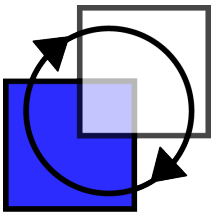
CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Prof. Bernd Finkbeiner, Ph.D.
Jana Hofmann, M.Sc.
Reactive Systems Group



Programmierung 1 (WS 2020/21)
Aufgaben für die Übungsgruppe I (Lösungsvorschläge)

Hinweis: Diese Aufgaben wurden von den Tutoren für die Übungsgruppe erstellt. Sie sind für die Klausur weder relevant noch irrelevant. 🤔 markiert potentiell schwerere Aufgaben.

Induktion

Aufgabe TI.1 *(Strukturelle Rekursion)*
Betrachten Sie die folgende Prozedur:

$$\begin{aligned} &dou : \mathscr{L}(X) \to \mathbb{N} \\ &dou\ (nil) = 2 \\ &dou\ (x :: xr) = 2 \cdot dou(xr) \end{aligned}$$

(a) Geben Sie für *dou* eine natürliche Terminierungsfunktion an.

(b) Geben Sie für *dou* eine strukturelle Terminierungsfunktion an.

Lösungsvorschlag TI.1

(a) $\lambda xs \in \mathscr{L}(X).\,|xs|$

(b) $\lambda xs \in \mathscr{L}(X).\,xs$

Aufgabe TI.2 *(Strukturelle Terminierungsfunktionen)*
Bestimmen Sie zu den folgenden Prozeduren jeweils eine strukturelle Terminierungsfunktion:

(a) $q : \mathscr{L}(X) \to \mathbb{N}$
$q\ nil = 0$
$q\ (x :: xr) = 1 + q\ xr$

(b) $s : \mathscr{L}(X) \to \mathscr{L}(\mathbb{N})$
$s\ nil = [42]$
$s\ (x :: xr) = (s\ xr)\ @\ (s\ xr)$

(c) $rev : \mathscr{L}(X) \to \mathscr{L}(X)$
$rev\ nil = nil$
$rev\ (x :: xr) = rev\ xr\ @\ [x]$

Lösungsvorschlag TI.2

(a) $\lambda xs \in \mathscr{L}(X).xs$

(b) $\lambda xs \in \mathscr{L}(X).xs$

(c) $\lambda xs \in \mathscr{L}(X).xs$

Aufgabe TI.3 *(Hin und her...)*
Gegeben sei folgende Prozedur:

$$
\begin{aligned}
&foo : \mathbb{N} \times \mathscr{L}(X) \times \mathscr{L}(X) \to \mathscr{L}(X) \\
&foo\ (0, nil, ys) = ys \\
&foo\ (n, nil, ys) = foo(n-1, ys, nil) && \text{für } n > 0 \\
&foo\ (n, (x :: xs), ys) = foo(n, xs, (x :: ys))
\end{aligned}
$$

(a) Geben Sie für foo eine natürliche Terminierungsfunktion an.

(b) Warum können Sie für foo nicht ohne Weiteres eine strukturelle Terminierungsfunktion finden?

Lösungsvorschlag TI.3

(a) $\lambda\ n \in \mathbb{N} \times xs \in \mathscr{L}(X) \times ys \in \mathscr{L}(X)\ .\ n + (n-1) \cdot (|xs| + |ys|) + |xs|$

(b) Wenn wir eine strukturelle Terminierungsfunktion angeben, muss für jeden Rekursionsschritt (x, x') gelten, dass $f\ x'$ eine Konstituente von $f\ x$ ist. Dies ist bei Listen sehr praktisch, da der Rest einer Liste immer eine Konstituente der gesamten Liste ist. Bei dieser Prozedur hängt die Terminierung allerdings nicht nur von Listen ab, sondern auch von $n \in \mathbb{N}$. Somit können wir nicht ohne Weiteres eine strukturelle Terminierungsfunktion angeben. Natürlich ist es immer möglich eine strukturelle Terminierungsfunktion anzugeben, wenn man auch eine natürliche angeben kann (wir haben ja gesehen, dass man auch natürliche Zahlen über Mengen definieren kann), allerdings wäre diese Lösung hier deutlich komplexer.

Aufgabe TI.4 *(Sind zwei Bäume schon ein Wald?)*
Gegeben sei folgende Prozedur:

$$
\begin{aligned}
&forest : \mathscr{T} \times \mathscr{T} \to \mathbb{N} \\
&forest\ ([], []) = 0 \\
&forest\ ([t_1], [t_2]) = 2 \\
&forest\ ([t_1, ..., t_n], [t'_1, ..., t'_m]) = \text{if } n > m \text{ then } m + forest(t_{n-1}, t_n) \text{ else } n + forest(t'_{m-1}, t'_m) && \text{für } n + m > 2
\end{aligned}
$$

Geben Sie für $forest$ eine strukturelle Terminierungsfunktion an.

Lösungsvorschlag TI.4
$\lambda\ t \in \mathscr{T} \times t' \in \mathscr{T}\ .\ [t, t']$

Aufgabe TI.5 *(last)*
Sei die folgende Prozedur gegeben:

$$
\begin{aligned}
&last : \mathscr{L}(X) \setminus \{nil\} \to X \\
&last\ (x :: xr) = foldl\ (\lambda(a, b).\ a, x, xr)
\end{aligned}
$$

Zeigen Sie, dass *last* das letzte Element einer nicht-leeren Liste zurückgibt. Zeigen Sie dazu mit struktureller Induktion, dass *last* die folgende Funktion berechnet:

$$
\begin{aligned}
&f \in \mathscr{L}(X) \setminus \{nil\} \to X \\
&f\ (x :: nil) = x \\
&f\ (x :: y :: xr) = f(y :: xr)
\end{aligned}
$$

Lösungsvorschlag TI.5

Beweis. Wir zeigen $\forall xs \in \mathscr{L}(X) \setminus \{nil\} : last(xs) = f(xs)$ durch strukturelle Induktion über $xs \in \mathscr{L}(X) \setminus \{nil\}$.
Wir unterscheiden zwei Fälle:

- Sei $xs = x :: nil$.

$$\begin{aligned} last\ xs &= last\ (x :: nil) \\ &= foldl\ (\lambda(a,b).\ a, x, nil) && \text{Definition von } last \\ &= x && \text{Definition von } foldl \\ &= f\ (x :: nil) && \text{Definition von } f \\ &= f\ xs \end{aligned}$$

- Sei $xs = x :: y :: xr$.

  Induktionsannahme: $last\,(y :: xr) = f\,(y :: xr)$

$$\begin{aligned} last\ xs &= last\ (x :: y :: xr) \\ &= foldl\ (\lambda(a,b).\ a, x, y :: xr) && \text{Definition von } last \\ &= foldl\ (\lambda(a,b).\ a, y, xr) && \text{Definition von } foldl \\ &= f\ (y :: xr) && \text{Induktion für } y :: xr \\ &= f\ (x :: y :: xr) && \text{Definition von } f \\ &= f\ xs \end{aligned}$$

■

Aufgabe TI.6 *(map)*
Zeigen Sie, dass *map*, definiert über strukturelle Rekursion, und *map′*, definiert mit *foldr*, semantisch äquivalent sind. 🤔

$$\begin{aligned} &map : (X \to Y) \to \mathscr{L}(X) \to \mathscr{L}(Y) \\ &map\ f\ nil = nil \\ &map\ f\ (x :: xr) = f(x) :: map\ f\ xr \end{aligned}$$

$$\begin{aligned} &foldr : (X \times Y \to Y) \to Y \to \mathscr{L}(X) \to Y \\ &foldr\ f\ y\ nil = y \\ &foldr\ f\ y\ (x :: xr) = f\ (x, foldr\ f\ y\ xr) \end{aligned}$$

$$\begin{aligned} &map' : (X \to Y) \to \mathscr{L}(X) \to \mathscr{L}(Y) \\ &map'\ f\ xs = foldr\ (\lambda(x, xr) \in X \times \mathscr{L}(X)\ .\ f(x) :: xr)\ nil\ xs \end{aligned}$$

Lösungsvorschlag TI.6
Beweis.
Wir zeigen $\forall xs \in \mathscr{L}(X) : map\ f\ xs = map'\ f\ xs$ durch strukturelle Induktion über $xs \in \mathscr{L}(X)$.

Wir unterscheiden zwei Fälle:

- Sei $xs = nil$.

$$\begin{aligned} map\ f\ nil &= nil && \text{Definition } map \\ &= foldr\ (\lambda(x, xr) \in X \times \mathscr{L}(X)\ .\ f(x) :: yr)\ nil\ nil && \text{Definition } foldr \\ &= map'\ f\ nil && \text{Definition } map' \end{aligned}$$

- Sei $xs = x :: xr$.

  Induktionsannahme: $map\ f\ xr = map'\ f\ xr$

  Sei im Folgenden $g \coloneqq (\lambda(x, xr) \in X \times \mathscr{L}(X)\,.\, f(x) :: xr)$

$$\begin{aligned}
map\ f\ (x :: xr) &= f(x) :: (map\ f\ xr) && \text{Definition } map\\
&= f(x) :: (map'\ f\ xr) && \text{Induktion für } xr\\
&= g\ (x, map'\ f\ xr) && \text{Prozeduranwendung}\\
&= g\ (x, foldr\ g\ nil\ xr) && \text{Definition } map'\\
&= foldr\ g\ nil\ (x :: xr) && \text{Definition } foldr\\
&= map'\ f\ (x :: xr) && \text{Definition } map'
\end{aligned}$$

Aufgabe TI.7
Gegeben seien die folgenden Definitionen:

$$\begin{aligned}
nil@B &= B & |nil| &= 0\\
(x :: A)@B &= x :: (A@B) & |x :: xr| &= 1 + |xr|
\end{aligned}$$

Beweisen Sie die Aussage $|xs \mathbin{@} [z]| = |xs| + 1$.

Lösungsvorschlag TI.7
Beweis durch Induktion über $xs \in \mathscr{L}(X)$. Fallunterscheidung.

(a) Sei $xs = nil$.

$$\begin{aligned}
|nil \mathbin{@} [z]| &= |[z]| && \text{Definition } @\\
&= |z :: nil| && \text{Definition } ::\\
&= 1 + |nil| && \text{Definition } length\\
&= |nil| + 1 && \text{Kommutativität } +
\end{aligned}$$

(b) Sei $xs = x :: xr$.
Induktionsannahme: $|xr \mathbin{@} [z]| = |xr| + 1$

$$\begin{aligned}
|(x :: xr) \mathbin{@} [z]| &= |x :: (xr) \mathbin{@} [z]| && \text{Definition } @\\
&= 1 + |(xr) \mathbin{@} [z]| && \text{Definition } length\\
&= 1 + |(xr)| + 1 && \text{Induktion für } xr\\
&= |x :: xr| + 1 && \text{Definition } length
\end{aligned}$$

Aufgabe TI.8 *(Aufwärmen mit Listen)*
Sei *filter* wie folgt definiert:

$$\begin{aligned}
filter &: (X \to \mathbb{B}) \times \mathscr{L}(X) \to \mathscr{L}(X)\\
filter(p, nil) &= nil\\
filter(p, x :: xr) &= \text{if } p\ x \text{ then } x :: filter(p, xr) \text{ else } filter(p, xr)
\end{aligned}$$

Beweisen Sie die Aussage $\forall xs \in \mathscr{L}(X)\, \forall p \in X \to \mathbb{B} : |filter(p, xs)| \leq |xs|$.

Sie dürfen die Definitionen der vorherigen Aufgabe verwenden.

Lösungsvorschlag TI.8
Beweis durch Induktion über $xs \in \mathscr{L}(X)$. Sei $p \in X \to \mathbb{B}$ beliebig. Fallunterscheidung.

(a) Sei $xs = nil$.

$$\begin{aligned} |\mathit{filter}(p, xs)| &= |\mathit{filter}(p, nil)| && xs = nil \\ &= |nil| && \text{Definition } \mathit{filter} \\ &= |xs| && xs = nil \end{aligned}$$

(b) Sei $xs = x :: xr$ und $p(x) = 1$.
Induktionsannahme: $|\mathit{filter}(p, xs)| \leq |xs|$

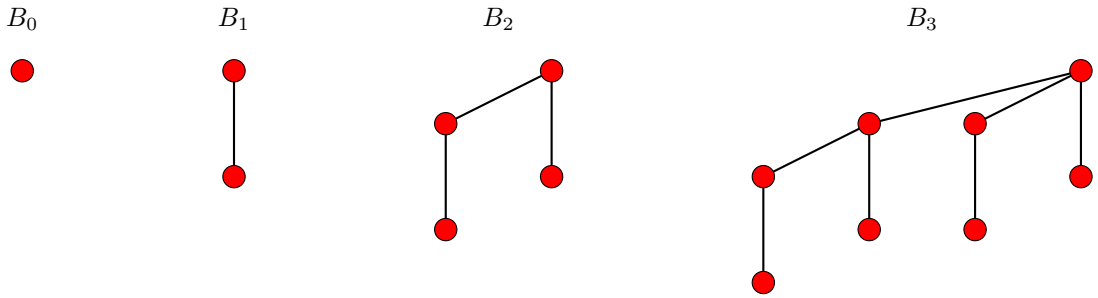
$$\begin{aligned} |\mathit{filter}(p, xs)| &= |\mathit{filter}(p, x :: xr)| && xs = x :: xr \\ &= |\text{if } p\ x \text{ then } x :: \mathit{filter}(p, xr) \text{ else } \mathit{filter}(p, xr)| && \text{Definition } \mathit{filter} \\ &= |x :: \mathit{filter}(p, xr)| && p(x) = 1 \\ &= 1 + |\mathit{filter}(p, xr)| && \text{Definition } |\_| \\ &\leq 1 + |xr| && \text{Induktion für } xr \\ &= |x :: xr| && \text{Definition } |\_| \\ &= |xs| && xs = x :: xr \end{aligned}$$

(c) Sei $xs = x :: xr$ und $p(x) = 0$.
Induktionsannahme: s.o.

$$\begin{aligned} |\mathit{filter}(p, xs)| &= |\mathit{filter}(p, x :: xr)| && xs = x :: xr \\ &= |\text{if } p\ x \text{ then } x :: \mathit{filter}(p, xr) \text{ else } \mathit{filter}(p, xr)| && \text{Definition } \mathit{filter} \\ &= |\mathit{filter}(p, xr)| && p(x) = 0 \\ &\leq |xr| && \text{Induktion für } xr \\ &\leq 1 + |xr| && \text{Mathe} \\ &= |x :: xr| && \text{Definition } |\_| \\ &= |x :: xr| && xs = x :: xr \end{aligned}$$

Aufgabe TI.9 *(Binomialbäume)*
Ein Binomialbaum $B_k \in \mathcal{B}$ der Ordnung k ist ein geordneter Baum und rekursiv wie folgt definiert:

- B_0 ist der Baum mit einem Knoten: $B_0 = [\,]$.
- B_k besteht aus zwei Kopien von B_{k-1}. Die Wurzel der einen Kopie wird das linkeste Kind der Wurzel der anderen Kopie: $B_k = [B_{k-1}, B'_1, \ldots, B'_n]$ wenn $B_{k-1} = [B'_1, \ldots, B'_n]$.



Graphische Darstellung der Binomialbäume B_0, B_1, B_2 und B_3

Seien die Prozeduren s, b, d wie in Kapitel 10.5 des Buches gegeben. Es sei außerdem die Prozedur $a : \mathcal{B} \to \mathbb{N}$ mit $a\ [t_1, \ldots, t_n] = n$ gegeben.

Beweisen Sie:

(a) $s\,(B_k) = 2^k$.

(b) $d\,(B_k) = k$.

(c) $a\,(B_k) = k$.

Lösungsvorschlag TI.9

(a) Beweis von $\forall k \in \mathbb{N} : s\,(B_k) = 2^k$ durch Induktion über $k \in \mathbb{N}$. Fallunterscheidung.

- Sei $k = 0$.

$s\,(B_0) = s\ [\,]$	Definition B_0
$= \text{if } n = 0 \text{ then } 1 \text{ else } 1$	Definition s
$= 1$	Definition if, $n = 0$
$= 2^0$	Arithmetik

- Sei $k = 1$.

$s\,(B_1) = s\ [B_0]$	Definition B_1
$= \text{if } n = 0 \text{ then } 1 \text{ else } 1 + s\,(B_0)$	Definition s
$= 1 + s\,(B_0)$	Definition if, $n = 1$
$= 1 + 2^0$	$s\,(B_0) = 2^0$, s. o.
$= 2^1$	Arithmetik

- Sei $k > 1$.

  Induktionsannahme: $s\,(B_\ell) = 2^\ell$ gelte für alle $\ell < k$.

$s\,(B_k) = s\ [B_{k-1}, B'_1, \ldots, B'_n]$	Definition B_k
$= \text{if } n + 1 = 0 \text{ then } 1 \text{ else } 1 + s\,(B_{k-1}) + s\,(B'_1) + \ldots + s\,(B'_n)$	Definition s
$= 1 + s\,(B_{k-1}) + s\,(B'_1) + \ldots + s\,(B'_n)$	Definition if, $n + 1 \neq 0$
$= s\,(B_{k-1}) + 1 + s\,(B'_1) + \ldots + s\,(B'_n)$	Assoziativität Addition
$= s\,(B_{k-1}) + \text{if } n = 0 \text{ then } 1 \text{ else } 1 + s\,(B'_1) + \ldots + s\,(B'_n)$	Definition if, $n \geq 1$
$= s\,(B_{k-1}) + s\,(B_{k-1})$	Definition s
$= 2^{k-1} + 2^{k-1}$	Induktion für $k - 1$
$= 2^k$	Arithmetik

(b) Beweis von $\forall k \in \mathbb{N} : d\,(B_k) = k$ durch Induktion über $k \in \mathbb{N}$. Fallunterscheidung.

- Sei $k = 0$.

$d\,(B_0) = d\ [\,]$	Definition B_1
$= \text{if } n = 0 \text{ then } 0 \text{ else } 1 + \max\{\,\}$	Definition d
$= 0$	Definition if, $n = 0$

- Sei $k = 1$.

$d\,(B_1) = d\ [B_0]$	Definition B_1
$= \text{if } n = 0 \text{ then } 0 \text{ else } 1 + \max\{d(B_0)\}$	Definition d
$= 1 + \max\{d(B_0)\}$	Definition if, $n = 1$
$= 1 + \max\{0\}$	$d\,(B_0) = 0$, s. o.
$= 1 + 0$	Definition max
$= 0$	Arithmetik

- Sei $k > 1$.

  Induktionsannahme: $s\,(B_\ell) = \ell$ gelte für alle $\ell < k$.

$$
\begin{aligned}
d\,(B_k) &= d\ [B_{k-1}, B'_1, \ldots, B'_n] && \text{Definition } B_k \\
&= \text{if } n+1 = 0 \text{ then } 0 \text{ else } 1 + \max\{d\,(B_{k-1}), d\,(B'_1), \ldots, d\,(B'_n)\} && \text{Definition } d \\
&= 1 + \max\{d\,(B_{k-1}), d\,(B'_1), \ldots, d\,(B'_n)\} && \text{Definition if, } n+1 \neq 0 \\
&= 1 + \max\{d\,(B_{k-1}), \max\{d\,(B'_1), \ldots, d\,(B'_n)\}\} && \text{Definition max} \\
&= \max\{1 + d\,(B_{k-1}), 1 + \max\{d\,(B'_1), \ldots, d\,(B'_n)\}\} && \text{Definition max} \\
&= \max\{1 + d\,(B_{k-1}), \text{if } n = 0 \text{ then } 0 \text{ else } 1 + \max\{d\,(B'_1), \ldots, d\,(B'_n)\}\} && \text{Definition if, } n \geq 1 \\
&= \max\{1 + d\,(B_{k-1}), d\,(B_{k-1})\} && \text{Definition } d \\
&= 1 + d\,(B_{k-1}) && \text{Definition max} \\
&= 1 + (k-1) && \text{Induktion für } k-1 \\
&= k && \text{Arithmetik}
\end{aligned}
$$

(c) Beweis von $\forall k \in \mathbb{N} : a\,(B_k) = k$ durch Induktion über $k \in \mathbb{N}$. Fallunterscheidung.

- Sei $k = 0$.

$$
\begin{aligned}
a\,(B_0) &= a\ [\,] && \text{Definition } B_0 \\
&= 0 && \text{Definition } a
\end{aligned}
$$

- Sei $k > 0$.

  Induktionsannahme: $a\,(B_\ell) = \ell$ gelte für alle $\ell < k$.

$$
\begin{aligned}
a\,(B_k) &= a\ [B_{k-1}, B'_1, \ldots, B'_n] && \text{Definition } B_k \\
&= |[B_{k-1}, B'_1, \ldots, B'_n]| && \text{Definition } a \\
&= 1 + |[B'_1, \ldots, B'_n]| && \text{Definition } |\cdot| \\
&= 1 + a\ [B'_1, \ldots, B'_n] && \text{Definition } a \\
&= 1 + a\,(B_{k-1}) && \text{Definition } B_k \\
&= 1 + (k-1) && \text{Induktion für } k-1 \\
&= k && \text{Arithmetik}
\end{aligned}
$$

Aufgabe TI.10 *(Der kleine Gauß)*
Sei die Prozedur *iterdn* wie folgt gegeben:

$$
\begin{aligned}
&\mathit{iterdn} : \mathbb{N} \times \mathbb{N} \times X \times (\mathbb{N} \times X \to X) \to X \\
&\mathit{iterdn}(n, m, s, f) = \text{if } n < m \text{ then } s \text{ else } \mathit{iterdn}(n-1, m, f(n, s), f)
\end{aligned}
$$

Weiterhin sei $f = \lambda(a, b) \in \mathbb{N} \times \mathbb{N}.\ a + b$.

Zeigen Sie mittels Induktion die folgende Aussage. Überlegen Sie sich zuvor, ob Sie die Aussage verstärken müssen: $\forall n \in \mathbb{N}.\ \mathit{iterdn}(n, 1, 0, f) = \sum_{i=1}^{n} i$

Lösungsvorschlag TI.10
Wir verstärken die Aussage:

Behauptung: $\forall n \in \mathbb{N} : \forall a \in \mathbb{N} : \mathit{iterdn}(n, 1, a, f) = a + \sum_{i=1}^{n} i$ mit $f = \lambda(a, b) \in \mathbb{N} \times \mathbb{N}.a + b$

Beweis. Durch Induktion über $n \in \mathbb{N}$. Fallunterscheidung:

- Sei $n = 0$ und $a \in \mathbb{N}$. Dann gilt:

$$\begin{aligned} iterdn(n, 1, a, f) &= iterdn(0, 1, a, f) && n = 0 \\ &= a && \text{Def. von } iterdn \\ &= a + \sum_{i=1}^{0} i && \text{Def. von } \sum \\ &= a + \sum_{i=1}^{n} i && n = 0 \end{aligned}$$

- Sei $n > 0$ und $a \in \mathbb{N}$. Induktionsannahme: Für alle $m < n$ gilt $\forall a \in \mathbb{N} : iterdn(m, 1, a, f) = a + \sum_{i=1}^{m} i$ mit $f = \lambda(a, b) \in \mathbb{N} \times \mathbb{N}.a + b$.
  Dann gilt:

$$\begin{aligned} iterdn(n, 1, a, f) &= iterdn(n-1, 1, f(n, a), f) && \text{Def. von } iterdn \\ &= f(n, a) + \sum_{i=1}^{n-1} i && \text{Induktion für } m = n-1 \\ &= n + a + \sum_{i=1}^{n-1} i && \text{Def. von } f \\ &= a + \sum_{i=1}^{n} i && \text{Def. von } \sum \end{aligned}$$

■

Aufgabe TI.11 *(Es wird noch stärker)*
Beweisen sie: $\forall n \in \mathbb{N} : \forall x \in \mathbb{Z} : x^n = iter(n, 1, \lambda a.\ a \cdot x)$

Benutzen Sie allein die Definition von *iter* und verzichten Sie auf jegliche Propositionen aus dem Buch.
Hinweis: Verstärken sie zunächst die Aussage.

Lösungsvorschlag TI.11
Wir benötigen eine Verstärkung der Korrektheitsaussage und zeigen deshalb zunächst:

$$\forall n \in \mathbb{N} : \forall x \in \mathbb{Z} : \forall s \in \mathbb{Z} : s \cdot x^n = iter(n, s, \lambda a.\ a \cdot x)$$

Beweis durch Induktion über $n \in \mathbb{N}$ Fallunterscheidung:

(a) Seien $n = 0$ und $x, s \in \mathbb{Z}$ beliebig.

$$\begin{aligned} iter(n, s, \lambda a.\ a \cdot x) &= iter(0, s, \lambda a.\ a \cdot x) && n = 0 \\ &= s && \text{Definition von iter} \\ &= s \cdot x^0 && \text{Arithmetik} \\ &= s \cdot x^n && n = 0 \end{aligned}$$

(b) Sei $n > 0$ und $x, s \in \mathbb{Z}$ beliebig.
Induktionsannahme: $\forall m < n : \forall x, s \in \mathbb{Z} : s \cdot x^m = iter(m, s, \lambda a.\ a \cdot x)$

$$\begin{aligned} iter(n, s, \lambda a.\ a \cdot x) &= iter(n-1, s \cdot x, \lambda a.\ a \cdot x) && \text{Definition von iter für } n > 0 \\ &= (s \cdot x) \cdot x^{n-1} && \text{Induktion für } n-1 \\ &= s \cdot x^n && \text{Artihmetik} \end{aligned}$$

Damit gilt die Behauptung auch für $s = 1$.

Aufgabe TI.12 *(Abstraktes Verstärken)*
Betrachten Sie folgende Prozeduren und Aussagen. Finden Sie passende Verstärkungen, mit welchen sich die Aussagen beweisen lassen.

(a) $fac : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$
$fac\ (a, 0) = a$
$fac\ (a, n) = fac\ (n \cdot a, n - 1) \quad n > 0$

$\forall n \in \mathbb{N} : fac\ (1, n) = n!$

(b) $fill : \mathscr{L}(X) \times \mathbb{N} \times X \to \mathscr{L}(X)$
$fill\ (xs, 0, x) = xs$
$fill\ (xs, n, x) = fill\ (x :: xs, n - 1, x) \qquad n > 0$

$\forall x \in X, n \in \mathbb{N} : fill\ (nil, n, x) = [\underbrace{x, \ldots, x}_{n \text{ mal}}]$

Lösungsvorschlag TI.12

(a) $\forall n, a \in \mathbb{N} : fac\ (a, n) = n! \cdot a$

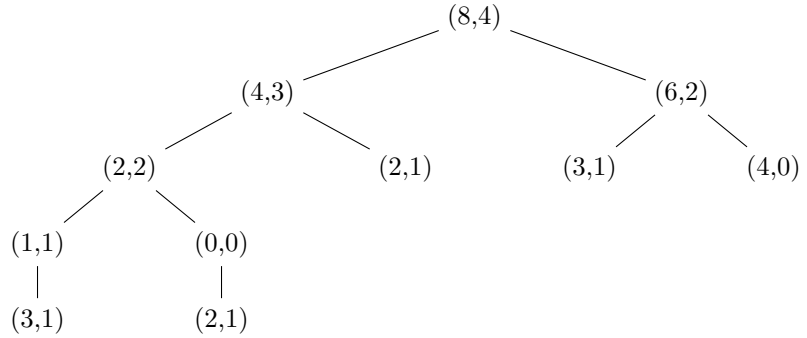
(b) $\forall x \in X, n \in \mathbb{N}, xs \in \mathscr{L}(X) : fill\ (xs, n, x) = [\underbrace{x, \ldots, x}_{n \text{ mal}}] \mathbin{@} xs$

Laufzeit

Aufgabe TI.13 *(Intuition)*
Bestimmen Sie die Laufzeit folgender Prozedur für das Argument $(8, 4)$, indem Sie den Rekursionsbaum zeichnen.

$t : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$
$t\ (m, n) = t\ (m + 2, 1) \qquad m \leq 1$
$t\ (m, n) = 2 \cdot m \qquad m > 1 \wedge n \leq 1$
$t\ (m, n) = t\ (m \operatorname{div} 2, n - 1) + t\ (m - 2, n - 2) \qquad m > 1 \wedge n > 1$

Lösungsvorschlag TI.13



Die Laufzeit beträgt somit 11.

Aufgabe TI.14 *(Größen- und Laufzeitencamp)*
Bestimmen Sie Größen- und Laufzeitfunktionen für die folgenden Prozeduren:

(a) $p : \mathbb{N}^2 \to \mathbb{N}$
$p\ (x, 0) = x$
$p\ (x, y) = 2 \cdot p\ (x, y - 1) \quad \text{für } y > 0$

(b) $p : \mathbb{N} \to \mathbb{N}$
$p\ 0 = 0$
$p\ 1 = 0$
$p\ n = \text{if } n \mod 2 = 0 \text{ then } p\ (n - 2) \text{ else } p\ (n - 2) + p\ (n - 2) \quad \text{für } n > 1$

(c) $p : \mathscr{L}(X) \times \mathscr{L}(X) \to \mathbb{N}$
$p\ (nil, nil) = 0$
$p\ (nil, y :: yr) = p\ (nil, yr)$
$p\ (x :: xr, nil) = p\ (xr, nil)$
$p\ (x :: xr, y :: yr) = \text{if } |xr| > |yr| \text{ then } p\ (xr, y :: yr) \text{ else } p\ (x :: xr, yr)$

Lösungsvorschlag TI.14

(a) Mit der Größenfunktion $\lambda(x, y) \in \mathbb{N}^2.\ y$ ergibt sich die Laufzeitfunktion $\lambda n \in \mathbb{N}.\ n + 1$.

(b) Mit der Größenfunktion $\lambda n \in \mathbb{N}.\ n$ ergibt sich die Laufzeitfunktion

$$\lambda n \in \mathbb{N}.\ \text{if } n \mod 2 = 0 \text{ then } \frac{n}{2} + 1 \text{ else } 2^{\lfloor \frac{n}{2} \rfloor + 1} - 1$$

(c) Mit der Größenfunktion $\lambda(xs, ys) \in \mathscr{L}(X) \times \mathscr{L}(X).\ |xs \mathbin{@} ys|$ ergibt sich die Laufzeitfunktion $\lambda n \in \mathbb{N}.\ n+1$.

Aufgabe TI.15 *(Früher war mehr Laufzeit)*
Bestimmen Sie rekursive und explizite Laufzeitfunktionen für folgende Prozeduren unter Beachtung der gegebenen Größenfunktionen:

(a) $a : \mathbb{N} \to \mathbb{N}$
$a\ (n) = 3 \qquad n = 0$
$a\ (n) = 2 \cdot a\ (n-1) \quad n > 0$
Größenfunktion: $\lambda n \in \mathbb{N}.\ n$

(b) $b : \mathbb{N} \to \mathbb{N}$
$b\ (n) = 12 \cdot n \qquad n = 0$
$b\ (n) = 12 + 8 \cdot b\ (n-1) \quad n > 0$
Größenfunktion: $\lambda n \in \mathbb{N}.\ n + 10$

(c) $c : \mathbb{N} \to \mathbb{N}$
$c\ (n) = 3 \qquad n = 0$
$c\ (n) = 2 \cdot c\ (n-1) \quad n > 0$
Größenfunktion: $\lambda n \in \mathbb{N}.\ n^2$

(d) $d : \mathbb{N} \to \mathbb{N}$
$d\ (n) = 0 \qquad n = 0$
$d\ (n) = d\ (n-1) \cdot d\ (n-1) \quad n > 0$
Größenfunktion: $\lambda n \in \mathbb{N}.\ n$

(e) $e : \mathscr{L}(X) \to \mathbb{N}$
$e\ (nil) = 5$
$e\ (x :: xr) = 3 \cdot e\ (xr)$
Größenfunktion: $\lambda xs \in \mathscr{L}(X).\ |xs|$

Lösungsvorschlag TI.15

(a) Rekursiv:

$$\begin{aligned} r &: \mathbb{N} \to \mathbb{N} \\ r\ 0 &= 1 \\ r\ n &= 1 + r(n-1) \qquad n > 0 \end{aligned}$$

Explizit: $\lambda n \in \mathbb{N}.\ n + 1$

(b) Rekursiv:

$$\begin{aligned} r &: \mathbb{N} \to \mathbb{N} \\ r\ n &= 1 \qquad n < 10 \\ r\ n &= 1 + r(n-1) \qquad n \geq 10 \end{aligned}$$

Explizit: $\lambda n \in \mathbb{N}.\ \text{if } n < 10 \text{ then } 1 \text{ else } n - 9$

(c) Rekursiv:

$$
\begin{aligned}
& r : \mathbb{N} \to \mathbb{N} \\
& r\ 0 = 1 \\
& r\ n = 1 + r(n-1) && n > 0 \wedge \exists k \in \mathbb{N} : n = k^2 \\
& r\ n = r(n-1) && n > 0 \wedge \nexists k \in \mathbb{N} : n = k^2
\end{aligned}
$$

Explizit: $\lambda n \in \mathbb{N}.\ \lfloor \sqrt{n} \rfloor + 1$

Hinweis: Das Abrunden ist hier notwendig, damit eine natürliche Zahl zurückgegeben wird. Außerdem wird so sichergestellt, dass für Argumente n die keine Quadratzahlen sind gilt, dass $c\ n = c(n-1)$ (siehe Buch S. 218).

(d) Rekursiv:

$$
\begin{aligned}
& r : \mathbb{N} \to \mathbb{N} \\
& r\ 0 = 1 \\
& r\ n = 1 + r(n-1) + r(n-1) && n > 0
\end{aligned}
$$

Explizit: $\lambda n \in \mathbb{N}.\ 2^{n+1} - 1$

(e) Rekursiv:

$$
\begin{aligned}
& r : \mathbb{N} \to \mathbb{N} \\
& r\ 0 = 1 \\
& r\ n = 1 + r(n-1) && n > 0
\end{aligned}
$$

Explizit: $\lambda n \in \mathbb{N}.\ n + 1$

Aufgabe TI.16 *(Merge)*
Betrachten Sie die folgende Prozedur *merge*, welche beim Sortieren durch Mischen verwendet wird:

$$
\begin{aligned}
& \mathit{merge} : \mathscr{L}(\mathbb{N}) \times \mathscr{L}(\mathbb{N}) \to \mathscr{L}(\mathbb{N}) \\
& \mathit{merge}\ (\mathit{nil}, \mathit{ys}) = \mathit{ys} \\
& \mathit{merge}\ (\mathit{xs}, \mathit{nil}) = \mathit{xs} \\
& \mathit{merge}\ (x :: \mathit{xr}, y :: \mathit{yr}) = \text{if } x \leq y \text{ then } x :: \mathit{merge}\ (\mathit{xr}, y :: \mathit{yr}) \text{ else } y :: \mathit{merge}\ (x :: \mathit{xr}, \mathit{yr})
\end{aligned}
$$

(a) Geben Sie eine Größenfunktion für *merge* an.

(b) Geben Sie für die Größe 4 ein Argument für *merge* mit minimaler Laufzeit und eines mit maximaler an. Geben Sie die entsprechenden Laufzeiten an.

(c) Welche minimale / maximale Laufzeit hat *merge* für Argumente der Größe n?

Lösungsvorschlag TI.16

(a) $\lambda(\mathit{xs}, \mathit{ys}) \in \mathscr{L}(\mathbb{N}) \times \mathscr{L}(\mathbb{N})\ .\ |\mathit{xs}| + |\mathit{ys}|$

(b) minimale Laufzeit: $(\mathit{nil}, [1,2,3,4])$ mit Laufzeit 1.

maximale Laufzeit: $([1,2,3], [4])$ mit Laufzeit 4.

(c) minimale Laufzeit: 1

maximale Laufzeit: n

Aufgabe TI.17 *(insert)*
Betrachten Sie die Prozedur *insert*, die ein neues Element in eine Liste einfügt (siehe Sortieren durch Einfügen, Abschnitt 5.1 im Buch):

$$
\begin{aligned}
&insert: \ \mathbb{Z} \times \mathscr{L}(\mathbb{Z}) \to \mathscr{L}(\mathbb{Z}) \\
&insert(x, nil) = [x] \\
&insert(x, y :: yr) = \text{if } x \leq y \text{ then } x :: y :: yr \text{ else } y :: insert(x, yr)
\end{aligned}
$$

(a) Wählen Sie eine Größenfunktion für *insert*.

(b) Geben Sie für die Größe 4 ein Argument mit minimaler Laufzeit und ein Argument mit maximaler Laufzeit an (in Bezug auf Argumente der Größe 4).

(c) Welche minimale und welche maximale Laufzeit hat die Prozedur *insert* für Argumente der Größe n?

(d) Geben Sie die Laufzeitfunktion von *insert* an.

Lösungsvorschlag TI.17

(a) Eine Größenfunktion ist gegeben durch $\lambda(x, xs) \in \mathbb{Z} \times \mathscr{L}(\mathbb{Z}) . \ |xs|$.

(b) für $(0, [0, 1, 2, 3])$ hat *insert* minimale Laufzeit (nämlich 1); für $(4, [0, 1, 2, 3])$ hat *insert* maximale Laufzeit (nämlich 5).

(c) Allgemein gilt: *insert* hat minimale Laufzeit 1 und maximale Laufzeit $n + 1$ für Argumente der Größe n.

(d) Aus c) folgt, dass die Laufzeitfunktion durch $\lambda n \in \mathbb{N}. \ n + 1$ gegeben ist.

Aufgabe TI.18 *(Auf die Größe kommt es an)*
Sei eine Prozedur p wie folgt gegeben:

$$
\begin{aligned}
&p : \mathbb{N} \to \mathbb{N} \\
&p \ 0 = 7 \\
&p \ n = p \ (n - 1) \qquad\qquad \text{für } n > 0
\end{aligned}
$$

Bestimmen Sie die rekursive Darstellung der Laufzeitfunktion von p bezüglich folgender Größenfunktionen:

(a) $\lambda n. \ n$

(b) $\lambda n. \ n + 5$

(c) $\lambda n. \ 5n$

(d) $\lambda n. \ 2^n$

Lösungsvorschlag TI.18

(a) Rekursive Darstellung der Laufzeitfunktion $r : \mathbb{N} \to \mathbb{N}_+$ von p für die Größenfunktion $\lambda n. \ n$.

$$
\begin{aligned}
&r(0) = 1 \\
&r(n) = r(n - 1) + 1 \qquad\qquad \text{für } n \geq 1
\end{aligned}
$$

(b) Rekursive Darstellung der Laufzeitfunktion $s : \mathbb{N} \to \mathbb{N}_+$ von p für die Größenfunktion $\lambda n. \ n + 5$.

$$
\begin{aligned}
&s(n) = 1 \qquad\qquad \text{für } n \leq 5 \\
&s(n) = s(n - 1) + 1 \qquad\qquad \text{sonst}
\end{aligned}
$$

(c) Rekursive Darstellung der Laufzeitfunktion $t : \mathbb{N} \to \mathbb{N}_+$ von p für die Größenfunktion $\lambda n. \ 5n$.

$$
\begin{aligned}
&t(n) = 1 \qquad\qquad \text{für } n \leq 4 \\
&t(n) = t(n - 5) + 1 \qquad\qquad \text{sonst}
\end{aligned}
$$

(d) Rekursive Darstellung der Laufzeitfunktion $u : \mathbb{N} \to \mathbb{N}_+$ von p für die Größenfunktion $\lambda n.\ 2^n$.

$$u(n) = 1 \qquad \text{für } n \leq 1$$
$$u(n) = u\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 1 \qquad \text{sonst}$$

Komplexitätstheorie

Aufgabe TI.19 *(Komplexitäten-Quiz)*
Welche der folgenden Aussagen gelten? Wenn nicht, warum nicht?

(a) $\lambda n \in \mathbb{N}.\ n - 1337 \in OF$

(b) $\mathcal{O}\left(\lambda n \in \mathbb{N}.\ 3n^2 - 2n + 8\right) = \mathcal{O}\left(n^2\right)$

(c) $\mathcal{O}(0) \subsetneq \mathcal{O}(1)$

(d) $\lambda n \in \mathbb{N}.\ n^2 \in \mathcal{O}(2^n)$

(e) $\mathcal{O}(0) \in \mathcal{O}(1)$

(f) $\forall f \in OF : f \preceq f$

(g) $\mathcal{O}(1) \subsetneq \mathcal{O}(n)$

(h) $\lambda n \in \mathbb{N}.\ -n \in OF$

(i) $\mathcal{O}(1) = \mathcal{O}(n)$

Lösungsvorschlag TI.19

(a) gilt

(b) gilt

(c) gilt

(d) gilt

(e) gilt nicht, da $\mathcal{O}(0)$ eine Menge ist und daher nicht in $\mathcal{O}(1)$ als Element sein kann, da $\mathcal{O}(1)$ nur Funktionen enthält

(f) gilt

(g) gilt

(h) gilt nicht, da $\lambda n \in \mathbb{N}.\ -n$ für alle Werte größer 0 negative Werte annimmt.

(i) gilt nicht, $\lambda n \in \mathbb{N}.\ n$ zwar in $\mathcal{O}(n)$, aber nicht in $\mathcal{O}(1)$ liegt.

Aufgabe TI.20 *(Inklusionsordnung)*
Ordnen Sie nach Inklusion:

$\mathcal{O}(n!)$	$\mathcal{O}(\log(n))$	$\mathcal{O}\left(n^1\right)$	$\mathcal{O}\left(2^n\right)$	$\mathcal{O}(\pi)$
$\mathcal{O}\left(n^5\right)$	$\mathcal{O}\left(2^{10} \cdot \sqrt{n}\right)$	$\mathcal{O}\left(n^n + 4 \cdot n!\right)$	$\mathcal{O}\left(n^2\right)$	$\mathcal{O}(1)$
$\mathcal{O}(\log(n) \cdot n)$	$\mathcal{O}(12 \cdot n!)$	$\mathcal{O}\left(12n^0 + 3 \cdot \log(n)\right)$	$\mathcal{O}(0)$	$\mathcal{O}\left(2 \cdot \log^2(n) \cdot n\right)$

Lösungsvorschlag TI.20
$\mathcal{O}(0) \subset \mathcal{O}(1) = \mathcal{O}(\pi) \subset \mathcal{O}(\log(n)) = \mathcal{O}\left(12n^0 + 3 \cdot \log(n)\right) \subset \mathcal{O}\left(2^{10} \cdot \sqrt{n}\right) \subset \mathcal{O}\left(n^1\right) \subset \mathcal{O}(\log(n) \cdot n) \subset \mathcal{O}\left(2 \cdot \log^2(n) \cdot n\right) \subset \mathcal{O}\left(n^2\right) \subset \mathcal{O}\left(n^5\right) \subset \mathcal{O}\left(2^n\right) \subset \mathcal{O}(n!) = \mathcal{O}(12 \cdot n!) \subset \mathcal{O}\left(n^n + 4 \cdot n!\right)$

Aufgabe TI.21 *($\mathcal{O}(0)$)*

(a) Erklären Sie, warum $\mathcal{O}(0) \neq \mathcal{O}(1)$ gelten muss.

(b) Kann die Laufzeit einer Prozedur in $\mathcal{O}(0)$ liegen? Begründen Sie.

Lösungsvorschlag TI.21

(a) Die Funktion $f\ x = 1$ liegt in $\mathcal{O}(1)$. Ebenso liegt $g\ x = 3$ in $\mathcal{O}(1)$, denn $4 \cdot (f\ x) \geq g\ x$ für alle außer endlich viele x und $f\ x \leq g\ x$ für alle außer endlich viele x (d. h. f dominiert g und g dominiert f).

Wenn man jetzt $h\ x = 0$ betrachtet, so wird zwar h von g dominiert ($h\ x \leq g\ x$ für alle außer endlich viele x), aber man findet keinen Faktor $c \neq 0$, sodass $c \cdot h\ x \geq g\ x$ für alle außer endlich viele x (denn $c \cdot h\ x$ ist immer 0, aber $c \cdot g\ x$ ist immer $3 \cdot x$).

(b) Ist p eine Prozedur, so wird für jedes Argument x die Laufzeit von p auf x mindestens 1 sein, weil jeder Rekursionsbaum mindestens einen Knoten enthält. Ist die Komplexität durch $\mathcal{O}(f)$ gegeben, so wird daher stets $\mathcal{O}(f) \supset \mathcal{O}(1)$ gelten.

Intuition: Eine Komplexität von $\mathcal{O}(0)$ würde bedeuten, dass die Prozedur keine Zeit zum Ausführen bräuchte. Da das auf einen realen Rechner aber nie vorkommen kann, ist die kleinstmögliche Laufzeit einer Prozedur 1 und damit liegt ihre Komplexität schon in $\mathcal{O}(1)$.

Aufgabe TI.22 *(Echte Handarbeit)*
Beweisen Sie: $\lambda n \in \mathbb{N}.\ 3n + 7 \in \mathcal{O}(n)$. Verwenden Sie keine Lemmas oder Propositionen, die in Kapitel 11 oder später eingeführt wurden.

Lösungsvorschlag TI.22

$\lambda n \in \mathbb{N}.\ 3n + 7 \in \mathcal{O}(n)$	
$\Leftrightarrow \lambda n \in \mathbb{N}.\ 3n + 7 \in \{g \in OF \mid g \preceq \lambda n \in \mathbb{N}.\ n\}$	Definition $\mathcal{O}$
$\Leftrightarrow \lambda n \in \mathbb{N}.\ 3n + 7 \preceq \lambda n \in \mathbb{N}.\ n$	Mengenlehre, $\lambda n \in \mathbb{N}.\ 3n + 7 \in OF$ mit $n_0 = 0$
$\Leftrightarrow \exists n_0 \in \mathbb{N} : \exists c \in \mathbb{N} : \forall n \geq n_0 : 3n + 7 \leq c \cdot n$	Definition $\preceq$
$\Leftarrow \forall n \geq 7 : 3n + 7 \leq 4n$	Wähle $n_0 \coloneqq 7$, $c \coloneqq 4$
$\Leftrightarrow \forall n \geq 7 : 3n + 7 \leq 3n + n$	Arithmetik
$\Leftrightarrow \top$	$n \geq 7$