Kapitel 6

Korrektheit

6.1 Abstrakte Prozeduren

Unter einer abstrakten Prozedur verstehen wir eine Vorschrift, die zu jedem Wert eines vorgegebenen Argumentbereichs eine Berechnung definiert, die, wenn sie terminiert, ein eindeutig bestimmtes Ergebnis in einem vorgegebenen Ergebnisbereich liefert. Hier ist ein Beispiel für eine abstrakte Prozedur:

$$sum\colon \mathbb{Z} \to \mathbb{N}$$
$$sum(x) = \text{if } x < 1 \text{ then } 0 \text{ else } sum(x-1) + x$$

Die Prozedur *sum* besteht aus einer **Typzusicherung** (erste Zeile) und einer **definierenden Gleichung** (zweite Zeile). Die Typzusicherung legt den **Argumentbereich** ($\mathbb{Z}$) und den **Ergebnisbereich** ($\mathbb{N}$) der Prozedur fest.

Da die durch eine abstrakte Prozedur beschriebenen Berechnungen gedanklich ausgeführt werden, gibt es anders als bei konkreten Prozeduren weder Speicherbeschränkungen noch Größenbeschränkungen für Zahlen. Außerdem gehen wir davon aus, das abstrakte Prozeduren so formuliert sind, dass die durch sie beschriebenen Berechnungen nicht zu undefinierten Operations- und Funktionsanwendungen führen (z. B. Division durch null). Für die durch eine abstrakte Prozedur und ein Argument aus dem Argumentbereich beschriebene Berechnung gibt es daher nur zwei Möglichkeiten: Entweder bricht die Berechnung nicht ab (Divergenz), oder sie terminiert mit einem Ergebnis im Ergebnisbereich.

Wenn eine abstrakte Prozedur mit Ausdrucksmitteln formuliert ist, die in einer Programmiersprache verfügbar sind, kann sie als konkrete Prozedure realisiert und mit einem Computer ausgeführt werden. Wir können dann davon ausgehen, dass die folgenden Korrektheitsbedingungen gelten:

1. Wenn die konkrete Prozedur für ein Argument regulär terminiert, terminiert die abstrakte Prozedur für dieses Argument und liefert dasselbe Ergebnis wie die konkrete Prozedur.
2. Wenn die abstrakte Prozedur für ein Argument divergiert, divergiert die Ausführung der konkreten Prozedur für dieses Argument oder wird wegen Überlauf oder Speicherplatzerschöpfung abgebrochen.

Dabei setzen wir natürlich voraus, dass Operationen auf reellen Zahlen nicht durch Gleitkommaoperationen realisiert werden.

Wir werden uns darauf beschränken, Eigenschaften von abstrakten Prozeduren zu beweisen. Wegen der gerade formulierten Korrektheitsbedingungen übertragen sich die Eigenschaften abstrakter Prozeduren weitgehend auf ihre konkreten Schwestern. Wenn man darüber hinaus beispielsweise sicherstellen will, dass es zu keinem Überlauf kommen kann, sind weitere Beweise fällig.

Wenn wir in diesem Kapitel das Wort Prozedur ohne den Zusatz abstrakt oder konkret verwenden, wird es sich stets um eine abstrakte Prozedur handeln.

Prozeduren können als mathematische Objekte formalisiert werden, aber dafür ist sehr viel mehr Aufwand erforderlich als für Funktionen. Funktionen sind also in dieser Hinsicht sehr viel einfachere Objekte als Prozeduren. Wir werden auf eine formale Darstellung von Prozeduren verzichten.

Wir sagen, dass eine Prozedur p eine **Prozedur** $X \to Y$ ist, wenn $X \to Y$ die Typzusicherung von p ist.

Sei p eine Prozedur $X \to Y$. Ein Paar $(x, y) \in X^2$ heißt **Rekursionspaar von** p, wenn für die Bestimmung von $p(x)$ gemäß der definierenden Gleichung von p die rekursive Bestimmung von $p(y)$ erforderlich ist. Die **Rekursionsrelation** von p ist die Menge aller Rekursionspaare von p.

Die Rekursionsrelation unserer Prozedur *sum* ist

$$\{ (n, n-1) \mid n \in \mathbb{N}^+ \}$$

Wir werden nur Prozeduren betrachten, bei denen, so wie bei *sum*, eine nicht rekursive Beschreibung der Rekursionsrelation direkt aus dem Argumentbereich und der definierenden Gleichung abgelesen werden kann.

Beachten Sie, dass die Rekursionsrelation einer Prozedur $X \to Y$ immer eine binäre Relation auf X ist.

Die Rekursionsrelation einer Prozedur enthält genügend Information, um zu entscheiden, ob die Prozedur für ein bestimmtes Argumente terminiert. Da die Rekursionsrelation einfacher als die Prozedur ist, bietet sie uns einen guten Ausgangspunkt für die Analyse des Terminierungsverhaltens der Prozedur.

Sei r eine binäre Relation. Eine **unendliche Kette in** r ist eine unendliche Folge $x_0, x_1, x_2, \ldots$, für die gilt: $\forall n \in \mathbb{N}\colon\ (x_n, x_{n+1}) \in r$. Wir sagen, dass r **für** x **terminiert**, wenn es keine unendliche Kette in r gibt, die mit x beginnt; und wir sagen, dass r **terminiert**, wenn es keine unendliche Kette in r gibt.

Beachten Sie, dass die Rekursionsrelation von *sum* terminiert.

Sei p eine Prozedur $X \to Y$. Wir sagen, dass p **auf** $x \in X$ **terminiert**, wenn die Rekursionsrelation von p auf x terminiert. Die Menge aller $x \in X$, für die p terminiert, bezeichnen wir mit $Dom\ p$ und nennen sie den **Definitionsbereich von** p. Wenn $Dom\ p = X$ gilt, sagen wir, dass p **terminiert**.

Sei p eine Prozedur $X \to Y$. Wenn p für ein **Argument** $x \in X$ terminiert, bezeichnen wir das eindeutig bestimmte **Ergebnis**, das wir mit der definierenden Gleichung von p für x bestimmen können, mit $p(x)$. Die Funktion

$$\{\, (x, p(x)) \mid x \in Dom\ p \,\} \in X \to Y$$

bezeichnen wir als die **Ergebnisfunktion der Prozedur** p.

Proposition 6.1.1 *Sei p eine Prozedur $X \to Y$ und $x \in X$. Dann ist die Ergebnisfunktion von p genau dann auf x definiert, wenn die Rekursionsrelation von p für x terminiert.*

Proposition 6.1.2 *Sei p eine terminierende Prozedur $X \to Y$. Dann ist die Ergebnisfunktion von p eine Funktion $X \to Y$, die die definierende Gleichung von p erfüllt.*

Wir zeigen am Beispiel unserer Prozedur *sum*, was mit „eine Funktion erfüllt die definierende Gleichung einer Prozedur“ gemeint ist: Eine Funktion $f \in \mathbb{Z} \to \mathbb{N}$ erfüllt die definierende Gleichung von *sum* genau dann, wenn gilt:

$$\forall x \in \mathbb{Z}\colon\ f(x) = \text{if } x < 1 \text{ then } 0 \text{ else } f(x-1) + x$$

Diese Aussage erhält man aus *sum* und f, indem man die Variable x der definierenden Gleichung von *sum*

$$sum(x) = \text{if } x < 1 \text{ then } 0 \text{ else } sum(x-1) + x$$

gemäß dem Argumentbereich von *sum* quantifiziert und die Auftreten des Namens *sum* in der definierenden Gleichung von *sum* durch den Namen f ersetzt.

Hier ist ein zweites Beispiel für eine abstrakte Prozedur:

$$sum'\colon \mathbb{Z} \to \mathbb{N}$$
$$sum'(x) = \text{if } x < 1 \text{ then } 0 \text{ else } \frac{x}{2}(x-1)$$

Da sum' nicht rekursiv ist, ist die Rekursionsrelation von sum' die leere Menge. Wir werden später zeigen, dass $sum(x) = sum'(x)$ für alle $x \in \mathbb{Z}$ gilt. Das bedeutet, dass *sum* und sum' beide dieselbe Funktion beschreiben. Beachten Sie aber, dass es sich bei *sum* und sum' um verschiedene Prozeduren handelt (*sum* ist rekursiv, sum' ist nicht rekursiv).

Beispiel: Fibonacci-Funktion

Die folgende Prozedur liefert für $n \in \mathbb{N}$ die so genannten **n-te Fibonacci-Zahl**:

$fib\colon \mathbb{N} \to \mathbb{N}$

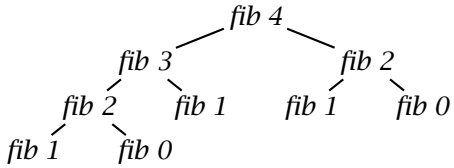
$fib(n) = \text{if } n < 2 \text{ then } n \text{ else } fib(n-1) + fib(n-2)$

Die Rekursionsrelation von *fib* ist

$$\{\, (n, n-i) \mid n \in \mathbb{N} \wedge n \geq 2 \wedge i \in \{1,2\} \,\}$$

Da diese Relation terminiert, terminiert *fib*.

Es ist aufschlussreich, die Rekursionsrelation einer Prozedur mit ihren Rekursionsbäumen in Zusammenhang zu bringen (siehe Abschnitt 1.8). Hier ist der Rekursionsbaum für *fib* und das Argument 4:



Wir stellen fest, dass jeder Kante des Baums ein Rekursionspaar der Prozedur entspricht.

Eine rekursive Prozedur heißt **linear-rekursiv**, wenn alle ihre Rekursionsbäume linear sind, und **binär-rekursiv**, wenn alle ihre Rekursionsbäume binär sind. Nichtlineare Rekursion wird als **Baumrekursion** bezeichnet. Die Prozedur *fib* ist binär-rekursiv, und die zuvor betrachtete Prozedur *sum* ist linear-rekursiv.

Beispiel: Länge von Listen

Sei X eine Menge. Die in Kapitel 4 definierte Menge $\mathcal{L}(X)$ aller Listen über X erfüllt die folgende Gleichung:

$$\mathcal{L}(X) = \{\langle\rangle\} \cup (X \times \mathcal{L}(X))$$

Wenn wir wollen, können wir diese Gleichung als eine rekursive Definition der Menge $\mathcal{L}(X)$ ansehen. Wenn wir uns die Definition von $\mathcal{L}(X)$ in Kapitel 4 nochmals ansehen, stellen wir fest, dass diese ebenfalls rekursiv ist. Rekursion spielt also auch bei der Definition von Mengen eine wichtige Rolle.

Wir betrachten eine Prozedur, die die Länge von Listen liefert:

$$\begin{aligned}&\mathit{length}\colon \mathcal{L}(X) \to \mathbb{N}\\ &\mathit{length}(\mathit{nil}) = 0\\ &\mathit{length}(x :: xr) = 1 + \mathit{length}(xr)\end{aligned}$$

Diese Prozedur hat zwei definierende Gleichungen, die eine disjunkte und erschöpfende Fallunterscheidung für die zulässigen Argumente der Prozedur bilden. Die Rekursionsrelation von *length* ist

$$\{\, (x :: xr, xr) \mid (x :: xr) \in \mathcal{L}(X) \,\} \;=\; \{\, ((x, xr), xr) \mid (x, xr) \in \mathcal{L}(X) \,\}$$

Diese Relation ist terminierend, da für jedes Paar $((x, xr), xr)$ gilt, dass xr ein echtes Teilobjekt von (x, xr) ist (siehe das Wohlfundierungsaxiom in Abschnitt 2.1).

6.2 Beweis durch Nachrechnen

Unter einem Beweis versteht man eine vollständige und nachvollziehbare mathematische Argumentation, die zweifelsfrei zeigt, dass eine mathematische Aussage gültig ist. Unter einer mathematischen Aussage versteht man dabei eine präzise formulierte Eigenschaft eines mathematischen Objekts.

Wir werden einige Eigenschaften der Ergebnisfunktion unserer Prozedur *sum* beweisen. Weil es bequem ist, werden wir die Ergebnisfunktion mit demselben Namen wie die Prozedur bezeichnen. Da die Prozedur *sum* terminiert, liefert uns Proposition 6.1.2 die folgenden Eigeschaften der Ergebnisfunktion:

(1) $\mathit{sum} \in \mathbb{Z} \to \mathbb{N}$

(2) $\forall x \in \mathbb{Z}\colon\ \mathit{sum}(x) = \text{if } x < 1 \text{ then } 0 \text{ else } \mathit{sum}(x-1) + x$

Behauptung 1: $\mathit{sum}(2) = 3$.

Beweis Wir können die Behauptung durch einfaches Nachrechnen zeigen:

$$\mathit{sum}(2) = \mathit{sum}(1) + 2 = (\mathit{sum}(0) + 1) + 2 = (0 + 1) + 2 = 3$$

Für die ersten 3 Gleichungen benötigen wir jeweils die Eigenschaft (2) der Funktion *sum*. □

In Zukunft werden wir von den durch Proposition 6.1.2 garantierten Eigenschaften einer Ergebnisfunktion f frei Gebrauch machen und das durch die Anmerkung „gemäß Definition von f“ dokumentieren.

Die nächste Behauptung ist mithilfe eines Allquantors und einer Implikation formuliert.

Behauptung 2: $\forall x \in \mathbb{Z}\colon\ x < 1 \Longrightarrow \mathit{sum}(x) = 0$.

Beweis Sei $x \in \mathbb{Z}$ und $x < 1$. Dann folgt $\mathit{sum}(x) = 0$ gemäß der Definition von *sum*. □

Wir merken uns, dass die Gültigkeit einer allquatifizierten Aussage $\forall x \in X\colon A$ nach dem folgenden Schema bewiesen werden kann:

> Sei $x \in X$.
> *Beweis, dass A für dieses allgemein gewählte x gilt.*

Die linke Teilaussage einer implikativen Aussage $A \Longrightarrow B$ nennt man **Prämisse** und die rechte **Konklusion**. Hier ist das im obigen Beweis verwendete Beweisschema für implikative Aussagen $A \Longrightarrow B$:

> Sei A gültig.
> *Beweis, dass B gilt;*
> *dabei kann verwendet werden, dass A gilt.*

Ein anderes Beweisschema für implikative Aussagen $A \Longrightarrow B$ sieht wie folgt aus:

> *Beweis, dass* $\neg A$ *gilt.*

Dieses Schema ignoriert die Konklusion und zeigt lediglich, dass die Prämisse nicht gilt. Wenn Ihnen die Gültigkeit dieses Beweisschemas nicht einleuchtet, müssen Sie ihre logische Intuition umprogrammieren. Eine logische Grundannahme besagt nämlich, dass eine implikative Aussage $A \Longrightarrow B$ genau dann gültig ist, wenn mindestens eine der Aussagen $\neg A$ und B gilt. Diese Grundannahme bedeutet, dass für alle Aussagen A und B die folgende Äquivalenz gilt:

$$(A \Longrightarrow B) \iff (\neg A \vee B)$$

Die folgende Proposition liefert die Grundlage für eine wichtige Beweistechnik.

Proposition 6.2.1 *Sei p eine terminierende Prozedur* $X \to Y$ *und f eine Funktion* $X \to Y$*, die die definierenden Gleichungen von p erfüllt. Dann ist f die Ergebnisfunktion von p.*

Die Anwendung dieser Proposition demonstrieren wir mit dem Beweis der folgenden Behauptung.

Behauptung 3: $\forall x \in \mathbb{Z}\colon\ \mathit{sum}(x) = \text{if } x < 1 \text{ then } 0 \text{ else } \frac{x}{2}(x+1)$.

Diese Behauptung ist interessant, da sie sagt, dass wir die Ergebnisfunktion der Prozedur *sum* auch ohne Rekursion berechnen können.

Beweis Zunächst definieren wir eine Funktion:

$$f \in \mathbb{Z} \to \mathbb{N}$$
$$f(x) = \text{if } x < 1 \text{ then } 0 \text{ else } \frac{x}{2}(x+1)$$

Die Behauptung gilt, wenn wir zeigen können, das f die Ergebnisfunktion der Prozedur *sum* ist. Gemäß Proposition 6.2.1 genügt es dafür zu zeigen, dass f die definierende Gleichung der Prozedur *sum* erfüllt. Das ist der Fall, wenn die Gleichung

$$f(x) = \text{if } x < 1 \text{ then } 0 \text{ else } f(x-1) + x$$

für alle $x \in \mathbb{Z}$ gilt. Sei $x \in \mathbb{Z}$. Wir unterscheiden zwei Fälle.

Sei $x < 1$. Dann lässt sich die Gleichung mithilfe der Definition von f nachrechnen.

Sei $x \geq 1$. Dann gilt:

$$\begin{aligned} f(x) &= \frac{x}{2}(x+1) && \text{gemäß Definition von } f \\ &= \frac{x-1}{2}(x-1+1) + x \\ &= f(x-1) + x && \text{gemäß Definition von } f \\ &= \text{if } x < 1 \text{ then } 0 \text{ else } f(x-1) + x && \text{da } x \geq 1 \end{aligned}$$

□

6.3 Ordnungsrelationen

Das klassische Beispiel für eine Ordnungsrelation ist die Ordnungsbeziehung für reelle Zahlen. Für je zwei Zahlen x und y gilt entweder $x < y$ (x kleiner y), $x = y$ oder $x > y$ (x größer y).

Wir beschäftigen uns hier mit Ordnungsrelationen, da ein enger Zusammenhang zu terminierenden Relationen besteht, die wir als Hilfsmittel für Terminierungsbeweise für Prozeduren benötigen. Beispielsweise ist $\{ (m,n) \in \mathbb{N}^2 \mid m > n \}$ eine terminierende Relation.

Die Inklusionsbeziehung für Mengen kann als eine Ordnungsbeziehung für Mengen aufgefasst werden. Beispielsweise gilt $\{1\} \subseteq \{1,2\}$. Für Inklusionsaussagen über Mengen benutzen wir die Symbole $\subsetneq$, $\subseteq$, $\supsetneq$, $\supseteq$. Nicht alle Mengen stehen in einer Inklusionsbeziehung: Beispielsweise gilt weder $\{1\} \subseteq \{2\}$ noch $\{2\} \subseteq \{1\}$.

Zunächst führen wir einige Schreibweisen für binäre Relationen ein, die sich besonders für Ordnungsrelationen eignen.

Für jede binäre Relation r vereinbaren wir die so genannte **Infixschreibweise**:

$$xry \stackrel{\text{def}}{=} (x,y) \in r$$

Die Infixschreibweise ist besonders dann bequem, wenn eine Relation durch ein Infixsymbol wie $\leq$, $\preceq$ oder $\sim$ bezeichnet wird.

Sei $\preceq$ eine binäre Relation. Wir vereinbaren die folgenden Schreib- und Sprechweisen:

$x \preceq y$			x **kleiner-gleich** y	
$x \succeq y$	$\stackrel{\text{def}}{\Longleftrightarrow}$	$y \preceq x$	x **größer-gleich** y	
$x \prec y$	$\stackrel{\text{def}}{\Longleftrightarrow}$	$x \preceq y \wedge x \neq y$	x **kleiner** y	x **Vorgänger von** y
$x \succ y$	$\stackrel{\text{def}}{\Longleftrightarrow}$	$y \prec x$	x **größer** y	x **Nachfolger von** y

Wenn wir eine binäre Relation $\preceq$ vorgeben, bekommen wir gemäß der obigen Schreibweisen drei **assoziierte Relationen** $\succeq$, $\prec$ und $\succ$, die wir wie folgt charakterisieren können:

$$\begin{aligned} \succeq &= \preceq^{-1} \\ \prec &= \{\,(x,y) \mid x \preceq y \wedge x \neq y\,\} \\ \succ &= \prec^{-1} \end{aligned}$$

Eine binäre Relation $\preceq$ heißt

- **reflexiv**, wenn $Dom(\preceq) = Ran(\preceq)$ und $\forall x \in Dom(\preceq)\colon\ x \preceq x$.
- **transitiv**, wenn für alle $x,y,z \in Dom(\preceq)$ gilt: Wenn $x \preceq y$ und $y \preceq z$, dann $x \preceq z$.
- **antisymmetrisch**, wenn für alle $x,y \in Dom(\preceq)$ gilt: Wenn $x \preceq y$ und $y \preceq x$, dann $x = y$.
- **partielle Ordnung**, wenn sie reflexiv, transitiv und antisymmetrisch ist.
- **linear**, wenn für alle $x,y \in Dom(\preceq)$ gilt: $x \preceq y \ \vee\ y \preceq x$.
- **wohlfundiert**, wenn $\succ$ terminiert.

Der Begriff partielle Ordnung formuliert die üblichen Mindestanforderungen an eine Ordnungsrelation. Eine partielle Ordnung $\preceq$ heißt

- **partielle Ordnung für eine Menge** X, wenn $Dom(\preceq) = X$ ist.
- **lineare Ordnung**, wenn sie linear ist.

- **wohlfundierte Ordnung**, wenn sie wohlfundiert ist.
- **Wohlordnung**, wenn sie linear und wohlfundiert ist.

Sei $X \subseteq \mathbb{R}$. Dann ist

$$NO(X) \stackrel{\text{def}}{=} \{ (x, y) \in X^2 \mid x \leq y \}$$

eine lineare Ordnung für X. Wir bezeichnen $NO(X)$ als die **natürliche Ordnung für** X.

Eine Menge $X \subseteq \mathbb{R}$ heißt **nach unten beschränkt**, wenn es ein $y \in \mathbb{R}$ gibt mit $\forall x \in X\colon\ y \leq x$. Wenn X eine nach unten beschränkte Teilmenge von $\mathbb{Z}$ ist, dann ist $NO(X)$ eine Wohlordnung.

Sei X eine Menge. Dann ist

$$SO(X) \stackrel{\text{def}}{=} \{ (x, y) \in X^2 \mid x \text{ Teilobjekt von } y \}$$

eine wohlfundierte Ordnung für X. Die Wohlfundierung ist durch das Wohlfundierungsaxiom gewährleistet (siehe Abschnitt 2.1). Wir bezeichnen $SO(X)$ als die **strukturelle Ordnung für** X.

Sei X eine Menge. Dann ist

$$IO(X) \stackrel{\text{def}}{=} \{ (Y, Z) \mid Y \subseteq Z \text{ und } Z \subseteq X \}$$

eine partielle Ordnung für $\mathcal{P}(X)$. Wir bezeichnen $IO(X)$ als die **Inklusionsordnung für** $\mathcal{P}(X)$. Wenn X endlich ist, dann ist $IO(X)$ wohlfundiert. Wenn X mindestens zwei Elemente enthält, dann ist $IO(X)$ keine lineare Ordnung.

Sei X eine Menge. Dann ist

$$Id(X) \stackrel{\text{def}}{=} \{ (x, x) \mid x \in X \}$$

eine wohlfundierte Ordnung für X. Wir bezeichnen $Id(X)$ als die **Identität für** X.

6.4 Terminierungsbeweise

Wir haben die Rekursionsrelation einer Prozedur so definiert, dass die Prozedur genau dann terminiert, wenn ihre Rekursionsrelation terminiert. Um die Terminierung von Prozeduren zu zeigen, benötigen wir also Techniken, mit denen wir die Terminierung von binären Relationen zeigen können. Zunächst ist es hilfreich, einige bereits als terminierend bekannte Relationen zu kennen.

Nach unseren Definitionen ist eine binäre Relation $\preceq$ genau dann wohlfundiert, wenn ihre assoziierte Relation $\succ$ terminiert. Damit wir von diesem Zusammenhang bequem Gebrauch machen können, vereinbaren wir für binäre Relationen r die folgende Notation:

$$r^{\succ} \stackrel{\text{def}}{=} \{\, (x,y) \mid (y,x) \in r \wedge x \neq y \,\}$$

Proposition 6.4.1 *Sei X eine nach unten beschränkte Teilmenge der ganzen Zahlen. Dann ist $NO(X)^{\succ}$ eine terminierende Relation.*

Proposition 6.4.2 *Sei X eine Menge. Dann ist $SO(X)^{\succ}$ eine terminierende Relation.*

Proposition 6.4.3 *Jede Teilmenge einer terminierenden Relation ist eine terminierende Relation.*

Das durch die letzte Proposition formulierte Kriterium lässt sich allgemeiner fassen. Wir sagen, dass eine Funktion f eine binäre Relation r in eine binäre Relation r' **einbettet**, wenn gilt:

1. $Dom\, r \cup Ran\, r \subseteq Dom\, f$.
2. $\forall (x,y) \in r\colon\ (fx, fy) \in r'$.

Wir sagen, dass eine binäre Relation r in eine binäre Relation r' **einbettbar** ist, wenn es eine Funktion gibt, die r in r' einbettet.

Proposition 6.4.4 *Jede Relation, die in eine terminierende Relation einbettbar ist, terminiert.*

Beweis Angenommen, f bettet r in r' ein und r' terminiert. Wir zeigen durch Widerspruch, dass r terminiert. Angenommen r terminiert nicht. Dann gibt es eine unendliche Kette $x_0, x_1, x_2, \ldots$ in r. Also ist $f(x_0), f(x_1), f(x_2), \ldots$ eine unendliche Kette in r'. Widerspruch, da r' terminiert. □

Eine Prozedur p heißt

- **natürlich-rekursiv**, wenn es eine nach unten beschränkte Teilmenge X der ganzen Zahlen gibt, so dass die Rekursionsrelation von p in $NO(X)^{\succ}$ eingebettet werden kann.
- **strukturell-rekursiv**, wenn es eine Menge X gibt, sodass die Rekursionsrelation von p in $SO(X)^{\succ}$ eingebettet werden kann.

Die in Abschnitt 6.1 betrachteten Prozeduren *sum* und *fib* sind natürlich-rekursiv, da ihre Rekursionsrelationen mit der Identitätsfunktion in $NO(\mathbb{N})^{\succ}$ eingebettet werden. Die im selben Abschnitt betrachtete Prozedur *length* ist

strukturell-rekursiv, da ihre Rekursionsrelation mit der Identitätsfunktion in $SO(\mathcal{L}(X))^{\succ}$ eingebettet wird.

Proposition 6.4.5 *Jede natürlich- oder strukturell-rekursive Prozedur terminiert.*

Beweis Folgt aus den Propositionen 6.4.1, 6.4.2 und 6.4.4. □

Beispiel: Größe von Bäumen

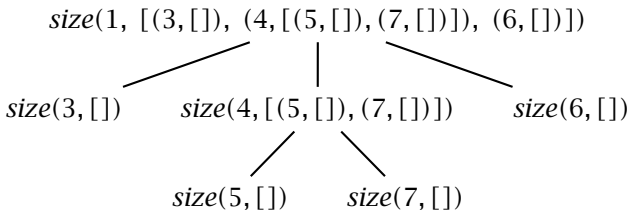
Sei X eine Menge. Die Menge $\mathcal{T}(X)$ der Bäume über X können wir wie folgt rekursiv definieren (vergleiche Abschnitt 5.8):

$$\mathcal{T}(X) \stackrel{\text{def}}{=} X \times \mathcal{L}(\mathcal{T}(X))$$

Wir betrachten eine rekursive Prozedur, die die Größe eines Baums liefert:

$$\begin{aligned} &\mathit{size}\colon \mathcal{T}(X) \to \mathbb{N} \\ &\mathit{size}(x, [t_1, \ldots, t_n]) = 1 + \mathit{size}(t_1) + \cdots + \mathit{size}(t_1) \end{aligned}$$

Hier ist ein Rekursionsbaum:



Die Rekursionsrelation der Prozedur *size* ist

$$\{\, ((x, [t_1, \ldots, t_n]),\ t_i) \mid (x, [t_1, \ldots, t_n]) \in \mathcal{T}(X) \wedge n \geq 1 \wedge i \in \{1, \ldots, n\} \,\}$$

Da es sich um eine Teilmenge von $SO(\mathcal{T}(X))^{\succ}$ handelt, ist *size* strukturell-rekursiv und damit terminierend.

Beispiel: Potenzen

Die Prozedur

$$\begin{aligned} &\mathit{power}\colon \mathbb{R} \times \mathbb{N} \to \mathbb{R} \\ &\mathit{power}(x, n) = \text{if } n = 0 \text{ then } 1 \text{ else } \mathit{power}(x, n-1) \cdot x \end{aligned}$$

liefert für x und n die Potenz x^n. Ihre Rekursionsrelation

$$\{\, ((x, n),\ (x, n-1)) \mid x \in \mathbb{R} \wedge n \in \mathbb{N}^+ \,\}$$

wird von $\lambda(x,n) \in \mathbb{R} \times \mathbb{N}\,.\, n$ in $NO(\mathbb{N})^{\succ}$ eingebettet. Also ist *power* natürlich-rekursiv und terminiert.

Beispiel: Natürliche Quadratwurzel

Wir werden später zeigen, dass die Prozedur

$$sqrt\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$
$$sqrt(n,x) = \text{if } n^2 > x \text{ then } n-1 \text{ else } power(n+1,x)$$

für alle $x \in \mathbb{N}$ die Gleichung

$$sqrt(0,x) = \max\{\, k \in \mathbb{N} \mid k^2 \leq x \,\}$$

erfüllt. Vorerst begnügen wir uns damit, zu zeigen, dass *sqrt* natürlich rekursiv ist. Dazu betrachten wir die Rekursionsrelation von *sqrt*:

$$\{\, ((n,x),\ (n+1,x)) \mid n \in \mathbb{N} \wedge x \in \mathbb{N} \wedge n^2 \leq x \,\}$$

Diese wird von $\lambda(n,x) \in \mathbb{N}^2\,.\, x - n + 1$ in $NO(\mathbb{N})^{\succ}$ eingebettet. Also ist *sqrt* natürlich-rekursiv und terminiert.

6.5 Induktion

Wenn X eine Menge ist, sagen wir, dass ein $x \in X$ ein **minimales Element von** X **für** $\preceq$ ist, wenn gilt: $\forall y \in X\colon\ y \preceq x \Longrightarrow y = x$.

Proposition 6.5.1 *Eine Relation $\preceq$ ist genau dann wohlfundiert, wenn jede nichtleere Menge ein minimales Element für $\preceq$ enthält.*

Satz 6.5.2 (Wohlfundierte Induktion) *Sei $\preceq$ eine wohlfundierte Relation, X eine Menge und $A \subseteq X$. Dann $A = X$, wenn gilt:*

$$\forall x \in X\colon\ (\forall y \in X\colon\ y \prec x \Longrightarrow y \in A) \implies x \in A$$

Beweis Gelte $\forall x \in X\colon\ (\forall y \in X\colon\ y \prec x \Longrightarrow y \in A) \implies x \in A$. Es genügt zu zeigen, dass $X - A = \emptyset$. Wir beweisen das durch Widerspruch. Sei also $X - A \neq \emptyset$. Dann hat $X - A$ wegen der obigen Proposition ein minimales Element x_0 für $\preceq$. Das bedeutet, dass $\forall y \in X\colon\ y \prec x_0 \Longrightarrow y \in A$ gilt. Also folgt mit der den Beweis einleitenden Annahme $x_0 \in A$. Das ist ein Widerspruch zu $x_0 \in X - A$. □

Für den Beweis von Eigenschaften von rekursiv definierten Funktionen benötigt man in vielen Fällen eine Beweistechnik, die als **Induktion** bezeichnet wird. Es

gibt verschiedene Varianten von Induktion. Wir erklären hier die allgemeinste, die als **wohlfundierte Induktion** bezeichnet wird.

Wir beginnen mit einer Auflistung des Inventars eines Induktionsbeweises:

Grundmenge: X

Aussagemenge: $A \subseteq X$

Zielbehauptung: $\forall x \in X\colon x \in A$

Induktionsrelation: $\preceq$ wohlfundiert

Induktionsbehauptung: $\forall x \in X\colon\ (\forall y \in X\colon\ y \prec x \Longrightarrow y \in A) \implies x \in A$

Induktionsannahme für $x \in X$: $\forall y \in X\colon\ y \prec x \Longrightarrow y \in A$

Die Ausgangssituation für einen Induktionsbeweis ist durch eine Grundmenge X und eine Aussagemenge $A \subseteq X$ gegeben. Bewiesen werden soll die Zielbehauptung $\forall x \in X\colon x \in A$.

Ein Induktionsbeweis für die Zielbehauptung führt zunächst eine wohlfundierte Relation $\preceq$ ein, die als Induktionsrelation bezeichnet wird. Damit steht die Induktionsbehauptung

$$\forall x \in X\colon\ (\forall y \in X\colon\ y \prec x \Longrightarrow y \in A) \implies x \in A$$

die jetzt bewiesen werden muss. Wenn dies gelingt, ist der Induktionsbeweis fertig. Aus Satz 6.5.2 folgt nämlich, dass die Gültigkeit der Zielbehauptung aus der Gültigkeit der Induktionsbehauptung folgt, unabhängig davon, welche Induktionsrelation gewählt wurde.

Damit der Beweis der Induktionsbehauptung gelingt, muss die „richtige“ Induktionsrelation gewählt werden. Wenn die Grundmenge eine nach unten beschränkte Menge von ganzen Zahlen ist, ist die natürliche Ordnung die naheliegende Wahl. Man spricht dann von **natürlicher Induktion**. Wenn die Elementen der Grundmenge dagegen zusammengesetzte Objekte sind (z. B. Listen oder Bäume), bietet sich die strukturelle Ordnung an. Man spricht dann von **struktureller Induktion**.

Die Induktionsbehauptung ist eine allquantifizierte Implikation der Form

$$\forall x \in X\colon\ (\text{Induktionsannahme für } x) \implies x \in A$$

Sie kann also nach dem folgenden Schema bewiesen werden:

> Sei $x \in X$ und gelte die Induktionsannahme für x.
> *Beweis für $x \in A$;*
> *dabei kann die Induktionsannahme für x verwendet werden.*

Wie bei der Zielbehauptung muss also für jedes $x \in X$ die Gültigkeit von $x \in A$ bewiesen werden. Anders als bei Zielbehauptung ist es aber erlaubt, die Gültigkeit der Induktionsannahme für x zu verwenden:

$$\forall y \in X\colon\ y \prec x \Longrightarrow y \in A$$

Diese besagt, dass alle Vorgänger von x Elemente von A sind. Für viele Probleme kann $x \in A$ nur unter Verwendung der Induktionsannahme bewiesen werden.

Insgesamt werden wir einen Induktionsbeweis für $\forall x \in X\colon x \in A$ gemäß dem folgenden Schema aufschreiben:

> Induktion über $x \in X$ mit $\preceq$.
> *Beweis für $x \in A$;*
> *dabei kann die Induktionsannahme $\forall y \in X\colon\ y \prec x \Longrightarrow y \in A$*
> *verwendet werden.*

Beispiel: Summenfunktion

Als Beispiel beweisen wir eine Eigenschaft der Summenfunktion:

$$sum \in \mathbb{Z} \to \mathbb{N}$$
$$sum(x) = \text{if } x < 1 \text{ then } 0 \text{ else } sum(x-1) + x$$

Proposition 6.5.3 $\forall n \in \mathbb{N}\colon\ sum(n) = \frac{n}{2}(n+1)$.

Wir beweisen diese Eigenschaft mithilfe von natürlicher Induktion. Dabei ist die Grundmenge $\mathbb{N}$ und die Aussagemenge $\{\, n \in \mathbb{N} \mid sum(n) = \frac{n}{2}(n+1) \,\}$. Als Induktionsrelation verwenden wir, wie bereits gesagt, die natürlichen Ordnung für $\mathbb{N}$. Kompakt können wir den Beweis wie folgt aufschreiben.

Beweis Durch natürliche Induktion über $n \in \mathbb{N}$. Wir unterscheiden zwei Fälle.

Sei $n = 0$. Dann folgt $sum(n) = 0 = \frac{n}{2}(n+1)$ durch Nachrechnen mithilfe der Definition von *sum*.

Sei $n > 0$. Dann gilt:

$$\begin{aligned} sum(n) &= sum(n-1) + n && \text{Definition von } sum \\ &= \frac{n-1}{2}(n-1+1) + n && \text{Induktionsannahme für } n-1 \\ &= \frac{n}{2}(n+1) \end{aligned}$$

□

Die Fallunterscheidung des Beweises folgt der Fallunterscheidung der Definition von *sum*. Nur für den Fall $n > 0$ ist für die Bestimmung des Funktionswerts eine

Rekursion erforderlich. Diese Rekursion wird im Beweis mithilfe der Induktionsannahme behandelt.

Die gerade beobachtete enge Entsprechung zwischen der rekursiven Funktionsdefinition und dem Induktionsbeweis einer Funktionseigenschaft ist kein Zufall. Allgemein gilt, dass die Terminierungsrelation einer rekursiven Funktionsdefinition als Induktionsordnung für die Beweise von Funktionseigenschaften verwendet werden kann, und dass sich die Fallunterscheidungen der Definition im Beweis der Induktionsbehauptung wiederfinden.

6.6 Strukturelle Induktion für Listen

Sei X eine Menge. Wir betrachten die Menge der Listen über X:

$$\mathcal{L}(X) = \{\langle\rangle\} \cup (X \times \mathcal{L}(X))$$

Funktionen, die die Konkatenation, Länge und Reversion von Listen über X liefern, definieren wir wie folgt:

$$\begin{aligned} &@ \in \mathcal{L}(X) \times \mathcal{L}(X) \to \mathcal{L}(X) && \text{Konkatenation}\\ &\quad nil@ys = ys\\ &(x::xr)@ys = x::(xr@ys) \end{aligned}$$

$$\begin{aligned} &|\_| \in \mathcal{L}(X) \to \mathbb{N} && \text{Länge}\\ &\quad |nil| = 0\\ &|x::xr| = 1 + |xr| \end{aligned}$$

$$\begin{aligned} &rev \in \mathcal{L}(X) \to \mathcal{L}(X) && \text{Reversion}\\ &\quad rev(nil) = nil\\ &rev(x::xr) = rev(xr)@[x] \end{aligned}$$

Proposition 6.6.1 (Assoziativität der Konkatenation) *Sei X eine Menge und seien $xs, ys, zs \in \mathcal{L}(X)$. Dann gilt: $(xs@ys)@zs = xs@(ys@zs)$.*

Beweis Seien $ys, zs \in \mathcal{L}(X)$. Wir beweisen

$$\forall xs \in \mathcal{L}(X)\colon\ (xs@ys)@zs = xs@(ys@zs)$$

durch strukturelle Induktion über $xs \in \mathcal{L}(X)$. Wir unterscheiden zwei Fälle.

Sei $xs = nil$. Dann

$$\begin{aligned}(xs@ys)@zs &= ys@zs && \text{Definition von } @\\ &= xs@(ys@zs) && \text{Definition von } @\end{aligned}$$

Sei $xs = x::xr$. Dann

$$\begin{aligned}(xs@ys)@zs &= (x::(xr@ys))@zs && \text{Definition von } @\\ &= x::((xr@ys)@zs) && \text{Definition von } @\\ &= x::(xr@(ys@zs)) && \text{Induktionsannahme für } xr\\ &= xs@(ys@zs) && \text{Definition von } @\end{aligned}$$

□

Proposition 6.6.2 *Sei X eine Menge und seien $xs, ys, zs \in \mathcal{L}(X)$. Dann gilt:*

1. $xs@nil = xs$
2. $|xs@ys| = |xs| + |ys|$
3. $|rev(xs)| = |xs|$
4. $rev(xs@ys) = rev(ys)@rev(xs)$
5. $rev(rev(xs)) = xs$

Beweis Übungsaufgabe. Für den Beweis von (3) benötigt man (2). Für den Beweis von (4) benötigt man (1) und Proposition 6.6.1. Für den Beweis von (5) benötigt man (4). □

Die Definitionen der Funktionen $@$, $|\_|$ und *rev* entsprechen den Prozedurdeklarationen für `append`, `length` und `rev` in Abschnitt 4.3. Allerdings handelt es sich um polymorphe Prozeduren. Wenn wir uns aber auf eine Typ t für die Typvariable festlegen, und X als die Menge aller Werte von t wählen, bekommen wir monomorphe Prozeduren, denen genau die obigen Funktionen zugeordnet sind. Also gelten die in Proposition 7.5.1 formulierten Eigenschaften auch für die Prozeduren.

6.7 Größenverhältnisse in Bäumen

Sei X eine Menge. Wir betrachten wieder die Menge der Bäume über X:

$$\mathcal{T}(X) \;=\; X \times \mathcal{L}(\mathcal{T}(X))$$

Funktionen, die die Größe, Tiefe und Breite von Bäumen liefern, definieren wir wie folgt:

$s \in \mathcal{T}(X) \to \mathbb{N}$ Größe
$s(x, [t_1, \ldots, t_n]) = 1 + s(t_1) + \cdots + s(t_n)$

$d \in \mathcal{T}(X) \to \mathbb{N}$ Tiefe
$d(x, [t_1, \ldots, t_n]) = 1 + \max\{-1, d(t_1), \ldots, d(t_n)\}$

$b \in \mathcal{T}(X) \to \mathbb{N}^+$ Breite
$b(x, [t_1, \ldots, t_n]) = \text{if } n = 0 \text{ then } 1 \text{ else } b(t_1) + \cdots + b(t_n)$

Die **inneren Knoten** eines Baums sind die Knoten, die keine Blätter sind. Ein Baum ist genau dann zusammengesetzt, wenn er mindestens einen inneren Knoten hat.

Proposition 6.7.1 *Sei $t \in \mathcal{T}(X)$ ein Baum, bei dem jeder innere Knoten mindestens zwei Nachfolger hat. Dann gilt $2b(t) > s(t)$. Mit anderen Worten, t hat mehr Blätter als innere Knoten.*

Wir beweisen die Behauptung durch strukturelle Induktion. Als Grundmenge verwenden wir

$$\{\, t \in \mathcal{T}(X) \mid \text{jeder innere Knoten von } t \text{ hat mindestens 2 Nachfolger} \,\}$$

Beweis Durch strukturelle Induktion über $t \in \mathcal{T}(X)$, wobei jeder innere Knoten von t mindestens 2 Nachfolger hat. Wir unterscheiden zwei Fälle.

Sei $t = (x, [])$. Dann folgt $2b(t) = 2 > 1 = s(t)$ durch Nachrechnen mit den Definitionen von b und s.

Sei $t = (x, [t_1, \ldots, t_n])$ mit $n \geq 2$. Dann gilt:

$$\begin{aligned}
2b(t) &= 2(b(t_1) + \cdots + b(t_n)) && \text{Definition von } b \\
&= 2b(t_1) + \cdots + 2b(t_n) \\
&\geq (s(t_1) + 1) + \cdots + (s(t_n) + 1) && \text{Induktionsannahme für } t_1, \ldots, t_n \\
&> s(t_1) + \cdots + s(t_n) + 1 && n \geq 2 \\
&= s(t) && \text{Definition von } s
\end{aligned}$$

Bei der Anwendung der Induktionsannahme auf die Unterbäume t_i haben wir von der Äquivalenz

$$2b(t_i) > s(t_i) \iff 2b(t_i) \geq s(t_i) + 1$$

Gebrauch gemacht. □

Proposition 6.7.2 *Für jeden balancierten Binärbaum $t \in \mathcal{T}(X)$ gilt:* $b(t) = 2^{d(t)}$.

Beweis Durch strukturelle Induktion über $t \in \mathcal{T}(X)$, wobei t ein balancierter Binärbaum ist. Wir unterscheiden zwei Fälle.

Sei $t = (x, [])$. Dann folgt $b(t) = 1 = 2^{d(t)}$ durch Nachrechnen mit den Definitionen von b und d.

Sei $t = (x, [t_1, t_2])$. Dann gilt:

$$\begin{aligned}
b(t) &= b(t_1) + b(t_2) && \text{Definition von } b \\
&= 2^{d(t_1)} + 2^{d(t_2)} && \text{Induktionsannahme für } t_1 \text{ und } t_2 \\
&= 2 \cdot 2^{d(t_1)} && t \text{ balanciert, also } d(t_1) = d(t_2) \\
&= 2^{1+d(t_1)} \\
&= 2^{1+\max\{d(t_1),d(t_2)\}} && t \text{ balanciert, also } d(t_1) = d(t_2) \\
&= 2^{d(t)} && \text{Definition von } d
\end{aligned}$$

□

Proposition 6.7.3 *Für jeden balancierten Binärbaum $t \in \mathcal{T}(X)$ gilt:* $s(t) = 2^{d(t)+1} - 1$.

Beweis Durch strukturelle Induktion über $t \in \mathcal{T}(X)$, wobei t ein balancierter Binärbaum ist. Wir unterscheiden zwei Fälle.

Sei $t = (x, [])$. Dann folgt $s(t) = 1 = 2^{d(t)+1} - 1$ durch Nachrechnen mit den Definitionen von b und d.

Sei $t = (x, [t_1, t_2])$. Dann gilt:

$$\begin{aligned}
s(t) &= 1 + s(t_1) + s(t_2) && \text{Definition von } s \\
&= 1 + 2^{d(t_1)+1} - 1 + 2^{d(t_2)+1} - 1 && \text{Induktionsannahme für } t_1 \text{ und } t_2 \\
&= 2 \cdot 2^{d(t_1)+1} - 1 && t \text{ balanciert, also } d(t_1) = d(t_2) \\
&= 2^{1+d(t_1)+1} - 1 \\
&= 2^{1+\max\{d(t_1),d(t_2)\}+1} - 1 && t \text{ balanciert, also } d(t_1) = d(t_2) \\
&= 2^{d(t)+1} - 1 && \text{Definition von } d
\end{aligned}$$

□

Glossar

Abstrakte Prozeduren (Typzusicherung, Argumentbereich, Ergebnisbereich, definierende Gleichung; Rekursionspaare, Rekursionsrelation, Rekursionsbäume,

Terminierung, Definitionsbereich; Ergebnisfunktion; natürliche und strukturelle Rekursion).

Ordnungsrelationen (Infixschreibweise, assoziierte Relationen ($\succeq$, $\prec$, $\succ$); reflexiv, transitiv, antisymmetrisch, linear, wohlfundiert; partielle Ordnung (für X); lineare Ordnung, wohlfundierte Ordnung, Wohlordnung; natürliche Ordnung $NO(X)$, strukturelle Ordnung $SO(X)$).

Terminierende Relationen (unendliche Kette, Terminierung für x, Einbettung; $NO(X)^{\succ}$, $SO(X)^{\succ}$).

Induktion (wohlfundierte, natürliche, strukturelle; Grundmenge, Aussagemenge, Zielbehauptung, Induktionsrelation, Induktionsbehauptung, Induktionsannahme für x).