

Kapitel 3a Termalgebren

- 3.1 Formeln
- 3.2 Boolesche Algebra
- 3.3 Algebraische Strukturen und Algebren
- 3.4 Abbildungen zwischen Algebren
- 3.5 Termalgebren
- 3.6 Termalgebren mit Variablen
- 3.7 Termersetzungssysteme
- 3.8 Vertiefung Algorithmusbegriff



3. Motivation

1/63

- Was ist „3+5“ ?
Antwort: „drei plus fünf“
→ Wir unterscheiden zwischen der **algebraischen Formel** „3+5“ und dem **Ergebnis** dieser Formel.
 - die Formel heißt eine **Rechenvorschrift** (Funktion, Prozedur,...)
 - das Ergebnis erhält man durch **Auswertung (Berechnung)** der Rechenvorschrift
- Formeln heißen auch **Terme**.
- In einer **Algebra A** bildet auch die Menge der Terme eine Algebra (**Termalgebra**).



3. Ziele

2/63

- Unterscheide Formel – Ergebnis** der Berechnung der Formel
- Texte und Textersetzung wie bei Markov-Algorithmen nehmen auf die Struktur der Texte keine Rücksicht (aber unser Gehirn tut das!), also
 - strukturierte Texte statt Zeichenreihen** betrachten
 - einfachste und am häufigsten benötigte Struktur: **Bäume**
 - Terme sind geordnete Bäume mit benannten (= markierten) Ecken
- Termersetzung**: Umformung von Termen bei gleichem Ergebnis
- Termalgebren**: Datenstrukturen und Datentypen in Programmen sind Termalgebren
- überdies: die boolesche Algebra ist ein schon bekannter Spezialfall, der Umgang mit Formeln der Aussagen- und Prädikatenlogik im nächsten Kapitel folgt formal den gleichen Gesetzen



3. Wozu das alles?

3/63

- Alle Datentypen und Datenstrukturen sind Termalgebren
- Unterscheide Spezifikation (Schnittstellenbeschreibung) und Implementierung:
 - Spezifikation nennt die an der Schnittstelle vorhandenen und gültigen
 - Operationen (Signatur)
 - Gesetze (Axiome)
 - beides zusammen kennzeichnet eine abstrakte Algebra
 - Implementierung beschreibt eine mögliche Realisierung
 - konkrete Algebra, die auch die Gesetze erfüllt
 - konkrete Algebra ist homomorphes Bild der abstrakten Algebra
 - sonst wäre die Implementierung fehlerhaft
- Termersetzung ist Berechnungsmethode auf der Spezifikationsebene
 - beschreibt auch die Berechnung von Funktionsaufrufen
 - grundlegendes Rechenmodell für funktionale und dann auch für imperative Programmiersprachen



3.1 Formeln

4/63

n-stellige Operation auf einer Menge A: Abbildung $f: A^n \rightarrow A$

Stelligkeit der Operation $f: A^n \rightarrow A$:

- Notation $f^{(n)}$ oder f/n macht Stelligkeit explizit
- unäre Operation: $n = 1$
- binäre Operation: $n = 2$

Argumente einer Anwendung der Operation f:

- Operanden a_1, \dots, a_n der Anwendung der Operation $f(a_1, \dots, a_n)$

Beispiele:

- $A = \mathbb{N}$, Addition $+$: binäre Operation
- $A = \mathcal{P}(U)$ über Grundmenge U:
 - Komplement $\bar{\cdot}$: unäre Operation
 - Vereinigung \cup , Durchschnitt \cap : binäre Operationen
- Reguläre Ausdrücke über Alphabet Σ :
 - $\emptyset, a \in \Sigma, \epsilon$: nullstellige Operationen
 - Wiederholung $*$: unäre Operation
 - Konkatenation \cdot , Vereinigung $+$: binäre Operationen



3.1 Signaturen

5/63

Signatur Σ Menge von Operationen $\Sigma = \Sigma^{(0)} \cup \Sigma^{(1)} \cup \Sigma^{(2)} \cup \dots$ mit

- $\Sigma^{(n)}$ ist Menge der n-stelligen Operationen
- $\Sigma^{(n)}$ sind paarweise disjunkt

Beispiel:

- $+, \cup, \cap \in \Sigma^{(2)}$
- Signatur einer booleschen Algebra $\Sigma = \{\tau, \perp, \bar{\cdot}, \wedge, \vee\}$
 - $\Sigma^{(0)} = \{\tau, \perp\}$
 - $\Sigma^{(1)} = \{\bar{\cdot}\}$
 - $\Sigma^{(2)} = \{\wedge, \vee\}$

Konstante: Operationen $c \in \Sigma^{(0)}$, häufige Annahme: $\Sigma^{(0)} \neq \emptyset$

Beispiele:

- Alle natürlichen Zahlen $0, 1, 2, \dots$
- Alle Zeichen $a \in \Sigma$ eines Zeichenvorrates
- Alle Teilmengen $M \in \mathcal{P}(U)$



3.2 Boolesche Algebra

9/63

Satz: \mathcal{B} ist boolesche Algebra, wenn gilt:

1. Von V1-V5 gilt das erste (oder jeweils das zweite) Gesetz.
2. Die Gesetze V8 oder die Gesetze V10 gelten.
3. \mathcal{B} ist vollständig, d.h. $\inf(X), \sup(X) \in U$ für alle $X \subseteq U$.

Also: zum Nachweis, daß eine boolesche Algebra vorliegt, braucht man nicht alle Gesetze beweisen. Insbesondere folgen V6, V7 und V9 aus den anderen Gesetzen (siehe auch die früheren Beweise).



3.2 Mengenalgebra und Unteralgebra

10/63

Mengenalgebra: Algebra $\mathcal{A} \triangleq (\mathcal{P}(U), \cup, \cap, C)$ mit Menge U , Vereinigung \cup , Durchschnitt \cap und Komplement C , wobei $CM \triangleq U \setminus M$

Satz: \mathcal{A} ist boolesche Algebra.

Beweis: Nachrechnen von V1-V10 (Übung)

Beobachtung: Für $\mathcal{L} \subseteq \mathcal{P}(U)$ gilt im Allgemeinen nicht mehr $M \cap N \in \mathcal{L}, M \cup N \in \mathcal{L}$ und $CM \in \mathcal{L}$

\mathcal{L} ist **algebraisch abgeschlossen** und $\mathcal{U} = (\mathcal{L}, \cup, \cap, C)$ **Unteralgebra**.

- wenn für alle $M, N \in \mathcal{L}$ gilt:
 - $M \cap N \in \mathcal{L}, M \cup N \in \mathcal{L}$ und $CM \in \mathcal{L}$
- äquivalente Bedingungen:
 - $\emptyset \in \mathcal{L}, U \in \mathcal{L}$
 - $M, N \in \mathcal{L}$, folgt $M \cap N \in \mathcal{L}$, und $CM \in \mathcal{L}$ } $\Rightarrow M \cup N \in \mathcal{L}$



3.2 Mengenalgebren und boolesche Algebren

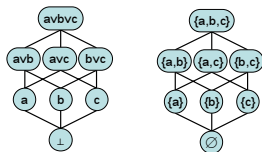
11/63

Satz von Stone:

Sei $\mathcal{B} = (A, \vee, \wedge, C)$ boolesche Algebra mit **endlicher** Grundmenge A . Dann gibt es eine Menge U und eine bijektive Abbildung $f: A \rightarrow \mathcal{P}(U)$, so daß für alle $x, y \in A$ gilt:

1. $f(\perp) = \emptyset$ und $f(\top) = U$
2. $f(x \vee y) = f(x) \cup f(y)$
3. $f(x \wedge y) = f(x) \cap f(y)$
4. $f(Cx) = U \setminus f(x)$

Beispiel:



3.2 Satz von Stone: Beweis

12/63

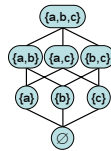
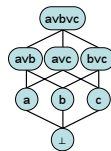
- Sei $|A| = 1$, wir bezeichnen das (einzige) Element von A als a
 - Es gilt:
 - $\perp = a = \top$
 - $a \wedge a = a$
 - $a \vee a = a$
 - Wähle $U = \emptyset$ und somit $f: A \rightarrow \emptyset$
 1. $f(\perp) = \emptyset$ und $f(\top) = \emptyset = U$
 2. $f(a \vee a) = f(a) = \emptyset = \emptyset \cup \emptyset = f(a) \cup f(a)$
 3. $f(a \wedge a) = f(a) = \emptyset = \emptyset \cap \emptyset = f(a) \cap f(a)$
 4. Es gilt:
 - $a \wedge a = \perp$
 - $a \vee a = \top$
 - Daher: $\mathcal{C}a = a$
 - $f(\mathcal{C}a) = f(a) = \emptyset = \emptyset \setminus \emptyset = U \setminus f(a)$



3.2 Satz von Stone: Beweis

13/63

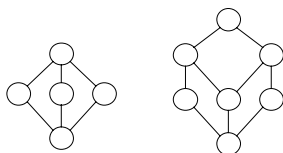
- Sei $|A| > 1$:
 - Wähle $U = \{a \in A \mid \perp < a \text{ und } b < a \Rightarrow b = \perp\}$
 - Definiere $f: A \rightarrow U$ für ein $a \in A$ als
 - $f(\perp) = \emptyset$
 - $f(a) = \{u \in U \mid u \leq a\}$, wenn $a \neq \perp$
 - $f(\mathcal{C}a) = \{u \in U \mid u \not\leq a\}$
 - Nachrechnen ergibt (vgl. Buch, S. 119):
 - f ist injektiv und surjektiv, also bijektiv
 - Die Forderungen 1-4 des Satzes sind erfüllt.
 - **Beispiele:**
 - $f(a \wedge b) = f(\perp) = \emptyset = \{a\} \cap \{b\} = f(a) \cap f(b)$
 - $f(\mathcal{C}b) = \{a, c\} = \{a\} \cup \{c\} = f(a) \cup f(c) = f(a \vee c)$
 - $f(\mathcal{C}(\mathcal{C}bvc)) = f(a) = \{a\} = U \setminus \{b, c\} = U \setminus f(bvc)$



3.2 Satz von Stone: Folgerungen

14/63

- Konklusion: alle endlichen booleschen Algebren sind isomorph zu Mengenalgebren
- Frage: Woran erkennt man sofort, daß die folgenden Verbände keine booleschen Algebren sind?



3.2 Weitere Eigenschaften

15/63

Korollar: Sei $\mathcal{B} = (U, \vee, \wedge, \complement)$ boolesche Algebra mit endlicher Grundmenge A , dann ist $|A| = 2^n$ für ein $n \geq 0$.

Beweis: \mathcal{B} ist isomorph zu $(\mathcal{P}(U), \cup, \cap, \complement)$ für ein endliches U .
 $\Rightarrow |A| = |\mathcal{P}(U)| = 2^{|U|}$

Spezialfall: $U = \{a\}$

$\Rightarrow \mathcal{P}(U) = \{\emptyset, \{a\}\}$

\Rightarrow Führt zu einer booleschen Algebra über Binärcode $\mathbb{B} \triangleq \{0, 1\}$

Korollar: Sei $\mathcal{B} = (U, \vee, \wedge, \complement)$ boolesche Algebra mit endlicher Grundmenge A . Dann ist \mathcal{B} isomorph zu einer Algebra über $\mathbb{B}^n = \underbrace{\mathbb{B} \times \dots \times \mathbb{B}}_{n \text{ mal}}$ wobei

$$\begin{aligned}(x_1, \dots, x_n) \vee (y_1, \dots, y_n) &= (x_1 \vee y_1, x_2 \vee y_2, \dots, x_n \vee y_n) \\ (x_1, \dots, x_n) \wedge (y_1, \dots, y_n) &= (x_1 \wedge y_1, x_2 \wedge y_2, \dots, x_n \wedge y_n) \\ \complement(x_1, x_2, \dots, x_n) &= (\complement x_1, \complement x_2, \dots, \complement x_n)\end{aligned}$$

Man sagt, die Operationen $\wedge, \vee, \complement$ sind **elementweise** definiert.



3.3 Algebraische Strukturen und Algebren

16/63

▪ Gegeben:

• Signatur Σ (also Menge von Operationen $\Sigma = \Sigma^{(0)} \cup \Sigma^{(1)} \cup \Sigma^{(2)} \cup \dots$)

• Elementaroperanden X

• Menge \mathcal{T} der korrekten Terme zu Σ und X

• Menge von Gesetzen (bzw. Axiome) Q , die bedeutungstreue Umformungen $f \rightarrow f'$ ($f, f' \in \Sigma$) definieren

Beispiele: die Halbgruppen- und Monoidgesetze HG1, HG2 (nur bei kommutativen HG), die Gesetze V1-V10 der booleschen Algebra

Das Tripel $\mathcal{A} = (\mathcal{T}, \Sigma, Q)$ bildet eine **algebraische Struktur**, man spricht auch von einer **Abstrakten Algebra**. Dabei gilt für zwei Terme $t, t' \in \mathcal{T}$: $t = t'$ gdw. t nach den Gesetzen Q in t' umgeformt werden kann.

Beispiele:

▪ Halbgruppen, Monoide, Verbände, Gruppen, Ringe, Körper, Vektorräume, boolesche Algebren

▪ alle Datenstrukturen der Informatik (**abstrakte Datentypen**)



3.3 Konkrete Algebren

17/63

Gegeben: Signatur Σ , Menge $X \ni \Sigma^{(0)}$ von Elementaroperanden, abstrakte Algebra $\mathcal{A} = (\mathcal{T}, \Sigma, Q)$

konkrete Algebra (\mathcal{C} -Algebra): Paar $A = (T, \Phi)$ wobei:

• T ist eine Menge mit $X \subseteq T$ (**Trägermenge**)

• $\Phi = \{f_r; T^n \rightarrow T \mid f^{(n)} \in \Sigma\}$, d.h. jedem n -stelligen Operationssymbol f wird eine Funktion $f_r; \underbrace{T \times \dots \times T}_{n \text{ mal}} \rightarrow T$ zugeordnet

• die Gesetze Q sind durch die Funktionen f_r erfüllt

Man sagt auch: A ist ein **Exemplar der algebraischen Struktur \mathcal{A}** .

oder: A ist eine **Implementierung von \mathcal{A}** .

Beispiel: die $(\mathbb{N}, +)$ implementiert ein kommutatives Monoid:

▪ die Signatur entspricht einem Monoid

▪ die Gesetze eines kommutativen Monoids sind erfüllt



3.3 Weitere Beispiele konkreter Algebren

18/63

Weitere Beispiele:

- $\mathbf{R, Q, C}$ sind Exemplare der algebraischen Struktur Körper.

also:

- Es kann mehrere verschiedene Exemplare (Implementierungen) derselben algebraischen Struktur geben.
- Manche konkreten Algebren sind identisch mit ihren abstrakten Algebren, z.B. die Mengenalgebra und boolesche Algebra.
⇒ unterscheide abstrakte und konkrete Algebra nur bei Bedarf
 - in der Mathematik sehr häufig keine Unterscheidung
 - in der Informatik wegen unterschiedlicher Implementierungen Unterscheidung meist unvermeidlich
- Zitat zum Nachdenken:
 - Nimmst du die blaue Pille wirst du in deinem Bett aufwachen und an das glauben, an das du glauben willst.
 - Nimmst du die rote Pille werde ich dich in die tiefsten Tiefen des Kaninchenbaus führen.
 - Bedenke: Alles was ich dir anbiete ist die Wahrheit. Nicht mehr.



3.3 Mehrsortige Signaturen

19/63

Beispiel:

Verbände und boolesche Algebren besitzen eine Ordnungsrelation \leq , die eine Halbordnung definiert.

- \leq ist eine zweistellige Funktion
- Problem: Ergebnis dieser Relation ist kein Verbandselement sondern ein Wert aus $\mathbb{B} = \{0, 1\}$
⇒ Einführung von Typen (**Sorten**)

Mehrsortige Signatur: Paar $\Sigma = \langle U, \Sigma_A \rangle$, wobei:

- S Menge von Trägermengen, der „Sorten“
- $\Sigma_A = \{f: s_1 \times \dots \times s_n \rightarrow A \mid n \geq 0 \text{ und } s_i \in S\}$
(Menge der Funktionen, die auf die Sorte A abbilden)



3.3 Mehrsortige Algebren: Beispiel

20/63

Beispiel:

Signatur eines Verbands über einem Universum U mit der Ordnungsrelation \leq :

- **Sorten:** U, B

Operationsymbole:

- $a: \rightarrow U$, für alle Atome a
- $C: U \rightarrow U$
- $\wedge: U \times U \rightarrow U$
- $\vee: U \times U \rightarrow U$
- $\leq: U \times U \rightarrow B$



3.3 Mehrsortige Algebren: Terme

21/63

Term der Sorte $s \in S$ über $\Sigma = \cup_{A \in S} \Sigma_A$ und $X_s \ni \Sigma_s^{(0)}$ ist induktiv definiert:

- jede Konstante $f \in X_s$ ist Term der Sorte s
- mit $f: s_1 \times \dots \times s_n \rightarrow s \in \Sigma_s$ und Termen t_1, \dots, t_n der Sorten s_1, \dots, s_n ist $f(t_1, \dots, t_n)$ Term der Sorte s

Beispiel:

▪ **Sorten:** U, B

▪ **Operationssymbole:**

- $a: \rightarrow U$, für alle Atome a
- $C: U \rightarrow U$
- $\wedge: U \times U \rightarrow U$
- $\vee: U \times U \rightarrow U$
- $\leq: U \times U \rightarrow B$

▪ Atom a , $a \wedge b$ sind Terme der Sorte U

▪ $a \leq Ca$ ist Term der Sorte B

▪ **($a \leq Ca$) \vee Ca ist weder Term der Sorte U noch der Sorte B**



3.3 Mehrsortige Algebren

22/63

Gegeben: mehrsortige Signatur $\Sigma = \cup_{A \in S} \Sigma_A$ mit Mengen von Elementaroperanden $X_s \ni \Sigma_s^{(0)}$, $s \in S$, Gesetze Q

Beispiel: Signaturen von Verbänden mit Ordnungsrelation

- Sorte U wird durch Menge V der Verbandselemente interpretiert
- Sorte B wird durch Binärcode $\mathbb{B} = \{0, 1\}$ interpretiert

Mehrsortige Algebra (Σ -Algebra): Paar $\mathcal{A} = (T, \Phi)$ wobei:

- $T = (T_s)$, wobei T_s ist Menge mit $X_s \subseteq T_s$ (Trägermenge der Sorte s)
- $\Phi = \{f: T_{s_1} \times \dots \times T_{s_n} \times T_s \mid f: s_1 \times \dots \times s_n \rightarrow s \in \Sigma\}$,
d.h. jedem Operationssymbol f der Stelligkeit s_1, \dots, s_n, s wird eine Funktion $f_T: T_{s_1} \times \dots \times T_{s_n} \rightarrow T_s$ zugeordnet
- die Gesetze Q sind erfüllt

Mehrsortige Algebren heißen auch **heterogene Algebren**

(Gegensatz: **homogene Algebra**)



3.3 Beispiel: Keller

23/63

Beispiel: Keller (engl. *stack*)

der Keller ist eine abstrakte mehrsortige Algebra mit folgenden Eigenschaften:

- Keller ist Datenstruktur zum Speichern von Werten der Sorte T , z.B. natürliche Zahlen $m \in \mathbb{N}$.
- Keller ist anfangs leer.
Elemente können oben auf den Keller gelegt werden.
- oberstes Element kann ausgelesen werden.
- oberstes Element kann entfernt werden.
- es kann nachgeprüft werden, ob ein Keller leer ist.



3.3 Beispiel: Keller

24/63

Signatur:

- createStack: $\rightarrow \text{stack}(T)$
- push: $\text{stack}(T) \times T \rightarrow \text{stack}(T)$
- top: $\text{stack}(T) \rightarrow T$
- pop: $\text{stack}(T) \rightarrow \text{stack}(T)$
- isEmpty: $\text{stack}(T) \rightarrow B$

• Axiome:

- K1: $\text{isEmpty}(\text{createStack}) = L$
- K2: $\text{isEmpty}(\text{push}(k, t)) = O$
- K3: $\text{top}(\text{push}(k, t)) = t$
- K4: $\text{pop}(\text{push}(k, t)) = k$





3.3 Beispiel: Keller

25/63

Beobachtung: Elemente der abstrakten Algebra sind Terme.

Beispiele:

- createStack 
- push(createStack, 5) 
- push(push(createStack, 5), 2) 

Beobachtung: Diese Terme können durch die Gesetze K1-K4 nicht vereinfacht werden.

Beispiel zur Vereinfachung eines Terms:

- $\text{pop}(\text{push}(\text{push}(\text{createStack}, 5), 2)) \stackrel{K4}{=} \text{push}(\text{createStack}, 5)$



3.3 Homogene und mehrsortige Algebren

26/63

- auf mehrsortige Algebren könnte man auch verzichten
- alle Datentypen zusammennehmen: $N, R, Q, C, \{\text{wahr, falsch}\}, \text{Listen, Zeichenreihen}, \dots$
- alle Operationen nur partiell definieren (keine Division für Wahrheitswerte oder Zeichenreihen, ...)
- in Maschinsprachen tut man das; keine Unterscheidung, ob eine Folge von 32 Bit eine ganze Zahl oder eine Gleitpunktzahl darstellt
- aber:
 - dann sind die Möglichkeiten, die Richtigkeit eines Programms anhand des Programmtexts einzusehen, eingeschränkt:
 - Laufzeitfehler wegen falscher Typen
 - riesiger Testaufwand bei großen Programmen
 - Keine Möglichkeit für Polymorphie ($+, +$ bei Bedarf Festpunkt-/Gleitpunktaddition)
- also:
 - mehrsortige Algebren eine praktische Notwendigkeit, in der Theorie reichen homogene Algebren



3.3 Spezifikation und Implementierung

27/63

- Die Gesetze oder Axiome eines Kellers **spezifizieren** das Verhalten des Kellers
 - jede Implementierung eines Kellers auf einem Rechner muß sich entsprechend diesen Gesetzen der abstrakten Algebra verhalten
- ⇒ Eine Implementierung eines Kellers auf einem Rechner ist ein Exemplar dieser abstrakten Algebra, also eine konkrete Algebra
- Ausblick: in funktionalen Sprachen kann auch die Spezifikation selbst als Implementierung benutzt werden
- ⇒ eine abstrakte Algebra heißt **Spezifikation** der konkreten Algebra (**Implementierung**)

Beispiel SQL: die Grundoperationen der relationalen Algebra und zugehörige Gesetze definieren eine abstrakte Algebra

- Die Operationen können ohne Kenntnis der Implementierung in einer konkreten Datenbank benutzt werden



3.4 Abbildungen zwischen Algebren

28/63

Beispiel:

- Sei $\Delta = \{a, b\}$ Zeichenvorrat.
- Signatur $\Sigma = \{a/0, b/0, \epsilon/0, /2\}$

Algebra $\mathcal{A} = \{\Delta^*, \Phi\}$ mit

- $\epsilon_{\mathcal{A}}: \Delta^* \rightarrow \Delta^* \quad \epsilon_{\mathcal{A}} = \epsilon$
- $a_{\mathcal{A}}: \Delta^* \rightarrow \Delta^* \quad a_{\mathcal{A}} = a$
- $b_{\mathcal{A}}: \Delta^* \rightarrow \Delta^* \quad b_{\mathcal{A}} = b$
- $\cdot_{\mathcal{A}}: \Delta^* \times \Delta^* \rightarrow \Delta^* \quad x \cdot_{\mathcal{A}} y = xy$

Unterschied \mathcal{A}, \mathcal{B} :
Vertauschung a und b

Algebra $\mathcal{B} = \{\Delta^*, \Phi\}$ mit

- $\epsilon_{\mathcal{B}}: \Delta^* \rightarrow \Delta^* \quad \epsilon_{\mathcal{B}} = \epsilon$
- $a_{\mathcal{B}}: \Delta^* \rightarrow \Delta^* \quad a_{\mathcal{B}} = b$
- $b_{\mathcal{B}}: \Delta^* \rightarrow \Delta^* \quad b_{\mathcal{B}} = a$
- $\cdot_{\mathcal{B}}: \Delta^* \times \Delta^* \rightarrow \Delta^* \quad x \cdot_{\mathcal{B}} y = xy$

⇒ \mathcal{A} und \mathcal{B} erfüllen beide die Gesetze eines Monoids.



3.4 Abbildungen zwischen Algebren

29/63

betrachte Abbildung $h: \Delta^* \rightarrow \Delta^*$ mit $h(x_1 \dots x_n) = \sum_{i=1}^n x_i, a = b, \underline{a} = a$:

- $h(\epsilon) = \epsilon \quad h(a) = b \quad h(b) = a$
- $h(\epsilon_a) = \epsilon_b \quad h(a_a) = a_b \quad h(b_a) = b_b$
- $h(x \cdot_a y) = h(x) \cdot_b h(y)$

betrachte Abbildung $h': \Delta^* \rightarrow \Delta^*$ mit $h(x_1 \dots x_n) = x_n \dots x_1$:

- $h'(abb \cdot ba) = h'(abbba) = abbba$
- aber:** $h'(abb) \cdot h'(ba) = bba \cdot ab = bbaab \neq abbba = h'(abb \cdot ba)$

Beobachtung: Abbildung h ist kompatibel mit den Operationen der Algebren \mathcal{A} und \mathcal{B} .



für die Abbildung h' gilt dies nicht



3.4 Homomorphismen

30/63

seien $\mathcal{A} = (M, \Phi)$ und $\mathcal{B} = (N, \Psi)$ zwei Σ -Algebren zur Signatur $\Sigma = (S, F)$.

Σ -Algebra-Homomorphismus $h: \mathcal{A} \rightarrow \mathcal{B}$:

Familie von Abbildungen $h_s: M_s \rightarrow N_s, s \in S$, so daß für alle $f: s_1 \times \dots \times s_n \rightarrow s \in F$ gilt:

$$h_s(f_M(x_{s_1}, \dots, x_{s_n})) = f_N(h_{s_1}(x_{s_1}), \dots, h_{s_n}(x_{s_n}))$$

diese Gleichung heißt **Homomorphiebedingung**

Spezialfall: zweistellige Operation f in homogenen Algebren:
 $h(f(x, y)) = f(h(x), h(y))$

Statt Σ -Algebra-Homomorphismus sagt man meist nur **Homomorphismus** oder **Morphismus**



3.4 Abbildungen – Homomorphismen

31/63

Σ -Monomorphismus: injektiver Σ -Homomorphismus, d.h. alle $h_s, s \in S$ sind injektiv.

Σ -Epimorphismus: surjektiver Σ -Homomorphismus, d.h. alle $h_s, s \in S$ sind surjektiv.

Σ -Isomorphismus: bijektiver Σ -Homomorphismus, d.h. alle $h_s, s \in S$ sind bijektiv.

Σ -Endomorphismus: Σ -Homomorphismus $h: \mathcal{A} \rightarrow \mathcal{A}$

Σ -Automorphismus: Σ -Isomorphismus $h: \mathcal{A} \rightarrow \mathcal{A}$

Eigenschaft

Wenn in einer Σ -Algebra \mathcal{A} die Gesetze Q gelten und ein Homomorphismus $h: \mathcal{A} \rightarrow \mathcal{B}$ existiert, dann gelten die Gesetze Q auch im Bild von \mathcal{A} (in der Σ -Algebra \mathcal{B}).



3.4 Beispiel: Keller implementiert mit Listen

32/63

Beispiel: betrachte die abstrakte Algebra Keller

sowie die konkrete Algebra $K = (L, \Psi)$ mit

$L_T = T^*$; $L_{\text{stack}(T)} = T^* \times T$; $L_B = \mathbb{B}$.

• Signatur:

• $\text{createStack}_T:$	$T^* \rightarrow T^*$	• Signatur der abstrakten Algebra:	• $\text{createStack}:$	$\rightarrow \text{stack}(T)$
• $\text{push}_T:$	$T^* \times T \rightarrow T^*$		• $\text{push}:$	$\text{stack}(T) \times T \rightarrow \text{stack}(T)$
• $\text{top}_T:$	$T^* \rightarrow T$		• $\text{top}:$	$\text{stack}(T) \rightarrow T$
• $\text{pop}_T:$	$T^* \rightarrow T^*$		• $\text{pop}:$	$\text{stack}(T) \rightarrow \text{stack}(T)$
			• $\text{isEmpty}:$	$\text{stack}(T) \rightarrow \mathbb{B}$

• Funktionen:

• createStack_T	$= []$	• Axiome:	• K1:	$\text{isEmpty}(\text{createStack}) = \text{true}$
• $\text{push}_T(s, x)$	$= s++[x]$		• K2:	$\text{isEmpty}(\text{push}(k, t)) = \text{false}$
• $\text{top}_T([x_1, \dots, x_n])$	$= x_n$		• K3:	$\text{top}(\text{push}(k, t)) = t$
• $\text{pop}_T([x_1, \dots, x_n])$	$= [x_1, \dots, x_{n-1}]$		• K4:	$\text{pop}(\text{push}(k, t)) = k$



3.4 Beispiel: Keller

33/63

Homomorphismus h von abstrakter Kelleralgebra in Algebra K .

- $h_T: T \rightarrow T$ ist die identische Abbildung
- $h_{\text{stack}(T)}: \mathcal{F}_{\text{stack}(T)} \rightarrow T^*$ ist definiert durch

$$h_{\text{stack}(T)}(\text{push}(\dots(\text{push}(\text{createStack}, x_1)\dots), x_n)) = [x_1, \dots, x_n]$$

Beweis der Homomorphiebedingungen:

z.B.

$$\begin{aligned} x_n &= h(\text{top}(\text{push}(\dots(\text{push}(\text{createStack}, x_1)\dots), x_n))) \\ &= \text{top}(h(\text{push}(\dots(\text{push}(\text{createStack}, x_1)\dots), x_n))) \\ &= \text{top}([x_1, \dots, x_n]) = x_n \end{aligned}$$

Die konkrete Algebra K implementiert die abstrakte Algebra der Keller.



3.4 Konstruktion homomorpher Bilder von Algebren

34/63

Ziel: auf einer Mengenfamilie $\{N_s\}_{s \in S}$ Funktionen $f \in F$ so definieren, daß eine vorgegebene Abbildung h einen Homomorphismus darstellt

- Sei $\Sigma = (S, F)$ eine Signatur,
- $\mathcal{A} = (M, \Phi)$ eine Σ -Algebra, die Gesetze Q erfüllt
- $N_s, s \in S$ eine Familie von Mengen
- $h_s: M_s \rightarrow N_s, s \in S$ surjektive Abbildungen
- für alle $f: s_1 \times \dots \times s_n \rightarrow s \in F$ sei f_N so definiert, daß die Homomorphiebedingung gilt, d.h. $f_N(h_{s_1}(a_1), \dots, h_{s_n}(a_n)) = h_s(f_M(a_1, \dots, a_n))$

Behauptung: $\mathcal{B} = (N, \Psi)$ mit $\Psi = \{f_N \mid f \in F\}$ ist eine Σ -Algebra, die die Gesetze Q erfüllt und $h: \mathcal{A} \rightarrow \mathcal{B}$ ist ein Σ -Algebra-Homomorphismus.

Man sagt, h induziert eine algebraische Struktur auf N .

praktische Anwendung: N soll zur Implementierung von \mathcal{A} dienen, enthält aber zuerst noch nicht einmal die nötigen Grundoperationen



3.4 Konstruktion homomorpher Bilder von Algebren

35/63

Beispiel: Betrachte die Algebra $\mathcal{Z} = (\mathbb{Z}, \{0/0, +/2\})$ der ganzen Zahlen.

$$\text{Sei } h: \mathbb{Z} \rightarrow \{0, 1\} \text{ mit } h(n) = \begin{cases} 0 & \text{falls } n \text{ gerade} \\ 1 & \text{falls } n \text{ ungerade} \end{cases}$$

- $0 + 0 = 0$ zwei gerade Zahlen addieren ergibt gerade Zahl
- $0 + 1 = 1$ gerade und ungerade Zahl addieren ergibt ungerade Zahl
- $1 + 0 = 1$ ungerade und gerade Zahl addieren ergibt ungerade Zahl
- $1 + 1 = 0$ zwei ungerade Zahlen addieren ergibt gerade Zahl

$\Rightarrow 0$ ist neutrales Element.

Man rechnet leicht nach, daß auch das Assoziativgesetz gilt.



3.4 Äquivalenzrelationen

36/63

Eine Äquivalenzrelation $\equiv \subseteq U \times U$ ist

- reflexiv: $\forall x \in U: x \equiv x$
- symmetrisch: $\forall x, y \in U: (x \equiv y) \leftrightarrow (y \equiv x)$
- transitiv: $\forall x, y, z \in U: ((x \equiv y) \wedge (y \equiv z)) \rightarrow x \equiv z$

Beispiel (fortgesetzt):

$U = \mathbb{N}$ und $n \equiv m$ genau dann, wenn

- entweder n und m beide gerade
- oder n und m beide ungerade

alternative Formulierung:

- betrachte $h: \mathbb{N} \rightarrow \{0, 1\}$ mit

$$h(n) = \begin{cases} 0 & \text{falls } n \text{ gerade} \\ 1 & \text{falls } n \text{ ungerade} \end{cases}$$

- $n \equiv m$ genau dann, wenn $h(n) = h(m)$

Einsicht:

Jeder Homomorphismus
 $f: A \rightarrow B$, A Algebra, definiert
eine Äquivalenzrelation auf A :
 $a \equiv b$ gdw. $f(a) = f(b)$



3.4 Äquivalenzrelationen

37/63

Äquivalenzklassen zu Äquivalenzrelation $\equiv \subseteq U \times U$:

$$[x] = \{y \in U \mid x \equiv y\}$$

Notation: $U/\equiv = \{[x] \mid x \in U\}$ bezeichnet die Menge aller Äquivalenzklassen von U bzgl. \equiv .

Beispiel (fortgesetzt):

Betrachte vorherige Äquivalenzrelation h auf \mathbb{Z} .

Es gibt zwei Äquivalenzklassen:

- $[0] = \{n \mid n \text{ ist gerade}\}$
- $[1] = \{n \mid n \text{ ist ungerade}\}$

Also ist $\mathbb{Z}/\equiv = \{[0], [1]\}$.



3.4 Quotientenalgebra

38/63

Sei

- Σ eine (homogene) Signatur
- $A = (M, \Phi)$ eine homogene Σ -Algebra
- $h: M \rightarrow N$ eine Abbildung von M in eine Menge N
- Äquivalenzrelation \equiv_h definiert durch $m \equiv m'$ gdw. $h(m) = h(m')$

Quotientenalgebra:

Σ -Algebra $A/\equiv_h = (M/\equiv_h, \Psi)$, wobei in Ψ für jedes $f/n \in \Sigma$ die Funktion

$$f_{M/\equiv_h}: (M/\equiv_h)^n \rightarrow M/\equiv_h \text{ durch}$$

$$f_{M/\equiv_h}([x_1], \dots, [x_n]) = [f(x_1, \dots, x_n)]$$

definiert ist.



3.4 Beispiel: Zusammenfassung

39/63

Zusammenfassung des **Beispiels**:

- Algebra $Z = (Z, \Phi)$ mit
 - neutralem Element $0_Z: \rightarrow Z \in \Phi$
 - Addition $+_Z: Z \times Z \rightarrow Z \in \Phi$
- Algebra $G = (G, \Psi)$ mit $G = \{0, 1\}$ und
 - neutralem Element $0_G: \rightarrow G \in \Psi$
 - Addition $+_G: G \times G \rightarrow G \in \Psi$
- $h: Z \rightarrow G$ mit $h(n) = \begin{cases} 0 & \text{falls } n \text{ gerade} \\ 1 & \text{falls } n \text{ ungerade} \end{cases}$
definiert Epimorphismus $h: Z \rightarrow G$

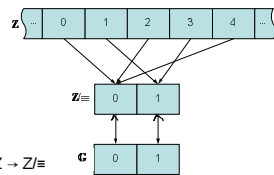


3.4 Beispiel: Zusammenfassung Forts.

40/63

- \equiv mit $n \equiv m$ genau dann, wenn $h(n) = h(m)$ gilt, definiert eine Äquivalenzrelation

- $Z/\equiv = (Z/\equiv, \Phi')$ ist Algebra mit
 - $0_{Z/\equiv} = [0_Z]$
 - $[x] +_{Z/\equiv} [y] = [x +_Z y]$



- $h: Z \rightarrow Z/\equiv$ mit $h(n) = [n]$
definiert Homomorphismus $h: Z \rightarrow Z/\equiv$

- G und Z/\equiv sind Exemplare der gleichen abstrakten Algebra



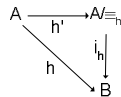
3.4 Homomorphiesatz der Algebra

41/63

Satz: Gegeben seien Σ -Algebren $\mathcal{A} = (M, \Phi)$, $\mathcal{B} = (N, \Psi)$ und ein Homomorphismus $h: \mathcal{A} \rightarrow \mathcal{B}$. Dann gibt es einen Isomorphismus $i_h: M/\equiv_h \rightarrow \text{Bild}(h)$ mit $h = i_h \circ \pi_h$, wobei $\pi_h: M \rightarrow M/\equiv_h$ die Abbildung $\pi_h(x) = [x]$ ist.

Beweis:

- Definiere $i_h: M/\equiv_h \rightarrow N$ durch $i_h([x]) = h(x)$
- i_h ist bijektive Funktion (Übung)
- i_h ist Homomorphismus
- $h = i_h \circ \pi_h$ nach Konstruktion



- Folgerung: Falls $h: \mathcal{A} \rightarrow \mathcal{B}$ ein Epimorphismus ist, so ist \mathcal{B} isomorph zu M/\equiv_h

- praktische Anwendung: Die Gesetze Φ einer abstrakten Datenstruktur \mathcal{A} definieren eine Äquivalenzrelation \equiv und alle Implementierungen \mathcal{B} von \mathcal{A} sind auch Implementierungen von M/\equiv . Wenn man M/\equiv studiert, hat man die Eigenschaften, die allen Implementierungen gemeinsam sind.



3.4 Morphismen mit Signaturwechsel

42/63

Homomorphismen erlauben keinen Signaturwechsel. Wenn es aber eine Zuordnung $\Sigma_A \rightarrow \Sigma_B$ so gibt, daß für alle Operationen aus Σ_A gilt $h(f_A(t_1, \dots, t_n)) = f_B(h(t_1), \dots, h(t_n))$, so gilt der Homomorphiesatz analog.

Beispiel: Dualisierung von booleschen Verbänden:
betrachte boolesche Algebra \mathcal{B} mit Gesetzen V1-V10:
duale Abbildung ersetzt \wedge durch \vee sowie \perp durch \top und umgekehrt.
 \Rightarrow liefert wegen Dualisierungsprinzip auch booleschen Verband.

$$d(\perp) = \top \quad d(\top) = \perp$$

$$d(\vee) = \wedge \quad d(\wedge) = \vee$$

$$d(C) = C$$

Beobachtung: Mit dieser Dualisierung gelten die Gesetze V1-V10.
 d ist ebenfalls ein Homomorphismus (mit Signaturwechsel)



3.5 Termalgebren

43/63

Beispiel: Betrachte Signatur $0: \rightarrow \mathbb{N}$
 $\text{succ}: \mathbb{N} \rightarrow \mathbb{N}$

Terme: $0, \text{succ}(0), \text{succ}(\text{succ}(0)), \dots$

Menge der Terme beschreibt die natürlichen Zahlen.
 \Rightarrow Charakterisierung von \mathbb{N} nach Peano

Weitere **Beobachtung:** Diese abstrakte Algebra hat keine Gesetze.
Terme lassen sich nicht vereinfachen.

Sei

- $\Sigma = (S, F)$ eine Signatur
- $X = (X_s), s \in S$ Familie von Elementaroperanden mit $\Sigma_{t,s} \subseteq X_s$
- $T = (T_s), s \in S$ Familie der Mengen der korrekten Terme der Sorte s

Eine abstrakte Algebra $A = (T, \Sigma, \emptyset)$ heißt **freie Termalgebra** zur Signatur Σ

Unterscheidung verschiedener freien Termalgebren: unterschiedliche Mengen von Konstanten X_s



3.5 Initiale oder Grundtermalgebren

44/63

Sei gegeben:

- $\Sigma = (S, F)$ eine Signatur
- $X = (X_s), s \in S$ Familie der zur Signatur gehörigen Elementaroperanden

Grundterm der Sorte $s \in S$:

Term der Sorte s , wobei **alle** Elementaroperanden zur Signatur gehören (d.h. $X_s = \Sigma_{t,s}$) für alle Sorten $s \in S$.

Notation: $G_0(\Sigma)$ ist die Menge aller Grundterme über der Signatur Σ .

Beispiel: Betrachte Signatur Σ mit Sorte \mathbb{N} und den Operationen

$$0: \rightarrow \mathbb{N}$$

$$\text{succ}: \mathbb{N} \rightarrow \mathbb{N}$$

Dann ist $G_0(\Sigma) = \{0, \text{succ}(0), \text{succ}(\text{succ}(0)), \dots\}$

Grundtermalgebra (initiale Algebra): Σ -Algebra $G = (G_0(\Sigma), \emptyset)$

In der Informatik sind fast alle freien Algebren initiale Algebren



3.5 Algebren mit Axiomen

45/63

Beispiel: Peano Arithmetik mit Axiomen

▪ Signatur:

- 0: $\rightarrow \mathbb{N}$
- succ: $\mathbb{N} \rightarrow \mathbb{N}$
- plus: $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

▪ Axiome:

- P1: plus(n, 0) = n
- P2: plus(n, succ(m)) = succ(plus(n, m))

▪ Grundtermalgebra kennt keine Gesetze.

⇒ **Beispiel:** plus(0, 0) ≠ 0 in Grundtermalgebra

Wie hängt die Grundtermalgebra mit anderen Algebren, insbesondere abstrakten Algebren, zusammen?



3.5 Anwendung des Homomorphiesatzes

46/63

Beispiel: Betrachte Grundtermalgebra G zu folgender Signatur:

- 0: $\rightarrow \mathbb{N}$
- succ: $\mathbb{N} \rightarrow \mathbb{N}$
- plus: $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

Sei A = (N, Φ) Algebra mit

- $0_N: \rightarrow \mathbb{N}$ $0_N = 0$
- $\text{succ}_N: \mathbb{N} \rightarrow \mathbb{N}$ $\text{succ}_N(n) = n + 1$
- $\text{plus}_N: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ $\text{plus}_N(n, m) = n + m$

Homomorphismus h: I → A:

- h(0) = 0_N
- h(succ) = succ_N
- h(plus) = plus_N



3.5 Anwendung des Homomorphiesatzes

47/63

Satz: Sei $\Sigma = (S, F)$ eine Signatur und $\mathcal{A} = (T, \Phi)$ eine Σ -Algebra. Dann existiert ein **Homomorphismus** h: $G_0(\Sigma) \rightarrow \mathcal{A}$.

Beweis:

- alle Konstanten $c \in \Sigma_0$ gehören zu \mathcal{I} , da \mathcal{A} Σ -Algebra
- alle Terme $t = f(t_1, \dots, t_n) \in G_0(\Sigma)$ gehören zu \mathcal{I}
- definiere h: $G_0(\Sigma) \rightarrow \mathcal{A}$ durch h(t) = t (Identität)
- es gilt h(t) = h(t'), wenn in \mathcal{A} gilt t = t' nach den Gesetzen Φ
 - h induziert eine Äquivalenzrelation \equiv in $G_0(\Sigma)$
- Anwendung des Homomorphiesatzes liefert das Ergebnis
 - $\text{Im}(G_0(\Sigma)/\equiv) = \text{Bild}(h)$ ist Unteralgebra von \mathcal{A}

▪ **Einsicht** \mathcal{A} kann nur dann mehr Elemente als Bild(h) umfassen, wenn es in \mathcal{A} Konstante gibt, die nicht Bild von Konstanten in $G_0(\Sigma)$ sind



3.6 Termalgebren mit Variablen

48/63

- Problem: Wie formuliert man Terme M , k , t so, daß gilt
 - für alle Mengen M gilt $M \cup M = M$ oder
 - für alle Keller k und alle Elemente t gilt $\text{top}(\text{push}(k,t)) = t$
- Lösung: Zulassen von Variablen in Termen
 - sie heißen auch logische Variable oder Unbestimmte
 - sie werden durch **Variablenbezeichner** notiert
- **Initiale Termalgebra** $\mathcal{T}(\Sigma, V)$ zur Signatur Σ mit Variablenmenge V
 - Definition der Konstantenmenge X als $X = \Sigma^{(0)} \cup V$
 - V bezeichnet eine abzählbar unendliche Menge von Variablen
 - wenn es noch weitere Konstante gibt, entfällt der Zusatz „initial“
- Einsicht: in jeder Menge von Formeln (Termen), die man hinschreiben kann, kommen nur endlich viele Variable vor, niemals abzählbar unendlich viele!
 - V abzählbar, damit man immer noch eine neue Variable finden kann



3.6 Terme mit Variablen

49/63

Beispiel: Keller

- Signatur:
 - createStack: $\rightarrow \text{stack}(T)$
 - push: $\text{stack}(T) \times T \rightarrow \text{stack}(T)$
 - top: $\text{stack}(T) \rightarrow T$
 - pop: $\text{stack}(T) \rightarrow \text{stack}(T)$
 - isEmpty: $\text{stack}(T) \rightarrow B$
- Axiome:
 - K1: $\text{isEmpty}(\text{createStack}) = \perp$
 - K2: $\text{isEmpty}(\text{push}(k, t)) = \perp$
 - K3: $\text{top}(\text{push}(k, t)) = t$
 - K4: $\text{pop}(\text{push}(k, t)) = k$
 - K5: $\text{top}(\text{createStack}) = \perp$
 - K6: $\text{pop}(\text{createStack}) = \perp$



3.6 Terme mit Variablen

50/63

Beobachtung: Die Axiome benutzen k und t

- diese stehen als Platzhalter für Terme, sog. **Variablen**
- k hat die Sorte $\text{stack}(T)$
- t hat die Sorte T
- Axiom K3 bedeutet z.B., daß für alle Keller k und alle Elemente t $\text{top}(\text{push}(k, t)) = t$ ist.
 - ⇒ gilt für jeden konkreten Keller, den man einsetzt und jedes konkrete Element, das man einsetzt.

Variablen der Sorte s : abzählbare Menge V_s

Variablen $V = \bigcup_{s \in S} V_s$: Wir schreiben $x: s \in V$ statt $x \in V_s$

Terme der Sorte s mit Variablen: Terme der Sorte s mit Konstanten und Variablen als Elementaroperanden.

Termalgebren mit Variablen: Algebra mit Trägermenge $\mathcal{T}(\Sigma, V)$ der Menge aller Terme mit Variablen V über Signatur Σ .



3.6 Substitution

51/63

Ziel: Einsetzen von Termen für Variablen

Substitution: Sei $\Sigma = (S, F)$ Signatur und V Menge von Variablen.

Eine Substitution ist eine Abbildung $\sigma: V \rightarrow \mathcal{T}(\Sigma, V)$ wobei

1. $\sigma(v)$ ist ein Term der Sorte $s \in S$, falls $v: s \in V$
2. $\sigma(v) \neq v$ nur für endlich viele Variablen v

Notation: $\sigma = [t_1/v_1, \dots, t_n/v_n]$: ersetze Variable v_i durch Term t_i .

Warum Bedingung 1? Betrachte z.B. Substitution $\sigma = [1/k, 2/t]$

- Anwenden auf $\text{pop}(\text{push}(k, t))$ ergibt $\text{pop}(\text{push}(1, 2))$
- ⇒ kein korrekter Term, da push als ersten Operanden einen Term der Sorte $\text{stack}(T)$ erwartet.

Warum Bedingung 2? Wir wollen Substitution auf Terme anwenden

- Terme sind endlich
- ⇒ Terme enthalten nur endlich viele Variablen, unendlich viele Variable werden nicht benötigt



3.6 Anwenden einer Substitution

52/63

Ziel: Durchführen des Einsetzens

Beobachtung: Anwendung einer Substitution auf einen Term mit Variablen ergibt einen Term mit Variablen

- ⇒ Einsetzen ist eine Abbildung $\mathcal{T}(\Sigma, V) \rightarrow \mathcal{T}(\Sigma, V)$, wobei Σ eine Signatur und V eine Menge von Variablen ist.

Notation: $t' = t\sigma$ ist der Term, der durch Anwenden der Substitution σ auf den Term t entsteht.

- ⇒ Wir verwenden Postfixnotation



3.6 Anwenden einer Substitution

53/63

Wie wendet man eine Substitution $\sigma = [t'/v]$ auf einen Term t an?

1. Ist t die Variable $v: s \in V$, so ist $t\sigma = t'[t'/v] = v[t'/v] = t'$
2. Ist t eine Konstante c oder eine Variable $\neq v$, so ist $t\sigma = t$
3. Ist $t = f(t_1, \dots, t_n)$ eine Funktion $f: s_1 \times \dots \times s_n \rightarrow s \in \Sigma$ und t_1, \dots, t_n Terme der Sorten s_1, \dots, s_n , so ist $t\sigma = f(t_1\sigma, \dots, t_n\sigma)$

Beobachtung: Die Abbildung „Anwendung einer Substitution“ definiert einen Algebra-Endomorphismus $\sigma: A(\Sigma, V) \rightarrow A(\Sigma, V)$ der Termalgebra A mit Variablen in sich selbst.

- ⇒ Axiome der Urbildalgebra gelten weiterhin!

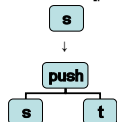


3.6 Anwenden einer Substitution

54/63

Beispiel:

- Substitution: $\sigma = [\text{push}(s, t)/s, 1/t]$



Durchführen der Substitution:

$$\text{push}(\text{push}(s, t), t)\sigma$$

$$= \text{push}(\text{push}(s, t)\sigma, t\sigma)$$

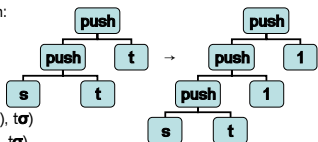
$$= \text{push}(\text{push}(\text{push}(s, t), t\sigma), t\sigma)$$

$$= \text{push}(\text{push}(\text{push}(\text{push}(s, t), t\sigma), t\sigma), t\sigma)$$

$$= \text{push}(\text{push}(\text{push}(\text{push}(s, t), 1), t\sigma), t\sigma)$$

$$= \text{push}(\text{push}(\text{push}(\text{push}(s, t), 1), 1), t\sigma)$$

$$= \text{push}(\text{push}(\text{push}(\text{push}(s, t), 1), 1), 1)$$



3.6 Passen (engl. *matching*)

55/63

Beispiel: Keller

$$\text{pop}(\text{push}(\text{push}(\text{createStack}, 1), 2)) = \text{push}(\text{createStack}, 1)$$

(Erinnerung: K4: $\text{pop}(\text{push}(k, t)) = k$)

Beobachtung: Es gibt eine Substitution der linken Seite des Axioms, so daß ein Unterterm von $\text{pop}(\text{push}(\text{push}(\text{createStack}, 1), 2))$ entsteht.

Term t paßt auf Term t'

genau dann, wenn es eine Substitution σ gibt, so daß $t\sigma = t'$ gilt.
(man sagt, t' ist **allgemeiner** als t bzw. t ist eine **Spezialisierung** von t')

Beispiel:

- $\text{pop}(\text{push}(\text{push}(\text{createStack}, 1), 2))$ paßt auf $\text{pop}(\text{push}(s, t))$
- $\text{pop}(\text{pop}(\text{push}(\text{push}(\text{createStack}, 1), 2)))$ paßt nicht auf $\text{pop}(\text{push}(s, t))$



3.6 mathematische Eigenschaften der Substitution

56/63

Eigenschaften:

- Assoziativgesetz gilt
- Neutrales Element: leere Substitution $\chi = []$
 - Substitutionen bilden bzgl. Hintereinanderausführung ein Monoid
 - Relation „paßt auf“ ist reflexiv, transitiv und Quasiordnung
 - Relation „paßt auf“ ist Quasiordnung
- Aber keine Halbordnung: für Variable x, y paßt $f(x)$ auf $f(y)$ und umgekehrt
 - starke Zusammenhangskomponenten: Terme, die sich durch Umbenennen von Variablen ineinander überführen lassen
 - „paßt auf“ ist Halbordnung bis auf Umbenennungen
 - größtes Element ist eine einzelne Variable x
 - kleinste obere Schranken existieren



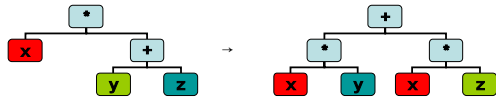
3.7 Beispiel: Distributivgesetz

60/63

Beispiel: Betrachte Ring Z der ganzen Zahlen

▪ Termersetzungsregel $l \rightarrow r$:

- $x * (y + z) \rightarrow x * y + x * z$ (Distributivgesetz)



▪ Wende Regel auf Term $t = (a+b) * ((c+d) + (e*f)) + g$ an:

- $l = x * (y+z)$ paßt auf $t' = (a+b) * ((c+d) + (e*f))$ mit $\sigma = [(a+b)/x, (c+d)/y, (e*f)/z]$
- Damit ergibt sich: $r\sigma = (a+b) * (c+d) + (a+b) * (e*f)$

- Insgesamt folgt $t \rightarrow s$ mit:
 $s = (a+b) * (c+d) + (a+b) * (e*f) + g$

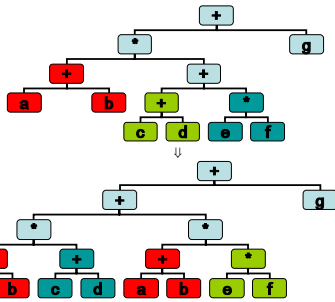


3.7 Beispiel: Distributivgesetz

61/63

▪ Ergebnis der Termersetzung:

$$(a+b) * ((c+d) + (e*f)) + g \Rightarrow (a+b) * (c+d) + (a+b) * (e*f) + g$$



3.7 Beispiel: Sortieren

62/63

Gegeben: Liste L über einer mit \leq total geordneten Menge U

Gesucht: sortierte Liste mit den gleichen Elementen wie L

▪ **Signatur:**

- O : $\rightarrow \mathcal{B}$ Wert falsch
- L : $\rightarrow \mathcal{B}$ Wert wahr
- \leq : $U \times U \rightarrow \mathcal{B}$
- nil : $\rightarrow List(U)$ leere Liste
- $cons$: $U \times List(U) \rightarrow List(U)$ fügt Element vorne ein
- $sort$: $List(U) \rightarrow List(U)$ sortiert Liste
- $insert$: $U \times List(U) \rightarrow List(U)$ fügt in sortierte Liste ein
- if : $B \times List(U) \times List(U) \rightarrow List(U)$ Verzweigung



3.7 Beispiel: Sortieren durch Termersetzung

63/63

▪ Termersetzungsregeln:

- $\text{sort}(\text{nil}) \rightarrow \text{nil}$
- $\text{sort}(\text{cons}(x,l)) \rightarrow \text{insert}(x, \text{sort}(l))$
- $\text{insert}(x, \text{nil}) \rightarrow \text{cons}(x, \text{nil})$
- $\text{insert}(x, \text{cons}(y, l)) \rightarrow \text{if}(x \leq y, \text{cons}(x, \text{cons}(y, l)), \text{cons}(y, \text{insert}(x, l)))$
- $\text{if}(O, l_1, l_2) \rightarrow l_2$
- $\text{if}(L, l_1, l_2) \rightarrow l_1$

- Der Term $x \leq y$ kann direkt durch O und L ersetzt werden.
- auch das kann man durch Termersetzungsregeln ausdrücken

Beispielauswertung:

$\text{sort}(\text{cons}(2, \text{cons}(1, \text{cons}(3, \text{nil}))))$