } a \in H_0 \\ b(\text{cons}(x_1, x_2)) = b(x_2) + b(x_1) + 2. \end{cases}$$

Jeder echte Vorgänger eines $x = \text{cons}(x_1, x_2)$ ist Vorgänger entweder von x_1 oder von x_2. Bezeichne für solches x und für $j < b(x)$ (andere Fälle sind belanglos) $i(x, j)$ den Wert 2 oder 1, je nachdem $\bar{x}_j \leqq x_2$ oder $\bar{x}_j \leqq x_1$ gilt; und $l(x, j)$ einen solchen Index, mit dem $\bar{x}_j$ in der zu $x_{i(x,j)}$ gehörigen Liste $(*)$ vorkommt. In [2] wurde mit allgemeiner Fassung der hier verwendeten Begriffe bewiesen, daß falls der Entsprechende des *Satzes* in Nr. **11** besteht, die Primitiv-Rekursivität der Entsprechenden von $i(x, j)$ und $l(x, j)$ die Primitiv-Rekursivität der entsprechenden Wertverlaufsrekursion nach sich zieht (deshalb sind diese notwendigerfalls zu den Ausgangfunktionen hinzuzunehmen).

In unserem Spezialfall können aber diese wie folgt als primitiv-rekursive Funktionen definiert werden:

$$i(x, j) = \begin{cases} 2, \text{ falls } x = \text{cons}(x_1, x_2) \,\&\, o(j) \leqq b(x_2) \\ 1 \text{ sonst (also auch für den uns allein} \\ \quad \text{interessierenden Fall } b(x_2) \prec o(j) \prec b(x) \end{cases}$$

und

$$l(x, j) = \begin{cases} o(j) \dot{-} (b(x_2) + 1), \text{ falls } x = \text{cons}(x_1, x_2) \,\&\, i(x, j) = 1 \\ o(j) \text{ sonts (also auch für } i(x, j) = 2). \end{cases}$$

So führt die Wertverlaufsrekursion nicht von der Klasse der in H primitiv-rekursiven Funktionen hinaus.

13. In Nr. **7** wurde erwähnt, daß eine derartige Rekursion, wie dort für equal angegeben war:

$$\text{(D)} \qquad \begin{cases} \text{equal}(a, y) = \text{eq}(a, y), \text{ falls } a \in H_0 \\ \text{equal}(\text{cons}(x_1, x_2), y) = g\big(\text{equal}(x_1, \text{car}(y)), \text{equal}(x_2, \text{cdr}(y))\big), \end{cases}$$

wobei für den Parameter y Einsetzungen erfolgten, nach eventueller Aufnahme neuer Ausgangsfunktionen auf Wertverlaufsrekursionen zurückgeführt werden kann.

Obwohl zu den hinzuzunehmenden Ausgangsfunktionen auch equal (x, y) selber gehört, möchte ich den Gedankengang des Beweises an diesem einfachen Beispiel andeuten.

Das erste Ziel ist eine Funktion $f(x, y)$ mit folgenden Eigenschaften zu definieren:

(1) Zu jedem y gibt es ein x mit

$$f(x, y) = y.$$

(2) Zu jedem $a \in H_0$, y und z gibt es ein x mit

$$f(x, y) = \text{eq}\,(a, f(z, y)).$$

(3) Zu jedem y und z gibt es ein x mit

$$f(x, y) = \text{car}\,(f(z, y)).$$

(4) Zu jedem y und z gibt es ein x mit

$$f(x, y) = \mathrm{cdr}\,(f(z, y)).$$

(5) Zu jedem y_1, z_1 und z_2 gibt es ein x mit

$$f(x, y) = g(f(z_1, y), f(z_2, y)).$$

Ein derartiges $f(x, y)$ nimmt sozusagen sämtliche „Bausteine" an, die in der Bestimmung je eines durch (D) definierten Wertes von equal (x, y) teilnehmen.

Nun kann ein solches $f(x, y)$ z. B. mit Verwendung der in Nr. **11** definierten primitiv-rekursiven Funktionen

$$k_i(u),\ c_n(u_0, \ldots, u_n) \quad \text{für} \quad i = 0, 1, 2;\ n = 1, 2,$$

wobei für $i \leqq n$

$$k_i(c_n(u_0, \ldots, u_n)) = u_i$$

gilt, wie folgt definiert werden:

$$\text{(D*)} \qquad \begin{cases} f(a, y) = y, \text{ falls } a \in H_0 \\ f(\mathrm{cons}(x_1, x_2), y) = \begin{cases} \mathrm{eq}\big(k_0(x_1), f(k_1(x_1), y)\big), \text{ falls } o(k_0(x_1)) = h_0 \\ \mathrm{car}\big(f(k_1(x_1), y)\big), \text{ falls } o(k_0(x_1)) = h_1 \\ \mathrm{cdr}\big(f(k_1(x_1), y)\big), \text{ falls } o(k_0(x_1)) = h_2 \\ g\big(f(k_1(x_1), y), f(k_2(x_1), y)\big) \text{ sonst.} \end{cases} \end{cases}$$

Tatsächlich werden so die Forderungen (1)—(5) der Reihe nach z. B. mit

$$\begin{aligned} x &= h_0, \\ x &= \mathrm{cons}\,(c_1(a, z), x_2), \quad \text{falls} \quad a \in H_0, \\ x &= \mathrm{cons}\,(c_1(h_1, z), x_2), \\ x &= \mathrm{cons}\,(c_1(h_2, z), x_2), \\ x &= \mathrm{cons}\,(c_2(h_3, z_1, z_2), x_2) \end{aligned}$$

bei beliebigem x_2, z. B. $x_2 = h_0$ erfüllt.

Definitionen der Art (D*) können zu Wertverlaufsrekursionen umformuliert werden.

Nach einem Hilfsatz in [2] kann nun durch eine ähnliche Definition eine Funktion $w(x, z)$ der Eigenschaft

$$f(x, f(z, y)) = f(w(x, z), y)$$

angegeben werden. (Diese hat besonders dann eine entscheidende Rolle, wenn es sich statt einer so einfachen Definition wie (D) um eine sog. „eingeschachtelte Rekursion" handelt, wobei für Parameter von (an Vorgänger-Stellen angenommenen) Werten der zu definierenden Funktion abhängige Ausdrücke eingesetzt werden; der genannte Hilfsatz ermöglicht die Auflösung der „Einschachtelungen".)

Mit Benutzung dieser Funktion $w(x, z)$ kann endlich durch eine primitive Rekursion eine Funktion $q(x)$ definiert werden, welche aus den Werten vom durch (D*)

definierten $f(x, y)$ die Werte von equal (x, y) sozusagen „herausschöpft", d.h., für welche bei jedem x, y

$$\text{equal}\,(x, y) = f(q(x), y)$$

gilt.

Nach der Definition von $k_{o(y)}(x)$ sind die in (D*) auftretenden

$$k_i(x_1) \qquad (i=0, 1, 2)$$

Vorgänger von x_1. Ist $h_i(x)$ die $v(x, i)$-te in der Liste $(*)$ der Vorgänger von x, und wird die Wertverlaufsfunktion von $f(x, y)$ mit $f^*(x, y)$ bezeichnet, so gilt nach Nr. **12**

$$f(k_i(x_1), y) = k_{v(x_1, i)}(f^*(x_1, y)).$$

Werden für $i=0, 1, 2$ die rechten Seiten dieser Identitäten für ihre linken Seiten in (D*) eingesetzt, so sieht man, daß falls $v(x, i)$ primitiv-rekursiv ist, (D*) zu einer Wertverlaufsrekursion umformuliert werden kann. Deshalb wurde in [2] allgemein gefordert, das Entsprechende von $v(x, i)$ notwendigerfalls zu den Ausgangsfunktionen hinzuzunehmen.

In unserem Spezialfall kann aber $v(x, i)$ (von natürlichen Zahlen auf sämtliche S-Ausdrücke für i in belangloser Weise erweitert) als eine primitiv-rekursive Funktion definiert werden.

Die Liste $(*)$ der Vorgänger von x besteht für $x \in H_0$ aus dem einzigen Glied $\bar{x}_0 = x$, und für ein $x = \text{cons}\,(x_1, x_2)$ aus den Gliedern

$$\bar{x}_0, \ldots, \bar{x}_{b(x_2)} = x_2, \bar{x}_{b(x_2)+1}, \ldots, \bar{x}_{b(x_2)+b(x_1)+1} = x_1, \bar{x}_{b(x_2)+b(x_1)+2} = x,$$

wo die ersten $b(x_2)+1$ Glieder die Liste der Vorgänger von x_2, und die darauf folgenden $b(x_1)+1$ Glieder die Liste der Vorgänger von x_1 bilden.

So gilt erstens immer

$$v(x, i) = 0, \quad \text{falls} \quad x \in H_0.$$

Sei nun

$$x = \text{cons}\,(x_1, x_2).$$

Wenn x zu keiner Liste gehört, oder wenn $i \leqq \text{long}(x)$ nicht besteht, so gilt

$$k_i(x) = x,$$

also

$$v(x, i) = b(x_2)+b(x_1)+2.$$

Sonst ist x der Form:

$$x = \text{cons}\Big(\underbrace{s_0}_{x_1}, \underbrace{\text{cons}\big(s_1, \ldots, \text{cons}\,(s_{\text{long}(x)}, h_0)\ldots\big)}_{x_2}\Big),$$

daher gilt

$$k_0(x) = s_0 = x_1 = \bar{x}_{b(x_2)+b(x_1)+1},$$

also

$$v(x, 0) = b(x_2)+b(x_1)+1;$$

und für $0 < i \leqq \text{long}\,(x)$

$$k_i(x) = k_{i-1}(x_2),$$

also

$$(D_0) \qquad v(x, i) = v(x_2, i-1).$$

Dies bietet für $i>1$ eine Rekursion, wobei für den Parameter i die Einsetzung $i-1$ erfolgt. Doch so einfache Fälle lassen sich leicht auflösen: Für $i>1$ kann (D_0) wiederholt werden. Da

$$x_2 = \mathrm{cdr}\,(x) \qquad (\text{und } x_1 = \mathrm{car}\,(x))$$

ist, ergibt sich so

$$v(x, i) = v\big(\mathrm{cdr}(x), i-1\big) = v\big(\mathrm{cdr}^2(x), i-2\big) = \ldots = v\big(\mathrm{cdr}^{(i)}(x), 0\big) =$$

$$= b\big(\mathrm{cdr}^{(i+1)}(x) + b\big(\mathrm{car}\,(\mathrm{cdr}^{(i)}(x))\big) + 1.$$

Daher ergibt sich $v(x, i)$ durch die Fallunterscheidung

$$v(x, i) = \begin{cases} h_0, \text{ falls } x \in H_0 \\ b\big(\mathrm{cdr}^{(o(i)+1)}(x)\big) + b\big(\mathrm{car}\,(\mathrm{cdr}^{(o\,(i))}(x))\big) + 1, \\ \quad \text{falls } x \notin H_0 \;\&\; \mathrm{list}\,(x) = h_0 \;\&\; o(i) \leqq \mathrm{long}(x) \\ b\big(\mathrm{cdr}(x)\big) + b\big(\mathrm{car}(x)\big) + 2 \text{ sonst} \end{cases}$$

als eine primitiv-rekursive Funktion.

Nach dem angedeuteten Gedankengang führen daher die Rekursionen, wobei für die Parameter Einsetzungen erfolgen (sogar die derart zustandekommenden „eingeschachachtelten Rekursionen") nicht von der Klasse der primitiv-rekursiven Funktionen von H hinaus.

14. Als eine einfache Verwendung der Funktionen $c_{o\,(x)}$ und $k_{o\,(y)}(x)$ erwähne ich noch die in [2] behandelte Zurückführung der „simultan-rekursiven" Definition mehrerer Funktionen

$$f_0, f_1, \ldots, f_n$$

auf die rekursive Definition der einzigen Funktion

$$f = c_n(f_0, f_1, \ldots, f_n)$$

woraus die einzelnen Funktionen durch die folgenden Substitutionen zurückerhalten werden können:

$$f_0 = k_0(f),\ f_1 = k_1(f),\ \ldots, f_n = k_n(f).$$

Z. B. ergibt

$$\begin{cases} f_0(a) = g_a, \text{ falls } a \in H_0 \\ f_0\big(\mathrm{cons}(x_1, x_2)\big) = g\big(x_1, x_2, f_0(x_1), f_0(x_2), f_1(x_1), f_1(x_2)\big) \end{cases}$$

und

$$\begin{cases} f_1(a) = g'_a, \text{ falls } a \in H_0 \\ f_1\big(\mathrm{cons}(x_1, x_2)\big) = g'\big(x_1, x_2, f_0(x_1), f_0(x_2), f_1(x_1), f_1(x_2)\big) \end{cases}$$

eine simultan-rekursive Definition der Funktionen $f_0(x), f_1(x)$.

(Des würde auf dasselbe hinausgehen, wenn statt der letzten Definitionsgleichung

$$f_1\,(\mathrm{cons}\,(x_1, x_2)) = g''\big(x_1, x_2, f_0(x_1), f_0(x_2), f_0\big(\mathrm{cons}\,(x_1, x_2)\big), f_1(x_1), f_1(x_2)\big)$$

stehen würde; man hätte dann nur g' als

$$g'(x_1, x_2, u_1, u_2, u_3, u_4) = g''(x_1, x_2, u_1, u_2, g(x_1, x_2, u_1, u_2, u_3, u_4), u_3, u_4)$$

zu definieren.)

In diesem Fall kann

$$f(x) = c_1(f_0(x), f_1(x))$$

durch folgende primitive Rekursion definiert werden:

$$\begin{cases} f(a) = c_1(g_a, g'_a), \text{ falls } a \in H_0 \\ f(\text{cons}(x_1, x_2)) = c_1\Big(g\big(x_1, x_2, k_0(f(x_1)), k_0(f(x_2))\big), g'\big(x_1, x_2, k_1(f(x_1)), k_1(f(x_2))\big)\Big), \end{cases}$$

und daraus erhält man durch Substitutionen:

$$f_0(x) = k_0(f(x)) \quad \text{und} \quad f_1(x) = k_1(f(x)).$$

15. In [2] wurde auch gezeigt, daß die allgemein-rekursiven Funktionen (im allgemeinen H) ähnlich wie im zahlentheoretischen Fall definiert werden können, als Funktionen, deren Werte aus Definitionsgleichungsystemen — durch endlichmal verwendeter Einsetzung von Elementtermen (in unserem Spezialfall von S-Ausdrücken) für Variablen, und durch Ersetzung einer Gleichungseite die als ein Teilausdruck auftritt, durch die andere Seite der betreffenden Gleichung — überall eindeutig erhalten werden können; mit Weglassung der Forderung „überall" entstehen derart die partiell-rekursiven Funktionen in H. (All diese können auch durch primitive Rekursionen und „unbeschränkte μ-Operationen" definiert werden.)

16. Einige Beispiele für Definitionen von im „Lisp 1.5" verwendeten primitivrekursiven Funktionen in unserem H (diese sind in etwas abweichender Form auch in [1] zu finden):

① Der Ausdruck, der aus x entsteht, wenn für jedes Vorkommen von y darin z gesetzt wird (in [1] wurde das nur im Fall $y \in H_0$ definiert, doch ohne Belang kann die Definition für beliebige y erweitert werden; ähnliches gilt auch in den folgenden Definitionen):

$$\begin{cases} \text{subst}(a, y, z) = \begin{cases} z, \text{ falls } a \in H_0 \,\&\, a = y \\ a, \text{ falls } a \in H_0 \,\&\, a \neq y \end{cases} \\ \text{subst}(\text{cons}(x_1, x_2), y, z) = \text{cons}(\text{subst}(x_1, y, z), \text{subst}(x_2, y, z)). \end{cases}$$

② Der Ausdruck, der jener Liste entspricht, die aus einer Liste durch Danachfügung der Reihe nach der Glieder einer anderen Liste entsteht (der Fall

$$\text{list}(x) = h_1 \vee \text{list}(y) = h_1$$

ist für uns ohne Belang; und für $a \in H_0$ interessiert uns nur $a = h_0$ als das Entsprechende der leeren Liste):

$$\begin{cases} \text{append}(a, y) = y, \text{ falls } a \in H_0 \\ \text{append}(\text{cons}(x_1, x_2), y) = \text{cons}(x_1, \text{append}(x_2, y)). \end{cases}$$

③ Die charakteristische Funktion der Relation „unter den Gliedern einer gewissen Liste vorkommen" (für list $(x)=h_1$ ohne Belang):

$$\begin{cases}\text{member}(a, y) = h_1, \text{ falls } a \in H_0 \\ \text{member}\big(\text{cons}(x_1, x_2), y\big) = \begin{cases} h_0, \text{ falls } x_1 = y \\ \text{member}(x_2, y) \text{sonst.}\end{cases}\end{cases}$$

④ Der Ausdruck, der jener Liste entspricht, die durch linkseitiger Hinzufügung zu einer Liste der Glieder einer „Paarliste" entsteht, die aus den cons-Funktionen („Paare" genannt) der ersten, zweiten, ... Glieder zweier Listen derselben Länge entsteht:

$$\begin{cases}\text{pairlis}(a, y, u) = u \text{ falls } a \in H_0 \\ \text{pairlis}\big(\text{cons}(x_1, x_2), y, u\big) = \text{cons}\big(\text{cons}(x_1, \text{car}(y)), \text{pairlis}(x_2, \text{cdr}(y), u)\big).\end{cases}$$

Wegen dem Vorkommen von

$$\text{pairlis}\,(x_2, \text{cdr}\,(y), u)$$

auf der rechten Seite haben wir hier bereits mit einer solchen Rekursion zu tun, wobei für den Parameter y eine Einsetzung erfolgt ist; worüber es sich in Nr. **12—13** handelte.

Noch zwei Definitionen bezüglich Paarlisten:

⑤ Das erste Paar einer Paarliste, dessen erstes Glied (d. h. car-Funktion) x ist (und etwa h_0, falls kein solches Paar vorhanden ist):

$$\begin{cases}\text{assoc}(a, x) = h_0, \text{ falls } a \in H_0 \\ \text{assoc}\big(\text{cons}(u_1, u_2), x\big) = \begin{cases} u_1, \text{ falls car}(u_1) = x \\ \text{assoc}(u_2, x) \text{ sonst.}\end{cases}\end{cases}$$

⑥ Gehört u zu einer Paarliste, worin die ersten Glieder Paare Atome sind, so bedeutet

$$\text{sublis}\,(x, u)$$

den Ausdruck, der aus dem Ausdruck x entsteht, falls jedes Vorkommen der ersten Glieder der Paare in x durch das zweite Glied (d. h. cdr-Funktion) des betreffenden Paars ersetzt wird.

Erst kann dies mit der Bezeichnung „suba" für $x \in H_0$ durch Rekursion nach u definiert werden:

$$\begin{cases}\text{suba}(a, x) = x, \text{ falls } a \in H_0 \\ \text{suba}\big(\text{cons}(u_1, u_2), x\big) = \begin{cases} \text{cdr}(u_1), \text{ falls car}(u_1) = x \\ \text{suba}(u_2, x) \text{ sonst.}\end{cases}\end{cases}$$

Damit ergibt sich die primitive Rekursion nach x:

$$\begin{cases}\text{sublis}(a, u) = \text{suba}(u, a), \text{ falls } a \in H_0 \\ \text{sublis}\big(\text{cons}(x_1, x_2), u\big) = \text{cons}\big(\text{sublis}(x_1, u), \text{sublis}(x_2, u)\big).\end{cases}$$

Рекурсивность программного языка "Lisp 1,5" в специальных случаях упорядоченных свободных голоморфных множеств

Понятие, имеющееся в заглавии (понятие «множеств, которые могут быть построены по методу натуральных чисел») автор ввёл на варшавском симпозиуме в 1959 году; и для таких множеств он обобщил теорию рекурсивных функций; разрабатывая как частный случай, теорию рекурсивных функций во множествах, находящихся выше любого алфавита. И другие авторы исследовали и применяли в основном этот специальный случай. Легко показать, что основные понятия программного языка „Lisp 1.5" являются примитивно-рекурсивными во множестве, находящемся выше подходящегося алфавита. В то же время множество симболических выражений языка „Lisp 1.5", построенное исходя из некоторых «атомов» с помощью единственного (двухпеременного) действия, само собой даётся как специальный случай упорядоченных свободных голоморфных множеств, не совпадающих с множествами слов. В статье даётся рекурсивная теория для этого случая.

EÖTVÖS LORÁND UNIVERSITÄT
BUDAPEST, UNGARN

(Eingegangen am 1. Februar 1973)