Kapitel 3: Funktionale Algorithmen i

1. Sorten und abstrakte Datentypen

2. Ausdrücke

3. Funktionale Algorithmen

4. Variablen, Anweisungen, Prozeduren

5. Prozeduraufrufe

6. Variablen, Anweisungen und Prozeduren in Java

Kapitel 3: Funktionale Algorithmen ii

7. Bedingte Anweisungen und Iteration

8. Verzweigung/Iteration in Java

9. Strukturierung von Programmen

- Das Konzept der Ausdrücke erlaubt uns nun, einfache funktionale Algorithmen zu entwerfen.
- Zur Veranschaulichung dient folgendes Beispiel aus der elementaren Physik:

  Ein frei beweglicher Körper mit der Masse m (> 0) werde aus der Ruhelage eine Zeit t lang mit einer auf ihn einwirkenden konstanten Kraft k bewegt. Gesucht ist ein Algorithmus, der (in Abhängigkeit von m, t und k) die Strecke s bestimmt, um die der Körper aus seiner ursprünglichen Lage fortbewegt wird.

- Die Datendarstellung ist in diesem Beispiel sehr einfach: Die Größen m, t und k sind offensichtlich aus $\mathbb{R}$ (mit $m > 0$ und $t \geq 0$).
- Der gesuchte Algorithmus kann damit als Abbildung (Funktion)

$$\mathit{strecke} : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

formuliert werden, wobei die Berechnungsvorschrift explizit (als Ausdruck) unter Verwendung von Grundoperationen angegeben werden soll.

- Die Idee für diesen Algorithmus ergibt sich aus folgenden einfachen Gesetzen:
  - Die auf einen Körper mit Masse m einwirkende Kraft k erzeugt eine konstante Beschleunigung

$$b = \frac{k}{m}$$

  - Der in der Zeit t zurückgelegte Weg bei einer Bewegung mit konstanter Beschleunigung b ist

$$s = \frac{1}{2} \cdot b \cdot t^2$$

  - Einsetzen liefert den fertigen (funktionalen) Algorithmus:

$$\mathit{strecke}(m, t, k) = (k \cdot t^2)/(2 \cdot m)$$

Erweiterung des Modulkonzepts

- Wir können nun das Modulkonzept wie angekündigt erweitern.
- Zur Erinnerung: ein Modul ist eine Menge von Sorten mit einer Menge von Operationen über diesen Sorten.
- Wir können nun die Menge der Sorten $\{\mathbb{R}\}$ und die Menge der Operationen bestehend aus *strecke* als Modul (das wir z.B. *Bewegung* nennen könnten) auffassen.
- Anders als bisher, sind die Operationen (in diesem Fall die Funktion *Strecke*) nun nicht mehr abstrakt, sondern durch einen (funktionalen) Algorithmus explizit angegeben.

Erweiterung des Modulkonzepts

MODULE BEWEGUNG

SORTS $\mathbb{R}$

OPS

FUNCTION $strecke(m : \mathbb{R}, t : \mathbb{R}, k : \mathbb{R}) \rightarrow \mathbb{R}$

OUTPUT Bewegte Strecke eines Körpers mit Masse m bei Krafteinwirkung k nach Zeit t

PRE $m > 0, t \geq 0$

BODY $(k \cdot t^2)/(2 \cdot m)$

Erweiterung des Modulkonzepts

- Unter **`OPS`** stehen nach wie vor alle Funktionsvereinbarungen des Moduls (hier nun nur die von *strecke*).
- Für die einzelnen Operationen sind angegeben (am Bsp. *strecke*):
  - Die Signatur der Operation (nach dem Schlüsselwort **`FUNCTION`**).
  - Optionale Beschreibung des Ergebnis (nach **`OUTPUT`**).
  - Evtl. *Vorbedingungen*, zur Einschränkung des Def.bereichs (nach Schlüsselwort **`PRE`**).
  - Der eigentliche *Funktionsrumpf* (Berechnungsvorschrift, Algorithmus) hinter dem Schlüsselwort **`BODY`**.

Erweiterung des Modulkonzepts

- *m*, *t* und *k* sind Variablen der Sorte $\mathbb{R}$, die sog. (*formalen*) *(Eingabe-) Parameter* von *strecke*.
- Im Funktionsrumpf darf ein beliebiger Ausdruck stehen; als Variablen für diesen Ausdruck sind nur die formalen Parameter des Algorithmus erlaubt.
- Der Vollständigkeit halber: Im Rumpf der Funktion *strecke* haben wir eine implizite Sortenanpassung verwendet!

Benutzerdefinierte Funktionen in Ausdrücken

- Allgemein können Module mehrere Funktionen zusammenfassen, d.h. auch unser Modul *Bewegung* könnte weitere Funktionen (z.B. die Endgeschwindigkeit ($k/m \cdot t$) oder die verrichtete Arbeit ($k \cdot \mathit{strecke}(m, t, k)$)) bereitstellen.
- Das können auch 0-stellige Funktionen sein, z.B.

  FUNCTION *PI* $:\to \mathbb{R}$
  OUTPUT Annährung der Zahl π
  BODY 3.14159

  die als „Synonym“ für Literale der Sorte des Bildbereichs (hier $3.14159 \in \mathbb{R}$) aufgefasst werden können.

Benutzerdefinierte Funktionen in Ausdrücken

- Die Funktion *strecke* kann dabei auch innerhalb der anderer Algorithmen *verwendet* (*aufgerufen*) werden genauso wie die bisherigen Basisoperationen.
- Dazu kann die Syntax von Ausdrücken entsprechend erweitert werden:

  Sind $a_1, \ldots, a_n$ Ausdrücke der Sorten $S_1, \ldots, S_n$ und f eine benutzerdefinierte Funktion (Algorithmus) eines konkreten Moduls mit der Signatur $f : S_1 \times \ldots \times S_n \rightarrow S_{n+1}$ (wobei $S_1, \ldots, S_n \in S$), dann ist $f(a_1, \ldots, a_n)$ ein Ausdruck der Sorte S_{n+1}.
- D.h. ist eine Funktion einmal implementiert, kann sie in Ausdrücken (also insbs. Rümpfen anderer Funktionen) wie jedes andere Operationssymbol verwendet werden.

Erweiterung des Modulkonzepts

Als Beispiel: das Modul BEWEGUNG mit weiteren Funktionen:

MODULE BEWEGUNG

OPS

FUNCTION *strecke*$(m : \mathbb{R}, t : \mathbb{R}, k : \mathbb{R}) \to \mathbb{R}$
OUTPUT Strecke
PRE $m > 0, t \geq 0$
BODY $(k \cdot t^2)/(2 \cdot m)$

FUNCTION *endgeschwindigkeit*$(m : \mathbb{R}, t : \mathbb{R}, k : \mathbb{R}) \to \mathbb{R}$
OUTPUT Endgeschwindigkeit
PRE $m > 0, t \geq 0$
BODY $(k/m) \cdot t$

FUNCTION *arbeit*$(m : \mathbb{R}, k : \mathbb{R}, t : \mathbb{R}) \to \mathbb{R}$
OUTPUT Arbeit
PRE $m > 0, t \geq 0$
BODY $k \cdot$ *strecke*(m, t, k)

Funktionsausführung/-aufruf

- Nun, da wir verschiedene Algorithmen „programmiert“ haben, wollen wir sie natürlich auch benutzen können, z.B. wollen wir die Strecke, die ein Körper mit Masse 1.0 zurücklegt, wenn er mit einer Kraft von 3.0 über 2.0 Zeiteinheiten bewegt wird[6], mit Hilfe der Funktion *strecke* berechnen.
- Die Ausführung eines Algorithmus ist ein *Funktionsaufruf* mit konkreten Eingabewerten: $strecke(1.0, 2.0, 3.0)$
- Was passiert bei so einem Funktionsaufruf?

[6] Über die physikalischen Maß-Einheiten haben wir uns bisher nicht gekümmert und tun dies hier auch nicht

Funktionsausführung/-aufruf

- Der Aufruf des Algorithmus *strecke* mit konkreten Werten ist nichts anderes als eine Substitution
  - Der Algorithmus wird durch einen Ausdruck definiert.
  - Der *Wert* dieses Audrucks ergibt sich aus der Substitution der Eingabevariablen durch die entsprechenden Werte, mit denen der Algorithmus aufgerufen wird.
  - Diese Substitution σ wird auch *Variablenbelegung* genannt; die Eingabevariablen werden jeweils mit einem Literal der entsprechenden Sorte belegt.
  - $V(\sigma)$ bezeichnet die Menge der (Eingabe-)Variablen in dieser Substitution.
  - z.B. beim Aufruf von *strecke*$(1.0, 2.0, 3.0)$ werden die Variablen m,t,k mit den Literalen 1.0,2.0,3.0 $\in \mathbb{R}$ belegt, d.h. $V(\sigma) = \{m, t, k\}$ $\sigma = [m/1.0, t/2.0, k/3.0]$.

Definition (Wert eines Ausdrucks)

Sei σ eine Substitution der Variablen $V(\sigma)$. Der Wertes $W_\sigma(u)$ eines Ausdrucks u bzgl. σ ist rekursiv (über den Aufbau von u) definiert:

- *Wenn u eine Variable $v \in V(\sigma)$ ist, so ist $W_\sigma(u) = u\,\sigma$*
- *Wenn u ein Literal op ist, so ist $W_\sigma(u) = op$*
- *Wenn u eine Anwendung eines n-stelligen Operators $op(a_1, \ldots, a_n)$ ist, so ist $W_\sigma(u) = op(W_\sigma(a_1), \ldots, W_\sigma(a_n))$*
- *Wenn u ein Funktionsaufruf $f(a_1, \ldots, a_n)$ mit Funktionsrumpf r ist, so ist $W_\sigma(u) = W_\sigma(r)$*

Wert eines Ausdrucks

- Bemerkungen:
  - $W_\sigma(u)$ ergibt als Wert ein Objekt (Literal) der Sorte von u.
  - Ergibt sich σ aus dem Kontext schreiben wir $W(u)$ statt $W_\sigma(u)$.
  - Wenn u eine Variable $v \notin V(\sigma)$ ist, so ist $W_\sigma(u)$ *undefiniert*.
  - Um nur definierte Werte für Ausdrücke zu erhalten, fordern wir:
    - Der Ausdruck im Rumpf einer Funktion f darf nur die formalen (Eingabe-) Parameter der Funktion enthalten (z.B. im Rumpf der Funktion *strecke* ist die Menge der Variablen $V = \{m, k, t\}$).
    - Beim Aufruf einer Funktion müssen alle formalen Parameter mit konkreten Werten belegt sein, d.h. für alle Variablen in V ist eine entsprechende Substitution mit einem konkreten Wert in σ definiert.

Beispiel

Der Aufruf von $\mathit{strecke}(1.0, 2.0, 3.0)$ entspricht einer Substitution $\sigma = [m/1.0, t/2.0, k/3.0]$ mit $V(\sigma) = \{m, t, k\}$:

$$
\begin{aligned}
W(\mathit{strecke}(1.0, 2.0, 3.0)) = & \\
& = W((k \cdot t \cdot t)/(2 \cdot m)) \\
& = W(k \cdot t \cdot t)/W(2 \cdot m) \\
& = (W(k) \cdot W(t) \cdot W(t))/(W(2) \cdot W(m)) \\
& = (k[k/3.0] \cdot t[t/2.0] \cdot t[t/2.0])/(2 \cdot m[m/1.0]) \\
& = (3.0 \cdot 2.0 \cdot 2.0)/(2 \cdot 1.0) \\
& = 12.0/2.0 \\
& = 6.0
\end{aligned}
$$

(Beachten Sie die implizite Sortenanpassung)

Beispiel

Der Aufruf von *endgeschwindigkeit*$(1.0, 2.0, 3.0)$ entspricht ebenfalls einer Substitution $\sigma = [m/1.0, t/2.0, k/3.0]$ mit $V(\sigma) = \{m, t, k\}$

$W($*endgeschwindikeit*$(1.0, 2.0, 3.0)) =$

$$
\begin{aligned}
&= W((k/m) \cdot t) \\
&= W(k/m) \cdot W(t) \\
&= (W(k)/W(m)) \cdot W(t) \\
&= (k[k/3.0]/m[m/1.0]) \cdot t[t/2.0] \\
&= (3.0/1.0) \cdot 2.0 \\
&= 6.0
\end{aligned}
$$

Beispiel

Der Aufruf von $\mathit{arbeit}(1.0, 2.0, 3.0)$ entspricht einer Substitution $\sigma = [m/1.0, t/2.0, k/3.0]$ mit $V(\sigma) = \{m, t, k\}$

$$
\begin{aligned}
W(\mathit{arbeit}(1.0, 2.0, 3.0)) = & \\
& = W(k \cdot \mathit{strecke}(m, t, k)) \\
& = W(k) \cdot W(\mathit{strecke}(m, t, k)) \\
& = k[k/3.0] \cdot W((k \cdot t \cdot t)/(2 \cdot m)) \\
& = 3.0 \cdot \ldots \\
& = 3.0 \cdot 6.0 \\
& = 18.0
\end{aligned}
$$