Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg

Ausarbeiten eines Vortrages über eine Arbeit von

Ernie G. Manes:

Implementierung von Behälter-Klassen

mit Monaden

Seminararbeit im SS 2005 von Deng ,Yuchen

Betreuer : Prof. Dr. H. Peter Gumm

Juni 2005

1.Einführung

Behälter sind abstrakte Datentypen. Verschiedene Behälter organisieren gleichartige Elemente mit unterschiedlichen Strukturen, und besitzen unterschiedliche Eigenschaften. Listen, Mengen, Multimengen und Bäume werden oft als Behälter in der Programmierung angewendet. Listen organisieren Elemente in einer Reihenfolge, wobei die Positionen verschiedener Elementen unvertauschbar sind. Im Gegensatz dazu spielen die Positionen der Elementen bei Mengen keine Rolle. Keines Element darf aber in einer Menge vielmals vorkommen. Multimengen sind Mengen ohne die obige Einschränkung. Bäume organisieren Elemente in einer Baumstruktur, und werden induktiv definiert. Ein Blatt ist ein Baum. Falls mehrere Bäume mit einem Wurzel verknüpft sind, entsteht ein neuer Baum. Die Frage ist nun, was Behälter sein sollen. Mit anderen Worten was ist eine axiomatische Definition für Behälter.

2. Intuitive Axiome

Ein Behälter für Elemente einer Menge X soll die folgende intuitive Axiome erfüllen.

2.1 Jeder X-Behälter organisiert eine endliche Menge von Elementen aus X.

2.2 Jedes Element aus X kann in einem einelementigen Behälter eingekapselt werden.

2.3 Ein Behälter von Behältern kann in einem einfachen Behälter ausgeschüttet werden.

Die 3 Axiome sind nur intuitiv zusammengefasst, und werden später in Monaden formalisiert.

3. ∑-Bäume

∑-Bäume mit Gleichungen werden später zur Implementierung von Behälter-Klassen verwendet .

Definition (∑-Bäume) : Sei X eine Menge Für $i \in N$ sei Σ_i eine Menge von i-stelligen Funktionssymbolen

$\mathbf{T_\Sigma X}$, die Menge aller ∑-Bäume mit Blättern aus X, ist wie folgt definiert:

1. $X \subset T_\Sigma X$
2. Gegeben $\tau \in \Sigma_i$, $\tau_1, \ldots, \tau_i \in T_\Sigma X$ Dann ist $\tau(\tau_1, \ldots, \tau_i) \in T_\Sigma X$

Beispiel :

Sei eine Menge X { x_1 , x_2 , x_3 }, Funktionssymbole $\Sigma_2 = \{*\}$, $\Sigma_1 = \{-1\}$, $\Sigma_0 = \{e\}$ gegeben.

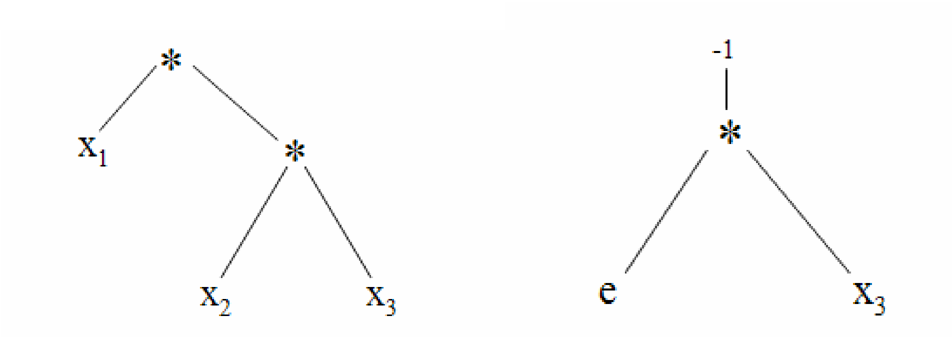


Abbildung 1

Die obigen zwei Bäume sind offensichtlich ∑-Bäume bzgl. der obigen Signatur.

∑-Bäume erfüllen die drei intuitiven Axiomen (2.1, 2.2, 2.3). Ein ∑-Baum enthält nur endlich viele Elemente, weil die Stelligkeit eines Funktionssymbols und die Anzahl der Schichten eines gegebenem ∑-Baum endlich sind. Laut Definition ist jedes einzelne Element $x \in X$ ein ∑-Baum, damit ist das intuitive Axiom (2.2) erfüllt. Wenn i ∑-Bäume durch ein i-stelliges Funktionssymbol verknüpft werden, entsteht ein neuer ∑-Baum. Der neu entstehende Baum ist ein ∑-Baum mit i ∑-Bäumen als Blättern. Solche ∑-Bäume können sich weiter auspacken lassen, so dass ihre innere Struktur sich zeigen kann. Der ∑-Baum ist nun ein ∑-Baum, dessen Blätter die Vereinigung der Blättern der i ∑-Bäumen sind.

4. Monaden

Eine Monade ist ein 3-Tupel (T, η, μ) ,wobei T ein Funktor ist, und η, μ natürliche Transformationen sind. Außerdem müssen η, μ die Monaden-Gleichungen erfüllen. Das bedeutet ,dass μ assoziativ ist, und η unit ist.

4.1 Funktor

Definition (Kategorie) : Eine **Kategorie C** besteht aus:

- ✧ einer Klasse Ob(C) von Objekten.
- ✧ je einer Menge $Mor_C(X,Y)$ zu jedem Paar (X,Y) von Objekten. Zu einem Morphismus *f* in Mor(X, Y) kann man auch *f*: X → Y schreiben.
- ✧ Verknüpfungsabbildungen $\circ$: $Mor_C(X,Y) \times Mor_C(Y, Z) \rightarrow Mor_C(X, Z)$
- ✧ einem Identitätsmorphismus id_X: X → X zu jedem Objekt X.

In der Kategorie der **Mengen** ist Ob(C) die Klasse aller Mengen. Zu zwei Mengen X, Y ist $Mor_{\mathbf{Set}}(X, Y)$ die Menge der Abbildungen von X nach Y. Der Identitätsmorphismus eines Objektes X ist die identische Abbildung X → X.

Definition (Funktor) : Ein **Funktor** ist eine Abbildung zwischen Kategorien. Ein **Funktor** *T* von einer Kategorie *C* in eine Kategorie *D* besteht aus:

- ✧ einer Zuordnung *T*: Ob(*C*) → Ob(*D*) ,
- ✧ Abbildungen *T:* $Mor_C(X, Y) \rightarrow Mor_D(T(X), F(Y))$ für je zwei Objekte X, Y von C.

Darüber hinaus müssen die Abbildungen zwischen den Morphismenmengen folgende Eigenschaften haben:

Seien Morphismen $f : X \rightarrow Y$ und $g : Y \rightarrow Z$,

- ✧ $T(g \circ f) = Tg \circ Tf$
- ✧ $T_{\mathrm{idX}} = \mathrm{id}_{TX}$.

Ein Behälter soll ein Funktor sein, damit jede Abbildung *f* : X → Y zu einer Abbildung *Tf*: Behälter <X> → Behälter <Y> fortgesetzt werden kann. Listen sind beispielsweise Funktor. Zu einer Menge X existiert immer die Menge aller Listen über dieser Menge X. Zu einer Abbildung $f : X \rightarrow Y$ erhält man auch eine Abbildung

Tf : Listen <X> → Listen <Y>, die jedes Element x in einer Liste zu *f*(x) abbildet, ohne die Positionen der Elementen zu ändern. In funktionalen Programmiersprachen wird diese Abbildung als *map f* bezeichnet. ∑-Bäume bilden auch einen Funktor.

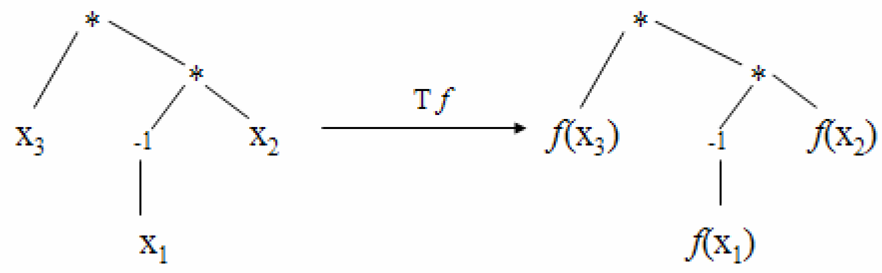


Abbildung 2

Für einen ∑-Baum t ersetzt T*f* alle Blätter x aus t durch f(x). Die Baumstruktur bleibt unverändert.

4.2 η und μ

Durch die Formalisierung der intuitiven Axiome (2.2, 2.3) erhält man direkt zwei Abbildungen η_X : X → Behälter < X > und μ_X : Behälter < Behälter < X > > → Behälter < X > . Die Abbildung η_X kapselt jedes Element aus der Menge X in einem einelementigen Behälter ein. Die Abbildung μ_X schüttet einen Behälter von Behältern in einen einfachen Behälter aus. Weiterhin sollen die beide Abbildungen natürliche Transformationen sein, und die Monaden-Gleichungen erfüllen.

Definition (natürliche Transformationen) : Unter **natürliche Transformationen** versteht man Abbildungen zwischen Funktoren. Sind *F* und *G* Funktoren von einer Kategorie *C* nach einer Kategorie *D*, so besteht eine natürliche Transformation *t* von *F* nach *G* aus Morphismen t_X: $F(X) \to G(X)$ für jedes Objekt *X* von *C*, so dass für jeden Morphismus $f: X \to Y$ zwischen Objekten von *C* das folgende Diagramm kommutativ ist.

$$\begin{tikzcd} F(X) \arrow[r, "t_X"] \arrow[d, "Ff"'] \arrow[dr, phantom, "\circ"] & G(X) \arrow[d, "Gf"] \\ F(Y) \arrow[r, "t_Y"'] & G(Y) \end{tikzcd}$$

Das bedeutet, dass $t_Y \circ F(f) = G(f) \circ t_X$ gilt.

4.2.1 η soll eine natürliche Transformation vom Identitätsfunktor nach T sein.

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[d, "f"'] \arrow[dr, phantom, "\circ"] & TX \arrow[d, "Tf"] \\ Y \arrow[r, "\eta_Y"] & TY \end{tikzcd}$$

Abbildung 3

Die Abbildung η_X packt ein Element x aus der Menge X in einen einelementigen Behälter $\eta_X(x)$ ein. Dann bilden die Abbildung Tf das Element $\eta_X(x)$ zu $Tf(\eta_X(x))$ ab. Auf dem anderem Weg wird das Element x zuerst durch die Abbildung f auf $f(x)$ abgebildet. Danach kapselt die Abbildung $f(x)$ in den Behälter $\eta_Y(f(x))$ ein. Es soll also $\eta_Y(f(x)) = Tf(\eta_X(x))$ sein.

4.2.2 μ soll eine natürliche Transformation von Funktor TT nach T sein.

$$\begin{tikzcd} TTX \arrow[r, "\mu_X"] \arrow[d, "TTf"'] \arrow[dr, phantom, "\circ"] & TX \arrow[d, "Tf"] \\ TTY \arrow[r, "\mu_Y"] & TY \end{tikzcd}$$

Abbildung 4

Ein Funktor TT ordnet eine Menge X einer Menge aller Behältern von Behältern über der Menge X zu, falls man ein Funktor T als ein Behälter vorstellt. Das obige

Diagramm soll kommutieren. Das heißt, dass das Ausschütten eines Behälters von Behältern in einem einfachen Behälter vor oder nach dem Abbilden jedes Elementes x in Behälter zu *f*(x) das gleiche Ergebnis haben soll. Zum Schluss soll man ein Behälter von *f*(x) erhalten.

4.2.3 μ soll assoziativ sein.

$$\begin{tikzcd}
TTTX \arrow[r, "T\mu_X"] \arrow[d, "\mu_{TX}"'] \arrow[dr, phantom, "\circ"] & TTX \arrow[d, "\mu_X"] \\
TTX \arrow[r, "\mu_X"] & TX
\end{tikzcd}$$

Abbildung 5

Die Abbildung μ_X gibt die Zuordnung zwischen *TT*X und *T*X an. Deshalb geht T μ_X von TTTX nach TTX, falls vor der Urbildmenge TTX sowie der Bildmenge TX jeweils ein T hinzugefügt werden. Bei μ_{TX} wird ein Behälter von Elementen als ein Objekt gesehen.

Für ein TTTX Struktur kann man sich so vorstellen. Man hat ein paar Beutel von Kartoffeln. Solche Beutel werden weiter in Körbe eingesteckt. Solche Körbe werden endlich im Lagerraum abgelagert. Beutel, Körbe und Lagerraum sind Behälter und entsprechen den drei Ts in TTTX. T μ_X nimmt Kartoffeln aus den Beuteln, und wirft sie direkt in Körben rein. Dann μ_X schüttet Kartoffeln von Körben in Largeraum aus. Am Ende erhält man einen Lageraum von Kartoffeln ohne Beutel und Körbe. Man kann auch auf andere Weise vorgehen. μ_{TX} schüttert zuerst Beutel von Kartoffeln von Körben raus. Danach nimmt μ_X Kartoffel aus den Beutel heraus.

4.2.4 Neutral.

$$
\begin{tikzcd}
TX \arrow[r, "T\eta_X"] \arrow[dr, "id"'] & TTX \arrow[d, "\mu_X"] & TX \arrow[l, "\eta_{TX}"'] \arrow[dl, "id"] \\
& TX &
\end{tikzcd}
$$

Abbildung 6

Die Abbildungen μ und η sollen Links bzw. Rechtsneutral folgen. Das bedeutet, dass die 2 Dreiecke im obigen Diagramm jeweils kommutativ sein sollen.

Der linke Dreieck zeigt, dass die Hintereinanderausführung von $T\eta_X$ und μ_X selben wie die identische Abbildung ist. $T\eta_X$ kapselt jedes Element in einem einelementigen Behälter ein. Anschließend schüttert μ_X solche Elemente von einelementigen Behältern wieder aus, so dass der Endzustand von Behälter sich nicht von dem Anfangzustand unterschiedet.

Im rechten Dreieck wird ein Behälter von Elementen als ein Objekt betrachtet . η_{TX} steckt ihn in einem einelementigen Behälter. Dann schüttert μ_X die im inneren Behälter liegenden Elementen aus. Man erhält schließlich einen Behälter von Elementen. Deswegen ist die Abbildung $\mu_X \circ \eta_{TX}$ gleich die identische Abbildung.

Zusammengefasst soll ein Behälter mit zwei Abbildungen μ und η eine Monade sein, damit ein Behälter geeignet mit jeder Menge X ist, und zu jeder Abbildung f : X → Y man eine Abbildung *Tf*: Behälter <X> → Behälter <Y> erhalten kann. Dass μ und η natürliche Transformationen sind, sichert, dass die μ , η Operationen und das Abbilden jedes Elementes x in Behälter zu f(x) keinen Einfluss aufeinander haben. Für einen vielschichtigen Behälter soll man frei wählen können, von welcher Schicht Anfang auszupacken, ohne das Ergebnis zu ändern. Deswegen soll μ assoziativ sein.

5. Finitäre Monade

Die Abbildungen μ und η formalisieren die intuitiven Axiomen(2.2, 2.3). Aber es bleibt noch das erste Axiom übrig, dass ein Behälter eine endliche Menge von

Elementen organisieren soll. Dafür wird der Begriff „Finitäre Monade“ eingeführt.

Definition (Finitäre Monaden) : T heißt finitär, falls für jede Menge X und jedes $\tau \in \mathit{T}X$ eine endliche Teilmenge $P \subseteq X$ existiert mit $\tau \in \mathit{T}P$.

Ist T eine finitäre Monade, kann man für jedes $\tau \in \mathit{T}X$ eine kleinste Teilmenge $P \subseteq X$ finden, so dass τ in $\mathit{T}P$ liegt. Mit anderen Worten umfasst dieses P genau die in τ liegenden Elemente aus der Menge X, und kann als mem(τ) bezeichnet werden. Die genaue Definition von mem(τ) ist wie folgt.

Definition (Mem) : Seien X eine Menge, *T* eine finitäre Monade,

Dann $\mathrm{mem}(\tau) = \cap\{ P \subseteq X \mid \tau \in TP \}$

Man muss noch zeigen, dass $\tau \in \mathit{T}(\mathrm{mem}(\tau))$ ist. Dazu kann man einfach davon ausgehen, dass $T(P \cap Q) = T(P) \cap T(Q)$ gilt. Das Problem liegt darin, dass mem(τ) ein Durchschnitt von unendlich vielen Mengen sein kann. In diesem Fall kann man die Voraussetzung, dass *T* eine finitäre Monade , anwenden. Sei $\mathrm{mem}(\tau) = \cap P_i$ mit $i \in \mathbf{I}$, existiert eine endliche Menge P_k mit $\tau \in TP_k$, weil *T* finitär ist. Die beide Menge mem(τ) und P_k sind endlich, so dass die Menge $P_k \setminus \mathrm{mem}(\tau)$ auch endlich ist. Für jedes Element x_j ($j \in \{1 ,..., n\}$, $n = \# P_k \setminus \mathrm{mem}(\tau)$) aus der Menge $P_k \setminus \mathrm{mem}(\tau)$ existiert eine Menge P_{i_j} mit $x_j \notin P_{i_j}$. Die Anzahl der Mengen P_{i_j} ist endlich, und der Durchschnitt von P_k und solcher P_{i_j} ist genau die Menge mem(τ). Deshalb gilt $\mathrm{mem}(\tau) = P_k \cap P_{i_1} \cap \ldots \cap P_{i_n}$, wobei $\tau \in \mathit{T}(P_k)$ und $\tau \in \mathit{T}(P_{i_j})$ sind. Daraus folgt, dass $\tau \in \mathit{T}(P_k) \cap \mathit{T}(P_{i_1}) \cap \ldots \cap T(P_{i_n})$ ist. Da $T(P \cap Q) = T(P) \cap T(Q)$ gilt, ist $\tau \in \mathit{T}(P_k \cap P_{i_1} \cap \ldots \cap P_{i_n})$. Also ist $\tau \in \mathit{T}(\mathrm{mem}(\tau))$.

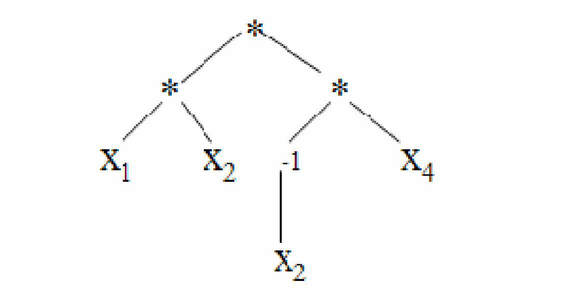


Abbildung 6

Für den in Abbildung 6 gezeigten $\sum$-Baum τ ist mem(τ) die Menge $\{x_1, x_2, x_4\}$, also alle Blätter von $\sum$-Baum τ.

Ein finitärer Behälter soll eine finitäre Monade sein, damit das erste intuitive Axiom erfüllt werden kann. Aber es liegt noch die Frage, ob man finitäre Monaden klassifizieren kann. Um diese Frage zu antworten, wird an dieser Stelle $T_{\sum,E}$ vorgestellt.

6. $T_{\sum,E}$

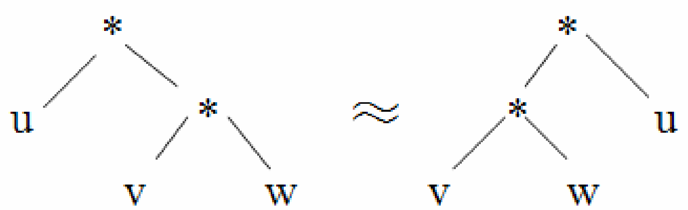
Definition ($T_{\sum,E}$) : Sei $\sum$ eine Menge von Funktionssymbolen, E eine Menge von Gleichungen.

$T_{\sum,E}\ X = T_{\sum}X\ /\ E$ die Menge der Äquivalenz-Klassen von $T_{\sum}X$ Bäumen nach Gleichheit in E.

Die Definition der Signatur $\sum$ ist wie bei $\sum$-Bäumen. $T_{\sum,E}X$ bezeichnet die Menge der Äquivalenz-Klassen von $\sum$-Bäumen nach Gleichheit in E. $T_{\sum,E}$ ist eine Monade.

Beispiel 3:

$\Sigma = \{ *, e\}$ $\quad E = \{ x*y = y*x \}$ $\quad X=\{ u, v, w \}$



Die Gleichung in E gibt die Kommutativität-Regel an. Die oben beide ∑-Bäume sind deutlich äquivalent.

Aus der Literatur [1] ist folgender Satz bekannt.

Satz : Jede finitäre Monade ist äquivalent zu $T_{\sum,E}$ für geeignete ∑,E.

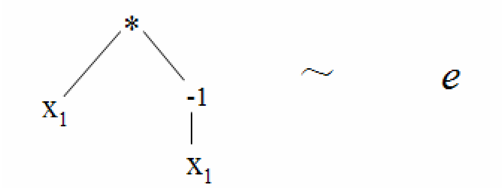
Der Satz gibt an, dass man finitäre Monade als Monaden von Typ $T_{\sum,E}$ klassifizieren kann.

7. Collection Monade

Mit finitären Monaden formalisiert man die drei intuitive Axiome. Eine finitäre Monade scheint schon ein geeigneter Behälter zu sein. Aber in der Tat nicht ist jede finitäre Monade ein geeigneter Behälter. Das folgende Beispiel zeigt, dass Elemente in einem Behälter in bestimmten Fällen verloren gehen oder entstehen können.

Beispiel 4:

$\sum = \{ *, -1, e \}$ $\quad$ $E = \{a * a^{-1} = e\}$ $\quad$ $X = \{x_1\}$



$$\mathrm{mem}(x_1 * x_1^{-1}) = \{x_1\} \quad \neq \quad \mathrm{mem}(e) = \{\,\}$$

Der linke ∑-Baum t_1 hat 2 identische Blätter, so dass $\mathrm{mem}(t_1) = \{x\}$ ist. Der rechte ∑-Baum ist einfach das Konstantsymbol e. Nach der Gleichung in E kann man den linken ∑-Baum in den rechten ∑-Baum reduzieren. mem(e) ist aber die leere Menge. Bei der Reduktion geht das Element x_1 einfach verloren. Um solche Fälle zu vermeiden muss die folgende Anforderung erfüllt werden.

Seien $*$ ein i-stellige Funktionssymbol, $t_1 \ldots t_i$ ∑-Bäume,

$$t = *(t_1, \ldots, t_i) \quad \Rightarrow \quad \mathrm{mem}(t) = \cup \{\mathrm{mem}(t_k) \mid k \in \{1 \ldots i\}\}$$

Eine Collection Monade ist eine finitäre Monade, die die obige Anforderung erfüllen.

Definition (Collection Monaden) : *T* ist eine Collection Monade, falls

1. *T* ist eine finitäre Monade,
2. für alle $\alpha : X \rightarrow TY$ und $\omega \in TX$ gilt :

   $\text{mem}\,(\alpha^{\#}(\omega)) = \cup\{\text{mem}\,(\alpha(x)) \mid x \in \text{mem}\,(\omega)\}$

In der Definition kommt die Abbildung $\alpha^{\#}$ vor. Sie ist wie folgt definiert.

Ist (T, η, μ) eine Monade, so läßt sich jede Abbildung $\alpha: X \rightarrow TY$ fortsetzen zu einer Abbildung $\alpha^{\#}: TX \rightarrow TY$ durch die Hintereinanderausführung von Tα und μ_Y (siehe Abbildung 7)

$$\begin{tikzcd} TX \arrow[r, "T\alpha"] \arrow[dr, "\alpha^{\#}"'] & TTY \arrow[d, "\mu_Y"] \\ & TY \end{tikzcd}$$

Abbildung 7

Man kann sich eine Abbildung α einfach als eine Substitution vorstellen. Für ein Term τ aus *TX* ersetzt *T*α jedes Element in τ durch einen Term aus *TY*. Dann schüttert μ_Y einen Behälter von Behältern *TTY* in einem einfachen Behälter *TY* aus.

Die zweite Bedingung in der Definition entspricht der oben erwähnten Anforderung. Für einen ∑-Baum t mit $t = *\,(t_1, \ldots, t_i)$ kann man ein Term $\omega = *\,(x_1, \ldots x_i)$ aus $T_{\sum}\{x_1, \ldots, x_i\}$ wählen und $\alpha : X \rightarrow TY$ so setzen, dass x_i einem ∑-Baum t_i zugeordnet wird. Dadurch ist die zweite Bedingung gleich wie die Anforderung.

$\text{mem}(t) = \text{mem}(*\,(t_1, \ldots, t_i)) = \text{mem}\,(\alpha^{\#}(*\,(x_1, \ldots x_i))$ falls jedes x_i entsprechendem t_i zugeordnet werden.

Sei $\text{mem}\,(\alpha^{\#}(*\,(x_1, \ldots x_i)) = \cup\{\text{mem}\,(\alpha(x)) \mid x \in \{x_1, \ldots x_i\}\}$

Dann ist $\text{mem}(t) = \cup\{\text{mem}\,(\alpha(x)) \mid x \in \{x_1, \ldots x_i\}\} = \cup\{\text{mem}(t_k) \mid k \in \{1 \ldots i\}\}$.

Es ist leicht zu ersehen, dass die im Beispiel definierte Monade $T_{\sum,E}$ keine Collection

Monade ist. Seien ein Term $\omega = ab \in T_{\sum,E}$ {a, b}, und $\alpha(a) = x_1$, $\alpha(b) = x_1^{-1}$, dann $\cup$ {mem (α(x)) | x $\in$ mem (ω)} = {x_1}, aber mem ($\alpha^{\#}(\omega)$) = $\varnothing$. Die zweite Bedingung für Collection Monaden ist nicht erfüllt.

Man kann jetzt die in Einführung gestellte Frage antworten. Ein Behälter soll eine Collection Monade sein.

8.Balancierte Gleichung:

Dadurch, dass das zusätzliche Axiom sich in der Definition für finitäre Monaden hinzufügen lässt, erhält man die Definition für Collection Monaden. Man kann auch eine weitere Bedingung für $T_{\sum,E}$ einsetzen, damit $T_{\sum,E}$ die obige Anforderung folgen kann. Diese Bedingung ist, dass die in einer Menge E angegebenen Gleichungen balanciert sein müssen. Die Definition für balancierte Gleichungen ist so:

1. Eine Gleichung ist balanciert, wenn die gleichen Variablen auf beiden Seiten der Gleichung auftreten.
2. Die Präsentation ($\sum$, E) ist balanciert, falls alle Gleichungen in E balanciert sind.

Beispiel:

ab = ba (balanciert) (ab) c = a(bc) (balanciert) $aa^{-1}=e$ (nicht balanciert)

Satz : Eine finitäre Monade (T, η, μ) ist eine Collection Monade, genau dann wenn sie isomorph zu $T_{\sum,E}$ mit balanciertem Gleichungssystem E ist.

Der Satz gibt an, dass jede Collection Monade eine isomorphe Darstellung im Form $T_{\sum,E}$ mit balanciertem Gleichungssystem E hat. Deshalb kann man viele uns bekannte Behälter durch geeignete $T_{\sum,E}$ implementieren.

Beispiel 5:

Listen: $\{ (a * b) * c = a * (b * a) \}$

Multimenge: $\{ a * b = b * a,$

$(a * b) * c = a * (b * a)\}$

Menge: $\{ a * b = b * a,$

$(a * b) * c = a * (b * a),$

$a * a = a \}$

Wiederholungsfreie Listen: $\{(a * b) * c = a * (b * a),$

$ab_1 \ldots b_n a = ab_1 \ldots b_n \}$

9 Fazit:

Ein Behälter soll die intuitive Axiome(2.1, 2.2, 2.3) erfüllen. Weiterhin soll ein Behälter ein Funktor sein. Durch die Formalisierung der intuitiven Axiomen (1.2, 1.3) erhält man zwei Abbildungen μ , η. Ein Behälter mit zwei Abbildungen μ , η soll eine Monade sein. Die Formalisierung von Axiom(2.1) erbringt die Definition für finitären Monaden. Aber eine finitäre Monade ist noch kein geeigneter Behälter. Obwohl eine finitäre Monade endlich viele Elementen organisiert, kann es passieren, dass Elemente verloren oder entstehen gehen. Demnach wird Collection Monaden definiert. Collection Monaden sind geeignete Behälter, und isomorph zu $T_{\Sigma,E}$ mit balanciertem Gleichungssystem E. Man kann also Behälter-Klassen mit geeignetem $T_{\Sigma,E}$ implementieren.

Literatur:

[1] Manes, E. G. (1976) Algebraic Theories, Springer-Verlag