Beispiel für Unifikationsverfahren

Unifikation zweier Terme s und t nach Robinson:

$s = +(*(2, x), 3)$
$t = +(z, x)$ $\sigma = [\,]$

Schritt

Abweichungspaar

1 $s\sigma = +(*(2, x), 3)$
$t\sigma = +(z, x)$

Fall 2b: $\sigma = [\,]\,[z / *(2, x)]$

Abweichungspaar

2 $s\sigma = +(*(2, x), 3)$
$t\sigma = +(*(2, x), x)$

Fall 2b: $\sigma = [\,]\,[z / *(2, x)]\,[x / 3]$

3 $s\sigma = +(*(2, 3), 3)$
$t\sigma = +(*(2, 3), 3)$

allgemeinster Unifikator: $\sigma = [z / *(2, x)]\,[x / 3] = [z / *(2, 3), x / 3]$

2.3 Algebren

Eine **Algebra** ist eine **formale Struktur** definiert durch eine Trägermenge, Operationen drüber und Gesetze zu den Operationen.

In der Modellierung der Informatik spezifiziert man mit Algebren
Eigenschaften veränderlicher Datenstrukturen und dynamische Systeme,
z. B. Datenstruktur *Keller* oder die Bedienung eines Getränkeautomaten.

Wir unterscheiden 2 Ebenen: abstrakte Algebra und konkrete Algebra:

Eine **abstrakte Algebra** spezifiziert Eigenschaften abstrakter Operationen,
definiert nur duch eine Signatur - Realisierung durch Funktionen bleibt absichtlich offen

Trägermenge: korrekte Terme zu einer Signatur

Gesetze identifizieren Terme mit gleicher Bedeutung:
z. B. $\neg$ false $\equiv$ true pop (push (k, t)) $\equiv$ k

Eine **konkrete Algebra** zu einer abstrakten Algebra

definiert **konkrete Funktionen** zu den Operationen der Signatur,
so dass die Gesetze auch für die Funktionsterme gelten.

Sie beschreibt so eine **Implementierung** der spezifizierten Datenstruktur,
bzw. des Systems

Abstrakte Algebra

Eine **abstrakte Algebra $\mathbf{A}$ = (τ, Σ, Q)** ist definiert durch die **Menge korrekter Terme** τ zur **Signatur** Σ und einer **Menge von Axiomen Q**.

Ein **Axiom** hat die Form $t_1 \equiv t_2$, wobei t_1, t_2, **korrekte Terme gleicher Sorte** sind und **Variable** enthalten können.

Die Algebra definiert alle Paare von Termen als **gleichbedeutend**, die man mit den Axiomen ineinander umformen kann.

Mit Axiomen umformen heißt:
Unter Anwenden eines Axioms $t_1 \equiv t_2$ kann man einen Term s_1 in einen Term s_2 oder s_2 in s_1 umformen, wenn gilt:

- s_1 und s_2 stimmen in ihren „äußeren" Strukturen überein und unterscheiden sich nur durch die Unterterme r_1 und r_2 an entsprechenden Positionen in s_1 und s_2, und
- es gibt eine Substitution σ , sodass gilt $t_1\ \sigma = r_1$ und $t_2\ \sigma = r_2$

$$
\begin{array}{lllllll}
\text{Terme} & s_1 & = & \ldots\ldots\ldots\ r_1\ \ldots\ldots\ldots & & \ldots\ldots\ldots\ r_2\ \ldots\ldots\ldots & = \quad s_2 \\
 & & & \shortparallel & & \shortparallel & \\
 & & & t_1\ \sigma & & t_2\ \sigma & \\
\text{Axiom} & & & t_1 & \equiv & t_2 &
\end{array}
$$

Beispiel: abstrakte Algebra Bool

Signatur Σ = ({BOOL}, F)

Operationen F:

true:		-> BOOL
false:		-> BOOL
$\wedge$:	BOOL x BOOL	-> BOOL
$\vee$:	BOOL x BOOL	-> BOOL
$\neg$:	BOOL	-> BOOL

Axiome Q: für alle x,y der Sorte BOOL gilt

Q_1:	$\neg$ true	$\equiv$ false
Q_2:	$\neg$ false	$\equiv$ true
Q_3:	true $\wedge$ x	$\equiv$ x
Q_4:	false $\wedge$ x	$\equiv$ false
Q_5:	$x \vee y$	$\equiv \neg (\neg x \wedge \neg y)$

Es wird außerdem gefordert, dass true und false **verschiedene** Konstante sind.

Konkrete Algebra

Zu einer abstrakten Algebra $\mathrm{A} = (\tau, \Sigma, Q)$, kann man

konkrete Algebren $\mathbf{A} = (W_A, F_A, Q)$

angeben, wobei

W_A eine **Menge von Wertebereichen**, je einer **für jede Sorte** aus S von Σ

F_A eine **Menge von Funktionen**, je eine **für jede Operation** aus F von Σ.

Die Definitions- und Bildbereiche der Funktionen müssen konsistent den Sorten der Operationen zugeordnet werden.

Die **Axiome Q** müssen mit den Funktionen F_A **für alle entsprechenden Terme aus τ gelten**.

Es können in der konkreten Algebra noch weitere Gesetze gelten.

Eine konkrete Algebra heißt auch **Modell der abstrakten Algebra**.

Beispiel für eine konkrete Algebra

Beispiel: eine konkrete Algebra FSet zur abstrakten Algebra Bool:

	konkrete Algebra FSet	**abstrakte Algebra Bool**
W_A:	$\{\varnothing, \{1\}\}$	Sorte BOOL
F_A:	$\{1\}$	true
	$\varnothing$	false
	Mengendurchschnitt $\cap$	$\wedge$
	Mengenvereinigung $\cup$	$\vee$
	Mengenkomplement bezüglich $\{1\}$	$\neg$

Axiome Q:
Man kann zeigen, dass die Axiome für die entsprechenden Terme mit F_A gelten:
z. B. $\varnothing \cap x \equiv \varnothing$ entspricht false $\wedge$ x $\equiv$ false

Die übliche boolsche Algebra ist natürlich auch eine konkrete Algebra zur abstrakten Algebra Bool.

Beispiel 2.2: Datenstruktur Keller

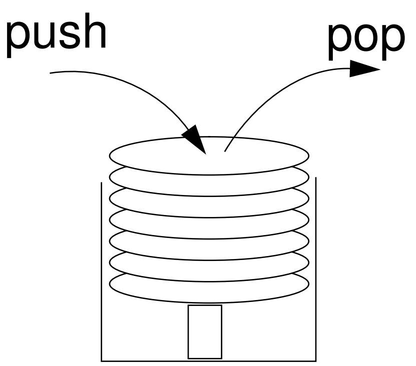
Die Eigenschaften einer **Datenstruktur Keller** beschreiben wir zunächst informell.
Folgende **Operationen** kann man mit einem Keller ausführen:

create Stack:	liefert einen leeren Keller
push:	fügt ein Element in den Keller ein
pop:	entfernt das zuletzt eingefügte Element
top:	liefert das zuletzt eingefügte und nicht wieder entfernte Element
empty:	gibt an, ob der Keller leer ist.

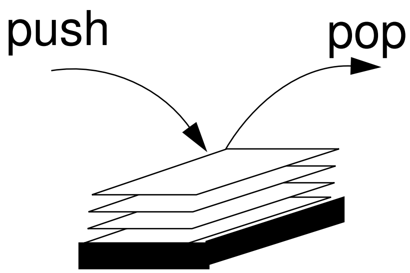
Die Eigenschaften der Datenstruktur Keller sollen präzise durch eine abstrakte Algebra spezifiziert werden.

Beispiele

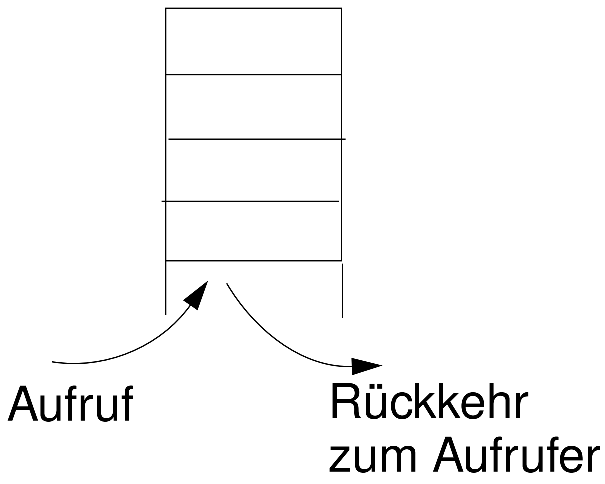
Tellerstapel



Aktenstapel



Laufzeitkeller



Beispiel: Abstrakte Algebra spezifiziert Keller

Abstrakte Algebra Keller:

Signatur Σ = (S, F),
Sorten S = {Keller, Element, BOOL},
Operationen F:

createStack:		-> Keller
push:	Keller x Element	-> Keller
pop:	Keller	-> Keller
top:	Keller	-> Element
empty:	Keller	-> BOOL

Axiome Q: für beliebige Terme t der Sorte Element und k der Sorte Keller gilt:

K1:	empty (createStack)	$\equiv$ true
K2:	empty (push (k, t))	$\equiv$ false
K3:	pop (push (k, t))	$\equiv$ k
K4:	top (push (k, t))	$\equiv$ t

Keller ist die Sorte, deren Terme Kellerinhalte modellieren.
Element und BOOL sind **Hilfssorten** der Algebra.

Implementierungen der abstrakten Algebra Keller können durch **konkrete Algebren** dazu beschrieben werden.

Klassifikation von Operationen

Die Operationen einer Algebra, die die **definierte Sorte als Ergebnissorte** (hier: Keller) haben, kann man in 2 disjunkte Mengen einteilen: **Konstruktoren** und **Hilfskonstruktoren**.

Die Konstruktoren müssen so gewählt sein, dass man **jeden Grundterm mit den Axiomen** in einen gleichwertigen **umformen** kann, der **nur Konstruktoren** enthält. (Ausnahme s. u.)

z. B. in der Keller-Algebra:
Konstruktoren: push, createStack; Hilfskonstruktor: pop
mit dem Axiom K3 kann man Vorkommen der Operation pop aus Termen entfernen

Wir sagen: Grundterme, in denen nur **Konstruktoren** vorkommen, sind in **Normalform**.
z. B. push (.... push (createStack, n_1) , ...), n_m), mit $m \geq 0$

Eigenschaften von Termen in Normalform, kann man auf gleichwertige Terme **übertragen**.

Operationen, die eine **andere Ergebnissorte** haben, heißen **Projektionen**. (hier: top, empty).

Undefinierte Terme:

Terme der definierten Sorte, die man **nicht in eine Normalform** umformen kann, werden als **undefiniert** angesehen. Sie modellieren eine **Fehlersituation**, z. B. pop (createStack)

Manche **Projektionen** sind nicht für jeden Term in Normalform definiert.
Dies modelliert auch **Fehlersituationen**, z. B.top (createStack)