

Beispiel für nicht-abbrechende Vervollständigung:

$$(1) \quad (x + y) + z \quad \rightarrow \quad x + (y + z)$$

$$(2) \quad f(x) + f(y) \quad \rightarrow \quad f(x + y)$$

Lexikographische Pfadordnung mit $+ > f$

Überlappung auf $(f(x) + f(y)) + z$

Kritisches Paar $(f(x) + (f(y) + z), f(x + y) + z)$

führt zu neuer Regel

$$(3) \quad f(x + y) + z \rightarrow f(x) + (f(y) + z)$$

Überlappung von (2) und (3): $f(f(x) + f(y)) + z$

führt zu neuem kritischem Paar und neuer Regel

$$(3^2) \quad f^2(x + y) + z \rightarrow f^2(x) + (f^2(y) + z)$$

... allgemein:

$$(3^n) \quad f^n(x + y) + z \rightarrow f^n(x) + (f^n(y) + z)$$

d.h. es entsteht ein unendliches Regelsystem

Bemerkung: Mit $f > +$ und Umorientierung der 2. Regel terminiert die Vervollständigung sofort.

Einige algebraische Grundbegriffe

Eine (einsortige) *Algebra* ist eine Menge mit Operationen auf der Menge.

Eine mehrsortige Algebra besteht entsprechend aus mehreren Mengen (jeweils eine pro Sorte) und Operationen auf den Mengen.

Eine Σ -*Algebra* ist eine Algebra, deren Operationen der Signatur Σ entsprechen.

D.h. Σ -*Algebren* sind die Strukturen, mit deren Hilfe Interpretationen von Termen und Gleichungen definiert werden.

Wie vorher setzen wir voraus, daß jede der Trägermengen nicht leer ist.

Die Term Mengen $(T_{\Sigma}^{(S)}(V))_{S \in \Sigma}$ bilden eine Σ -Algebra $T_{\Sigma}(V)$, die *Term-Algebra* zu Σ : die den Operationen zugeordneten Funktionen sind gerade die term-bildenden Operationen.

Für eine *sinnvolle* Signatur Σ , d.h. eine Signatur, die mindestens einen Grundterm für jede Sorte zuläßt, bilden auch die Mengen der Grundterme $T_{\Sigma}^{(S)}(\emptyset)$ eine Σ -Termalgebra, die *Grundterm-Algebra* T_{Σ} .

Algebraische Grundbegriffe (2)

Ein Σ -*Homomorphismus* ist eine Abbildung zwischen Σ -Algebren, die die Sorten und Operationen der Signatur respektiert.

Ein Σ -*Isomorphismus* ist ein bijektiver Σ -*Homomorphismus*.

Die zu einer Σ -Algebra \mathcal{A} gehörende Interpretationsfunktion, die Terme auf Elemente der Trägermengen von \mathcal{A} abbildet, ist für jede Variablenbelegung $\mathcal{V} : V \rightarrow \mathcal{A}$ ein Σ -Homomorphismus von $T_\Sigma(V)$ nach \mathcal{A} ; der Homomorphismus ist durch \mathcal{V} und \mathcal{A} eindeutig bestimmt.

Für einen Grundterm t ist das Bild unter dem Homomorphismus bereits durch \mathcal{A} eindeutig bestimmt.

Term-erzeugte Algebren

Satz: Ist Σ eine sinnvolle Signatur, so gilt für die Grundterm-Algebra T_Σ : Zu jeder Σ -Algebra \mathcal{A} gibt es genau einen Homomorphismus von T_Σ nach \mathcal{A} .

(In der Sprache der Kategorientheorie heißt dies, daß T_Σ ein *initiales Objekt* in der Kategorie der Σ -Algebren ist.)

Eine Σ -Algebra \mathcal{A} heißt *term-erzeugt* oder *erreichbar*, wenn es für jedes Element a einer Trägemenge A_S zur Sorte S einen Grundterm $t \in T_\Sigma^{(S)}$ gibt, dessen Interpretation in A_S das Element a ist.

Offensichtlich ist die Grundterm-Algebra T_Σ term-erzeugt.

Satz: Für eine Σ -Algebra \mathcal{A} sind die folgenden Aussagen äquivalent:

- \mathcal{A} ist term-erzeugt.
- Es gibt einen surjektiven Homomorphismus $h : T_\Sigma \rightarrow \mathcal{A}$

Gleichungsspezifizierte Algebren

Die Interpretation einer Σ -Gleichung ist ein *Modell* der Gleichung, wenn die Gleichung unter der Interpretation für *jede* Variablenbelegung erfüllt (d.h. wahr) ist; entsprechend für eine Menge von Gleichungen.

Jeder Interpretation entspricht eine Σ -Algebra (und umgekehrt), somit können wir auch von einer Algebra als einem Modell einer Menge von Gleichungen sprechen.

Eine Signatur Σ zusammen mit einer Menge E von Σ -Gleichungen über Σ kann aufgefaßt werden als *Spezifikation* der Klasse derjenigen Algebren, die Modelle für die Gleichungen sind.

Die Spezifikation repräsentiert die *Theorie* der Klasse von Algebren, wobei die Gleichungen die “Axiome” der Theorie darstellen.

Gleichungsspezifizierte Algebren (2)

Beispiele:

- Die bereits angegebenen Gleichungen definieren *Gruppentheorie* als eine gleichungsspezifizierte algebraische Theorie.
- Analog lassen sich viele andere algebraische Strukturen (Theorien) mit Hilfe von Gleichungen spezifizieren.
- Wichtiger für die Informatik sind Spezifikationen von *algebraischen Datenstrukturen* oder *abstrakten Datentypen* (s. später).

Gleichungstheorien

Semantische Folgerung für Gleichungen:

$$E \models s = t \quad s = t \text{ folgt semantisch aus } E$$

d.h. in jedem Modell der Gleichungsmenge E ist auch die Gleichung $s = t$ erfüllt.

$Th(\Sigma, E)$ – Menge der Gleichungen, die aus E semantisch folgen

d.h. die von E definierte “Gleichungstheorie”.

Ist $Alg(\Sigma, E)$ die Klasse aller Σ -Algebren, die die Gleichungen E erfüllen (d.h. aller (Σ, E) -Algebren), dann ist $Th(\Sigma, E)$ die Menge der Gleichungen, die von allen Algebren $\mathcal{A} \in Alg(\Sigma, E)$ erfüllt sind.

Eine Gleichungsspezifikation (Σ, E) definiert in diesem Sinn eine algebraische Gleichungstheorie.

Bemerkung: Die Axiomatisierung einer Gleichungstheorie ist nicht eindeutig; es kann durchaus verschiedene Gleichungsmengen geben, die als Axiome dieselbe Theorie definieren.

Initiale Algebren

Eine Algebra \mathcal{A} heißt *initial* in einer Klasse \mathcal{K} von Algebren, wenn es genau einen Homomorphismus von \mathcal{A} zu jeder Algebra \mathcal{B} in \mathcal{K} gibt.

Aus dieser Eigenschaft ergibt sich, daß alle initialen Algebren einer Klasse isomorph sind; in einer Klasse von Algebren ist die initiale Algebra (sofern sie existiert) ‘bis auf Isomorphie’ eindeutig bestimmt.

Initiale Algebren haben spezifische Eigenschaften:

- Sie sind minimal, d.h. es gibt keine Elemente, deren Existenz nicht durch die Spezifikation gefordert wird (“*no junk*”).
- Es gelten keine Gleichungen, die nicht durch die Spezifikation erzwungen werden; es werden nur so viele Elemente identifiziert wie unbedingt nötig (“*no confusion*”).

Für Spezifikationen ohne Gleichungen bilden die Termalgebren die initialen Algebren; für Spezifikationen mit Gleichungen werden zur Konstruktion von initialen Algebren *Quotienten* gebildet.

Kongruenzen und Quotienten

Σ -Kongruenz \equiv auf T_Σ : Äquivalenzrelation, die mit allen Operationen aus Σ verträglich ist:

Sind $u_i \equiv v_i$ ($1 \leq i \leq n$) und f ein n -stelliges Operationssymbol aus Σ , so ist auch $f(u_1, \dots, u_n) \equiv f(v_1, \dots, v_n)$.

(Genauer: für jede Sorte S gibt es eine Relation \equiv_S , d.h. \equiv ist eine S -indizierte Familie von Relationen)

Quotient \mathcal{A}/\equiv einer Σ -Algebra \mathcal{A} nach einer Kongruenz-Relation \equiv auf \mathcal{A} :
besteht aus Äquivalenzklassen

$$[a]_{\equiv} := \{b \in A \mid b \equiv a\}$$

Ein Quotient wird zu einer Σ -Algebra durch folgende Interpretation der Operationen in Σ :

$$f_{\mathcal{A}/\equiv}([a_1], \dots, [a_n]) := [f_{\mathcal{A}}(a_1, \dots, a_n)]$$

wohldefiniert wegen Kongruenzeigenschaft.

Quotienten und Gleichungsspezifikationen

Die Gleichungen einer Gleichungsspezifikation (Σ, E) definieren eine Kongruenzrelation \equiv_E auf der Term-Algebra $T_\Sigma(V)$ und damit eine Quotientenalgebra $T_\Sigma(V)/\equiv_E$.

Satz: T_Σ/\equiv_E ist initial in der Klasse der (Σ, E) -Algebren.

Satz: Jede term-erzeugte (Σ, E) -Algebra \mathcal{A} läßt sich als Quotient von T_Σ/\equiv_E darstellen, d.h. jede term-erzeugte (Σ, E) -Algebra \mathcal{A} ist isomorph zu einer Quotientenalgebra von T_Σ/\equiv_E .

Aufgrund der Eigenschaften der Termalgebra und der Initialität ergibt sich, daß die Gleichungen, die in allen Algebren einer gleichungsspezifizierten Klasse gelten, genau diejenigen sind, die in der initialen Algebra gelten:

Satz: Für ein beliebige Σ -Grundgleichung e gilt $E \models e$ genau dann, wenn e in T_Σ/\equiv_E gilt.

Für eine Σ -Gleichung e gilt $E \models e$ genau dann, wenn e in $T_\Sigma(V)/\equiv_E$ gilt.

Kalkül für Gleichungslogik

im wesentlichen: Umschreiben der Eigenschaften der Gleichheitsrelation in der Form von Regeln

Reflexivität: (Axiomenschema)

$$x = x$$

Symmetrie:

$$\frac{x = y}{y = x}$$

Transitivität:

$$\frac{x = y \quad y = z}{x = z}$$

Substitutivität:

$$\frac{x = y}{\sigma(x) = \sigma(y)}$$

für jede Substitution σ .

Kalkül für Gleichungslogik (2)

Kongruenz:

Für jedes n -stellige Funktionssymbol f und alle Terme x_i und y_i ($1 \leq i \leq n$):

$$\frac{x_i = y_i \quad (1 \leq i \leq n)}{f(x_1, \dots, x_n) = f(y_1, \dots, y_n)}$$

Ableitbarkeit in diesem Kalkül:

$$E \vdash s = t$$

‘Gleichung $s = t$ ist in dem Kalkül aus der Gleichungsmenmge E ableitbar’

Vollständigkeit und Korrektheit

Satz: Die folgenden Aussagen sind äquivalent

1. $E \models s = t$
2. $E \vdash s = t$
3. $s \leftrightarrow_R^* t$, wobei R das aus E entstehende Termersetzungssystem ist.

Beweisidee für die Äquivalenz der Aussagen (2) und (3): Induktion über die Länge der Ableitung.

Nach dem vorhergehenden Satz reicht es für (1) aus zu zeigen, daß die Gleichung $s = t$ in dem initialen Modell erfüllt ist.

Aussage (3) macht die Bedeutung konfluenten TES für Gleichungstheorien explizit.

Algebraische Datenstrukturen

In der Informatik, insbesondere der Programmierung, ist man nicht so sehr an algebraischen Strukturen wie Gruppen interessiert, sondern mehr an Strukturen wie Listen, Bäumen usw.

Solche Strukturen lassen sich mit Hilfe von Gleichungen als *algebraische Datentypen* spezifizieren.

Beispiel: natürliche Zahlen

Signatur:

Sorte Nat

Konstante $0 \in Nat$

Operationen:

$suc : Nat \rightarrow Nat,$

$pred : Nat \rightarrow Nat$

Gleichungen:

$pred(suc(x)) = x$

$pred(0) = 0$

Algebraische Datenstrukturen (2)

Im initialen Modell dieser Spezifikation gilt u.a., daß 0 verschieden von jedem Element der Form $suc^n(x)$ ist.

- ↪ von Interesse ist nur (oder primär) die zugehörige initiale Algebra, die den natürlichen Zahlen entspricht.
- ↪ Nat wird von 0 und suc erzeugt: jeder Term kann auf 0 oder $suc^n(x)$ für geeignetes n reduziert werden.

Algebraische Datenstrukturen: Stapel

Eine Theorie von Stapeln (Stacks) über Nat :

Sorten: Nat , $Stack$

Operationen:

Die Operationen zu Nat

$newst : Stack$

$push : Nat \times Stack \rightarrow Stack$

$top : Stack \rightarrow Nat$

$pop : Stack \rightarrow Stack$

Gleichungen:

$pop(push(x, s)) = s$

$top(push(x, s)) = x$

$top(newst) = 0$ (*)

$pop(newst) = newst$ (*)

Bemerkung: Die mit (*) markierten Gleichungen sind problematisch.

Algebraische Datenstrukturen: Mengen

Abstrakter Datentyp (“ADT”) für endliche Mengen über einer Element-Sorte S

Sorten: S , *Menge*

Operationen:

$leer : Menge$

$ins : S \times Menge \rightarrow Menge$

Gleichungen:

$ins(x, ins(x, m)) = ins(x, m)$ (Idempotenz)

$ins(x, ins(y, m)) = ins(y, ins(x, m))$ (Kommut.)

Die Signatur ist isomorph zu einer Teilsignatur der Theorie der linearen Listen. Die Gleichungen definieren eine Kongruenz; sie suggerieren eine “Implementierung” von endlichen Mengen durch Äquivalenzklassen von Listen.

Frage: wie kann Kommutativität auf Listen repräsentiert werden?

Generatoren:

Für eine term-erzeugte Algebra können i.a. unter den Operationen der Signatur *Generatoren* identifiziert werden:

Die Elemente der Algebra sind dann gerade die Bilder (unter dem kanonischen Homomorphismus) einer Teilmenge derjenigen Terme (Elemente der Term-Algebra T_Σ), die mit Hilfe nur der Generatoren gebildet werden können.

Beispiele:

- 0 und *suc* erzeugen die (Elemente der) Sorte *Nat*.
- Generatoren für die Theorie der Stapel sind die Operationen *newst* und *push*

Spezifikation von abstrakten Datentypen in PVS (Vorgriff):

Die PVS-Spezifikation eines ADT mit Hilfe des Konstrukts *datatype* erzwingt die Unterscheidung zwischen Generatoren, "Selektoren" und anderen, insbesondere definierten, Operationen: es können nur term-erzeugte Datentypen (Algebren) eingeführt werden.

Generatoren (Forts.)

Z.B. für Stapel über *Nat*:

```
Stack: DATATYPE
      BEGIN   newst: newst?
            push(top: Nat, pop: Stack): push?
      END Stack
```

Die oben angegebenen Gleichungen sind Teil der intendierten Semantik dieser Deklaration; außerdem wird impliziert:

- `newst` und `push` sind die Generatoren, die alle Werte des Datentyps `Stack` erzeugen.
- Die Generatoren sind disjunkt.
- Das “passende” Induktionsprinzip.

Details kommen später.

Termersetzung und funktionales Programmieren

- Spezifikation der Datenstrukturen:
Signatur, Gleichungen
- Spezifikation von Funktionen durch Mengen von Gleichungen, die sich am Aufbau der Datenstruktur orientieren.
- Gleichungen zu Termersetzungsregeln
- “Ausführung” von Funktionsaufrufen durch Termersetzung
- “Symbolische Auswertung” möglich
- Berechnung als Deduktion
- kann als eine Form des “logischen Programmierens” (in Gleichungslogik) angesehen werden

Beispiel: Lineare Listen:

Sorten A , LL

Konstante $nul : LL$

Operationen $cons : A \times LL \rightarrow LL$

$car : LL \rightarrow A$

$cdr : LL \rightarrow LL$

Gleichungen wie bei Stapel

Definierte Funktionen: z.B.

$$append(nul, l) = l$$

$$append(cons(a, x), y) = cons(a, append(x, y))$$

$append$ ist eine rekursive Funktion

\rightsquigarrow entspricht der Datenstruktur, die induktiv aufgebaut ist

\rightsquigarrow Induktion (kommt später)