Partielle Monomorphie algebraischer Spezifikationen: Definition und Nachweis

Studienarbeit von Martin Giese

Betreuer: Prof. W. Menzel
Arno Schönegge

11. Oktober 1996

Inhaltsverzeichnis

1 Einleitung

Als Grundlage für die Entwicklung von Software dienen Anforderungsspezifikationen, in denen (vom Auftraggeber) das gewünschte Verhalten des Softwareprodukts festgehalten wird. Im Rahmen der formalen Softwareentwicklung müssen auch diese Anforderungsspezifikationen in einer formalen Sprache geschrieben werden, etwa als algebraische Spezifikationen erster Stufe. Der Übergang von einer Intuition zu einer formalen Darstellung derselben birgt Fehlerquellen in sich: Die Klasse von Modellen einer Spezifikation ist möglicherweise nicht die, die man intuitiv erwartet. Dann entspricht ein Programm, das bezüglich der formalen Spezifikation korrekt ist, trotzdem nicht unbedingt den Bedürfnissen des Auftraggebers.

Die Erfahrung zeigt, daß bei algebraischen Spezifikationen die angegebenen Axiome in dem intendierten Modell, bzw. den intendierten Modellen meist korrekt sind. Die Fehlerquelle liegt vielmehr darin, daß die Spezifikation weitere, nicht-intendierte Modelle besitzen kann, in denen die Axiome auch gelten, die aber ein unerwünschtes Verhalten haben. (Siehe etwa [Sch95b] für Beispiele)

Eine Methode um sicherzustellen, daß eine Spezifikation keine nicht-intendierten Modelle besitzt, besteht darin, die Monomorphie der Spezifikation nachzuweisen. Dann gibt es bis auf Isomorphie nur höchstens *ein* Modell der Spezifikation. Dieser Ansatz hat jedoch den Nachteil, daß damit absichtlich in der Spezifikation gelassene „Lücken“ unmöglich werden. Es müssen also manchmal auch Eigenschaften des Modells spezifiziert werden, auf die es eigentlich nicht ankommt. Dadurch könnte dem Implementierer die Freiheit genommen sein, zwischen mehreren möglichen Modellen dasjenige auszuwählen, welches die effizienteste Implementierung erlaubt.

Einige Fälle könnten mittels partieller Algebren behandelt werden, wobei die Spezifikationslücken als „undefiniert“ spezifiziert werden. Aber erstens entspricht diese Undefiniertheit oft nicht der verfolgten Absicht, und zweitens bringt die Behandlung der Partialität einige Probleme mit sich, siehe etwa [Jon96, GS95].

Es ergibt sich also der Wunsch nach einem Begriff von „partieller Monomorphie“, d.h. eingeschränkter Eindeutigkeit von Spezifikationen. Dies mag an einer Reihe von Beispielen erläutert werden.

1. Die Spezifikation nat+pred (Abbildung 1) beschreibt die natürlichen Zahlen mit einer Nachfolgerfunktion $+1$ und einer Vorgängerfunktion -1. Dabei ist der Vorgänger von 0 nicht festgelegt. Die Auswertung von beliebigen Termen, in denen nicht der Vorgänger von 0 berechnet wird, ist jedoch eindeutig bestimmt. [1]

[1] In dieser Arbeit werden durchgehend nur termerzeugte Modelle betrachtet.

Abbildung 1: Spezifikation nat+pred

specification
  sorts nat
  functions

0	:		$\rightarrow$	nat
$.+1$	:	nat	$\rightarrow$	nat
$.-1$	:	nat	$\rightarrow$	nat

  variables x, y : nat

axioms

$0 \neq x+1$
$x+1 = y+1 \rightarrow x = y$
$x+1-1 = x$

end specification

Abbildung 2: Spezifikation store

specification
  sorts idx, elem, store
  functions

fst-idx	:		$\rightarrow$	idx
nxt-idx	:	idx	$\rightarrow$	idx
fst-elem	:		$\rightarrow$	elem
nxt-elem	:	elem	$\rightarrow$	elem
@	:		$\rightarrow$	store
put	:	idx $\times$ elem $\times$ store	$\rightarrow$	store
get	:	idx $\times$ store	$\rightarrow$	elem

  variables i, j: idx
    x, y: elem
    s: store

axioms

$get(i, put(i, x, s)) = x$
$i \neq j \rightarrow get(i, put(j, x, s)) = get(i, s)$

end specification

Abbildung 3: Spezifikation ord-list

specification

sorts elem, oli

functions

fst-elem	:		$\to$	elem
nxt-elem	:	elem	$\to$	elem
@	:		$\to$	oli
$. .<.$	:	elem $\times$ oli	$\to$	oli
car	:	oli	$\to$	elem
cdr	:	oli	$\to$	oli

predicates

$. <_e .$	:	elem $\times$ elem
$. \ll .$	:	elem $\times$ oli

variables l, l_0, l_1: oli
e, e_0, e_1: elem

axioms

$e \ll @$
$e_0 <_e e_1 \land e_1 \ll l \to e_0 \ll e_1 .< l$
$e \ll l \to e .< l \neq @$
$e_0 .< l_0 = e_1 .< l_1 \land e_0 \ll l_0 \land e_1 \ll l_1 \to e_0 = e_1 \land l_0 = l_1$
$e \ll l \to car(e .< l) = e$
$e \ll l \to cdr(e .< l) = l$

end specification

2. Die Spezifikation store (Abbildung 2) beschreibt Felder (stores) mit den Funktionen put und get zum Einfügen bzw. Auslesen von Elementen an bestimmten Positionen. Dabei wird nichts über die Gleichheit der store-Objekte ausgesagt. Ebensowenig ist das Verhalten der Index- und Elementtypen idx und elem festgelegt. Steht jedoch das Verhalten von fst-idx, nxt-idx, fst-elem und nxt-elem fest, so ist das Ergebnis eines get an einer Stelle, die zuvor mit put gesetzt worden ist, eindeutig bestimmt.

3. Die Spezifikation ord-list (Abbildung 3) beschreibt geordnete Listen über einem Elementtyp elem mit einer Konstruktorfunktion $.<$, sowie den Dekompositionsfunktionen car und cdr. Das Prädikat $\ll$ bestimmt ob es legal ist, ein bestimmtes Element zu einer Liste hinzuzufügen. Dabei sind weder car und cdr der leeren Liste bestimmt, noch das Resultat, falls ein Element zu einer Liste hinzugefügt wird, das nicht kleiner als das erste Element der Liste ist. Es ist auch das Prädikat $\ll$

Abbildung 4: Spezifikation pred-on-zero-and-succ

```
specification
    sorts s
    functions
        0     :       →  s
        n−1   :       →  s
        . +1  :  s    →  s
        . −1  :  s    →  s
    variables x: s

axioms

     x+1−1 = x
     x+1 = 0 → x = n−1

end specification
```

nicht eindeutig bestimmt, es könnte etwa immer erfüllt sein. Dies läßt Implementierungsfreiheiten: Es wird nur verlangt, daß *wenn* es erfüllt ist, sich die Operationen richtig verhalten, und daß es für legale Operationen auf geordneten Listen sicher erfüllt ist. Auch das Verhalten der Operationen auf dem Elementtyp ist nicht spezifiziert. Aber die Spezifikation bestimmt eindeutig das Verhalten von .< beim Hinzufügen von Elementen an geordnete Listen die kleiner als deren erstes Element sind, sowie von car und cdr auf nicht-leeren, geordneten Listen, sobald das Verhalten von fst-elem, nxt-elem und $<_e$ festgelegt ist.

4. Die Spezifikation pred-on-zero-and-succ (Abbildung 4) beschreibt eine Klasse von Algebren, die entweder eine zu den natürlichen Zahlen mit Vorgänger- und Nachfolgerfunktion isomorphe Teilmenge enthalten, oder zur additiven Gruppe eines endlichen Restklassenrings Z_n mit Vorgänger- und Nachfolgerfunktion und einer Konstante n−1, deren Nachfolger 0 ist, isomorph sind. Im ersten Fall ist das Verhalten von 0, +1 und −1 bis auf den Vorgänger von 0 festgelegt und n−1 ist unbestimmt. 0 −1 und n−1 brauchen nicht einmal durch 0 und +1 erzeugbar zu sein. Dies läßt dem Implementierer die Freiheit, etwa Fehlerobjekte einzuführen. Im zweiten Fall sind auch der Vorgänger von 0 (nämlich n−1) und die Konstante n−1 festgelegt.

In dieser Arbeit soll eine formale Definition für den Begriff der partiellen Monomorphie gegeben werden, der sich mit dem intuitiven Begriff in diesen Beispielen deckt. Weiter soll ein Verfahren vorgestellt werden, daß den formalen Nachweis dieser partiellen Monomorphie erlaubt.

Im Folgenden sollen zunächst einige grundlegende Begriffe und Notationen eingeführt werden (Abschnitt 2). In Abschnitt 3 wird eine Definition partieller Monomorphie gegeben werden, die es erlaubt, die vorgestellten Beispiele zu behandeln. In Abschnitt 4 wird ein Formalismus zur Beschreibung von Termmengen eingeführt. In Abschnitt 5 werden (notwendige und hinreichende) Beweisverpflichtungen für den Nachweis partieller Monomorphie hergeleitet und Beweisheuristiken erläutert. Es folgt in Abschnitt 6 eine Beschreibung der Durchführung der vorgestellten Beispiele in dem interaktiven Beweissystem KIV ([Rei95, HRS90]). Abschnitt 7 enthält eine abschließende Diskussion der Ergebnisse.

2 Grundlagen und Definitionen

Die folgenden Notationen und Schreibweisen werden durchgehend benötigt:

2.1 Notationen

2.1.1 Definition (Notationen)
Zu einer Menge I besteht eine **I-indizierte Familie** $M = (M_i)_{i \in I}$ aus je einem Objekt M_i für jedes $i \in I$.

Eine I-indizierte Familie M von Mengen M_i ist in einer I-indizierten Familie N von Mengen N_i **enthalten**, in Zeichen $M \subseteq N$, falls $M_i \subseteq N_i$ für alle $i \in I$.

Zu einer Menge M ist M^* die Menge aller endlichen **Wörter** über M mit dem leeren Wort λ.

Zu einer Relation $R \subseteq M \times M$ und einer Teilmenge $N \subseteq M$ bezeichnet $R|_N := R \cap (N \times N) \subseteq N \times N$ deren **Einschränkung** auf N.

2.2 Signaturen und Terme

2.2.1 Definition (Signaturen)
Eine **Signatur** $\Sigma = (S, F)$ besteht aus einer endlichen Menge S (von **Sorten**) und einer endlichen $S^* \times S$-indizierten Familie $F = (F_{w,s})_{(w,s) \in S^* \times S}$ endlicher, disjunkter[2] Mengen von **Funktionssymbolen**. Statt $f \in F_{w,s}$ wird manchmal die Schreibweise $f : w \to s \in F$ verwendet.

2.2.2 Definition (Σ-Terme)
Für eine Signatur $\Sigma = (S, F)$ und eine Familie von Variablen $V = (V_s)_{s \in S}$ bezeichnet $T(\Sigma, V) = (T(\Sigma, V)_s)_{s \in S}$ die **Terme über Variablen in** V. $T(\Sigma, V)$ ist die kleinste Familie von Mengen mit:

- $V \subseteq T(\Sigma, V)$,
- $F_{\lambda,s} \subseteq T(\Sigma, V)_s$ für alle $s \in S$,

[2] Dadurch wird *overloading* von Funktionssymbolen explizit ausgeschlossen.

- $f(t_1, \dots, t_n) \in T(\Sigma, V)_s$ für alle $f : s_1, \dots, s_n \to s \in F$ und $t_1 \in T(\Sigma, V)_{s_1}, \dots, t_n \in T(\Sigma, V)_{s_n}$.

Zu einer Signatur $\Sigma = (S, F)$ bezeichnet $T(\Sigma) := T(\Sigma, \emptyset)$ die Familie der **Σ-Grundterme**. $\mathcal{T}(\Sigma) := \{T | T = (T_s)_{s \in S} \subseteq T(\Sigma)\}$ ist die Menge aller S-indizierten Familien von Σ-Grundtermmengen.

Bislang sind Signaturen denkbar, die zu leeren Mengen von Grundtermen führen. Wir wollen diese ausschließen und schränken uns daher auf Signaturen ein, für die gilt: **Zu jeder Sorte $s \in S$ existiert mindestens ein Grundterm $t \in T(\Sigma)_s$.**

Signaturmorphismen bilden die Sorten und Funktionssymbole einer Signatur derartig auf Sorten und Funktionssymbole einer anderen Signatur ab, daß die Typen erhalten bleiben.

2.2.3 Definition (Signaturmorphismen)
Für zwei Signaturen $\Sigma_1 = (S_1, F_1)$, $\Sigma_2 = (S_2, F_2)$ ist ein **Signaturmorphismus** $\sigma : \Sigma_1 \to \Sigma_2$ ein Paar $\sigma = (\sigma_{\text{sorts}}, \sigma_{\text{fcts}})$ totaler Abbildungen $\sigma_{\text{sorts}} : S_1 \to S_2$ und $\sigma_{\text{fcts}} : F_1 \to F_2$, so daß für alle $f : w \to s \in F_1$ gilt[3] $\sigma_{\text{fcts}}(f) : \sigma_{\text{sorts}}(w) \to \sigma_{\text{sorts}}(s) \in F_2$

Signaturmorphismen sollen in der naheliegenden Weise auf Terme erweitert werden. Es ist dann $\sigma(t) \in T(\Sigma_2)_{\sigma_{\text{sorts}}(s)}$ für alle $t \in T(\Sigma_1)_s$.

2.2.4 Definition (Vereinigung von Signaturen)
Seien $\Sigma_1 = (S_1, F_1)$ und $\Sigma_2 = (S_2, F_2)$ disjunkte Signaturen, d.h. $S_1 \cap S_2 = F_1 \cap F_2 = \emptyset$. Die Signatur $\Sigma := (S_1 \cup S_2, F)$ mit $F_{w,s} = F_{1,w,s}$ falls $s \in S_1$ und $F_{w,s} = F_{2,w,s}$ falls $s \in S_2$ heißt **disjunkte Vereinigung** von Σ_1 und Σ_2, in Zeichen $\Sigma = \Sigma_1 \oplus \Sigma_2$.

Durch Inklusion entstehen Signaturmorphismen $\iota_1 : \Sigma_1 \to \Sigma$ und $\iota_2 : \Sigma_2 \to \Sigma$ mit $\iota_{1,\text{sorts}}(s) = s$, $\iota_{2,\text{sorts}}(s) = s$, $\iota_{1,\text{fcts}}(f) = f$, $\iota_{2,\text{fcts}}(f) = f$.

2.3 Algebren

2.3.1 Definition (Σ-Algebren)
Sei $\Sigma = (S, F)$ eine Signatur. Eine **Σ-Algebra** $\mathcal{A} = \Big((s^{\mathcal{A}})_{s \in S}, (f^{\mathcal{A}})_{f \in F} \Big)$ besteht aus einer S-indizierten Familie nichtleerer **(Träger-)**Mengen $s^{\mathcal{A}}$ und aus einer F-indizierten Familie totaler Funktionen $f^{\mathcal{A}}$, wobei $f^{\mathcal{A}} : s_1^{\mathcal{A}} \times \dots \times s_n^{\mathcal{A}} \to s^{\mathcal{A}}$ für $f : s_1, \dots, s_n \to s \in F$. Die Klasse aller Σ-Algebren bezeichnen wir mit $\mathrm{Alg}(\Sigma)$.

2.3.2 Definition (σ-Redukt)
Zu einem Signaturmorphismus $\sigma : \Sigma_1 \to \Sigma_2$ mit $\Sigma_1 = (S_1, F_1)$, $\Sigma_2 = (S_2, F_2)$ und einer Σ_2-Algebra $\mathcal{A}$ ist das **σ-Redukt** von $\mathcal{A}$, in Zeichen $\mathcal{A}|_\sigma$, die durch $s^{\mathcal{A}|_\sigma} := \sigma_{\text{sorts}}(s)^{\mathcal{A}}$ für $s \in S_1$ und $f^{\mathcal{A}|_\sigma} := \sigma_{\text{fcts}}(f)^{\mathcal{A}}$ für $f \in F_1$ erklärte Σ_1-Algebra.

[3] $\sigma_{\text{sorts}}(s_1, \dots, s_n)$ ist eine Kurzschreibweise für $(\sigma_{\text{sorts}}(s_1), \dots, \sigma_{\text{sorts}}(s_n))$.

2.3.3 Definition (Variablenbelegung)
Zu einer Signatur $\Sigma = (S, F)$, einer Algebra $\mathcal{A} \in \mathrm{Alg}(\Sigma)$ und einer Familie von Variablen $V = (V_s)_{s \in S}$ heißt $\alpha = (\alpha_s)_{s \in S}$, wobei $\alpha_s : V_s \to s^{\mathcal{A}}$ totale Funktionen sind, eine **Variablenbelegung** von V nach $\mathcal{A}$.

Häufig wird $\alpha = [v_1 := x_1, \ldots, v_n := x_n]$ geschrieben für eine Variablenbelegung von $\{v_1, \ldots, v_n\}$ mit $\alpha(v_i) = x_i$.

2.3.4 Definition (Termauswertung)
Zu einer Signatur $\Sigma = (S, F)$, einer Algebra $\mathcal{A} \in \mathrm{Alg}(\Sigma)$, einer Familie von Variablen $V = (V_s)_{s \in S}$ und einer Variablenbelegung α wird die **Termauswertungsfunktion** $\cdot^{\mathcal{A},\alpha} = (\cdot_s^{\mathcal{A},\alpha})_{s \in S}$ mit $\cdot_s^{\mathcal{A},\alpha} : T(\Sigma, V)_s \to s^{\mathcal{A}}$ definiert durch

- $t_s^{\mathcal{A},\alpha} = \alpha_s(v)$, falls $t = v \in V_s$.
- $t_s^{\mathcal{A},\alpha} = f^{\mathcal{A}}$, falls $t = f \in F_{\lambda,s}$.
- $t_s^{\mathcal{A},\alpha} = f^{\mathcal{A}}((t_1)_{s_1}^{\mathcal{A},\alpha}, \ldots, (t_n)_{s_n}^{\mathcal{A},\alpha})$, falls $t = f(t_1, \ldots, t_n)$ für ein Funktionssymbol $f : s_1, \ldots, s_n \to s \in F$ und $t_i \in T(\Sigma, V)_{s_i}$.

Die Sorte s wird normalerweise weggelassen, da sie sich aus dem Term ergibt. Ebenso wird für Grundterme $t \in T(\Sigma)$ die Variablenbelegung i.allg. nicht dazu geschrieben — es genügt also $t^{\mathcal{A}}$ — da sie keinen Einfluß auf die Auswertung des Terms hat.

Ist $U_s \subseteq T(\Sigma)_s$ eine Menge von Grundtermen, so wird die Bezeichnung $U^{\mathcal{A}} := \{t^{\mathcal{A}} | t \in U\}$ benutzt.

2.3.5 Definition (Termerzeugte Algebren)
Sei $\Sigma = (S, F)$ eine Signatur. Eine **Σ-Algebra** $\mathcal{A} \in \mathrm{Alg}(\Sigma)$ heißt **termerzeugt**, falls für alle $s \in S$ gilt $\{t^{\mathcal{A}} | t \in T(\Sigma)_s\} = s^{\mathcal{A}}$.

Die Klasse aller termerzeugten Σ-Algebren bezeichnen wir mit $\mathrm{Gen}(\Sigma)$.

Σ-Homomorphismen bilden Trägerelemente einer Σ-Algebra derart auf Trägerelemente einer anderen Σ-Algebra ab, daß die Deutung der Funktionssymbole erhalten bleibt.

2.3.6 Definition (Σ-Homomorphismen, Σ-Isomorphismen)
Seien $\Sigma = (S, F)$ und $\mathcal{A}, \mathcal{B} \in \mathrm{Alg}(\Sigma)$. Ein **$\Sigma$-Homomorphismus** $h : \mathcal{A} \to \mathcal{B}$ ist eine S-indizierte Familie $h = (h_s)_{s \in S}$ totaler Abbildungen $h_s : s^{\mathcal{A}} \to s^{\mathcal{B}}$, so daß für jedes $f : s_1, \ldots, s_n \to s \in F$ und alle $a_1 \in s_1^{\mathcal{A}}, \ldots a_n \in s_n^{\mathcal{A}}$ gilt:

$$h_s(f^{\mathcal{A}}(a_1, \ldots, a_n)) = f^{\mathcal{B}}(h_{s_1}(a_1), \ldots, h_{s_n}(a_n)).$$

Ein **Σ-Isomorphismus** ist ein Σ-Homomorphismus, bei dem alle h_s bijektiv sind. Zwei Algebren $\mathcal{A}, \mathcal{B} \in \mathrm{Alg}(\Sigma)$ heißen **Σ-isomorph**, in Zeichen $\mathcal{A} \simeq \mathcal{B}$, falls ein Σ-Isomorphismus $h : \mathcal{A} \to \mathcal{B}$ existiert.

2.3.7 Faktum (Charakterisierung der Homomorphie)
Seien $\mathcal{A}, \mathcal{B} \in \mathrm{Gen}(\Sigma)$ zwei termerzeugte Algebren über einer Signatur $\Sigma = (S, F)$. $h : \mathcal{A} \to \mathcal{B}$ ist Σ-Homomorphismus genau dann, wenn $h_s(t^{\mathcal{A}}) = t^{\mathcal{B}}$ für alle $t \in T(\Sigma)_s$ und $s \in S$.

Beweis. Siehe etwa [Wir90], Fact 2.2.6.(1) ■

2.4 Spezifikationen

In dieser Arbeit soll der Begriff der Spezifikation völlig abstrakt gehalten werden. Die Ergebnisse sind also von keinem konkreten Formalismus abhängig. In den Beispielen werden algebraische Spezifikationen erster Stufe verwendet.

2.4.1 Definition (Spezifikationen)
Eine **Spezifikation** $SP = (\Sigma, C)$ besteht aus einer Signatur Σ und einer gegenüber Σ-Isomorphie abgeschlossenen Menge von Algebren $C \subseteq \mathrm{Gen}(\Sigma)$, d.h. falls $\mathcal{A} \in C$ und $\mathcal{A} \simeq \mathcal{B}$, so ist auch $\mathcal{B} \in C$.

$\mathrm{Spec}(\Sigma) = \{(\Sigma, C) \mid C \subseteq \mathrm{Gen}(\Sigma)$ ggü. Isomorphie abgeschlossen$\}$ soll die Menge aller Spezifikationen zur Signatur Σ bezeichnen.

Für eine Spezifikation $SP = (\Sigma, C)$ soll $\mathrm{Gen}(SP) = C$ die Menge ihrer Modelle bezeichnen.

Jeder Signaturmorphismus $\sigma : \Sigma_1 \to \Sigma_2$ soll auf Spezifikationen erweitert werden, und zwar durch $\sigma(SP) = (\Sigma_2, \{\mathcal{A} \mid \mathcal{A}|_\sigma \in \mathrm{Gen}(SP)\})$ für alle $SP \in \mathrm{Spec}(\Sigma_1)$.

Damit ergibt sich folgende Definition der (totalen) Monomorphie:

2.4.2 Definition (Monomorphie)
Eine Spezifikation $SP \in \mathrm{Spec}(\Sigma)$ heißt **monomorph**, falls je zwei Algebren $\mathcal{A}, \mathcal{B} \in \mathrm{Gen}(SP)$ Σ-isomorph sind.

In Abschnitt 5 wird noch die folgende Operation auf Spezifikationen benötigt:

2.4.3 Definition (Vereinigung von Spezifikationen)
Gegeben seien disjunkte Signaturen Σ_1, Σ_2, sowie Spezifikationen $SP_1 \in \mathrm{Spec}(\Sigma_1)$ und $SP_2 \in \mathrm{Spec}(\Sigma_2)$. Dann wird die **disjunkte Vereinigung** von SP_1 und SP_2, in Zeichen $SP_1 \oplus SP_2$, definiert durch: $SP_1 \oplus SP_2 = (\Sigma_1 \oplus \Sigma_2, \{\mathcal{A} \mid \mathcal{A}|_{\iota_1} \in \mathrm{Gen}(SP_1)$ und $\mathcal{A}|_{\iota_2} \in \mathrm{Gen}(SP_2)\})$, wobei ι_1 und ι_2 die Inklusionsmorphismen aus Definition 2.2.4 sind.

2.5 Formeln

In den folgenden Definitionen soll vorerst auch von keinem konkreten Formalismus für Formeln mit Variablen ausgegangen werden, obwohl in den Beispielen Prädikatenlogik erster Stufe verwendet wird. Die folgenden Definitionen beinhalten einige Mindestanforderungen an den Formelbegriff, die später noch ergänzt werden, wo konkretere Anforderungen an den Formalismus entstehen.

2.5.1 Definition (Formeln)
Im Folgenden soll $\mathrm{Fml}(\Sigma, V)$ für jede Signatur Σ und jede Familie von Variablen $V = (V_s)_{s\in S}$ die Menge der **Σ-Formeln** mit freien Variablen in V bezeichnen.

Jeder Signaturmorphismus $\sigma : \Sigma_1 \to \Sigma_2$ soll auf Formeln erweitert werden, und zwar so, daß $\sigma(F) \in \mathrm{Fml}(\Sigma_2, V)$ für alle $F \in \mathrm{Fml}(\Sigma_1, V)$.

Es sei weiter eine **Gültigkeitsrelation** $\models$ gegeben, wobei $\mathcal{A}, \alpha \models \Phi$ die Gültigkeit von $\Phi \in \mathrm{Fml}(\Sigma, V)$ in der Algebra[4] $\mathcal{A} \in \mathrm{Gen}(\Sigma)$ unter der Variablenbelegung α bezeichnet. Dabei soll für jeden Signaturmorphismus $\sigma : \Sigma_1 \to \Sigma_2$, jede Algebra $\mathcal{A} \in \mathrm{Gen}(\Sigma_2)$, jede Variablenbelegung α von V nach $\mathcal{A}|_\sigma$ und jede Formel $\Phi \in \mathrm{Fml}(\Sigma_1, V)$ gelten: $\mathcal{A}|_\sigma, \alpha \models \Phi$ genau dann, wenn $\mathcal{A}, \alpha \models \sigma(\Phi)$.

3 Partielle Monomorphie auf Termmengen

Der gewöhnliche Monomorphiebegriff fordert die Existenz eines Isomorphismus zwischen je zwei Algebren aus $\mathrm{Gen}(SP)$. Diese Bedingung stellt wegen Faktum 2.3.7 sicher, daß die Interpretation *sämtlicher* Grundterme in allen Algebren aus $\mathrm{Gen}(SP)$ in gleicher Weise geschieht.

Für einen Begriff von partieller Monomorphie soll nur die Interpretation *bestimmter* Grundterme festgelegt sein. Daraus ergibt sich der folgende Begriff der partiellen Isomorphie zweier Algebren.

3.1 Partielle Isomorphie

3.1.1 Definition (Partielle Isomorphie 1)
Seien $\Sigma = (S, F)$ eine Signatur, $U \subseteq T(\Sigma)$ eine Familie von Grundtermmengen und $\mathcal{A}, \mathcal{B} \in \mathrm{Gen}(\Sigma)$ zwei Σ-Algebren. Ein **partieller Σ-Isomorphismus auf U**, $I : \mathcal{A} \to \mathcal{B}$ ist eine S-indizierte Familie $I = (I_s)_{s\in S}$ totaler, bijektiver Abbildungen $I_s : U_s^{\mathcal{A}} \to U_s^{\mathcal{B}}$, so daß für alle Sorten $s \in S$ und Terme $t \in U_s$ gilt: $I_s(t^{\mathcal{A}}) = t^{\mathcal{B}}$.

Zwei Algebren $\mathcal{A}, \mathcal{B} \in \mathrm{Gen}(\Sigma)$ heißen **partiell isomorph auf U**, in Zeichen $\mathcal{A} \overset{U}{\simeq} \mathcal{B}$, falls es einen partiellen Σ-Isomorphismus auf U von $\mathcal{A}$ nach $\mathcal{B}$ gibt.

[4] Hier sollen nur termerzeugte Modelle betrachtet werden.

Dieser Begriff ist eine Verallgemeinerung des bisherigen Begriffs der Isomorphie:

3.1.2 Faktum (Spezialfall Isomorphie)
$I : \mathcal{A} \to \mathcal{B}$ mit $\mathcal{A}, \mathcal{B} \in \mathrm{Gen}(\Sigma)$ ist ein partieller Σ-Isomorphismus auf $T(\Sigma)$ genau dann, wenn I ein Σ-Isomorphismus ist, also insbesondere $\overset{T(\Sigma)}{\simeq} = \simeq$.

Beweis. Direkte Folge von Faktum 2.3.7.

■

In den Beispielen wird sich herausstellen, daß der mittels Grundtermmengen definierte Begriff noch nicht allgemein genug ist. Es ist teilweise nötig, für verschiedene Algebren unterschiedliche Termmengen zu benutzen. Das führt zu den folgenden, allgemeineren Definitionen:

3.1.3 Definition (Termmengenfunktionen)
Sei Σ eine Signatur. Eine Funktion $\mathcal{U} : \mathrm{Gen}(\Sigma) \to \mathcal{T}(\Sigma)$, so daß für alle $\mathcal{A}, \mathcal{B} \in \mathrm{Gen}(\Sigma)$ mit $\mathcal{A} \simeq \mathcal{B}$ gilt $\mathcal{U}(\mathcal{A}) = \mathcal{U}(\mathcal{B})$ heißt **Termmengenfunktion**.

3.1.4 Definition (Partielle Isomorphie 2)
Seien Σ eine Signatur, $\mathcal{U} : \mathrm{Gen}(\Sigma) \to \mathcal{T}(\Sigma)$ eine Termmengenfunktion und $\mathcal{A}, \mathcal{B} \in \mathrm{Gen}(\Sigma)$ zwei Σ-Algebren. Zwei Algebren $\mathcal{A}, \mathcal{B} \in \mathrm{Gen}(\Sigma)$ heißen **partiell isomorph auf** $\mathcal{U}$, in Zeichen $\mathcal{A} \overset{\mathcal{U}}{\simeq} \mathcal{B}$, falls $\mathcal{U}(\mathcal{A}) = \mathcal{U}(\mathcal{B})$ und $\mathcal{A} \overset{\mathcal{U}(\mathcal{A})}{\simeq} \mathcal{B}$ (bzw. äquivalent $\mathcal{A} \overset{\mathcal{U}(\mathcal{B})}{\simeq} \mathcal{B}$).

3.1.5 Definition ($\boldsymbol{\emptyset}$)
$\boldsymbol{\emptyset} : \mathrm{Gen}(\Sigma) \to \mathcal{T}(\Sigma)$ bezeichne die Funktion mit $\boldsymbol{\emptyset}(\mathcal{A}) = \emptyset$ für alle $\mathcal{A}$.

3.1.6 Faktum (Eigenschaften von $\overset{\mathcal{U}}{\simeq}$)
Die Relation $\overset{\mathcal{U}}{\simeq} \subseteq \mathrm{Gen}(\Sigma) \times \mathrm{Gen}(\Sigma)$ hat die folgenden Eigenschaften:

- *$\overset{\mathcal{U}}{\simeq}$ ist eine Äquivalenzrelation.*
- $\overset{\boldsymbol{\emptyset}}{\simeq} = \mathrm{Gen}(\Sigma) \times \mathrm{Gen}(\Sigma)$.

Beweis.

- Für jede Familie von Termmengen U ist $\overset{U}{\simeq} \subseteq \mathrm{Gen}(\Sigma) \times \mathrm{Gen}(\Sigma)$ eine Äquivalenzrelation , wobei als Abbildung I für Reflexivität, Symmetrie und Transitivität jeweils Identität, Inverse und Verkettung genommen werden. Da die Gleichheit Äquivalenzrelation ist, ist auch $\overset{\mathcal{U}}{\simeq}$ für jede Funktion $\mathcal{U} : \mathrm{Gen}(\Sigma) \to \mathcal{T}(\Sigma)$ Äquivalenzrelation.

- Es ist $\mathcal{A} \overset{\emptyset}{\simeq} \mathcal{B}$ für alle Algebren $\mathcal{A}, \mathcal{B} \in \mathrm{Gen}(\Sigma)$, da die leere Abbildung $I : \emptyset \to \emptyset$ bijektiv ist.

■

3.2 Partielle Monomorphie

Die Definition soll zunächst der Deutlichkeit halber für Termmengen angegeben werden:

3.2.1 Definition (Partielle Monomorphie auf Termmengen)
Seien $\Sigma = (S, F)$ eine Signatur, $U \subseteq T(\Sigma)$ eine Familie von Grundtermmengen, und $SP \in \mathrm{Spec}(\Sigma)$ eine Spezifikation. Dann heißt SP **monomorph auf** U, falls $\mathcal{A} \overset{U}{\simeq} \mathcal{B}$ für alle Algebren $\mathcal{A}, \mathcal{B} \in \mathrm{Gen}(\Sigma)$.

3.2.2 Faktum (Charakterisierung von Monomorphie 1)
$SP \in \mathrm{Spec}(\Sigma)$ ist monomorph genau dann, wenn SP partiell monomorph auf $T(\Sigma)$ ist.

Beweis. Dies ergibt sich aus den Fakten 3.1.2 und 3.1.6. ∎

Während dieser Begriff zur Behandlung einfacher Beispiele wie nat+pred ausreicht, erfordern die anderen Beispiele ein mächtigeres Konzept. Die Folgende Definition stellt eine Verallgemeinerung auf Termmengenfunktionen dar und führt das Konzept der relativen Monomorphie ein.

3.2.3 Definition (Partielle Monomorphie)
Seien $\Sigma = (S, F)$ eine Signatur, $\mathcal{U}, \mathcal{V} : \mathrm{Gen}(\Sigma) \to \mathcal{T}(\Sigma)$ Termmengenfunktionen, und $SP \in \mathrm{Spec}(\Sigma)$ eine Spezifikation. Dann heißt SP

- **monomorph auf $\mathcal{U}$ relativ zu $\mathcal{V}$**, falls $\overset{\mathcal{V}}{\simeq}|_{\mathrm{Gen}(SP)} \subseteq \overset{\mathcal{U}}{\simeq}|_{\mathrm{Gen}(SP)}$.
- **monomorph auf $\mathcal{U}$**, falls SP monomorph auf $\mathcal{U}$ relativ zu $\emptyset$ ist.
- **monomorph relativ zu $\mathcal{V}$**, falls SP monomorph auf $\mathcal{U}$ relativ zu $\mathcal{V}$ ist, wobei $\mathcal{U}(\mathcal{A}) := T(\Sigma)$ für alle $\mathcal{A} \in \mathrm{Gen}(SP)$.

Das folgende Faktum stellt den Bezug zum totalen Monomorphiebegriff und zur partiellen Monomorphie auf einer Termmenge her.

3.2.4 Faktum (Charakterisierung von Monomorphie 2)
SP ist monomorph genau dann, wenn SP monomorph relativ zu $\emptyset$ ist.

SP ist monomorph auf U genau dann, wenn SP monomorph auf $\mathcal{U}$ ist, wobei $\mathcal{U}(\mathcal{A}) := U$ für alle $\mathcal{A} \in \mathrm{Gen}(\Sigma)$.

Beweis. Dies ergibt sich aus Faktum 3.2.2 und den Definitionen. ∎

3.2.5 Faktum (partielle Ordnung)
Die Relation $\sqsubseteq_{SP}$ auf den Termmengenfunktionen $\mathrm{Gen}(\Sigma) \to \mathcal{T}(\Sigma)$ *mit* $\mathcal{V} \sqsubseteq_{SP} \mathcal{U}$*, falls SP monomorph auf* $\mathcal{U}$ *relativ zu* $\mathcal{V}$ *ist, ist eine (partielle) Ordnung mit größtem Element* $\emptyset$*.*

Beweis. Folgt aus Def. 3.2.3 und 3.1.5 mit den Eigenschaften von $\subseteq$. ■

Zu dieser Definition von partieller Monomorphie sind zwei Bemerkungen angebracht. Erstens mutet das Benutzen von Termmengen, also *syntaktischen* anstelle von rein *semantischen* Entitäten in der Definition ungeschickt an. Man sollte sich jedoch klar machen, daß es keine andere naheliegende Möglichkeit gibt, die zumindest in der Lage ist die Beispiele in der Einleitung zu erfassen. Insbesondere kann nicht ohne weiteres von einem Isomorphismus zwischen den gesamten Trägermengen von Modellen ausgegangen werden, wie man sich am Beispiel store verdeutlichen kann. Auch ein reines Abschwächen der Verträglichkeitsbedingung für die Funktionen in der Monomorphie-Definition 2.4.2 erreicht noch nicht das gewünschte, da es zum Beispiel Modelle $\mathcal{A}$ von nat+pred gibt, in denen $0^{\mathcal{A}}\ -1^{\mathcal{A}} = 0^{\mathcal{A}}\ +1^{\mathcal{A}}$ ist, der Vorgänger von 0 also zu 1 ausgewertet wird. Diese Freiheit bei der Implementierung ist gerade erwünscht. Sicherlich soll nun der Vorgänger von 1, $0\ +1\ -1$ eindeutig festgelegt sein. Würde die Monomorphie auf Objekten definiert, müßte man dann aber — aufgrund der obigen Gleichung — verlangen, daß $0\ -1\ -1$ als Vorgänger von 1 eindeutig festgelegt ist, was sicherlich nicht sinnvoll ist.

Zweitens wurde die Einführung der Termmengen*funktionen* noch nicht motiviert. Der Anreiz, diese zu verwenden kommt aus zwei Richtungen: zum einen kann es speziell beim Begriff relativer Monomorphie — wie im Beispiel pred-on-zero-and-succ — vorkommen, daß die Termmenge, auf der Monomorphie behauptet wird, von bestimmten Aspekten der Algebra abhängt. Im Beweis zu ord-list (siehe Abschnitt 6.2.3) hängen zum Beispiel sowohl $\mathcal{U}$ als auch $\mathcal{V}$ von der Algebra ab. Andererseits müssen Termmengen auch beschrieben werden und es erweist sich als nützlich, wenn man bei der Beschreibung auf Formeln über der Signatur zurückgreifen kann. (Man siehe dazu etwa die Bemerkung am Ende von Abschnitt 3.3.1.) Diese Formeln müssen allerdings in einer Algebra interpretiert werden. Der Ansatz, die Gültigkeit der Formeln in allen Modellen zu verlangen, um unabhängig von einer bestimmten Algebra zu werden, wurde erprobt und hat sich als unergiebig erwiesen. Insbesondere konnten dann keine notwendigen sondern nur noch hinreichende Beweisverpflichtungen gewonnen werden. Die Vorteile der Verwendung der Termmengenfunktionen anstelle konstanter Termmengen scheint also die damit verbundene Verkomplizierung des Begriffs aufzuwiegen. Zur Verdeutlichung folgt eine Behandlung der in der Einleitung vorgestellten Beispiele im Rahmen der eben gegebenen Definition. Die Beschreibung der Termmengen ist hier noch nicht auf einen formalen Nachweis der Monomorphie ausgerichtet, dies geschieht in Abschnitt 4.

3.3 Beispiele

In den folgenden Beispielen tauchen mitunter Prädikate auf. In unseren Definitionen wurden diese der Übersichtlichkeit halber nicht berücksichtigt, dies kann aber leicht geschehen. Prädikate sollen hier einfach als Funktionen mit Ergebnistyp bool betrachtet werden, wobei der Typ bool, ebenso wie zwei Funktionssymbole true, false $:\to$ bool als implizit in jeder Signatur vorhanden angesehen wird. In jeder Algebra $\mathcal{A}$ soll $\mathrm{bool}^{\mathcal{A}} = \{tt, ff\}$ sein, und $\mathrm{true}^{\mathcal{A}} = tt$, sowie $\mathrm{false}^{\mathcal{A}} = ff$. Weiterhin sollen nur Termmengen U betrachtet werden, die zur Sorte bool mindestens die Terme true und false enthalten. Das führt dazu, daß für zwei Algebren $\mathcal{A} \overset{U_{\mathrm{bool}}}{\simeq} \mathcal{B}$ genau dann, wenn für alle $t \in U_{\mathrm{bool}}$ gilt: $t^{\mathcal{A}} = t^{\mathcal{B}}$, was dem entspricht, was man von dem Begriff der partiellen Monomorphie als eindeutiger Festgelegtheit eines Prädikats erwartet.

Die in den Beispielen erwähnten Monomorphieresultate wurden alle formal mit den in Abschnitt 5 beschriebenen Methoden bewiesen.

3.3.1 nat+pred

Es soll die Spezifikation nat+pred aus Abbildung 1 betrachtet werden. Sei $\mathcal{U}(\mathcal{A}) = U$ unabhängig von der Algebra $\mathcal{A} \in \mathrm{Gen}(\text{nat+pred})$, wobei U die Menge[5] aller Terme sein soll, in denen nie der Vorgänger von 0 genommen wird, also die kleinste Menge von Termen mit

- $0 \in U$,
- $t\ {+1} \in U$ für alle $t \in U$,
- $t\ {-1} \in U$ für alle $t \in U$, so daß $+1$ in t öfter vorkommt als -1. (*)

3.3.1 Satz (partielle Monomorphie von nat+pred)
Die Spezifikation nat+pred ist monomorph auf $\mathcal{U}$, d.h. zwei beliebige Modelle aus Gen(*nat+pred*) *sind isomorph auf U.*

Man bemerke, daß man die gleiche Termmenge erhält, falls man die Bedingung (*) im letzten Punkt dadurch ersetzt, daß $t^{\mathcal{A}} \neq 0^{\mathcal{A}}$ in einer beliebigen Algebra $\mathcal{A} \in \mathrm{Gen}(\text{nat+pred})$ sein soll.

3.3.2 store

Ein erstes Beispiel für relative Monomorphie liefert die Spezifikation store aus Abbildung 2. Sei $\Sigma_0 = (\{\text{idx,elem}\}, \{\text{fst-idx,nxt-idx,fst-elem,nxt-elem}\})$ die Signatur der „Parametertypen" und $\mathcal{V}(\mathcal{A}) := T(\Sigma_0)$ die Familie der Σ_0-Grundterme für alle $\mathcal{A} \in \mathrm{Gen}(\text{store})$. $\mathcal{U}(\mathcal{A})_{\text{elem}}$ soll alle Terme aus put und get enthalten, in denen get nur an Stellen angewandt wird, die vorher mit

[5] Der Index nat für den Typ soll hier der Einfachheit halber weggelassen werden.

put gesetzt wurden. Man kann etwa eine Familie von Hilfsmengen $ST(k)$ für $k \in \mathrm{idx}^{\mathcal{A}}$ definieren, so daß $ST(k)$ gerade alle store-Terme enthält, in denen die Stelle k (eindeutig) gesetzt worden ist.

Die $ST(k)$ und $\mathcal{U}(\mathcal{A})_{\mathrm{elem}}$ werden dann definiert als kleinstes System von Mengen, für das gilt:

- $\mathrm{put}(t_i,t_x,t_s) \in ST(k)$ für alle Terme $t_i \in T(\Sigma)_{\mathrm{idx}}$ mit $t_i^{\mathcal{A}} = k$, $t_x \in \mathcal{U}(\mathcal{A})_{\mathrm{elem}}$ und $t_s \in T(\Sigma)_{\mathrm{store}}$,
- $\mathrm{put}(t_i,t_x,t_s) \in ST(k)$ für alle Terme $t_i \in T(\Sigma)_{\mathrm{idx}}$, $t_x \in \mathcal{U}(\mathcal{A})_{\mathrm{elem}}$ und $t_s \in ST(k)$,
- $T(\Sigma_0)_{\mathrm{elem}} \subseteq \mathcal{U}(\mathcal{A})_{\mathrm{elem}}$,
- $\mathrm{get}(t_i,t_s) \in \mathcal{U}(\mathcal{A})_{\mathrm{elem}}$ für $t_i \in T(\Sigma)_{\mathrm{idx}}$ und $t_s \in ST(t_i^{\mathcal{A}})$.

Der erste Punkt betrifft den Fall, daß ein Stelle neu gesetzt wird, der zweite Punkt den Fall, daß die Stelle schon gesetzt war. Der dritte Punkt sorgt dafür, daß die elem-Terme aus Σ_0 auch in $\mathcal{U}$ enthalten sind, während der letzte Punkt gerade diejenigen get-Terme hinzufügt, in denen die abgefragte Stelle gesetzt worden ist.

Mit diesen Definitionen gilt:

3.3.2 Satz (partielle Monomorphie von store)
Die Spezifikation store ist monomorph auf $\mathcal{U}$ relativ zu $\mathcal{V}$. Das heißt: sind zwei Modelle $\mathcal{A}, \mathcal{B}$ isomorph auf $T(\Sigma_0)$, so ist $\mathcal{U}(\mathcal{A}) = \mathcal{U}(\mathcal{B})$ und die Modelle sind monomorph auf $\mathcal{U}(\mathcal{A}) = \mathcal{U}(\mathcal{B})$.

Man bemerke, daß $\mathcal{U}(\mathcal{A})_{\mathrm{store}}$ leer ist, was daran liegt, daß über die Gleichheit von store-Objekten gar keine Aussage gemacht werden soll.

In den vorliegenden Termmengen sind auch Terme enthalten, in denen get auf ungesetzte Stelle angewandt wird. Diese kommen durch die Bedingung $t_s \in T(\Sigma)_{\mathrm{store}}$ hinzu. Es wird also die Eindeutigkeit von get an gesetzten Stellen behauptet, auch wenn vorher an anderen Stellen illegale Operationen erfolgt sind. Darin unterscheidet sich dieses Beispiel von den anderen, in denen die „legalen“ Terme nur aus anderen „legalen“ Termen zusammengesetzt werden.

3.3.3 ord-list

Im Falle der Spezifikation ord-list aus Abbildung 3 haben wir — ähnlich wie bei store — einen „Parametertyp“ elem. Wir definieren hier $\Sigma_0 = (\{\mathrm{elem}\}, \{\text{fst-elem}, \text{nxt-elem}\})$ und $\mathcal{V}(\mathcal{A}) := T(\Sigma_0)$. Die Terme in $\mathcal{U}(\mathcal{A})$ hängen hier ganz explizit von der Algebra ab, da diese ja die Interpretation der Ordnung $<_e$ bestimmt. Es ist $\mathcal{U}(\mathcal{A}) = (\mathcal{U}(\mathcal{A})_{\mathrm{oli}}, \mathcal{U}(\mathcal{A})_{\mathrm{elem}})$, wobei $\mathcal{U}(\mathcal{A})_{\mathrm{oli}}$ und $\mathcal{U}(\mathcal{A})_{\mathrm{elem}}$ die kleinsten Termmengen sind mit

- $@ \in \mathcal{U}(\mathcal{A})_{\text{oli}}$
- t_e .< $t_l \in \mathcal{U}(\mathcal{A})_{\text{oli}}$ für alle $t_l \in \mathcal{U}(\mathcal{A})_{\text{oli}}$ und $t_e \in \mathcal{U}(\mathcal{A})_{\text{elem}}$ für die gilt: $\mathcal{A} \models t_e <_e \text{car}(t_l)$.
- $\text{cdr}(t_l) \in \mathcal{U}(\mathcal{A})_{\text{oli}}$ für alle $t_l \in \mathcal{U}(\mathcal{A})_{\text{oli}}$ mit $\mathcal{A} \models t_l \neq @$.
- $T(\Sigma_0)_{\text{elem}} \subseteq \mathcal{U}(\mathcal{A})_{\text{elem}}$.
- $\text{car}(t_l) \in \mathcal{U}(\mathcal{A})_{\text{elem}}$ für alle $t_l \in \mathcal{U}(\mathcal{A})_{\text{oli}}$ mit $\mathcal{A} \models t_l \neq @$.

3.3.3 Satz (partielle Monomorphie von ord-list)
Mit den vorangegangenen Definitionen ist ord-list monomorph auf $\mathcal{U}$ relativ zu $\mathcal{V}$.

3.3.4 pred-on-zero-and-succ

Bei der Spezifikation aus Abbildung 4 hängen die Termmengen von der Kardinalität des Modells ab. Mit $\Sigma_0 = (\{\text{s}\}, \{0, +1\})$ wird $\mathcal{V}(\mathcal{A}) := T(\Sigma_0)$ festgelegt. Im Falle endlicher Kardinalität ist durch die Interpretation der Terme in $\mathcal{V}(\mathcal{A})$ schon die Interpretation von -1 festgelegt. Bei unendlicher Kardinalität darf (wie bei nat+pred) -1 nicht auf 0 angewandt werden. Sei also $\mathcal{U}(A) := U$ die kleinste Menge mit:[6]

- $0 \in U$
- $\text{n}-1 \in U$ falls $\text{s}^{\mathcal{A}}$ endlich
- $t\ +1 \in U$ für alle $t \in U$
- $t\ -1 \in U$ für alle $t \in U$, so daß $\mathcal{A} \models t \neq 0$ oder $\text{s}^{\mathcal{A}}$ endlich.

Die Unterscheidung zwischen endlichem und unendlichem Modell ist auch formal durch die Bedingung $\mathcal{A} \models \text{n}-1\ +1 = 0$ zu erreichen.

Auch hier gilt:

3.3.4 Satz (partielle Monomorphie von pred-on-zero-and-succ)
pred-on-zero-and-succ ist monomorph auf $\mathcal{U}$ relativ zu $\mathcal{V}$.

4 Beschreibung von Termmengen

Um partielle Monomorphie formal nachzuweisen, ist ein Formalismus zur Beschreibung von Termmengen nötig. Der nachfolgend beschriebene Formalismus hat sich als ausreichend mächtig erwiesen, um die in den Beispielen vorkommenden Termmengen zu beschreiben. Es lassen sich aus Termmengenbeschreibungen in dieser Sprache automatisch Beweisverpflichtungen in dynamischer Logik generieren, um partielle Monomorphie nachzuweisen.

[6]Der Index s für den Typ soll auch hier weggelassen werden.

4.1 Syntax von Termmengenbeschreibungen

Sei $\Sigma = (S, F)$ eine Signatur. In den folgenden Abschnitten soll $\mathcal{T} = T(\Sigma) \in \mathrm{Gen}(\Sigma)$ die freie Termalgebra bezeichnen.

Sei $TMVar = (TMVar_{w,s})_{(w,s)\in S^*\times S}$ eine Menge von **Termmengenbezeichnern** und $TVar = (TVar_s)_{s\in S}$ eine Menge von **Termbezeichnern**. Statt $v \in TMVar_{(s_1,\dots,s_n),s}$ wird auch die Schreibweise $v : s_1, \dots, s_n \to s \in TMVar$ verwendet.

4.1.1 Definition (instantiierte Termmengenbezeichner)
Für einen Termmengenbezeichner $v : s_1, \dots, s_n \to s \in TMVar$ und Terme $t_i \in T(\Sigma, TVar)_{s_i}$ heißt $v(t_1, \dots, t_n)$ **instantiierter Termmengenbezeichner vom Typ** s.

4.1.2 Definition (Klauseln)
Seien für $i \in \{1, \dots, n\}$: $x_i \in TVar_{s_i}$ Termvariablen, für $i \in \{0, \dots, n\}$: w_i instantiierte Termmengenbezeichner vom Typ s_i, $t \in T(\Sigma, TVar)_{s_0}$ ein Term vom Typ s_0 über den Termbezeichnern und $\Phi \in \mathrm{Fml}(\Sigma, TVar)$ eine Formel mit freien Variablen in $TVar$. Dann ist

$$t \ \texttt{in} \ w_0 \ \texttt{for} \ x_1 \ \texttt{in} \ w_1\texttt{,} \ \dots\texttt{,} \ x_n \ \texttt{in} \ w_n \ \texttt{with} \ \Phi\texttt{.}$$

eine **Klausel**. Dabei darf `for` weggelassen werden, falls $n = 0$ und `with` Φ darf weggelassen werden, falls Φ allgemeingültig (also in jeder Algebra unter jeder Variablenbelegung gültig) ist. Eine **Termmengenbeschreibung** ist eine Menge von Klauseln.

4.2 Semantik von Termmengenbeschreibungen

Die Interpretation einer Termmengenbeschreibung ist eine Familie von Grundtermmengen, und zwar eine Teilmenge von $T(\Sigma)_s = s^{\mathcal{T}}$ für jeden instantiierten Termmengenbezeichner w vom Typ s. Die Interpretation hängt von einer Algebra $\mathcal{A} \in \mathrm{Gen}(\Sigma)$ ab, in der die Formeln Φ interpretiert werden.

4.2.1 Definition (Semantik von Termmengenbeschreibungen)
Gegeben seien eine Termmengenbeschreibung[7] K und eine Σ-Algebra $\mathcal{A} \in \mathrm{Gen}(\Sigma)$. Es wird für jedes $v : s_1, \dots, s_n \to s \in TMVar$ eine Funktion $v^{K,\mathcal{A}} : s_1^{\mathcal{T}} \times \cdots \times s_n^{\mathcal{T}} \to \mathcal{P}(s^{\mathcal{T}})$ definiert,[8] so daß das System ihrer Werte $v^{K,\mathcal{A}}(t_1, \dots, t_n)$ mit $t_i \in s_i^{\mathcal{T}}$ das kleinste System von Mengen mit der folgenden Eigenschaft ist:

[7] K für **K**lauselmenge.

[8] $\mathcal{P}(M)$ bezeichne die Potenzmenge von M.

Für jede Klausel

$$t \texttt{ in } w_0 \texttt{ for } x_1 \texttt{ in } w_1 \texttt{, ..., } x_n \texttt{ in } w_n \texttt{ with } \Phi\texttt{.}$$

in K mit Termbezeichnern $x_i \in TVar_{s_i}$, einem Term $t \in T(\Sigma, TVar)_{s_0}$, instantiierten Termmengenbezeichnern $w_i = v_i(p_{i1}, \ldots, p_{im_i})$, Termmengenbezeichnern $v_i : s_{i1}, \ldots, s_{im_i} \to s_i$ und Termen $p_{ij} \in T(\Sigma, TVar)_{s_{ij}}$, sowie für jede Belegung der Termbezeichner $\tau = (\tau_s)_{s \in S}$, $\tau_s : TVar_s \to s^{\mathcal{T}}$ gilt:

Wenn für alle $i \in \{1, \ldots, n\}$: $\tau(x_i) \in v_i^{K,\mathcal{A}}(p_{i1}^{\mathcal{T},\tau}, \ldots, p_{im_i}^{\mathcal{T},\tau})$ und $\mathcal{A}, \tau^{\mathcal{A}} \models \Phi$, dann gilt $t^{\mathcal{T},\tau} \in v_0^{K,\mathcal{A}}(p_{01}^{\mathcal{T},\tau}, \ldots, p_{0m_0}^{\mathcal{T},\tau})$. Dabei soll $\tau^{\mathcal{A}}(x) := (\tau(x))^{\mathcal{A}}$ sein für jeden Termbezeichner x.

Mit dieser Definition ist die Zuordnung von $\mathcal{A}$ zu $v^{K,\mathcal{A}}()$ für eine gegebene Klauselmenge K und einen (parameterlosen) Termmengenbezeichner $v \in TMVar_{\lambda,s}$ eine Termmengenfunktion, da die Semantik für isomorphe Algebren die gleichen Termmengen liefert.

4.3 Beispiele

Die in Abschnitt 3.3 definierten Termmengen können mit Hilfe des angegebenen Formalismus beschrieben werden:

4.3.1 nat+pred

Wir wählen $TMVar := \{\texttt{U}\}$, $TVar := \{\texttt{t}\}$. Die Termmengenbeschreibung K enthalte die Klauseln:

```
0    in U.
t +1 in U for t in U.
t -1 in U for t in U with t ≠ 0.
```

Dann ist $\texttt{U}^{K,\mathcal{A}} = \mathcal{U}(\mathcal{A})$ für alle $\mathcal{A} \in \mathrm{Gen}(\mathrm{nat} + \mathrm{pred})$.

4.3.2 store

Wir wählen die folgenden Bezeichner:

$TMVar_{\lambda,\mathrm{elem}} := \{\texttt{Velem,Uelem}\}$
$TMVar_{\lambda,\mathrm{idx}} := \{\texttt{Vidx}\}$
$TMVar_{\lambda,\mathrm{store}} := \{\texttt{ST0}\}$
$TMVar_{(\mathrm{idx}),\mathrm{store}} := \{\texttt{ST}\}$
$TVar_{\mathrm{elem}} := \{\texttt{x}\}$
$TVar_{\mathrm{idx}} := \{\texttt{i,j}\}$
$TVar_{\mathrm{store}} := \{\texttt{s}\}$

Die Termmengenbeschreibung K enthalte die Klauseln

```
fst-elem      in Velem.
nxt-elem(x)   in Velem for x in Velem.

fst-idx       in Vidx.
nxt-idx(i)    in Vidx for i in Vidx.

@             in STO.
put(i,x,s)    in STO for i in Vidx, x in Uelem, s in STO.

put(i,x,s)    in ST(i) for i in Vidx, x in Uelem, s in STO.
put(i,x,s)    in ST(j) for i in Vidx, x in Uelem, s in ST(j)
                       with i=j.

x             in Uelem for x in Velem.
get(i,s)      in Uelem for i in Vidx, s in ST(i).
```

Dann ist für alle $\mathcal{A} \in \text{Gen(store)}$:
$\texttt{Velem}^{K,\mathcal{A}} = T(\Sigma_0)_{\text{elem}}$
$\texttt{Vidx}^{K,\mathcal{A}} = T(\Sigma_0)_{\text{idx}}$
$\texttt{STO}^{K,\mathcal{A}} = T(\Sigma)_{\text{store}}$
$\texttt{ST}^{K,\mathcal{A}}(t) = ST(t^{\mathcal{A}})$
$\texttt{Uelem}^{K,\mathcal{A}} = \mathcal{U}(\mathcal{A})_{\text{elem}}$

Hier ist zu beachten, daß die Formel `i=j` nicht die Gleichheit der Terme bedeutet. Sie wird in $\mathcal{A}$ ausgewertet und bedeutet also $\texttt{i}^{\mathcal{A}} = \texttt{j}^{\mathcal{A}}$.

4.3.3 ord-list

Wir wählen
$TMVar_{\lambda,\text{elem}} := \{\texttt{Velem,Uelem}\}$,
$TMVar_{\lambda,\text{oli}} := \{\texttt{Uoli}\}$
$TVar_{\text{elem}} := \{\texttt{e,e1}\}$
$TVar_{\text{oli}} := \{\texttt{l}\}$

K enthalte die Klauseln

```
fst-elem    in Velem.
nxt-elem(e) in Velem for e in Velem.

e <e e1     in Vbool for e in Velem, e1 in Velem.

@      in Uoli.
e .< l in Uoli for e in Uelem, l in Uoli with e <e car(l).
cdr(l) in Uoli for l in Uoli with l≠@.

e      in Uelem for e in Velem.
car(l) in Uelem for l in Uoli with l≠@.
```

Dann ist für alle $\mathcal{A} \in \text{Gen(store)}$:
$\texttt{Velem}^{K,\mathcal{A}} = T(\Sigma_0)_{\text{elem}}$
$\texttt{Uelem}^{K,\mathcal{A}} = \mathcal{U}(\mathcal{A})_{\text{elem}}$
$\texttt{Uoli}^{K,\mathcal{A}} = \mathcal{U}(\mathcal{A})_{\text{oli}}$

4.3.4 pred-on-zero-and-succ

Wir wählen $TMVar := \{\mathtt{U}\}$, $TVar := \{\mathtt{t}\}$. K enthalte die Klauseln

```
0    in U.
n-1  in U with n-1 +1 = 0.
t +1 in U for t in U.
t -1 in U for t in U with t ≠ 0 ∨ n-1 +1 = 0.
```

Dann ist $\mathtt{U}^{K,\mathcal{A}} = \mathcal{U}(\mathcal{A})$ für alle $\mathcal{A} \in \mathrm{Gen}(\mathrm{nat{+}pred})$.

5 Nachweis partieller Monomorphie

In diesem Abschnitt soll eine Methode entwickelt werden, um die partielle Monomorphie von algebraischen Spezifikationen formal nachzuweisen. Die Methode ist eine Erweiterung der in [Sch95a] vorgestellten, die dem Nachweis der (totalen) Monomorphie dient.

5.1 Beweisverpflichtungen

Um Aussagen über das Verhältnis zweier Algebren zueinander machen zu können, werden Signatur und Spezifikation verdoppelt.

5.1.1 Definition (Σ',SP')
Sei $\Sigma = (S, F)$ eine Signatur. Dann bezeichnet $\Sigma' := (S', F')$ eine dazu disjunkte Signatur mit Sorten $S' = \{s' | s \in S\}$ und Funktionen $F' = (F'_{w',s'})_{(w',s') \in (S')^* \times S'}$ wobei $F'_{(s'_1,\ldots,s'_n),s'} = \{f' | f \in F_{(s_1,\ldots,s_n),s}\}$. Mit den Abbildungen $\sigma_{\mathrm{sorts}}(s) := s'$ und $\sigma_{\mathrm{fcts}}(f) := f'$ ist $\sigma = (\sigma_{\mathrm{sorts}}, \sigma_{\mathrm{fcts}}) : \Sigma \to \Sigma'$ ein Signaturmorphismus. Ist $SP \in \mathrm{Spec}(\Sigma)$ eine Spezifikation, so bezeichnet SP' die Spezifikation $\sigma(SP) \in \mathrm{Spec}(\Sigma')$.

Da die Definition der partiellen Monomorphie Termmengen benutzt, wird außerdem die Σ-Termalgebra benötigt:

5.1.2 Definition (Σ^t,SP^t)
Sei $\Sigma = (S, F)$ eine Signatur. Dann bezeichnet $\Sigma^t := (S^t, F^t)$ eine dazu disjunkte Signatur mit Sorten $S^t = \{s^t | s \in S\}$ und Funktionen $F^t = (F^t_{w^t,s^t})_{(w^t,s^t) \in (S^t)^* \times S^t}$ wobei $F^t_{(s^t_1,\ldots,s^t_n),s^t} = \{f^t | f \in F_{(s_1,\ldots,s_n),s}\}$.

$SP^t \in \mathrm{Spec}(\Sigma^t)$ sei die Spezifikation der freien Termalgebra über Σ^t, d.h. $SP^t = (\Sigma^t, \{\mathcal{A} \mid \mathcal{A} \simeq T(\Sigma^t)\})$

5.1.3 Definition (Σ^+,SP^+)
Sei $\Sigma = (S, F)$ eine Signatur. Es wird definiert: $\Sigma^+ := \Sigma \oplus \Sigma' \oplus \Sigma^t$. $SP^+ := SP \oplus SP' \oplus SP^t \in \mathrm{Spec}(\Sigma^+)$.

$\iota : \Sigma \to \Sigma^+$ und $\iota' : \Sigma' \to \Sigma^+$ seien die durch Inklusion definierten Signaturmorphismen.

Für $\mathcal{A}^+ \in \mathrm{Gen}(\Sigma^+)$ seien $\mathcal{A} := \mathcal{A}^+|_\iota$ und $\mathcal{A}' := \mathcal{A}^+|_{\iota'}$.

Die Beweisverpflichtungen für die partielle Monomorphie erfordern nun, daß in der zugrundeliegenden Logik Formeln existieren, die das Auftreten eines Terms in einer bestimmten Termmenge charakterisieren. Hier kommen also weitere Anforderungen an die Formelsprache hinzu, zusätzlich zu den Minimalanforderungen aus Abschnitt 2.5. Insbesondere wird in der nächsten Definition die Einbettung der Prädikatenlogik für die Formeln $FIsom(\mathcal{U})$ erforderlich.

5.1.4 Definition ($in_{\mathcal{U},s}$, $in'_{\mathcal{U},s'}$, val_s, $val'_{s'}$, $FIsom(\mathcal{U})$)
Für eine gegebene Termmengenfunktion $\mathcal{U} = (\mathcal{U}_s)_{s\in S}$ mit $\mathcal{U}_s : \mathrm{Gen}(\Sigma) \to \mathcal{T}(\Sigma)$ seien $in_{\mathcal{U},s}, in'_{\mathcal{U},s'} \in \mathrm{Fml}(\Sigma^+, \{\tau_s\})$ Formeln mit einer freien Variable τ_s der Sorte s^t, so daß für alle $\mathcal{A}^+ \in \mathrm{Gen}(SP^+)$ gilt $\mathcal{A}^+, [\tau_s := t^t] \models in_{\mathcal{U},s}$ für $t \in T(\Sigma)_s$ genau dann, wenn $t \in \mathcal{U}(\mathcal{A})_s$ und $\mathcal{A}^+, [\tau_s := t^t] \models in'_{\mathcal{U},s'}$ für $t \in T(\Sigma)_s$ genau dann, wenn $t \in \mathcal{U}(\mathcal{A}'|_\sigma)_s$.

Ferner sei $val_s \in \mathrm{Fml}(\Sigma^+, \{\tau_s, x_s\})$ eine Formel mit freien Variablen τ_s der Sorte s^t und x_s der Sorte s, so daß $\mathcal{A}^+, [\tau_s := t^t, x_s := x] \models val_s$ für $t \in T(\Sigma)_s$ und $x \in s^{\mathcal{A}}$ genau dann, wenn $t^{\mathcal{A}} = x$, sowie $val'_{s'} \in \mathrm{Fml}(\Sigma^+, \{\tau_s, x'_{s'}\})$ eine Formel mit freien Variablen τ_s der Sorte s^t und $x'_{s'}$ der Sorte s', so daß $\mathcal{A}^+, [\tau_s := t^t, x'_{s'} := x'] \models val'_{s'}$ für $t \in T(\Sigma)_s$ und $x' \in s'^{\mathcal{A}'}$ genau dann, wenn $\sigma(t)^{\mathcal{A}'} = x'$.

Schließlich wird die Formel

$$\begin{aligned} FIsom(\mathcal{U}) := \forall x, y, x', y', t_1, t_2. \quad & \bigwedge_{s\in S} (in_{\mathcal{U},s}(t_1) \leftrightarrow in'_{\mathcal{U},s'}(t_1)) \\ \wedge & \bigwedge_{s\in S} (in_{\mathcal{U},s}(t_1) \wedge in_{\mathcal{U},s}(t_2) \\ & \quad \wedge val_s(t_1, x) \wedge val_s(t_2, y) \\ & \quad \wedge val'_{s'}(t_1, x') \wedge val'_{s'}(t_2, y') \\ & \quad \to (x = y \leftrightarrow x' = y')) \end{aligned}$$

definiert.

Wenn der im vorangehenden Abschnitt beschriebene Termmengenbeschreibungs-Formalismus verwendet wird, können in dynamischer Logik etwa Programme uniform in den Klauselmengen angegeben werden, die als mögliche Ausgaben gerade die Terme aus den Interpretationen der jeweiligen Termmengenbezeichner besitzen. Dann können Programmformeln zur Definition der $in_{\mathcal{U},s}$ genommen werden, siehe Abschnitt 6.1.

Mit den definierten Formeln ergibt sich:

5.1.5 Lemma ($\mathcal{A} \overset{\mathcal{U}}{\simeq} \mathcal{A}'|_\sigma$ **mittels** $FIsom(\mathcal{U})$)
Seien $\Sigma = (S, F)$, $\mathcal{U} = (\mathcal{U}_s)_{s\in S}$ *mit* $\mathcal{U}_s : \mathrm{Gen}(\Sigma) \to \mathcal{T}(\Sigma)$ *und* $\mathcal{A}^+ \in \mathrm{Gen}(SP^+)$. *Dann gilt:* $\mathcal{A} \overset{\mathcal{U}}{\simeq} \mathcal{A}'|_\sigma$ *genau dann, wenn* $\mathcal{A}^+ \models FIsom(\mathcal{U})$.

Beweis. Der erste Teil der Konjunktion in $FIsom(\mathcal{U})$ entspricht der für die Isomorphie nötigen Gleichheit der Termmengen:

$\mathcal{U}(\mathcal{A}) = \mathcal{U}(\mathcal{A}'|_\sigma) =: U.$

$\Leftrightarrow$ $t \in \mathcal{U}(\mathcal{A})_s$ gdw. $t \in \mathcal{U}(\mathcal{A}'|_\sigma)_s$.

$\Leftrightarrow$ $\mathcal{A}^+, [\tau_s := t^t] \models in_{\mathcal{U},s}$ gdw. $\mathcal{A}^+, [\tau_s := t^t] \models in'_{\mathcal{U},s'}$

$\Leftrightarrow$ Gültigkeit des ersten Teils der Konjunktion.

Dem zweiten Teil entspricht die Existenz des Isomorphismus:

$I : U^{\mathcal{A}} \to U^{\mathcal{A}'|_\sigma}$ mit $I(t^{\mathcal{A}}) = t^{\mathcal{A}'|_\sigma}$ wohldefiniert und bijektiv

$\Leftrightarrow$ $\bar{I} : U^{\mathcal{A}} \to U^{\mathcal{A}'}$ mit $\bar{I}(t^{\mathcal{A}}) = \sigma(t)^{\mathcal{A}'}$ wohldefiniert und bijektiv

$\Leftrightarrow$ Falls $t_1, t_2 \in U$, $x = t_1^{\mathcal{A}}$, $x' = \sigma(t_1)^{\mathcal{A}'}$, $y = t_2^{\mathcal{A}}$ und $y' = \sigma(t_2)^{\mathcal{A}'}$, gilt $x = y$ genau dann, wenn $x' = y'$.

$\Leftrightarrow$ Falls $\mathcal{A}^+, [\tau_s := t_1^t] \models in_{\mathcal{U},s}$, $\mathcal{A}^+, [\tau_s := t_2^t] \models in_{\mathcal{U},s}$,
$\mathcal{A}^+, [\tau_s := t_1^t, x_s := x] \models val_s$, $\mathcal{A}^+, [\tau_s := t_1^t, x'_{s'} := x'] \models val'_{s'}$,
$\mathcal{A}^+, [\tau_s := t_2^t, y_s := y] \models val_s$ und $\mathcal{A}^+, [\tau_s := t_2^t, y'_{s'} := y'] \models val'_{s'}$,
gilt $\mathcal{A}^+ \models (x = y \leftrightarrow x' = y')$

$\Leftrightarrow$ Gültigkeit des zweiten Teils der Konjunktion.

■

5.1.6 Satz (Beweisverpflichtung)
Seien $\Sigma = (S, F)$, $\mathcal{U} = (\mathcal{U}_s)_{s \in S}, \mathcal{V} = (\mathcal{V}_s)_{s \in S}$ *mit* $\mathcal{U}_s, \mathcal{V}_s : \mathrm{Gen}(\Sigma) \to \mathcal{T}(\Sigma)$ *und* $SP \in \mathrm{Spec}(\Sigma)$*. Dann gilt: SP ist monomorph auf* $\mathcal{U}$ *relativ zu* $\mathcal{V}$ *genau dann, wenn* $\mathcal{A}^+ \models (FIsom(\mathcal{V}) \to FIsom(\mathcal{U}))$ *für alle* $\mathcal{A}^+ \in \mathrm{Gen}(\Sigma^+)$

Beweis. Der Satz folgt aus Lemma 5.1.5, unter Beachtung der Definitionen 2.4.3 und 5.1.3:

SP ist monomorph auf $\mathcal{U}$ relativ zu $\mathcal{V}$

$\Leftrightarrow$ Für alle $\mathcal{A}, \mathcal{B} \in \mathrm{Gen}(SP)$ mit $\mathcal{A} \overset{\mathcal{V}}{\simeq} \mathcal{B}$ gilt $\mathcal{A} \overset{\mathcal{U}}{\simeq} \mathcal{B}$.

$\Leftrightarrow$ Für alle $\mathcal{A}^+ \in \mathrm{Gen}(SP^+)$ mit $\mathcal{A} \overset{\mathcal{V}}{\simeq} \mathcal{A}'|_\sigma$ gilt $\mathcal{A} \overset{\mathcal{U}}{\simeq} \mathcal{A}'|_\sigma$.

$\Leftrightarrow$ Für alle $\mathcal{A}^+ \in \mathrm{Gen}(SP^+)$ mit $\mathcal{A}^+ \models FIsom(\mathcal{V})$ gilt $\mathcal{A}^+ \models FIsom(\mathcal{U})$

$\Leftrightarrow$ Für alle $\mathcal{A}^+ \in \mathrm{Gen}(SP^+)$ gilt $\mathcal{A}^+ \models (FIsom(\mathcal{V}) \to FIsom(\mathcal{U}))$

■

5.2 Beweisheuristiken

Hier sollen einige Beweisheuristiken erläutert werde, die sich beim formalen Beweisen der im letzten Abschnitt eingeführten Beweisverpflichtungen bewährt haben.

1. Aus Symmetriegründen reicht es, an Stelle der beiden Äquivalenzen in $FIsom(\mathcal{U})$ jeweils nur Implikationen von links nach rechts zu zeigen.

2. Die beiden Teile der Konjunktion von $FIsom(\mathcal{U})$ können getrennt aus der Prämisse $FIsom(\mathcal{V})$ abgeleitet werden. Dabei kann der zweite Teil, der die Existenz des Isomorphismus zeigt, häufig im Beweis des ersten Teils, also der Gleichheit der Termmengen verwendet werden.

3. Häufig existieren für einige Sorten $S_0 \subseteq S$ Formeln $R_s \in \mathrm{Fml}(\Sigma^+, \{\tau_s\})$ für $s \in S_0$ mit der Eigenschaft, daß wenn $\mathcal{A}^+, [\tau_s := t_1^t] \models in_{\mathcal{U},s}$, mit $t_1 \in T(\Sigma)_s$ auch ein $t_2 \in T(\Sigma)_s$ existiert mit $\mathcal{A}^+, [\tau_s := t_2^t] \models R_s$ und $t_1^{\mathcal{A}} = t_2^{\mathcal{A}}$. Wenn diese Formeln R_s einfacher als die $in_{\mathcal{U},s}$ sind, kann man für diese Sorten den Isomorphienachweis auf diese Formeln reduzieren:

   Man zeigt als Lemma, daß

$$
\begin{array}{l}
\mathcal{A}^+ \models \forall\ x, y, x', y', t_1, t_2. \\
\qquad FIsom(\mathcal{V}) \\
\quad \rightarrow \\
\qquad \bigwedge_{s \in S \setminus S_0} (in_{\mathcal{U},s}(t_1) \wedge in_{\mathcal{U},s}(t_2) \\
\qquad\qquad \wedge val_s(t_1, x) \wedge val_s(t_2, y) \\
\qquad\qquad \wedge val'_{s'}(t_1, x') \wedge val'_{s'}(t_2, y') \\
\qquad\qquad \rightarrow (x = y \rightarrow x' = y')) \\
\quad \wedge \bigwedge_{s \in S_0} ((in_{\mathcal{U},s}(t_1) \wedge val_s(t_1, x)) \rightarrow \\
\qquad\qquad \exists t_2.(R_s(t_2) \wedge val_s(t_2, x))) \\
\quad \wedge \bigwedge_{s \in S_0} (R_s(t_1) \wedge R_s(t_2) \\
\qquad\qquad \wedge val_s(t_1, x) \wedge val_s(t_2, y) \\
\qquad\qquad \wedge val'_{s'}(t_1, x') \wedge val'_{s'}(t_2, y') \\
\qquad\qquad \rightarrow (x = y \rightarrow x' = y'))
\end{array}
$$

   und benutzt dies für den Nachweis des zweiten Teils der Beweisverpflichtung.[9]

   Dieses Verfahren wurde in den beiden folgenden typischen Fällen erfolgreich angewandt:

   (a) Für jedes Objekt einer Sorte $s \in S$ das durch einen Term in U_s erreicht werden kann existiert auch ein Term in V_s, also $U_s^{\mathcal{A}} \subseteq V_s^{\mathcal{A}}$. Dann wird man $R_s := in_{\mathcal{V},s}$ benutzen. Der dritte Teil der Konjunktion im obigen Lemma ergibt sich dann gerade aus $FIsom(\mathcal{V})$.

   (b) Es existiert eine Untermenge von Termen $U_0 \subseteq \mathcal{U}$ mit $U_s^{\mathcal{A}} \subseteq (U_0)_s^{\mathcal{A}}$. Dies wird meist eine Menge von Konstruktortermen sein, die allein alle Objekte erzeugen. Man verwendet dann $R_s := in_{\mathcal{U}_0,s}$. Die Isomorphieeigenschaft für R_s muß dann allerdings immer noch gezeigt werden.

[9] Man beachte, daß die in Punkt 1 angesprochene Vereinfachung hier schon durchgeführt wurde!

4. Es ist manchmal einfacher, partielle Monomorphie auf $\mathcal{U}$ relativ zu $\mathcal{W}$ in mehreren Schritten nachzuweisen. Man hat dann etwa drei Termmengenfunktionen $\mathcal{U}$, $\mathcal{V}$ und $\mathcal{W}$ und zeigt dann zuerst, daß die Spezifikation monomorph auf $\mathcal{V}$ relativ zu $\mathcal{W}$ ist und danach, daß sie monomorph auf $\mathcal{U}$ relativ zu $\mathcal{V}$ ist. Daraus ergibt sich mittels Faktum 3.2.5, daß sie auch monomorph auf $\mathcal{U}$ relativ zu $\mathcal{W}$ ist.

5. Wie die Beweise im einzelnen geführt werden müssen, hängt von der Struktur der Formeln $in_{\mathcal{U},s}$ ab. Wenn diese aus Termmengenbeschreibungen wie sie im Abschnitt 4 eingeführt wurden erzeugt sind, werden sie die rekursive Struktur der Termmengenbeschreibung wiederspiegeln. Man wird dann den Beweis durch strukturelle Induktion über den Term t_1 oder — falls der Kalkül dies ermöglicht — über die Anwendung der Klauseln zu dessen Erzeugung führen. Meistens existiert für jeden Term nur eine mögliche Anwendungsreihenfolge der Klauseln um ihn zu erzeugen. Es sind allerdings auch Termmengenbeschreibungen möglich, in denen der selbe Term für zwei verschiedene Algebren mittels unterschiedlicher Klauseln erzeugt wird. In solchen Fällen müssen andere Heuristiken gefunden werden.

6 Fallstudien in KIV

Das Beweissystem KIV (*Karlsruhe Interactive Verifier*) ermöglicht interaktives formales Beweisen in dynamischer Logik. Termmengenbeschreibungen in Form von Klauselmengen lassen sich uniform in (indeterministische) Programme übersetzen, deren mögliche Ausgaben gerade die in den entsprechenden Termmengen enthaltenen Terme sind. Diese Programme können dann in den in Definition 5.1.4 eingeführten Formeln $in_{\mathcal{U},s}$ verwendet werden. Der folgende Unterabschnitt soll den Übergang zu Beweisverpflichtungen in dynamischer Logik erläutern, was einen weiteren Schritt zur Konkretisierung des Formalismus bedeutet. In 6.2 wird dann die Vorgehensweise in den einzelnen Fallstudien erläutert.

6.1 Beweisverpflichtungen in dynamischer Logik

Die genaue Form der Formeln $in_{\mathcal{U},s}$, $in'_{\mathcal{U},s'}$, val_s und $val'_{s'}$ in Definition 5.1.4 wurde bislang offengelassen. Wird dynamische Logik verwendet, können hier Programmformeln verwendet werden. Auf diesem Weg können beliebige (in den Algebren $\mathcal{A}$, bzw. $\mathcal{A}'$) rekursiv aufzählbare Termmengen behandelt werden.

Werden die im Abschnitt 4 eingeführten Termmengenbeschreibungen verwendet, bietet es sich an, aus den Klauselmengen eine indeterministische Prozedur Pv für jeden Termmengenbezeichner v zu erzeugen, so daß deren Interpretation über einer gegebenen Algebra als mögliche Ausgaben die Terme enthält, die in der Interpretation des Termmengenbezeichners auftreten. Für Terme $t \in T(\Sigma)$ und Algebren $\mathcal{A}^+ \in \mathrm{Gen}(SP^+)$ soll also gelten: $t \in v^{K,\mathcal{A}}$ genau dann, wenn $\mathcal{A}^+ \models \langle \mathrm{P}v(;\mathrm{res})\rangle\, \mathrm{res} = t^t$.

Dabei ist noch zu bedenken, daß bei parametrisierten Termmengenbezeichnern die Parameter an die Prozedur übergeben werden müssen. Außerdem werden je *zwei* Prozeduren für jeden Termmengenbezeichner benötigt, da die Formel $in'_{\mathcal{U},s'}$ sich auf $v^{K,\mathcal{A}'|_\sigma}$ bezieht. Schließlich kann man sich explizite *val* Formeln sparen, indem man die Prozeduren Pv gleich die Auswertung der Terme in $\mathcal{A}$ sowie $\mathcal{A}'$ mit ausgeben läßt.

Dann ergeben sich für jeden Termmengenbezeichner $v : s_1, \ldots, s_n \to s \in$ *TMVar* zwei Prozeduren mit Sortierung
Pv $(s_1, \ldots, s_n)$: (s^t, s, s')
und
Pv' $(s'_1, \ldots, s'_n)$: (s^t, s, s')
mit der Eigenschaft, daß für alle $\mathcal{A}^+ \in \mathrm{Gen}(SP^+)$ und alle Parameterinstantiierungen $p_1 \in T(\Sigma)_{s_1}, \ldots, p_n \in T(\Sigma)_{s_n}$ gilt:

$$\begin{gathered} t \in v^{K,\mathcal{A}}(p_1, \ldots, p_n) \text{ mit } t^{\mathcal{A}} = x \text{ und } \sigma(t)^{\mathcal{A}'} = x' \\ \text{genau dann, wenn} \\ \mathcal{A}^+ \models \left\langle \mathrm{P}v(p_1^t, \ldots, p_n^t; \mathrm{r^t}, \mathrm{r}, \mathrm{r}')\right\rangle \mathrm{r^t} = t^t \wedge \mathrm{r} = x \wedge \mathrm{r}' = x' \end{gathered}$$

und

$$\begin{gathered} t \in v^{K,\mathcal{A}'|_\sigma}(p_1, \ldots, p_n) \text{ mit } t^{\mathcal{A}} = x \text{ und } \sigma(t)^{\mathcal{A}'} = x' \\ \text{genau dann, wenn} \\ \mathcal{A}^+ \models \left\langle \mathrm{P}v'(p_1^t, \ldots, p_n^t; \mathrm{r^t}, \mathrm{r}, \mathrm{r}')\right\rangle \mathrm{r^t} = t^t \wedge \mathrm{r} = x \wedge \mathrm{r}' = x' \end{gathered}$$

Wird nun $\mathcal{U}(\mathcal{A}) = (\mathcal{U}(\mathcal{A})_s)_{s\in S}$ gesetzt mit $\mathcal{U}(\mathcal{A})_s = u_s^{K,\mathcal{A}}$ für entsprechende Termmengenbezeichner $u_s :\to s \in TMVar$, so kann die Formel

$$\begin{aligned} FIsom(\mathcal{U}) := \bigwedge_{s\in S} \forall\, x \in s, x', y' \in s', t \in s^t. & \\ & \left\langle \mathrm{P}u_s(;\mathrm{r^t}, \mathrm{r}, \mathrm{r}')\right\rangle (\mathrm{r} = x \wedge \mathrm{r}' = x') \\ & \wedge \left\langle \mathrm{P}u_s(;\mathrm{r^t}, \mathrm{r}, \mathrm{r}')\right\rangle (\mathrm{r} = x \wedge \mathrm{r}' = y') \\ & \to (x = y \leftrightarrow x' = y') \\ \wedge \quad & \left\langle \mathrm{P}u_s(;\mathrm{r^t}, \mathrm{r}, \mathrm{r}')\right\rangle \mathrm{r^t} = t \\ & \leftrightarrow \left\langle \mathrm{P}u_s{}'(;\mathrm{r^t}, \mathrm{r}, \mathrm{r}')\right\rangle \mathrm{r^t} = t \end{aligned}$$

für die Beweisverpflichtungen genommen werden.

Der Rumpf der Prozeduren Pu und Pu' für $u : s_1, \ldots, s_m \to s \in TMVar$ besteht jeweils aus einer **alternative**-Anweisung. Es wird eine Alternative erzeugt für jede Klausel

t `in` w_0 `for` x_1 `in` w_1`,` ...`,` x_n `in` w_n `with` Φ.

in K mit Termbezeichnern $x_i \in TVar_{s_i}$, einem Term $t \in T(\Sigma, TVar)_{s_0}$, instantiierten Termmengenbezeichnern $w_i = u_i(p_{i1}, \ldots, p_{im_i})$, Termmengenbezeichnern $u_i : s_{i1}, \ldots, s_{im_i} \to s_i$ und Termen $p_{ij} \in T(\Sigma, TVar)_{s_{ij}}$, die zur Beschreibung von u gehört, d.h. $u_0 = u$. Die Form der Alternative ist in Pu und Pu' leicht unterschiedlich und aus den Abbildungen 5 und 6 zu ersehen.

Abbildung 5: Struktur von $\mathrm{P}u(\mathrm{i}_1, \ldots, \mathrm{i}_m;\ \mathbf{var}\ \mathrm{r}^{\mathrm{t}}, \mathrm{r}, \mathrm{r}')$

```
Pu(i_1, ..., i_m; var r^t, r, r')
begin
  alternative
    ...
  or
      begin
        --  Belegung für alle vorkommenden Termbezeichner
        ...
        Ts(; x^t, x, x');
        ...
        --  Elemente der Termmengen
        ...
        Pu_i(p_i1, ..., p_im_i; v_i^t, v_i, v_i');
        ...
        if  v_1 = x_1 ∧ ··· ∧ v_n = x_n          --  Enthaltensein
          ∧ i_1 = p_01 ∧ ··· ∧ i_m0 = p_0m0      --  Parameter
          ∧ Φ  then
          r^t, r, r' := t^t, t, t'               --  Resultat setzen
        else
          abort
      end
  or
    ...
end
```

Die dabei benutzten Hilfsprozeduren $\mathrm{T}s$ liefern jeweils einen beliebigen Grundterm des Typs s, sowie dessen Auswertung in $\mathcal{A}$ und $\mathcal{A}'$ zurück. Ihre Definition ist in Abbildung 7 zu sehen.

6.2 Beweise in KIV

Um die Abschnitt 3.3 erwähnten Monomorphiebeweise zu führen, wurde in KIV jeweils die gleiche Projektstruktur aufgebaut. Diese ist in Abbildung 8 wiedergegeben. Dabei enthält SP die eigentliche Spezifikation. SP' ist die Umbenennung durch einen Signaturmorphismus, in dem an jeden Bezeichner ein Strich angehängt wird. SP-t ist die Spezifikation der freien Termalgebra über einer Signatur, die durch Anhängen von „-t“ an jeden Bezeichner aus SP entsteht. In + wird dann die Vereinigung der drei gebildet. Die Spezifikation empty ist leer. Sie dient nur als Export für das Modul PM in dem die partielle Monomorphie bewiesen wird.

In dem Modul PM befinden sich die im letzten Abschnitt eingeführten Prozedurdefinitionen und eine Reihe von Lemmata über diese, die die Beweisverpflichtungen für die partielle Monomorphie bilden. Dabei wurden

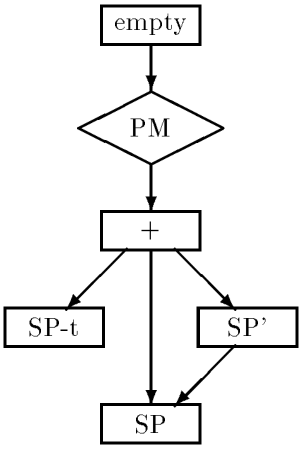
Abbildung 6: Struktur von $\mathrm{P}u'(\mathrm{i}_1, \ldots, \mathrm{i}_m;\ \mathbf{var}\ \mathrm{r}^{\mathrm{t}}, \mathrm{r}, \mathrm{r}')$

```
Pu′(i₁, ..., iₘ; var rᵗ, r, r′)
begin
  alternative
    ...
  or
      begin
        --   Belegung für alle vorkommenden Termbezeichner
        ...
        Ts(; xᵗ, x, x′);
        ...
        --   Elemente der Termmengen
        ...
        Puᵢ′(p_i1, ..., p_imᵢ; vᵢᵗ, vᵢ, vᵢ′);
        ...
        if   v₁′ = x₁′ ∧ ··· ∧ vₙ′ = xₙ′           --  Enthaltensein
          ∧ i₁ = p_01 ∧ ··· ∧ i_m0 = p_0m0       --  Parameter
          ∧ σ(Φ)   then
          rᵗ, r, r′ := tᵗ, t, t′                --  Resultat setzen
        else
          abort
      end
  or
    ...
end
```

Abbildung 7: Struktur von $\mathrm{T}s(;\mathbf{var}\ \mathrm{x}^{\mathrm{t}}, \mathrm{x}, \mathrm{x}')$

```
Ts(; var xᵗ, x, x′)
begin
  alternative
    ··· --   eine Alternative pro f : s₁, ..., sₙ → s ∈ F
  or
      begin
        --   Erzeugen der Unterterme
        ...
        Tsᵢ(; xᵢᵗ, xᵢ, xᵢ′);
        ...
        --   Term zusammenbauen & auswerten
        xᵗ := fᵗ(x₁ᵗ, ..., xₙᵗ);
        x := f(x₁, ..., xₙ);
        x′ := f′(x₁′, ..., xₙ′)
      end
  or
    ...
end
```

Abbildung 8: Projektstruktur für Monomorphiebeweise in KIV



die Beweisverpflichtungen gemäß den Punkten 1 und 2 in Abschnitt 5.2 vereinfacht. Teilweise wurden die Beweisverpflichtungen zusätzlich nach den verschiedenen Sorten aufgespalten, um sie übersichtlicher zu machen. Bei gegenseitig rekursiven Prozeduren mußten sie allerdings doch zusammen mit einer einzigen Induktion bewiesen werden.

In den nächsten Abschnitten folgt eine detailliertere Beschreibung der einzelnen Fallstudien zusammen mit einigen statistischen Daten über den Aufwand zu deren Durchführung.

6.2.1 nat+pred

Diese Fallstudie ist die kleinste, da die Monomorphie hier nicht relativ ist, also nur eine einzige Termmenge nötig war. Das Beweisen hat weniger als eine Stunde Arbeit und insgesamt 28 Interaktionen erfordert. Davon wurden 7 zum Beweis eines Reduktionslemmas im Sinne von Abschnitt 5.2 Punkt 3b benötigt. In Abwandlung des dort erwähnten Prinzips wurde eine deterministische Prozedur geschrieben, die wie folgt aussieht:

$\mathrm{DM}(\mathrm{r}; \textbf{var } \mathrm{r}')$
begin
  if $\mathrm{r} = .0$ **then**
    $\mathrm{r}' := .0'$
  else
    begin
      $\mathrm{DM}(\mathrm{r}. - 1; \mathrm{r}');$
      $\mathrm{r}' := \mathrm{r}'. + 1'$
    end
end

Nun reicht es zu zeigen, daß wann immer ein Term, der von PUnat ermittelt wird, in $\mathcal{A}$ zu n, in $\mathcal{A}'$ aber zu n' ausgewertet wird, die Prozedur DM bei Eingabe n als Ergebnis n' liefert. Dieses Lemma wurde durch Induktion über die Prozeduraufrufe von PUnat bewiesen. Dabei ergibt sich je ein Fall für jede der drei Alternativen in der Prozedur PUnat. Wird ein 0-Term erzeugt, so sieht man ohne weiteres, daß DM in den Basisfall läuft. Bei einem +1-Term wird die Induktionshypothese angewandt, woraus folgt, daß DM für einen Wert n die Ausgabe n' liefert, und zu zeigen bleibt, daß dann auch $n+1$ die Ausgabe $n'+1'$ liefert. Aufgrund der Axiome ist $n+1 \neq 0$ und $n+1-1=n$. Der rekursive Aufruf erhält also n als Parameter, liefert somit n' (da DM deterministisch ist) und es wird tatsächlich $n'+1'$ zurückgegeben. Im dritten Fall erzeugt PUnat einen −1-Term. Der Term wird also zu $n-1$ bzw. $n'-1'$ ausgewertet, wobei die Induktionshypothese wieder liefert, daß DM bei Eingabe n n' ausgibt. Des weiteren wird wegen der Formel $\Phi = \mathtt{t} \neq \mathtt{0}$ in der Termmengenbeschreibung in PUnat sichergestellt, daß $n \neq 0$. Zu zeigen ist, daß DM bei Eingabe $n-1$ auch $n'-1'$ liefert. Dies macht man sich klar, indem man den Aufruf DM(n)$\rightarrow n'$ betrachtet: wegen $n \neq 0$ wird DM rekursiv mit $n-1$ aufgerufen und liefert einen Wert m', der inkrementiert und ausgegeben wird, für den also $m'+1' = n'$ gelten muß. Laut Axiomen ist also $n'-1' = (m'+1')-1' = m'$ und der Beweis ist fertig. Diese Schlüsse wurden größtenteils automatisch gezogen, lediglich Induktion, Fallunterscheidung, Anwendung der Induktionshypothese und Aufrufe der Prozeduren erforderten Interaktionen.

Dieses Lemma liefert die Reduktion. Der Isomorphieteil für DM folgt sofort, da DM deterministisch ist. Die Beweise in den Fallstudien laufen alle nach diesem Schema von Induktion und Fallunterscheidung, wobei die Details allerdings variieren.

Das Lemma für die Gleichheit der Termmengen wird ebenfalls durch Induktion über die Rekursionstiefe des PUnat-Aufrufs und Fallunterscheidung gezeigt. Im ersten Fall muß gezeigt werden, daß es einen Aufruf von PUnat' gibt, der den 0-Term liefert, was einfach ist. Im zweiten Fall ist ein Aufruf von PUnat gegeben, der einen Term t liefert und laut Induktionshypothese gibt es auch einen solchen Aufruf von PUnat'. Zu zeigen ist, daß es einen Aufruf von PUnat' gibt, der den Term $t+1^t$ liefert, was ebenso einfach ist. Im letzten fall muß gezeigt werden, daß es einen Aufruf von PUnat' gibt, der den Term $t-1^t$ liefert, wobei zusätzlich $t^{\mathcal{A}} \neq 0$ gegeben ist. Wenn man die dritte Alternative im Aufruf von PUnat' betrachtet, muß noch gezeigt werden, daß auch $t^{\mathcal{A}'|_\sigma} \neq 0$ gilt. Aber im Reduktionslemma wurde gezeigt, daß DM bei Eingabe von $t^{\mathcal{A}}$ als Ausgabe $t^{\mathcal{A}'|_\sigma}$ liefert, und DM kann bei Eingabe 0 nur 0' ausgeben.

Dieser Beweis ist typisch für alle durchgeführten Termmengen-Gleichheitsbeweise. Im wesentlichen wird gezeigt, daß es einen Aufruf von Pu' gibt, der dem selben Pfad folgt wie der von Pu'. In den durchgeführten

Fallstudien gab es für jeden erzeugten Term immer nur einen möglichen Berechnungspfad. Es mußte also nur noch gezeigt werden, daß aus der Gültigkeit der Formeln Φ jeweils auch die der entsprechenden $\sigma(\Phi)$ folgt.

6.2.2 store

Die store-Fallstudie war etwas aufwendiger. Die Beweise haben insgesamt 120 Interaktionen erfordert und etwa zwei bis drei Stunden gebraucht. Isomorphiebedingung und Termmengen-Gleichheit wurden mit je einem Lemma bewiesen, das eine stärkere Aussage für die Induktion beinhaltete (Generalisierung).

Das Reduktionslemma für die Isomorphie sieht wie folgt aus:

$$
\begin{array}{l}
\quad FIsom(\mathcal{V}) \\
\rightarrow \\
\qquad (\quad \langle \kappa \ \# \ \text{PUelem(;re-t, re, re')} \rangle (\text{re} = \text{x} \wedge \text{re'} = \text{x'}) \\
\qquad\ \ \rightarrow \langle \text{PVelem(;re-t, re, re')} \rangle (\text{re} = \text{x} \wedge \text{re'} = \text{x'})) \\
\quad \wedge \ (\qquad \langle \kappa \ \# \ \text{PST(par;rs-t, rs, rs')} \rangle (\text{rs} = \text{s} \wedge \text{rs'} = \text{s'}) \\
\qquad\qquad \wedge \ \langle \text{PVidx(;ri-t, ri, ri')} \rangle (\text{ri-t} = \text{par} \wedge \text{ri} = \text{i} \wedge \text{ri'} = \text{i'}) \\
\qquad\ \ \rightarrow \langle \text{PVelem(;re-t, re, re')} \rangle (\text{re} = \text{get(i, s)} \wedge \text{re'} = \text{get'(i', s')}))
\end{array}
$$

Der erste Teil der Konjunktion ist gerade die Reduktion von $\mathcal{U}_{\text{elem}}$ auf $\mathcal{V}_{\text{elem}}$. Der zweite Teil behauptet, daß in $\mathtt{ST}^{K,\mathcal{A}}(t_i)$ nur solche Terme t_s liegen, daß auch Terme t_x in $\mathcal{V}_{\text{elem}}$ liegen mit $t_x^{\mathcal{A}} = get^{\mathcal{A}}(t_i^{\mathcal{A}}, t_s^{\mathcal{A}})$ und $t_x^{\mathcal{A}'|_\sigma} = get^{\mathcal{A}'|_\sigma}(t_i^{\mathcal{A}'|_\sigma}, t_s^{\mathcal{A}'|_\sigma})$. Diese beiden Behauptungen benötigen sich beim induktiven Beweis über die Rekursionstiefe κ gegenseitig. Der Beweis für dieses Lemma benötigte 41 Interaktionen.

Die erste Behauptung ist klar, falls der zur Klausel

```
x in Uelem for x in Velem.
```

gehörende Zweig von PUelem betreten wird. Im anderen Zweig wird ein Index ausgewählt und PST mit diesem aufgerufen. Es wird die Induktionshypothese angewandt, und zwar der zweite Teil der Konjunktion.

Für die zweite Behauptung ist zum einen der zur Klausel

```
put(i,x,s) in ST(i) for i in Vidx, x in Uelem, s in STO.
```

gehörende Fall zu betrachten. Hier erfolgen Aufrufe von PVidx, PUelem und PST0. Des weiteren ist der Index vom PVidx-Aufruf der selbe wie der in der Behauptung, da der Term der gleiche ist. Da das von PUelem gelieferte Element an der durch den Index angegebenen Stelle in den store geschrieben wird, liefern die gets in der Behauptung gerade die Auswertungen, die PUelem zurückgegeben hat. Laut erstem Teil der Induktionshypothese existiert ein zu dem Aufruf von PUelem gehörender Aufruf von PVelem.

Bleibt noch der Fall für die Klausel

```
put(i,x,s) in ST(j) for i in Vidx, x in Uelem, s in ST(j)
                        with i=j.
```

Hier muß unterschieden werden, ob der Index in der Behauptung der selbe ist wie der, der in PST gesetzt wird oder nicht. Im ersten Fall ist die Situation wie im Zweig für die erste Klausel. Sonst wird der zweite Teil der Induktionshypothese benutzt, zusammen mit dem zweiten Axiom, das besagt, daß der Wert vom get derselbe ist wie der auf dem vom rekursiven Aufruf von PST gelieferten store.

Die Beweisverpflichtung der Gleichheit der Termmengen wird ebenfalls mit einem stärkeren Lemma gezeigt, das die Gleichheit der generierten Termmengen bei PUelem , PST0 und PST'(par) behauptet und überdies noch die Gleichheit der Auswertungen:

$$
\begin{array}{l}
FIsom(\mathcal{V}) \\
\rightarrow \\
\quad (\quad \langle \kappa \ \# \ \text{PUelem(;re-t, re, re')} \rangle (\text{re-t} = \text{x-t} \wedge \text{re} = \text{x} \wedge \text{re'} = \text{x'}) \\
\qquad \rightarrow \langle \text{PUelem'(;re-t, re, re')} \rangle (\text{re-t} = \text{x-t} \wedge \text{re} = \text{x} \wedge \text{re'} = \text{x'})) \\
\quad \wedge (\quad \langle \kappa \ \# \ \text{PST0(;rs-t, rs, rs')} \rangle (\text{rs-t} = \text{s-t} \wedge \text{rs} = \text{s} \wedge \text{rs'} = \text{s'}) \\
\qquad \rightarrow \langle \text{PST0'(;rs-t, rs, rs')} \rangle (\text{rs-t} = \text{s-t} \wedge \text{rs} = \text{s} \wedge \text{rs'} = \text{s'})) \\
\quad \wedge (\quad \langle \kappa \ \# \ \text{PST(par;rs-t, rs, rs')} \rangle (\text{rs-t} = \text{s-t} \wedge \text{rs} = \text{s} \wedge \text{rs'} = \text{s'}) \\
\qquad \rightarrow \langle \text{PST'(par;rs-t, rs, rs')} \rangle (\text{rs-t} = \text{s-t} \wedge \text{rs} = \text{s} \wedge \text{rs'} = \text{s'}))
\end{array}
$$

Der Beweis hierzu benötigte 68 Interaktionen. Er beruht auf einer Induktion über die Rekursionstiefe κ und im Induktionsschritt entsteht ein Zweig für jede der drei Behauptungen. Die einzelnen Zweige verlaufen ähnlich wie bei der Beweisverpflichtung für nat+pred, nur führen die gegenseitigen rekursiven Aufrufe zur Anwendung jeweils entsprechender Teile der Induktionshypothese.

Die Vorgehensweise, Induktion über die Rekursionstiefe mehrerer Prozeduren gleichzeitig zu führen wird wohl bei allen Beweisen für mehrere sich gegenseitig referenzierende Termmengen nützlich sein. Im nächsten Beispiel wurde auch so verfahren. Bemerkenswert ist noch der zweite Teil der Konjunktion im Reduktionslemma, die gerade so gewählt ist, daß damit bei Anwendung der Induktionshypothese der erste Teil bewiesen werden kann.

6.2.3 ord-list

In der Fallstudie für die Spezifikation ord-list wurde der in Abschnitt 5.2, Punkt 4 beschriebene Weg eingeschlagen. Die Klauselmenge sah wie folgt aus:

```
fst-elem    in Welem.
nxt-elem(e) in Welem for e in Welem.

e <e e1 in Wbool for e in Welem, e1 in Welem.

@      in PVoli.
e .< l in Voli for e in Welem, l in Voli with e <_e car(l).

@      in Uoli.
e .< l in Uoli for e in Uelem, l in Uoli with e <_e car(l).
cdr(l) in Uoli for l in Uoli with l≠@.

e      in Uelem for e in Welem
car(l) in Uelem for l in Uoli with l≠@.
```

$\mathcal{V}_{\text{elem}}$ enthält also nur geordnete Listen mit Elementen aus $\mathcal{W}_{\text{elem}}$. Dadurch wird die Schwierigkeit der gegenseitigen Rekursion von PUelem und PUoli zunächst verschoben. Im Beweis für die partielle Monomorphie auf $\mathcal{U}$ kann dann $\mathcal{U}_{\text{oli}}$ auf $\mathcal{V}_{\text{oli}}$ und $\mathcal{U}_{\text{elem}}$ auf $\mathcal{W}_{\text{elem}}$ reduziert werden. Die beiden Monomorphiebeweise brauchten zusammen 270 Interaktionen und waren innerhalb eines Tages zu machen.

Für die Isomorphiebedingung von $\mathcal{V}$ waren drei Lemmata nötig. Eines davon besagt, daß PVoli nur vernünftige geordnete Listen l zurückgibt, in dem Sinne, daß wenn $e <_e car(l)$ oder $l = @$, dann auch $e \ll l$. Das zweite ist die entsprechende Aussage für die „gestrichene“ Signatur. Das dritte besagt, daß zwei Aufrufe von PWelem, die die selben Terme liefern auch die selben Auswertungen liefern. Die Beweise gingen alle durch Induktion über die Rekursionstiefe und brauchten zusammen 20 Interaktionen. Der eigentliche Beweis der Isomorphiebedingung brauchte dann noch 25 Interaktionen. Dabei wird wie üblich induziert und gezeigt, daß der Ablauf in den beiden Aufrufen von PVoli genau der selbe ist, was eben deshalb der Fall ist, weil die Elemente alle durch Aufrufe von PWelem entstehen und die Isomorphieeigenschaft für PWelem in der Prämisse steht.

Die Gleichheit der Termmengen erforderte drei weitere Lemmata. Eines besagt, daß gleiche Terme auch bei PVoli zu gleichen Auswertungen führen, wurde durch Zurückführung auf das entsprechende Lemma für PWelem bewiesen und benötigte 23 Interaktionen. Das zweite ist eine besser handhabbare Umformulierung der Isomorphie Aussage für PWbool und benötigte 12 Interaktionen. Das dritte charakterisiert die Eigenschaften von Aufrufen von PVoli und PVoli', die beide den selben Term liefern und besagt, daß dies entweder ein @-Term ist, worauf auch die Auswertungen @, bzw @' werden, oder es gibt einen Aufruf von PWelem, der den car-Unterterm zurückliefert, wobei dieser wiederum zu car bzw. car' der Werte des ursprünglichen Terms ausgewertet wird. Der Beweis dazu erforderte 23 Interaktionen. (Es sollte allerdings ein nicht-induktiver Beweis mit weniger Interaktionen möglich sein.) Der Beweis der Termmengen-Gleichheit selber benötigte dann noch 33 Interaktionen und geht genauso wie üblich.

Für die Monomorphie auf $\mathcal{U}$ wurden die Reduktionen von $\mathcal{U}_{\mathrm{oli}}$ auf $\mathcal{V}_{\mathrm{oli}}$ und $\mathcal{U}_{\mathrm{elem}}$ auf $\mathcal{W}_{\mathrm{elem}}$ benutzt, die wieder durch gleichzeitige Induktion über die Rekursionstiefe gezeigt werden können. Das Reduktionslemma erforderte 67 Interaktionen und das Termmengen-Gleichheitslemma, in dem die Gleichheit der Termmengen für $\mathcal{U}_{\mathrm{oli}}$ und $\mathcal{U}_{\mathrm{elem}}$ auf dem normalen Wege gezeigt werden, weitere 51 Interaktionen.

Insgesamt waren die vielen Interaktionen zu einem Großteil auf das Einfügen von Lemmata (36 mal) zurückzuführen. Dabei waren häufig drei bis vier Intaktionen pro Lemma-Benutzung nötig um die Prämissen zu beweisen, da das System diese automatisch oft auf falschem Wege zu schließen versucht hat. Mit einer Optimierung von KIV zum Behandeln derartiger Beweise könnten die Interaktionen wahrscheinlich auf etwa die Hälfte reduziert werden.

6.2.4 pred-on-zero-and-succ

Diese Fallstudie war wegen der sehr unterschiedlichen Modelle der Spezifikation interessant. Dabei war erstaunlicherweise beim Beweisen niemals eine explizite Fallunterscheidung nach der Existenz eines Vorgängers von 0 nötig. Insgesamt waren für die Beweise 92 Interaktionen notwendig, der Zeitbedarf lag bei etwa drei Stunden.

Da die durch Terme in $\mathcal{U}$ erreichbaren Elemente auch schon durch 0 und +1 erreicht werden können, liegt auch hier ein Beweis durch Reduktion nahe. Dabei wurden zwei Lemmata benötigt, die besagen, daß wenn $\mathcal{A}$ endlich ist, auch $\mathcal{A}'$ endlich ist, und daß in diesem Fall ein Term $t \in \mathcal{V}$ existiert mit $t^{\mathcal{A}} = \mathrm{n}{-}1^{\mathcal{A}}$ und $t^{\mathcal{A}'|_\sigma} = \mathrm{n}{-}1^{\mathcal{A}'|_\sigma}$. Dies wird bewiesen mit Hilfe der Feststellung, daß für Modelle mit n−1 +1=0 ein $t \in \mathcal{V}$ existieren muß mit $t \neq 0^{\mathrm{t}}$ aber $t^{\mathcal{A}} = 0^{\mathcal{A}}$ und $t^{\mathcal{A}'|_\sigma} = 0^{\mathcal{A}'|_\sigma}$. In dem entsprechenden Aufruf von PV muß es dann einen rekursiven Aufruf geben, der einen Term mit Wert n−1, bzw. n−1' liefert. Die Beweise dieser Lemmata benötigten 35 Interaktionen.

Die Beweise für die Reduktion und die Termmengen-Gleichheit stützen sich auf diese Lemmata und brauchen 25 bzw. 29 Interaktionen und unterscheiden sich in der Struktur nicht von den entsprechenden Beweisen in den anderen Fallstudien.

Die spezifischen Besonderheiten finden sich hier (wie auch bei ord-list) in den Lemmata wieder, die für die Hauptbeweise benötigt werden, welche ihrerseits in immer der gleichen Form ablaufen.

7 Diskussion

In dieser Arbeit wurde die Definition eines Begriffs partieller Monomorphie von (abstrakten) Spezifikationen auf Termmengen gegeben. Des weiteren wurde das Konzept der relativen Monomorphie eingeführt. Es wurden notwendige und hinreichende Beweisverpflichtungen angegeben, um diese Eigenschaften von Spezifikationen nachzuweisen und einige Heuristiken vorgeschlagen um diese zu beweisen. Es wurde ein Formalismus zur Beschreibung von Termmengen angegeben, und gezeigt wie man daraus automatisch Beweisverpflichtungen in dynamischer Logik generieren kann. Schließlich wurden die mit Hilfe des interaktiven Beweissystems KIV durchgeführten Fallstudien erläutert.

Ansätze zu einem Begriff partieller Monomorphie und eine Beweistechnik dazu im Rahmen des Spezifikationswerkzeugs Larch ([GH93]) findet man bei [LW96], allerdings wird nur Monomorphie von Nicht-Konstruktoren relativ zu Konstruktoren betrachtet. Ein anderer Ansatz um ähnliche Probleme zu lösen besteht in der Verwendung von partiellen Algebren, was allerdings einige erhebliche Nachteile hat, siehe etwa [GS95]. Der vorgestellte Formalismus zur Termmengenbeschreibung ist eine starke Erweiterung regulärer Baumgrammatiken. Die Erweiterungen betreffen erstens die Einführung von Parametern und zweitens die Verwendung von Formeln, die über einer Algebra interpretiert werden. In anderen Anwendungsbereichen wurden schon Formalismen entwickelt, um Termmengen zu beschreiben, jedoch mit anderen Zielsetzungen. (Abschlußeigenschaften, Entscheidbarkeit, etc. — siehe etwa [CCC+94])

Sowohl theoretisch wie praktisch bleiben auf diesem Gebiet noch Aufgaben: Eine nötige Erweiterung des vorgestellten Monomorphie-Begriffs betrifft parametrisierte Spezifikationen, die den Beispielen ord-list, bzw. store sicherlich angemessener gewesen wären. Auf dem Bereich der Termmengenbeschreibung stellt sich die Frage der Ausdrucksmächtigkeit. Kann hiermit jede benötigte „Spezifikationslücke“ beschrieben werden, bzw. geht dies sogar mit einem einfacheren Beschreibungsformalismus?

Beim Bearbeiten der Fallstudien ergab sich öfters der Wunsch nach Optimierungen. Künftig könnte also entweder an einer Anpassung der Heuristiken des KIV-Systems für derartige Aufgaben, oder aber der Entwicklung eines eigenen Kalküls für Monomorphiebeweise gearbeitet werden. Auch die Beweisverpflichtungen lassen sich in vielen Fällen wahrscheinlich optimieren, etwa indem mit enrichments gearbeitet wird, wobei in der Basisspezifikation die als isomorph gegebenen Teile enthalten sind, wodurch dann nur noch ein Teil der Spezifikation vervielfältigt werden müßte.

Bei alldem ist zu beachten, daß diese Begriffe dazu dienen sollen, die formale Spezifikation sicherer zu machen, ihr die Risiken des Formalen zu nehmen. Ein Begriff von Monomorphie, der sich nicht mit der intuitiven Vorstellung dessen deckt, der ihn benutzt, verschiebt die Unsicherheit der Softwareentwicklung von der Spezifikation auf den Monomorphiebegriff, so wie sie vom Programm zur Spezifikation gewandert ist. Wenn derartige Probleme gelöst werden, könnte die Möglichkeit, erwünschte Eigenschaften von Spezifikationen formal nachzuweisen, das Vertrauen in formale Softwareentwicklungsmethoden erheblich stärken.

Literatur

[CCC+94] Anne-Cécile Caron, Hubert Comon, Jean-Luc Coquidé, Max Dauchet, and Florent Jacquemard. Pumping, cleaning and symbolic constraints solving. In *Proc. ICALP*, volume 820 of *LNCS*, pages 436–449, 1994.

[GH93] J. Guttag and J. Horning. *Larch: Languages and Tools for Formal Specification.* Springer Verlag, 1993.

[GS95] D. Gries and F.B. Schneider. Avoiding the undefined by underspecification. In *Computer Science Today: Recent Trends and Developments*, volume 1000 of *Lecture Notes in Computer Science*, pages 366–373. Springer Verlag, 1995.

[HRS90] M. Heisel, W. Reif, and W. Stephan. Tactical theorem proving in program verification. In *Proceedings of the 10th International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Computer Science*, pages 117–131. Springer Verlag, 1990.

[Jon96] C.B. Jones. Tanstaafl (with partial functions). In *Proceedings of the CADE-13 Workshop on Mechanization of Partial Functions*, 1996.

[LW96] G.T. Leavens and J.M. Wing. Protection from the underspecified. Technical Report CMU-CS-96-129, School of Computer Science, Carnegie Mellon University, 1996.

[Rei95] W. Reif. The KIV-Approach to Software Verification. In M. Broy and S. Jähnichen, editors, *KORSO: Methods, Languages, and Tools for the Construction of Correct Software*, volume 1009 of *Lecture Notes in Computer Science.* Springer Verlag, 1995.

[Sch95a] A. Schönegge. Proof obligations for monomorphicity. Technical Report 34/95, Universität Karlsruhe, Fakultät für Informatik, 1995.

[Sch95b] A. Schönegge. Would you ever risk a non-monomorphic specification? Technical Report 33/95, Universität Karlsruhe, Fakultät für Informatik, 1995.

[Wir90] M. Wirsing. *Algebraic Specification*, volume B of *Handbook of Theoretical Computer Science*, chapter 13, pages 675–788. Elsevier Science Publishers B. V., 1990.