Überblick

3. Mathematische Grundlagen
3.1 Mengen und Abbildungen
3.2 Induktion und Rekursion
3.3 Ausdrücke

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" ein 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 konkreten Werten 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 $S_1 \times \ldots \times S_n \to S_0$ (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 die

- 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

Die Funktion f_2 aus Übungsaufgabe 2-1 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 $\neg$ höhere Präzedenz als $\wedge$, $\wedge$ hat höhere Präzedenz als $\vee$.
- Damit erhalten wir auch für den Ausdruck

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

die ursprünglich gewünschte, nun *implizite* Klammerung:

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

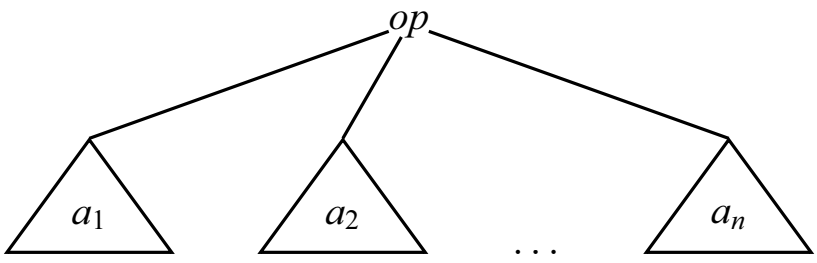
Links-/Rechts-Assoziativität

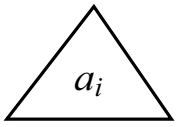
- Wenn Operatoren gleicher Bindungsstärke konkurrieren, muss außerdem entschieden werden, ob der linke oder der rechte "gewinnt".
- Man legt hierzu fest, ob die Operanden eines Operators *linksassoziativ* oder *rechtsassoziativ* binden.
- Beispiel: Der Ausdruck "$x \vee y \vee z$" bedeutet
  - linksassoziativ:
    $$(x \vee y) \vee z$$
  - 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 weitere wichtige Art der Darstellung von Ausdrücken mit eindeutiger Zuordnung von Operanden zu Operatoren (durch implizite Klammerung) ist die *Baumdarstellung von Ausdrücken*:

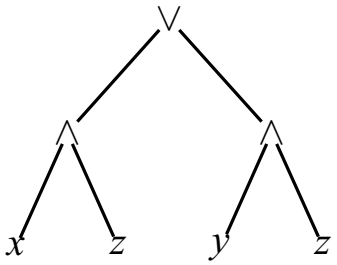
- Eine Variable v wird durch den Knoten v dargestellt.
- Ein n-stelliger Operator op wird durch den Knoten op mit den n Operanden als Teilbäumen dargestellt.



- 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 defniert 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. durchlaufe jeden Unterbaum links-abwärts
  3. 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 und einem Ausdruck $\sigma = [v/a]$ 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, d.h. Algorithmen spezifizieren die Abfolge von Aktionen statt den funktionalen Zusammenhang zwischen Eingabe und Ausgabe. 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 (obwohl beide Interpretationsprozesse z.T. sehr unterschiedlich ablaufen).
- Dasselbe ist grundsätzlich für Programme zu fordern: Ein Programm muß maschinenlesbar sein (also der vom Compiler zu überprüfenden Syntax folgen) als auch menschenlesbar, da es für Menschen auf die korrekte Semantik überprüfbar sein muss.