LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Skript zur Vorlesung:

Einführung in die Programmierung

WiSe 2009 / 2010

Prof. Dr. Christian Böhm
Annahita Oswald
Bianca Wackersreuther

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

Gliederung

Mengen

- Die Charakterisierung von Daten in der Vorlesung setzt den Mengen-Begriff voraus.
- Als informelle Definition genügt uns:
  Eine *Menge* M ist eine Zusammenfassung von verschiedenen Objekten, den *Elementen* der Menge. Die Notation $a \in M$ bedeutet: a ist ein Element der Menge M.
- Eine Menge kann beliebig viele Elemente enthalten, also z.B. auch gar keine. In diesem Fall spricht man von der leeren Menge, geschrieben $\emptyset$ oder $\{\}$.

Mengen

- Eine Menge kann z.B. *extensional* durch Aufzählung der Elemente definiert werden.
- Die Reihenfolge der Elemente spielt dabei keine Rolle.
- Man kann eine Menge aber auch *intensional* definieren, d.h. durch Angabe einer Bedingung, die alle Elemente und nur die Elemente der Menge erfüllen.
- Beispiel: Menge M der Quadratzahlen, die kleiner als 30 sind.
  - Extensional: $M = \{1, 4, 9, 16, 25\} = \{4, 1, 9, 25, 16\}$
  - Intensional: $M = \{a \mid a \in \mathbb{N}, a \text{ ist Quadratzahl und } a < 30\}$
- Beispiel: leere Menge
  - Extensional: $M = \{\}$
  - Intensional: $M = \{a \mid a \in \mathbb{N}, a < 2 \text{ und } a > 1\}$

Eigenschaften von Mengen

- Alle Elemente einer Menge sind verschieden.
- Man könnte zwar eine Menge {1, 2, 2, 3} angeben, dies wäre aber redundant. Die gleiche Menge wird durch {1, 2, 3} definiert.
- Mit dem Konzept einer Menge kann man also nicht mehrfaches Vorkommen eines gleichen Elementes (Wertes) modellieren.
- Hierzu dient z.B. das Konzept der Multimengen, die wie Mengen geschrieben werden, aber andere Eigenschaften und Rechenregeln haben.
- Soll auch die Reihenfolge der Elemente eine Rolle spielen, sind Folgen eine geeignete Modellierung (wie wir sie in den Algorithmen in Kapitel 2 verwendet haben).

Eigenschaften von Mengen

- Es gelten folgende wichtige Eigenschaften von Mengen und Beziehungen zwischen Mengen:

Bezeichnung	Notation	Bedeutung
M ist Teilmenge von N	$M \subseteq N$	aus $a \in M$ folgt $a \in N$
M ist echte Teilmenge von N	$M \subset N$	es gilt $M \subseteq N$ und $M \neq N$
Vereinigung von M und N	$M \cup N$	$\{x \mid x \in M$ oder $x \in N\}$
Schnittmenge von N und M	$M \cap N$	$\{x \mid x \in M$ und $x \in N\}$
Differenz M ohne N	$M \setminus N$	$\{x \mid x \in M$ und $x \notin N\}$
M und N sind disjunkt	$M \cap N = \emptyset$	M und N haben keine gemeinsamen Elemente
Kardinalität einer Menge M	$\lvert M \rvert$	Anzahl der Elemente von M

Tupel, kartesisches Produkt von zwei Mengen

- Die Elemente einer Menge können selbst zusammengesetzt sein aus verschiedenen Mengen.
- Ein *geordnetes Paar* (*Tupel*) (x, y) besteht aus zwei Werten x und y, wobei x die erste und y die zweite Komponente ist.
- Beispiel: Eine Spielkarte hat eine Farbe und ein Symbol: (*Karo*, *Bube*), (*Herz*, *Dame*).
- Das *kartesische Produkt* $M \times N$ zweier Mengen M und N ist die Menge aller geordneten Paare mit erster Komponente aus M und zweiter Komponente aus N:

$$M \times N := \{(x, y) | x \in M \text{ und } y \in N\}$$

- Beispiel: Für $S = \{7, 8, 9, 10, \textit{Bube, Dame, König, Ass}\}$ und $F = \{\textit{Kreuz}, \textit{Pik}, \textit{Herz}, \textit{Karo}\}$, können wir ein Kartenspiel als die Menge $F \times S$ definieren.

n-Tupel, kartesisches Produkt von n Mengen

- Die Konzepte “Tupel” und “kartesisches Produkt zweier Mengen” lassen sich leicht auf eine beliebige Anzahl n von Mengen verallgemeinern.
- Das allgemeine kartesische Produkt über n Mengen ist die Menge aller geordneten n-Tupel mit den Komponenten aus diesen Mengen:
  $M_1 \times M_2 \times \ldots \times M_n :=$
  $\{(a_1, a_2, \ldots, a_n) \mid a_1 \in M_1 \text{ und } a_2 \in M_2 \ldots \text{ und } a_n \in M_n\}$.
- Sind alle Mengen identisch ($M_i = M_j$ für alle $1 \leq i, j \leq n$), schreibt man für $M \times M \times \ldots \times M$ häufig auch M^n.

Potenzmenge

- Viele Objekte werden durch Mengen beschrieben.
- Der Wertebereich einer Datenmenge solcher Objekte ist dann eine Menge, die Mengen enthält.
- Eine spezielle Form solcher Mengen von Mengen ist die *Potenzmenge*:
- Die Potenzmenge einer Grundmenge U ist die Menge aller Teilmengen von U, geschrieben $\wp(U)$.
- Beispiel: $U = \{d, f, s\}$
  $\wp(U) = \{\emptyset, \{d\}, \{f\}, \{s\}, \{d, f\}, \{d, s\}, \{f, s\}, \{d, f, s\}\}$
- **Für eine Menge der Kardinalität *n*, welche Kardinalität hat ihre Potenzmenge?**

Potenzmenge

- Anwendungsbeispiel:
  Modellieren der Lösungsmenge einer Aufgabe.
- Aufgabe: Anzahl der Münzen, die nötig sind, einen bestimmten Geldbetrag auszuzahlen.
- Je nach Geldbetrag und den Werten der verfügbaren Münzarten gibt es keine, eine oder mehrere Zahlen als Antwort.
- Der Wertebereich der prinzipiell möglichen Lösungen ist $\mathcal{P}(\mathbb{N})$.

Relationen

- Eine n-stellige Relation ist eine Menge von n-Tupeln.
- Prädikate wie z.B. die Beziehung “kleiner” ($a < b$) sind Beispiele für Relationen. Wir können also z.B. schreiben:
  - $< \subseteq \mathbb{N} \times \mathbb{N}$
  - $(1, 2) \in <$
  - $(2, 1) \notin <$
- Eine Relation R ist erfüllt (oder wahr) für alle Tupel a mit $a \in R$ und nur für diese Tupel. Man schreibt auch: Ra.
- Für zweistellige Relationen schreibt man auch xRy (z.B.: $x < y$).

Eigenschaften zweistelliger Relationen

Sei $R \subseteq M \times M$ (d.h. $R \in \wp(M \times M)$).

- R ist *reflexiv*, wenn für alle $x \in M$ gilt: xRx.
- R ist *symmetrisch*, wenn für alle $x, y \in M$ gilt: aus xRy folgt yRx.
- R ist *antisymmetrisch*, wenn für alle $x, y \in M$ gilt:
  aus xRy und yRx folgt $x = y$.
- R ist *transitiv*, wenn für alle $x, y, z \in M$ gilt:
  aus xRy und yRz folgt xRz.
- R ist *alternativ*, wenn für alle $x, y \in M$ gilt: xRy oder yRx.

Äquivalenzrelation

- Sei $R \in \wp(M \times M)$.
- R ist eine *Äquivalenzrelation*, wenn R reflexiv, symmetrisch und transitiv ist.
- Beispiel: Wenn für ein Kartenspiel $F \times S$ mit
  $S = \{7, 8, 9, 10, \textit{Bube}, \textit{Dame}, \textit{König}, \textit{Ass}\}$ und
  $F = \{\textit{Kreuz}, \textit{Pik}, \textit{Herz}, \textit{Karo}\}$
  beim Vergleich zweier Karten die Farbe keine Rolle spielt, sondern zwei Karten mit dem gleichen Symbol als gleich gelten, ist die entsprechende Äquivalenzrelation:

$$\{((f_1, s_1), (f_2, s_2)) \mid f_1 \in F, f_2 \in F, s_1 \in S, s_2 \in S, s_1 = s_2\}$$

Ordnungsrelation

Bestimmte Kombinationen von Eigenschaften qualifizieren eine Relation zur *Ordnungsrelation,* durch deren Anwendung man beispielsweise eine Menge sortieren könnte:
Sei $R \in \wp(M \times M)$.

- R ist eine *partielle Ordnung*, wenn R reflexiv, antisymmetrisch und transitiv ist.
- R ist eine *totale Ordnung*, wenn R eine alternative partielle Ordnung ist.

Auf den ganzen Zahlen gibt es die Ordnungsrelation $\leq \in \wp(\mathbb{Z} \times \mathbb{Z})$. Ist diese Ordnungsrelation total?

Funktionen (Abbildungen)

- Eine *Funktion* ist eine Abbildung von einer bestimmten Menge auf eine bestimmte Menge.
- Funktionen können wir als Relationen mit speziellen Eigenschaften auffassen, sie stellen nämlich eine *rechtseindeutige* Beziehung zwischen Definitionsbereich und Bildbereich dar.
- Formal gesehen ist eine Funktion f eine 2-stellige Relation $f \subseteq D \times B$, für die gilt: Aus $(x, y) \in f$ und $(x, z) \in f$ folgt $y = z$, d.h. einem Element aus D ist höchstens ein Element aus B eindeutig zugeordnet.
- Die Menge D heißt *Definitionsbereich* von f.
- Die Menge B heißt *Bildbereich* von f.
- Als Schreibweisen sind gebräuchlich:

$$(x, y) \in f \Leftrightarrow y = f(x) \Leftrightarrow f(x) = y \Leftrightarrow f : x \mapsto y$$

Signatur von Funktionen

Da Funktionen spezielle Relationen sind, stammen sie aus Wertebereichen, die Teilmengen der Wertebereiche entsprechend strukturierter Relationen sind:

- Die Menge $D \rightarrow B$ ist die Menge aller Funktionen, die D als Definitionsbereich und B als Bildbereich haben. $D \rightarrow B$ enthält als Elemente alle Mengen von Tupeln $(d, b) \in (D \times B)$, die Funktionen sind.
- Es gilt: $D \rightarrow B \subseteq \wp(D \times B)$.
- Für eine Funktion $f \in D \rightarrow B$ gilt: $f \subseteq D \times B$.
- Man schreibt: $f : D \rightarrow B$, d.h. f hat die *Signatur* $D \rightarrow B$.
- "Signatur" ist ein zentrales Konzept in der Spezifikation von Programmen.

Eigenschaften von Funktionen

- Eine Funktion $f : D \rightarrow B$ ist
  - *total*, wenn es für jedes $x \in D$ ein Paar $(x, y) \in f$ gibt;
  - *surjektiv*, wenn es zu jedem $y \in B$ ein Paar $(x, y) \in f$ gibt;
  - *injektiv*, wenn es zu jedem $y \in B$ höchstens ein Paar $(x, y) \in f$ gibt;
  - *bijektiv*, wenn f zugleich surjektiv und injektiv ist.
- Man sagt auch, eine Funktion ist *partiell*, wenn es gleichgültig ist, ob sie total ist oder nicht.
- Vorsicht: Wenn ein Mathematiker nicht angibt, ob eine Funktion total oder partiell ist, meint er in der Regel eine totale Funktion. Für einen Informatiker ist die partielle Funktion der Normalfall.

Stelligkeit von Funktionen

- Funktionen aus $D \to B$ sind n-stellig, wenn der Definitionsbereich D eine Menge von n-Tupeln ist.
- Ist D nicht als kartesisches Produkt strukturiert, so sind die Funktionen aus $D \to B$ 1-stellig, wenn D nicht leer ist, und 0-stellig, wenn D leer ist.
- 0-stellige Funktionen sind *konstante Funktionen,* kurz *Konstanten,* für jeweils einen Wert aus B, d.h. jeder Wert b aus B ($b \in B$) kann als 0-stellige Funktion $b : \emptyset \to B$ aufgefasst werden.
- In diesem Sinne kann man den Ausdruck 1 + 2 als Verschachtelung von mehreren Funktionen auffassen:
  - $1: \emptyset \to \mathbb{N}$
  - $2: \emptyset \to \mathbb{N}$
  - $+: \mathbb{N} \times \mathbb{N} \to \mathbb{N}$

Nochmal: Schreibweisen

- Die Notation $f(x_1, \ldots, x_n)$, die die "Anwendung" von f auf die Argumente $x_1, \ldots, x_n$ beschreibt, heißt *Funktionsschreibweise*.
- Bei 2-stelligen Funktionen verwendet man häufig auch die *Infixschreibweise* $x_1 f x_2$.
  Beispiel: 17 + 8 statt +(17, 8) für die Addition.
- Bei 1-stelligen Funktionen verwendet man gerne die *Präfixschreibweise* $f x_1$.
  Beispiel log 13 statt log(13) für die Logarithmusfunktion.
- Manchmal ist jedoch auch die *Postfixschreibweise* $x_1 \ldots x_n f$ gebräuchlich.
  Beispiel: 21! statt !(21) für die Fakultätsfunktion.
- Bemerkung: In der Mathematik gibt es Schreibweisen, die sich nicht direkt einer dieser Schreibweisen unterordnen lassen, z.B. $\sqrt{x}$ (Quadratwurzel), x^y (Potenzierung), $\frac{x}{y}$ (Division).

Prädikate

- Ein *Prädikat* ist eine Funktion $D \rightarrow \mathbb{B}$ wobei $\mathbb{B} = \{True, False\}$.
- Prädikate sind also eine Abbildung auf die Menge der booleschen Werte, d.h. der Bildbereich dieser Funktionen ist $\mathbb{B}$.
- Beispiel:
  $=: \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{B}$ ist ein Prädikat (die Gleichheitsrelation auf $\mathbb{Z}$).
  Es ist z.B.
  $$= (-321, -321) = TRUE$$
  oder anders ausgedrückt
  $$= (-321, -321) \mapsto TRUE$$
- Wir lassen als Schreibweise "$= TRUE$" meist weg und schreiben bei 2-stelligen Prädikaten in Infixschreibweise

$$-321 = -321.$$

Folgen

- Eine *Folge* $(x_1, \ldots, x_n)$ der Länge n über die Elemente einer Menge M (d.h. $x_i \in M$) ist ein n-Tupel von Werten aus M, d.h.
$$(x_1, \ldots, x_n) \in M_n.$$
- Eine Folge $x \in M_0$ der Länge $n = 0$ wird leere Folge genannt.
- Die Menge aller nicht-leeren Folgen über M wird meist mit
$$M^+ = M_1 \cup M_2 \cup M_3 \cup \ldots$$
bezeichnet.
- Die Menge aller Folgen (auch leerer) über M ist dann
$$M^* = M_0 \cup M^+.$$
- Die Länge einer Folge x wird auch mit $|x|$ bezeichnet.

Folgen

- Die Konkatenation $\circ$ aus dem vorherigen Kapitel ist damit formal eine Abbildung
$$\circ : M^* \times M^* \to M^*.$$
- Für eine formale Definition von $\circ$ benötigen wir zunächst noch die *Projektion*, eine Abbildung
$$\pi : M^n \times I_n \to M.$$
- Die Menge $I_n = \{i \mid i \in \mathbb{N} \text{ und } 1 \leq i \leq n\}$ heißt *Indexmenge*.
- Die Abbildung π bildet eine Folge $x = (x_1, \ldots, x_n)$ der Länge n und ein i $(1 \leq i \leq n)$ auf die i-te Komponente x_i der Folge ab.
- Beispiel: $x = (4, 5, 6)$
$\pi(x, 1) = 4$, $\pi(x, 2) = 5$, $\pi(x, 3) = 6$.

Folgen

- Offensichtlich ist auch $x = (\pi(x, 1), \ldots, \pi(x, |x|))$ eine alternative Schreibweise für eine Folge x.
- Damit können wir die Konkatenation $\circ$ wie folgt formal definieren:

$$x \circ y = (\pi(x, 1), \ldots, \pi(x, |x|), \pi(y, 1), \ldots, \pi(y, |y|)),$$

oder anders ausgedrückt:

$$\pi((x \circ y), i) = \begin{cases} \pi(x, i) & \text{für } 1 \leq i \leq |x| \\ \pi(y, i - |x|) & \text{für } |x| + 1 \leq i \leq |x| + |y| \end{cases}$$

- Beispiel: Sei $M = \mathbb{N}_0$ und $x = (7, 0, 3, 18)$, $y = (21, 3, 7)$, dann ist

$x \circ y = (\pi(x, 1), \pi(x, 2), \pi(x, 3), \pi(x, 4), \pi(y, 5-4), \pi(y, 6-4), \pi(y, 7-4))$
$= (7, 0, 3, 18, 21, 3, 7).$

Beweisprinzip der “vollständigen Induktion”

Ein fundamentales mathematisches Beweisprinzip ist die *vollständige Induktion*:

Sei $p : \mathbb{N}_0 \to \mathbb{B}$ ein totales Prädikat.

Falls

1. $p(0)$ (*Induktionsanfang*) und
2. für beliebiges $n \in \mathbb{N}_0$ gilt der *Induktionsschluss*:
   “Falls $p(n)$ (Induktionsvoraussetzung), dann $p(n + 1)$.”

Dann: $p(n)$ für alle $n \in \mathbb{N}_0$.

Die vollständige Induktion (“nach n”) ermöglicht es zu beweisen, dass eine Aussage (“p”) für alle $n \in \mathbb{N}_0$ gilt.

Beweisprinzip der “vollständigen Induktion”

Beispiel:

- Sei $p(n) : \Leftrightarrow \sum_{i=0}^{n} i = \frac{n \cdot (n+1)}{2}$
- Zu beweisen: Gültigkeit von $p(n)$ für alle $n \in \mathbb{N}_0$
- Induktionsanfang:
  - Zu zeigen, dass $p(0)$, d.h. $\sum_{i=0}^{0} i = \frac{n \cdot (n+1)}{2}$
  - $\sum_{i=0}^{0} i = 0$
  - $\frac{0 \cdot (0+1)}{2} = 0 \surd$
- Für den Induktionsschluss können wir für $n \in \mathbb{N}_0$ als Induktionsvoraussetzung $p(n)$, d.h. $\sum_{i=0}^{n} i = \frac{n \cdot (n+1)}{2}$ annehmen.
- Zu zeigen ist die Gültigkeit von $p(n + 1)$, d.h. $\sum_{i=0}^{n+1} i = \frac{(n+1) \cdot (n+1+1)}{2}$

Beweisprinzip der “vollständigen Induktion”

Beweis (unter Verwendung der Induktionsvoraussetzung):

$$
\begin{aligned}
\sum_{i=0}^{n+1} i &= 0 + 1 + \ldots + n + (n+1) \\
&= \sum_{i=0}^{n} i + (n+1) \\
&= \frac{n \cdot (n+1)}{2} + n + 1 \\
&= \frac{n \cdot (n+1) + 2 \cdot (n+1)}{2} \\
&= \frac{(n+1) \cdot (n+2)}{2} \\
&= \frac{(n+1) \cdot (n+1+1)}{2} \qquad \surd
\end{aligned}
$$

Warum ist dieses Beweisprinzip gültig?

Wie kann man sicher sein, dass p für alle Zahlen aus $\mathbb{N}_0$ gilt, wenn $p(0)$ gilt und man für ein beliebiges festes $n \in \mathbb{N}_0$ von $p(n)$ auf $p(n + 1)$ schließen kann?

Induktive Definition der Menge der natürlichen Zahlen

Die Menge $\mathbb{N}_0$ lässt sich durch folgende Regeln *induktiv definieren*:

1. $0 \in \mathbb{N}_0$
2. Ist $n \in \mathbb{N}_0$ dann ist auch $n + 1 \in \mathbb{N}_0$.
3. Außer den Elementen gemäß Regeln 1 und 2 enthält $\mathbb{N}_0$ keine weiteren Objekte.

Aufbau der Menge der natürlichen Zahlen

Die Elemente der Menge $\mathbb{N}_0$ werden gemäß dieser induktiven Definition der Reihe nach "konstruiert":

- Zunächst wird 0 gemäß Regel 1 als Element von $\mathbb{N}_0$ festgelegt.
- Wegen Regel 2 ist dann 0 + 1 = 1 Element von $\mathbb{N}_0$.
- Erneute Anwendung von Regel 2 ergibt 1 + 1 = 2 als Element von $\mathbb{N}_0$ usw.

  "*n* + 1" können wir auch die *Nachfolger-Funktion* nennen:

  $$\begin{array}{llll} \textit{successor}: & \mathbb{N}_0 & \rightarrow & \mathbb{N}_0 \\ \text{mit} & n & \mapsto & n+1 \end{array}$$

  Jede Zahl aus $\mathbb{N}_0$ wird erzeugt durch endlich-oft-malige Anwendung von *successor* auf 0, z.B.:

  3 = *successor*(*successor*(*successor*(0))), also gilt: $3 \in \mathbb{N}_0$

Vollständige Induktion / Induktive Definition

- Dieser "induktive Aufbau" der Menge $\mathbb{N}_0$ ist der Grund für die Gültigkeit des Beweisprinzips der vollständigen Induktion.
- Das Prinzip der vollständigen Induktion vollzieht genau diesen Erzeugungsmechanismus der Menge $\mathbb{N}_0$ nach:
  - Der Induktionsanfang verifiziert $p(0)$.
  - Mit dem Induktionsschluss, angewendet auf $n = 0$, erhält man $p(0 + 1)$, d.h. $p(1)$.
  - Mit einem weiteren Induktionsschluss, angewendet auf $n = 1$, erhält man $p(1 + 1)$, d.h. $p(2)$ usw.
- Da $\mathbb{N}_0$ nur Elemente enthält, die gemäß der induktiven Definition von $\mathbb{N}_0$ konstruiert sind, gilt dann also p tatsächlich für alle Zahlen aus $\mathbb{N}_0$.

Rekursive Definition von Abbildungen

- Eine weitere Konsequenz der induktiven Definition von $\mathbb{N}_0$ ist die Ermöglichung *der rekursiven Definition* von Abbildungen von $\mathbb{N}_0$.
- Die rekursive Definition einer Funktion *f* mit Definitionsbereich $\mathbb{N}_0$ bedeutet intuitiv:
  - $f(0)$ wird explizit festgelegt.
  - $f(n + 1)$ für ein beliebiges $n \in \mathbb{N}_0$ wird auf $f(n)$ "zurückgeführt", d.h. in Abhängigkeit von $f(n)$ definiert.
  - Die Werte der Funktion $f(0), f(1), f(2)$ usw. sind dann wie zuvor erzeugbar, was $f(m)$ für alle $m \in \mathbb{N}_0$ festlegt.

Beispiel: Fakultäts-Funktion

- Die Fakultäts-Funktion $! : \mathbb{N}_0 \to \mathbb{N}_0$ ist rekursiv definiert wie folgt:
  - $0! = 1$
  - $(n + 1)! = (n + 1) \cdot (n\,!)$
- Oft wird äquivalent statt der Rückführung von $n + 1$ auf n der Fall $n \neq 0$ auf $n - 1$ zurückgeführt:

$$n! = \begin{cases} 1, & \text{falls } n = 0, \\ n \cdot (n-1)! & \text{sonst.} \end{cases}$$

Beispiel: Summen-Formel

- Die Summen-Formel aus dem obigen Beispiel zum Beweis durch vollständige Induktion lässt sich ebenfalls rekursiv definieren:

$$\sum_{i=0}^{n} i = \begin{cases} 0, & \text{falls } n = 0, \\ \sum_{i=0}^{n-1} i + n & \text{sonst.} \end{cases}$$

- Das Beweisprinzip der vollständigen Induktion eignet sich besonders gut, wenn in der zu beweisenden Aussage rekursiv definierte Abbildungen auftreten.

Beispiel: Summen-Formel

- Unabhängig von der Gestalt des Summanden lässt sich eine Summenformel grundsätzlich immer rekursiv definieren.
- Sei $a : \mathbb{N} \to \mathbb{N}$.

$$\sum_{i=0}^{n} a(i) = \begin{cases} 0, & \text{falls } n = 0, \\ \sum_{i=0}^{n-1} a(i) + a(n) & \text{sonst.} \end{cases}$$

Induktive Definition von Folgen

- Folgen haben wir vorher als n-Tupel definiert, also als Elemente aus M^*.
- Hierbei gilt offensichtlich, dass eine Folge der Länge 1 $(a) \in M$ mit ihrem einzigen Element $a \in M$ identisch ist.
- Unter Ausnutzung dieser Eigenschaft können wir Folgen auch induktiv definieren.
- Hilfsfunktionen hierzu ermöglichen das Anfügen eines Elements $a \in M$ an eine Folge $x \in M^*$:

  $$postfix : M^* \times M \to M^*$$
  mit $postfix(x, a) = x \circ (a)$

- Analog: Das Anfügen einer Folge $x \in M^*$ an ein Element $a \in M$:

  $$prefix : M^* \times M \to M^*$$
  mit $prefix(a, x) = (a) \circ x$

Induktive Definition von Folgen

- Damit kann eine Folge $x \in M^*$, $x = (x_1, \ldots, x_n)$ schrittweise aufgebaut werden.
- Ausgehend von der leeren Folge () werden die Elemente $x_1, x_2, \ldots, x_n$ angefügt:

$$x = postfix(\ldots postfix(postfix((), x_1), x_2), \ldots, x_n)$$
$$= () \circ (x_1) \circ (x_2) \circ \ldots \circ (x_n)$$

Induktive Definition von Folgen

Induktive Definition von M^*:

1. $() \in M^*$
2. Ist $x \in M^*$ und $a \in M$, dann ist *postfix* $(x, a) \in M^*$.

Analoge Definition unter Verwendung von *prefix*:

1. $() \in M^*$
2. Ist $a \in M$ und $x \in M^*$, dann ist *prefix* $(a, x) \in M^*$.

Rekursive Funktionen über Folgen

- Da nun Folgen induktiv definiert sind, liegt es nahe, dass wir sehr einfach rekursive Abbildungen über Folgen definieren können.
- $first : M^+ \rightarrow M$ mit $first(prefix(a, x)) = a$
- $rest : M^+ \rightarrow M^*$ mit $rest(prefix(a, x)) = x$
- Die Bedeutung dieser Funktionen ist offensichtlich. Für eine nicht leere Folge $(x_1, \ldots, x_n) = ()$ gilt:

$$first(x_1, x_2, \ldots, x_n) = x_1$$

$$rest(x_1, x_2, \ldots, x_n) = (x_2, \ldots, x_n)$$

Rekursive Funktionen über Folgen

Die *Projektion* auf Folgen

$$\pi : M^n \times I_n \rightarrow M.$$

mit $\pi(x, i) = x_i$ können wir nun auch rekursiv definieren, z.B.:

$$\pi(x, i) = \begin{cases} first(x), & \text{falls } i = 1, \\ \pi(rest(x), i - 1) & \text{sonst.} \end{cases}$$

Verallgemeinerung

- Die induktive Struktur von Folgen lässt sich leicht verallgemeinern.
- Jede nicht-leere Folge ist zusammengesetzt aus einem Element $a \in M$ und einer anderen Folge x:

$$y = prefix(a, x)$$

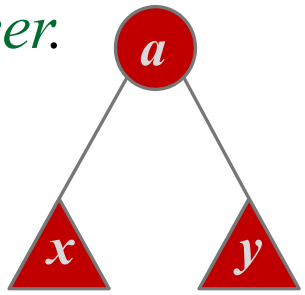
- Abstrahiert von der konkreten Funktion *prefix*, ist y durch das Paar (a, x) bestimmt, wobei x selbst wieder derartig bestimmt ist.
- Eine Verallgemeinerung kann darin bestehen, dass wir Objekte einführen, die aus einem Element a und mehreren "Resten" bestehen:

$$(a, x, y)$$

- Hierbei gilt induktiv, dass die "Reste" x und y selbst von der gleichen Art sind.

Binärbäume

- Solche Objekte heißen *Binärbäume* (über M).
- Analog zu den Folgen lassen wir den *leeren Baum* zu, den wir mit ε bezeichnen.
- Induktive Definition der Menge $binarytree_M$ über M:
  1. $\varepsilon \in binarytree_M$
  2. Wenn $a \in M$ und $x, y \in binarytree_M$, dann ist $(a, x, y) \in binarytree_M$.
- Hierbei heißt a *Wurzel*, x *linker Teilbaum* und y *rechter Teilbaum* eines Binärbaumes (a, x, y).
- Ein Binärbaum der Gestalt $(a, \varepsilon, \varepsilon)$ heißt *Blatt*.
- Ein von ε verschiedener Binärbaum heißt *nicht-leer*.
- Es ist auch üblich, Bäume graphisch darzustellen. Die kanonische Darstellung für (a, x, y) ist:



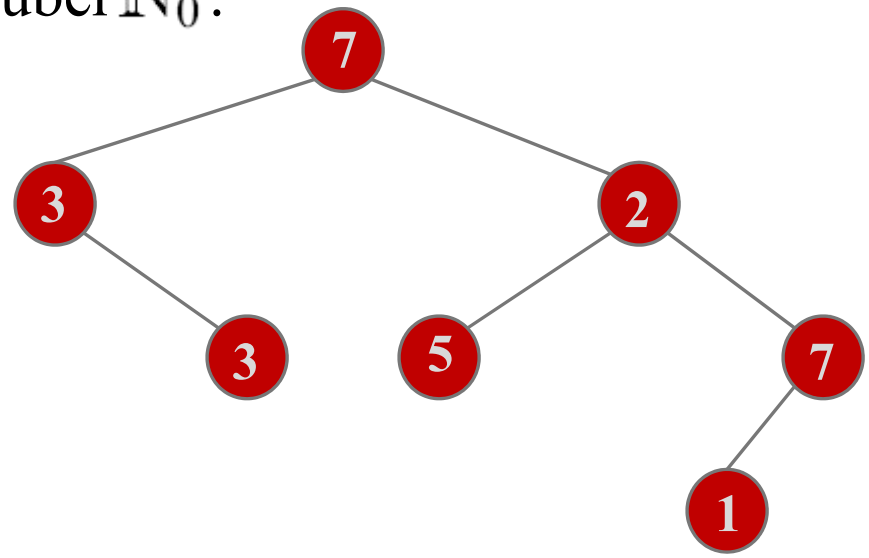
Binärbäume

- Beispiel: Das Objekt

$$(7, (3, \varepsilon, (3, \varepsilon, \varepsilon)), (2, (5, \varepsilon, \varepsilon), (7, (1, \varepsilon, \varepsilon), \varepsilon)))$$

ist ein Binärbaum über $\mathbb{N}_0$.

- Graphisch:



- Die Wurzel des Baumes und die Wurzeln von Teilbäumen (linker bzw. rechter Unterbaum) sind *Knoten*.

Rekursive Funktionen über Binärbäumen

- Entsprechend der induktiven Definition lassen sich auch leicht wieder rekursive Funktionen über Binärbäumen definieren, die auf die einzelnen Elemente zugreifen:
- $root : binarytree_M \setminus \{\varepsilon\} \rightarrow M$
  mit $root(a, x, y) = a$
- $left : binarytree_M \setminus \{\varepsilon\} \rightarrow binarytree_M$
  mit $left(a, x, y) = x$
- $right : binarytree_M \setminus \{\varepsilon\} \rightarrow binarytree_M$
  mit $right(a, x, y) = y$

Rekursive Funktionen über Binärbäumen

Damit können wir z.B. die Anzahl der Knoten bestimmen:

$$nodes : binarytree_M \to \mathbb{N}_0$$

$$nodes(z) = \begin{cases} 0, & \text{falls } z = \varepsilon, \\ 1 + nodes(left(z)) + nodes(right(z)) & \text{sonst.} \end{cases}$$

Ausdrücke

- Betrachten wir folgende Zeichenketten:
  - $a + 5$
  - blau & gelb
- Diese Zeichenketten können wir als Folgen von Symbolen begreifen.
- Eine solche Folge von Symbolen heißt auch *Ausdruck* oder *Term*.
- Das Konzept der Ausdrücke bildet einen Grundbaustein der meisten höheren Programmiersprachen, daher schauen wir uns dieses Konzept im Folgenden etwas genauer an.

Struktur und Bedeutung

- Für einen Ausdruck können wir Regeln aufstellen. Intuitiv ist klar, dass etwa die Symbolfolge “5+” nicht korrekt ist, “5 + 2” oder “*a* + 5” aber schon. Es gibt also offenbar eine *Struktur,* die den Aufbau von korrekten Symbolfolgen (Ausdrücken) beschreibt.
- Außerdem hat eine Folge von Symbolen (ein Ausdruck) eine *Bedeutung* (vorausgesetzt, die verwendeten Symbole haben ihrerseits auch eine Bedeutung). Andernfalls könnte man sagen, “den Ausdruck verstehe ich nicht”.
- Die Struktur von korrekten Ausdrücken können wir auch “Syntax” oder “Grammatik” nennen, die Bedeutung “Semantik”.
- Wir können eine korrekte Struktur beschreiben oder überprüfen, ohne etwas über die Bedeutung zu wissen.

Sorten von Ausdrücken

- Offenbar beschreibt der Ausdruck "*a* + 5" die Addition einer Variablen mit einer Konstanten, der Ausdruck "blau & gelb" eine Farbmischung.
- Die beiden Ausdrücke gehören damit zu unterschiedlichen *Sorten*.
- Wir könnten z.B. festlegen:
  - "*a* + 5" ist ein Ausdruck der Sorte $\mathbb{N}_0$.
  - "blau & gelb" ist ein Ausdruck der Sorte "Farbe".
- Die Bezeichnung "Sorte" anstelle von "Wertebereich" macht deutlich, dass wir zunächst nicht an den konkretenWerten interessiert sind.
- Die Werte sind erst interessant, wenn es um die Bedeutung (Semantik) der Ausdrücke geht.

Struktur von Ausdrücken

- Ausdrücke werden zusammengesetzt aus *Operatoren* und *Variablen*.
- Operatoren bezeichnen Funktionen und haben daher wie diese eine Stelligkeit.
- Wie bei Funktionen stehen 0-stellige Operatoren für Konstanten.
- Variablen sind Namen für Ausdrücke. Jede Variable hat eine Sorte.

Operatoren

- Ein 1-stelliger Operator wird auf einen Ausdruck angewendet.
- Ein *n*-stelliger Operator wird auf ein *n*-Tupel von Ausdrücken angewendet.
- Die *n* Komponenten des *n*-Tupels heißen *Operanden* des Operators, der auf das *n*-Tupel angewendet wird.
- Ein Operator bildet einen Ausdruck einer Sorte, die dem Bildbereich der vom ihm bezeichneten Funktion entspricht.

Einige Operator-Beschreibungen

$$
\begin{array}{rcl}
+ & : & \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0 \\
- & : & \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0 \\
= & : & \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{B} \\
> & : & \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{B} \\
< & : & \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{B} \\
\wedge & : & \mathbb{B} \times \mathbb{B} \to \mathbb{B} \\
\vee & : & \mathbb{B} \times \mathbb{B} \to \mathbb{B} \\
\neg & : & \mathbb{B} \to \mathbb{B} \\
\mathit{TRUE} & : & \emptyset \to \mathbb{B} \\
\mathit{FALSE} & : & \emptyset \to \mathbb{B} \\
0 & : & \emptyset \to \mathbb{N}_0 \\
1 & : & \emptyset \to \mathbb{N}_0
\end{array}
$$

Operatoren und Sorten

- Eine Operatorbeschreibung enthält ein Operatorsymbol und gibt dazu die Signatur der vom Operatorsymbol bezeichneten Funktion an, d.h. die Sorten der Operanden und die Sorte des vom Operator gebildeten Ausdrucks.
- In den Operatorbeschreibungen auf der vorherigen Folie kommen die Sorten $\mathbb{N}_0$ und $\mathbb{B}$ vor.
- Der Operator < verknüpft zwei Ausdrücke der Sorte $\mathbb{N}_0$ (seine Operanden) und bildet einen Ausdruck der Sorte $\mathbb{B}$.
- Der 0-stellige Operator 0 ist ein Ausdruck der Sorte $\mathbb{N}_0$ und hat keine Operanden.

Induktive Definition von Ausdrücken

Sei gegeben:

- Eine Menge von Sorten S,
- eine Menge von Operator-Beschreibungen F und
- eine Menge von Variablen V, die verschieden sind von allen Operator-Symbolen in F.
  Es gilt: $V = \bigcup_{S_i \in S} V_{S_i}$, wobei V_{S_i} die Menge aller Variablen der Sorte $S_i \in S$ bezeichnet.

Die Menge der Ausdrücke ist dann wie folgt induktiv definiert:

- Eine Variable $v \in V$ ist ein Ausdruck.
- Das Symbol op eines 0-stelligen Operators aus F ist ein Ausdruck.
- Sind $a_1, \ldots, a_n$ Ausdrücke der Sorten $S_1, \ldots, S_n$ und $op \in F$ ein Operator der Signatur $S_1 \times \ldots \times S_n \rightarrow S_0$ (wobei $S_0, \ldots, S_n \in S$), dann ist $op(a_1, \ldots, a_n)$ ein Ausdruck der Sorte S_0.

Schreibweisen

Als Schreibweisen für die Anwendung eines mehrstelligen Operators *op* gibt es wie für Funktionen folgendes:

- Funktionsform als $op(a_1, \ldots, a_n)$, wobei auch die a_i in Funktionsform geschrieben sind.
- Präfixform als $op\ a_1 \ldots a_n$, wobei auch die a_i in Präfixform geschrieben sind.
- Postfixform als $a_1 \ldots a_n op$, wobei auch die a_i in Postfixform geschrieben sind.
- Infixform als $(op\ a_1)$ für einstellige Operatoren *op*, $(a_1\ op\ a_2)$ für zweistellige Operatoren *op*, wobei auch die a_i in Infixform geschrieben sind. Bei höherer Stelligkeit als 2 kann der Operator *op* auch aus mehreren Teilen $op_1 \ldots op_{n-1}$ bestehen: $(a_1\ op_1\ a_2 \ldots op_{n-1} a_n)$. Wenn die Zuordnung der Operanden zu den Operatoren eindeutig ist, kann die Klammerung entfallen. Die hier definierte Form heißt *vollständig geklammert*.

Schreibweisen

Folgende Funktion *f* ist bestimmt durch den Ausdruck in Infixform "$(x \wedge z) \vee (y \wedge z)$". Hierbei sind x, y, z Variablen, $\vee$ und $\wedge$ Operatoren (aus der obigen Menge von Operator-Beschreibungen). Die äußeren Klammern sind weggelassen, da die Zuordnung der Teilausdrücke zu den Operatoren eindeutig ist. Übersetzt in die anderen Schreibweisen lautet der Ausdruck:

- Präfixform: $\vee \wedge x\ z \wedge y\ z$
- Postfixform: $x\ z \wedge y\ z \wedge \vee$
- Funktionsform: $\vee(\wedge(x, z), \wedge(y, z))$

Bindung

- Während man in der Präfixform und in der Postfixform auf Klammern verzichten kann (**Warum eigentlich?**), ist Klammerung in der Infixform grundsätzlich nötig, um Operanden ihren Operatoren eindeutig zuzuordnen.
- Lassen wir die Klammern im Ausdruck "$(x \wedge z) \vee (y \wedge z)$" weg, so erhalten wir $x \wedge z \vee y \wedge z$
- Nun wäre auch eine völlig andere Bindung möglich, z.B. $x \wedge ((z \vee y) \wedge z)$

Bindungsstärke (Präzedenz)

- Eindeutigkeit ohne Klammerung kann man in der Infixform erreichen, indem man den Operatoren unterschiedliche *Bindungsstärken* (Präzedenzen) zuweist.
- Beispielsweise hat unter den logischen Operatoren ¬ höhere Präzedenz als ∧ und ∧ hat höhere Präzedenz als ∨.
- Damit erhalten wir auch für den Ausdruck

  $$x \wedge z \vee y \wedge z$$

  die ursprünglich gewünschte, nun *implizite* Klammerung:

  $$(x \wedge z) \vee (y \wedge z)$$

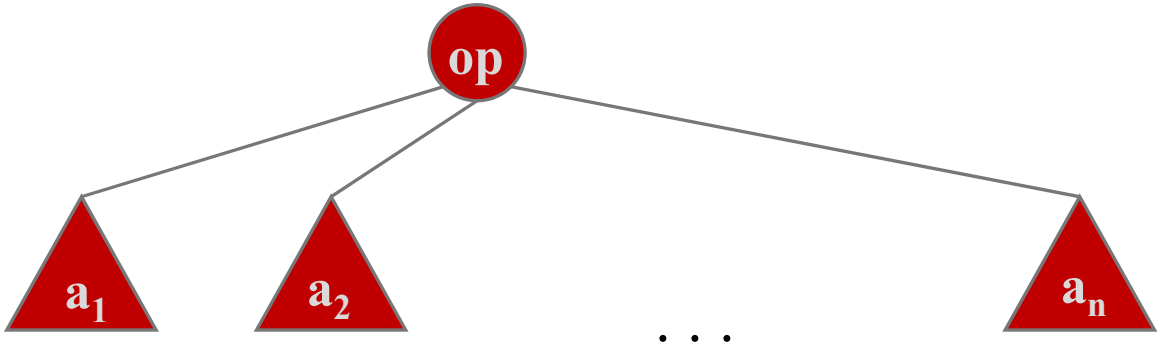
Links-/Rechts-Assoziativität

- Wenn Operatoren gleicher Bindungsstärke konkurrieren, muss außerdem entschieden werden, ob der linke oder der rechte "gewinnt".
- Man legt hierzu fest, ob die Operanden eines Operators *linksassoziativ* oder *rechtsassoziativ* binden.
- Beispiel: Der Ausdruck "$x \vee y \vee z$" bedeutet
  - linksassoziativ:
    $$(x \vee y) \vee z$$
  - rechtsassoziativ:
    $$x \vee (y \vee z)$$
- Die linksassoziative Bindung ist gebräuchlicher.
- Natürlich ist das Setzen von Klammern dennoch erlaubt und sogar sinnvoll, um anzuzeigen, dass die implizite Klammerung der erwünschten Klammerung entspricht.

Baumdarstellung

Eine wichtige Art der Darstellung von Ausdrücken mit eindeutiger Zuordnung von Operanden zu Operatoren (durch implizite Klammerung) ist die *Baumdarstellung von Ausdrücken*:

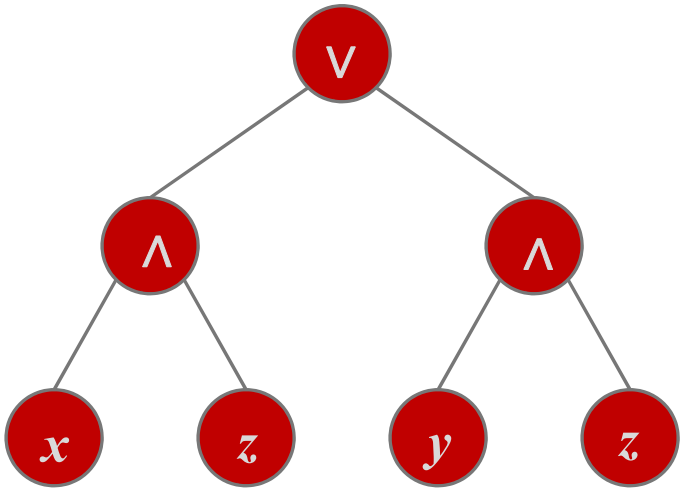
- Eine Variable *v* wird durch den Knoten *v* dargestellt.
- Ein *n*-stelliger Operator *op* wird durch den Knoten *op* mit den *n* Operanden als Teilbäumen dargestellt.



- Hierbei ist  die Baumdarstellung des Ausdrucks a_i.

Beispiel

Die Baumdarstellung des Ausdrucks "$(x \wedge z) \vee (y \wedge z)$" ist demnach:



Baumdurchlauf

- Da die Baumdarstellung (wie ja auch die Ausdrücke selbst) induktiv definiert ist, lassen sich auch naheliegende Funktionen leicht rekursiv definieren, die die Baumdarstellung auf die Präfixform, Postfixform bzw. Infixform abbilden.
- Man durchläuft dazu den Baum links-abwärts:
  1. Besuche den Wurzelknoten.
  2. Wenn der Wurzelknoten kein Blatt ist, führe für jeden Unterbaum *U* folgende Aktionen durch:
     a) Durchlaufe *U* links-abwärts
     b) Besuche dann wieder den Wurzelknoten.

Baumdurchlauf

- Man erhält die Präfixform, wenn man den Knotennamen beim ersten Besuch eines Knotens ausgibt.
- Man erhält die Postfixform, wenn man den Knotennamen beim letzten Besuch eines Knotens ausgibt.
- Man erhält die Infixform, wenn man beim ersten Besuch die öffnende, beim letzten die schließende Klammer ausgibt und bei den dazwischenliegenden Besuchen die Operatorteile.
- Bemerkung: Dieses Verfahren verdeutlicht, dass die verschiedenen Schreibweisen äquivalent sind.

Überladung von Operatoren

Betrachten Sie nun folgende Menge F von Operatorbeschreibungen:

$$
\begin{array}{rcl}
+ & : & \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{N}_0 \\
0 & : & \emptyset \to \mathbb{N}_0 \\
1 & : & \emptyset \to \mathbb{N}_0 \\
+ & : & \mathbb{R} \times \mathbb{R} \to \mathbb{R} \\
0.0 & : & \emptyset \to \mathbb{R} \\
1.0 & : & \emptyset \to \mathbb{R}
\end{array}
$$

- "0 + 1" ist ein gültiger Ausdruck.
- "0.0 + 1.0" ist ein gültiger Ausdruck.
- "0.0 + 1" ist **kein** gültiger Ausdruck.

Überladen

- Das Operatorsymbol “+” kommt in *F* zweimal vor, erhält aber jeweils eine unterschiedliche Beschreibung (Signatur).
- Das Operatorsymbol “+” beschreibt also in *F* zwei unterschiedliche Funktionen.
- Man sagt: Das Operatorsymbol “+” ist *überladen*.
- Durch die Erfüllbarkeit einer Signatur kann man entscheiden, welcher Operator tatsächlich zu verwenden ist.
- Der Ausdruck “0.0 + 1” erfüllt in *F* keine Signatur.

Bedeutung

- Um die *Bedeutung* (Semantik) eines Ausdrucks festzulegen, müssen einerseits die Funktionen, die von den Operatoren bezeichnet werden, definiert sein.
- Andererseits müssen alle Variablen ersetzt werden können durch den von ihnen bezeichneten Ausdruck.
- Dann können alle im Ausdruck vorkommenden Operationen ausgeführt werden.
- Schließlich kann man für einen Ausdruck einen Wert einsetzen.

Bedeutung

Beispiel:
Um den Ausdruck “2 + 5” zu interpretieren, vereinbaren wir:

- Der Operator 2 steht für die Zahl 2 $\in \mathbb{N}_0$.
- Der Operator 5 steht für die Zahl 5 $\in \mathbb{N}_0$.
- Der Operator + steht für die arithmetische Addition zweier natürlicher Zahlen.

Durch Anwendung der vereinbarten Funktionen erhalten wir als Wert des Ausdrucks die Zahl 7 $\in \mathbb{N}_0$.

Substitution

- Um Variablen durch die von ihnen bezeichneten Ausdrücke zu ersetzen, muss wiederum die Bedeutung einer Variablen vereinbart werden.
- Aus einer Liste von vereinbarten Bedeutungen von Variablen kann man dann die Ersetzung (*Substitution*) der Variablen in einem Ausdruck vornehmen.
- Als einfache Substitution kann man ein Paar σ aus einer Variablen v und einem Ausdruck a $\sigma = [v/a]$ ansehen, wobei v und a zur selben Sorte gehören müssen.
- Festzustellen, ob zwei Ausdrücke zur selben Sorte gehören, ist nicht trivial und wird unter dem Thema “Unifikation” studiert.

Anwendung einer Substitution

Die Anwendung einer Substitution $\sigma = [v/t]$ auf einen Ausdruck u wird geschrieben

$$u\sigma \text{ bzw. } u[v/t].$$

Das Ergebnis dieser Anwendung ist ein Ausdruck, der durch einen der folgenden drei Fälle bestimmt ist:

1. $u[v/t] = t$, falls u die zu ersetzende Variable v ist,
2. $u[v/t] = u$, falls u ein 0-stelliger Operator oder eine andere Variable als v ist,
3. $u[v/t] = f(u_1[v/t], u_2[v/t], \ldots, u_n[v/t]))$, falls $u = f(u_1, u_2, \ldots, u_n)$.

Anwendung einer Substitution

Beispiel:
Sei $\sigma = [x/2 * b]$, angewendet auf den Ausdruck "$x + x - 1$".
In Funktionsform notiert:

$$\begin{aligned} -(+(x,x),1)[x/*(2,b)] &= -(+(x,x)[x/*(2,b)],1[x/*(2,b)]) \\ &= -(+(x,x)[x/*(2,b)],1) \\ &= -(+(x[x/*(2,b)],x[x/*(2,b)]),1) \\ &= -(+(*(2,b),*(2,b)),1) \end{aligned}$$

Bemerkungen

- Ausdrücke stellen ein klassisches funktionales Konzept dar: Der Ausdruck 3 + 5 hat offenbar den Wert $8 \in \mathbb{N}_0$.
- Die Funktionen, die durch die Operatoren 3, + und 5 bezeichnet werden, sind in einer vollständigen Semantik definiert.
- Wie der Wert dieses Ausdrucks auf einem Rechner konkret berechnet wird, ist typischerweise aber nicht näher spezifiziert.
- Die formalere Beschäftigung mit Ausdrücken sowie die hier nur skizzierten Problembereiche
  - Sorten und Überladung,
  - Definition und Eigenschaften rekursiver Funktionen, z.B. für Baumdurchläufe,
  - Substitution und Unifikation

  bilden einen wichtigen Inhalt der Vorlesung “Programmierung und Modellierung”.

Bemerkungen

- Ausdrücke bilden einen wichtigen Grundbaustein in den meisten höheren Programmiersprachen, funktionalen wie imperativen, auch in Java.
- Die meisten höheren Programmiersprachen verwenden funktionale *und* imperative Konzepte. Java ist – neben der Möglichkeit, Daten objektorientiert zu modellieren – insbesondere als imperative Sprache angelegt. Wir werden aber sehen, dass Ausdrücke ein wichtiger Bestandteil von Anweisungen sind.
- Die Schreibweise, die wir hier für Ausdrücke kennengelernt haben, ist sowohl menschenlesbar als auch maschinenlesbar.
- Auch Programme müssen sowohl maschinenlesbar (also der vom Compiler zu überprüfenden Syntax folgen), als auch menschenlesbar sein, da es für Menschen auf die korrekte Semantik überprüfbar sein muss.