Wertebereiche zur Modellierung des Getränkeautomaten

Folgende Aspekte des Getränkeautomaten
können durch Wertebereiche Modelliert werden:

- Getränkevarianten
- Vorrat an Getränken und Zutaten
- Vorrat an Wechselgeld
- Eingeworfene Münzen
- Betätigte Wahltasten
- Anzeige des Automaten
- Zustand des Automaten
- weitere Aspekte ...

© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 214b

Ziele:

Anregung zur Modellierung des Getränkeautomaten

in der Vorlesung:

Erläuterungen dazu

2.2 Terme
Übersicht

In allen formalen Kalkülen benutzt man **Formeln als Ausdrucksmittel**.
Hier betrachten wir **nur ihre Struktur - nicht ihre Bedeutung**. Wir nennen sie **Terme**.

Terme bestehen aus **Operationen**, **Operanden**, **Konstanten und Variablen**:

a + 5 blau ? gelb = grün ♥ > ♦

Terme werden nicht „ausgerechnet“.
Operationen, Konstanten und Variablen werden als **Symbole ohne Bedeutung** betrachtet.

Notation von Termen:

Infix-, **Postfix**-, **Präfix**- und **Baum**-Form

Umformung von Termen:

Grundlage für die Anwendung von Rechenregeln, Gesetzen

Für **Variable** in Termen werden Terme **substituiert:**

in a + a = 2∗a substituiere a durch 3∗b 3∗b + 3∗b = 2∗3∗b

Unifikation: Terme durch Substitution von Variablen gleich machen,
z. B. um die Anwendbarkeit von Rechenregeln zu prüfen

© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 215

Ziele:

Informelle Übersicht

in der Vorlesung:

Erläuterungen dazu

- Terme als Strukturen erklären.
- Terme umformen ohne zu "rechnen".

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.1

Sorten und Signatur

Terme werden zu einer **Signatur** gebildet.
Sie legt die verwendbaren Symbole und die Strukturierung der Terme fest.

Signatur Σ := (S, F), S ist eine Menge von **Sorten**, F ist eine Menge von **Operationen**.

Sorte s $\in$ **S: Name** für einen Wertebereich; er braucht nicht weiter definiert zu sein.
z. B. STACK, ELEM, BOOL

Operation f $\in$ **F**: **Name** einer Funktion (Operatorsymbol), beschrieben durch
Anzahl der Operanden (**Stelligkeit)**, **Sorten der Operanden** und des **Ergebnisses**

Beispiele:

Name	**Operandensorten**	**Ergebnissorte**	
push:	STACK x ELEM	-> STACK	2-stellig
createStack:		-> STACK	0-stellig
$\wedge$:	BOOL x BOOL	-> BOOL	2-stellig
$\neg$:	BOOL	-> BOOL	1-stellig
T:		-> BOOL	0-stellig

0-stellige Operationen sind Konstante, z. B. T, F, createStack

Beispiel für eine Signatur: Σ_{BOOL} := (S_{BOOL}, F_{BOOL}) mit
S_{BOOL} := { BOOL },
F_{BOOL} := { T: -> BOOL, F: -> BOOL, $\wedge$: BOOL x BOOL -> BOOL, $\neg$: BOOL-> BOOL}

© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 216

Ziele:

Begriff der Signatur verstehen

in der Vorlesung:

- Erläuterung der Begriffe
- Beispiele für Terme zu Signaturen
- Signatur zu STACK-Operationen
- Hinweis: Der Name Signatur wird 2-fach verwendet: wie hier definiert und als "Signatur einer Funktion (siehe Folie Mod-2.11)".
- Hinweis: Im Buch werden die Sorten von Operationen erst später unterschieden.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.1

Korrekte Terme

In **korrekten Termen** muss jeweils die Zahl der Operanden mit der **Stelligkeit** der Operation und die **Sorten** der Operandenterme mit den Operandensorten der Operation übereinstimmen.

Die **Menge τ der korrekten Terme der Sorte s** wird induktiv definiert:
Zu einer Signatur $\Sigma = (S, F)$ ist ein **korrekter Term der Sorte $s \in S$**:

- der **Name einer Variablen** der Sorte s, oder
- die **Anwendung einer n-stelligen Operation**
  $f: s_1 \times s_2 \times \ldots \times s_n \rightarrow s \in \Sigma$ auf Terme t_i: $f(t_1, t_2, .., t_n)$
  wobei jedes **t_i ein korrekter Term der Sorte s_i** ist
  mit $n \geq 0$ (einschließlich Konstante bei $n = 0$) und $i \in \{1, ..., n\}$

Nur die nach diesen Regeln gebildeten Terme gehören zur Menge τ der korrekten Terme.

$f(t_1,..., t_n)$ ist ein **n-stelliger Term**; die t_i sind seine **Unterterme**.

Korrekte Terme**,** die **keine Variablen** enthalten, heißen **Grundterme**.

Beispiele:korrekte Terme zur Signatur Σ_{BOOL}: F $\neg T$ $T \wedge x$ $\neg(a \wedge b)$ $x \wedge \neg y$

nicht korrekt: $a \neg b$ $\neg(\wedge b)$

© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 217

Ziele:

Regeln zur Struktur von Termen

in der Vorlesung:

- Terme zu den Signaturen von Mod-2.16 konsturieren.
- Beispiele für falsche Terme.
- Vergleich mit Typregeln in Programmiersprachen.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.1

Verständnisfragen:

- Welche Terme kann man aus den Operationen 0: -> N0 und succ: N0 -> N0 bilden?
- Geben Sie einige Terme zu den Signaturen von Mod-2.16 an.

Notationen für Terme

Notation eines n-stelligen Terms mit Operation (Operator) f und Untertermen $t_1, t_2, ..., t_n$:

Bezeichnung	Notation	Beispiele
Funktionsform:	Operator **vor der geklammerten Folge** seiner Operanden	
	$f(t_1, t_2, ..., t_n)$	empty (push (createStack, c))
		$\wedge$ (< (0, a), < (a, 10))
Präfixform:	Operator **vor** seinen Operanden	
	$f\ t_1\ t_2 \ldots t_n$	$\wedge < 0\ a < a\ 10$
Postfixform:	Operator **nach** seinen Operanden	
	$t_1\ t_2 \ldots t_n\ f$	$0\ a < a\ 10 < \wedge$
Infixform	2-stelliger Operator **zwischen** seinen (beiden) Operanden	
	$t_1\ f\ t_2$	$0 < a \wedge a < 10$

Die **Reihenfolge der Operanden** ist in allen vier Notationen **gleich**.

© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 218

Ziele:

Verschiedene Notationen für denselben Term

in der Vorlesung:

An weiteren Beispielen erläutern:

- Struktur der Notationen
- Beispiele für 2-, 1- und 3-stellige Operationen
- Umformungen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.1

Verständnisfragen:

- Kennzeichnen Sie alle Teilterme eines Terms in den 4 Formen?
- Wie finden Sie in der Postfixform die Operanden zu einem Operator?
- Können in einem Term in Infixform die Operanden immer eindeutig zugeordnet werden?

Präzedenzen und Klammern

Die **Infixform** benötigt Klammern oder Präzedenzen, um Operanden an ihren Operator zu binden: Ist in x + 3 * y die 3 rechter Operand des + oder linker Operand des * ?

Klammern beeinflussen die Struktur von Termen in der Infixform:
z. B. (x + 3) * y oder x + (3 * y)
Redundante Klammern sind zulässig.
Ein Term ist **vollständig geklammert**, wenn er und alle seine Unterterme geklammert sind:
z. B. ((x) + ((3) * (y)))

Für die **Infixform** können den Operatoren unterschiedliche **Bindungsstärken (Präzedenzen)** zugeordnet werden, z. B. bindet * seine Operanden vereinbarungsgemäß stärker an sich als +, d. h. * hat **höhere Präzedenz** als +.
Damit sind x + 3 * y und x + (3 * y) verschiedene Schreibweisen für denselben Term.

Für **aufeinanderfolgende Operatoren gleicher Präzedenz** muss geregelt werden, ob sie ihre Operanden links-assoziativ oder rechts-assoziativ binden:

links-assoziativ:	x + 3 + y	steht für (x + 3) + y
rechts-assoziativ:	x ** 3 ** y	steht für x ** (3 ** y)

Funktionsform, Präfixform, Postfixform benötigen weder Präzedenzen noch zusätzliche Klammern!

© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 218a

Ziele:

Präzedenzen verstehen

in der Vorlesung:

An weiteren Beispielen erläutern:

- notwendige, redundante und vollständige Klammerung von Termen,
- Verwechselung mit Klammern 1-elementiger Folgen vermeiden,
- Präzedenzen und Assoziativität,
- Präzedenzen in Programmiersprachen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.1

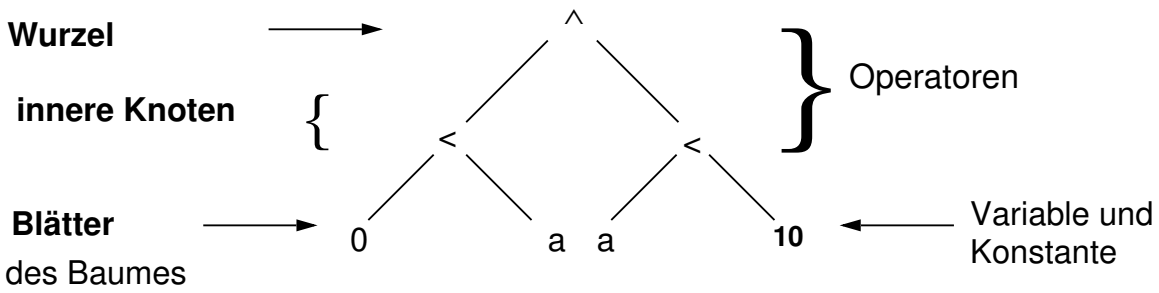
Verständnisfragen:

- Weshalb benötigen Präfix- und Postfixform keine Klammern?
- Welche Präzedenzen haben die Operatoren in Java?

Terme als Bäume

Terme kann man als Bäume darstellen (Kantorowitsch-Bäume):

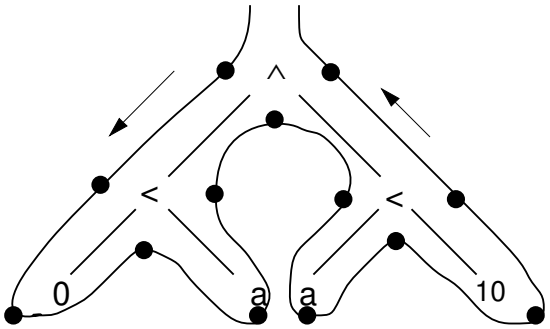
Term: $0 < a \wedge a < 10$



Aus einem Durchlauf des Baumes in Pfeilrichtung erzeugt man

- **Präfixform**, wenn man beim **ersten Besuch**
- **Postfixform**, wenn man beim **letzten Besuch**
- **Infixform**, wenn man beim **vorletzten Besuch** (bei 2-stelligen Operatoren)

den Operator aufschreibt.



© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 219

Ziele:

Zusammenhang der Darstellungen verstehen

in der Vorlesung:

- Bäume erläutern
- Baumdurchläufe erläutern
- Rekursive Definition der Notationen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.1

Verständnisfragen:

- Wie hängen Baumdarstellung und vollständig geklammerte Terme zusammen?

Substitution

Eine **Substitution** beschreibt, wie in einem Term vorkommende **Variable durch Terme ersetzt** werden.

Eine **einfache Substitution** σ **= [v** / **t]** ist ein Paar aus einer Variablen v und einem Term t zur Signatur Σ. v und t müssen **dieselbe Sorte** s haben.
Beispiel: σ = [x / 2*b]

Die **Anwendung einer Substitution** σ **auf einen Term** u schreibt man **u** σ, z. B. (x+1) [x / 2*b].

Die **Anwendung einer Substitution u** σ mit σ **= [v** / **t]**, ist **definiert** durch

- $u\,\sigma = t$, falls u die zu ersetzende Variable v ist,
- $u\,\sigma = u$, falls $u \neq v$ und u eine Konstante oder eine andere Variable ist,
- $u\,\sigma = f\,(u_1\sigma, u_2\sigma, ..., u_n\sigma)$, falls $u = f\,(u_1, u_2, ..., u_n)$

D. h. in u werden **alle Vorkommen der Variablen v gleichzeitig durch den Term t ersetzt**.

Kommt v auch in t vor, so wird es nicht nochmals ersetzt!

Beispiele: (x + 1) [x / 2*b] = (2*b + 1)

(x - x) [x / 3] = (3 - 3)

(x + y) [y / y*y] = (x + y*y)

© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 220

Ziele:

Formale Definition des Einsetzens für Variable

in der Vorlesung:

An Beispielen erlätern:

- konsistentes Ersetzten mehrerer Vorkommen
- gleichzeitiges Ersetzen
- Ersetzen wird nicht iteriert
- Variable können ungebunden bleiben

Hinweis: Wir haben hier **nicht die Notation aus dem Skript vom WS 2000/2001 und nicht die aus dem Buch von Goos verwendet! Dort werden die Paare in umgekehrter Reihenfolge angegeben: [Term/Variable].**

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.6

Verständnisfragen:

Geben Sie Beispiele für Substitutionen zu Termen der Signatur zu BOOL an.

Mehrfache Substitution

Eine **mehrfache Substitution** $\sigma = [\, x_1 / t_1, \ldots, x_n / t_n \,]$ mit paarweise verschiedenen Variablen x_i steht für die
gleichzeitige Substitution aller Vorkommen aller Variablen x_i jeweils durch die Terme t_i.

Beispiele: $\sigma = [\, x / 2*b,\ y / 3 \,]$

$(x + y)\ \sigma = (2*b + 3)$
$(y + a*y)\ \sigma = (3 + a*3)$
$(x * y)\ [\, x / y,\ y / y*y \,] = (y * (y * y))$

Auf einen Term können **mehrere Substitutionen hintereinander** ausgeführt werden,
z. B. $u\ \sigma_1\ \sigma_2\ \sigma_3 = ((u\ \sigma_1)\ \sigma_2)\ \sigma_3$

$(x+y)\ [x/y*x]\ [y/3]\ [x/a] = (y*x+y)\ [y/3]\ [x/a] = (3*a+3)$

Mehrere einfache **Substitutionen hintereinander** kann man **in eine mehrfache Substitution** mit gleicher Wirkung umrechnen:
Die Hintereinanderausführung $[\, x_1 / t_1, \ldots, x_n / t_n \,]\ [\, y / r \,]$

hat auf jeden Term die gleiche Wirkung wie

falls y unter den x_i vorkommt $[\, x_1 / (t_1[\, y / r \,]), \ldots, x_n / (t_n\ [\, y / r \,])]$
falls y nicht unter den x_i vorkommt $[\, x_1 / (t_1[\, y / r \,]), \ldots, x_n / (t_n\ [\, y / r \,]),\ y / r \,]$

Beispiel: $[\, x / y*x \,]\ [\, y / 3 \,]\ [\, x / a \,] = [\, x / 3*x,\ y / 3]\ [\, x / a \,] = [\, x / 3*a,\ y / 3 \,]$

© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 220a

Ziele:

Umgang mit Substitutionen verstehen

in der Vorlesung:

An Beispielen erlätern:

- konsistentes Ersetzten mehrerer Variablen,
- Hintereinanderausführung,
- Umrechnung.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.6

Verständnisfragen:

Begründen Sie die Regel für die Umrechnung.

Umfassende Terme

Rechenregeln werden mit **allgemeineren Termen** formuliert, die auf **speziellere Terme** angewandt werden,

z. B. Distributivgesetz:	$a * (b + c)$	=	$a * b + a * c$
angewandt auf	$2 * (3 + 4*x)$	=	$2 * 3 + 2 * 4*x$

Ein **Term s umfasst einen Term t**, wenn es eine Substitution σ gibt, die s in t umformt: $s\ \sigma = t$

s ist ein **allgemeineres** Muster; es umfasst den **spezielleren** Term t.

s umfasst t, ist eine **Quasiordnung**, d. h. die Relation **umfasst** ist

transitiv:	sei $r\ \sigma_1 = s$, $s\ \sigma_2 = t$, dann ist $r\ (\sigma_1\ \sigma_2) = t$
reflexiv:	$t\ [\ ] = t$, mit der leeren Substitution [ ]
nicht antisymmetrisch:	Terme, die sich nur in den Variablennamen unterscheiden, kann man ineinander umformen, z. B. $2*x\ [\ x / y\ ] = 2*y$ und $2*y\ [\ y / x\ ] = 2*x$

© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 221

Ziele:

Terme als Muster verstehen

in der Vorlesung:

- Substitution als Anwendung einer Rechenregel
- Erläuterung der Relation "umfasst"
- weitere Beispiele

Verständnisfragen:

Warum ist es nicht völlig korrekt, zu sagen, dass t spezieller ist als s, wenn s t umfasst?

Unifikation

Bei der Relation „umfasst“ wird ein Term mit einer Substitution in einen anderen transformiert.

Bei der **Unifikation** werden **zwei Terme so substituiert, dass sie gleich werden**.

Zwei Terme s und t sind unifizierbar, wenn es eine Substitution σ gibt mit s σ = t σ.
σ heißt **Unifikator von s und t**.

Beispiel: Terme: s = (x + y) t = (2 + z)
Unifikatoren: σ_1 = [x / 2, y / z] σ_2 = [x / 2, z / y],
σ_3 = [x / 2, y / 1, z / 1] σ_4 = [x / 2, y / 2, z / 2] ...

Ist σ **ein Unifikator** von s und t und τ **eine Substitution**, dann ist auch die Hintereinanderausführung $\sigma\ \tau = \sigma'$ auch ein Unifikator von s und t.
Wir sagen σ **ist allgemeiner als** σ'**,** denn t σ umfasst t σ'.

Im Beispiel sind σ_1 und σ_2 allgemeiner als σ_3 und σ_4 .

Ein Unifikator σ heißt **allgemeinster Unifikator** der Terme s und t, wenn es zu allen anderen Unifikatoren σ' eine Substitution τ gibt mit $\sigma\ \tau = \sigma'$.

Im Beispiel sind σ_1 und σ_2 allgemeinste Unifikatoren, z. B. σ_1 [z / 1] = σ_3

Es kann **mehrere allgemeinste Unifikatoren** geben. Sie können durch **Umbenennen von Variablen** ineinander überführt werden, z. B. σ_1 [z / y] = [x / 2, y / y, z / y] = σ_2

© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 222

Ziele:

Allgemeines Prinzip Unifikation verstehen

in der Vorlesung:

An Beispielen zeigen:

- Unifikatoren
- nicht unifizierbare Terme
- allgemeinste Unifikatoren machen keine unnötigen Festlegungen.
- Zur letzten Zeile der Folie: Eine Substitution [y/y] hat keine Wirkung, also [y/y] = []. In mehrefachen Substitutionen kann man Komponenten der Form y/y weglassen und Komponenten vertauschen, ohne die Wirkung der Substitution zu ändern.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.6

Verständnisfragen:

- Wie müssen 2 Terme beschaffen sein, damit es Unifikatoren gibt, die verschieden sind von den allgemeinsten Unifikatoren? Hinweis: Nur wenn ein allgemeinster Unifikator noch Variablen offen lässt, kann es speziellere geben.

Unifikationsverfahren

Unifikation zweier Terme s und t nach Robinson:

Seien s und t Terme in **Funktionsschreibweise**.
Dann ist das **Abweichungspaar A(s, t) = (u, v)** das erste Paar unterschiedlicher, korrespondierender Unterterme u und v, das man beim Lesen von links nach rechts antrifft.

Algorithmus:

1. Setze σ = [ ] (leere Substitution)
2. Solange es ein Abweichungspaar A(s σ, t σ) = (u, v) gibt wiederhole:
   a. ist **u eine Variable x**, die in v nicht vorkommt, dann ersetze σ durch σ [x / v], oder
   b. ist **v eine Variable x**, die in u nicht vorkommt, dann ersetze σ durch σ [x / u],
   c. **sonst** sind die Terme s und t **nicht unifizierbar; Abbruch** des Algorithmus.
3. Bei Erfolg gilt s σ = t σ und σ **ist allgemeinster Unifikator**.

Beachte, dass bei jeder Iteration die bisherige Substitution auf die vollständigen Terme s, t angewandt wird.

© 2001 bei Prof. Dr. Uwe Kastens

Vorlesung Modellierung WS 2001/2002 / Folie 223

Ziele:

Terme systematisch unifizieren

in der Vorlesung:

- Verfahren an Beispielen zeigen.
- Begründen, weshalb die Substitution immer wieder auf s und t angewandt wird.
- Begründen, weshalb in 2a und 2b geprüft wird, ob die Variable x in dem Unterterm vorkommt.

Verständnisfragen:

- Zeigen sie an einem Beispiel, dass es nötig ist, in 2a und 2b zu prüfen ob die Variable x in dem Unterterm vorkommt.