

Programmierung 1

Vorlesung 15

Livestream beginnt um 14:15 Uhr



Programmierung 1

1. Schnellkurs
2. Programmiersprachliches
3. Höherstufige Prozeduren
4. Listen und Strings
5. Sortieren
6. Konstruktoren und Ausnahmen
7. Bäume
8. Mengenlehre
- 9. Mathematische Prozeduren**

10. Induktive Korrektheitsbeweise

11. Laufzeit rekursiver Programme
12. Statische und dynamische Semantik
13. Konkrete Syntax
14. Datenstrukturen
15. Speicher und veränderliche Objekte
16. Stapelmaschinen und Übersetzer

Mathematische Objekte als reine Mengen

▶ Boolesche Werte:

false: \emptyset

true: $\{\emptyset\}$

▶ Natürliche Zahlen:

0: \emptyset 2: $\{\{\emptyset\}\}$ 4: $\{\{\{\{\emptyset\}\}\}\}$

1: $\{\emptyset\}$ 3: $\{\{\{\emptyset\}\}\}$...

▶ Zahlenmengen: $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{R}$

$\mathbb{B} := \{0, 1\}$

Boolesche Werte

$\mathbb{N} := \{0, 1, 2, \dots\}$

Natürliche Zahlen

$\mathbb{N}_+ := \{1, 2, 3, \dots\}$

Positive natürliche Zahlen

$\mathbb{Z} := \{\dots, -2, -1, 0, 1, 2, \dots\}$

Ganze Zahlen

$\mathbb{R} :=$ Menge der reellen Zahlen

Reelle Zahlen

▶ Paare:

$(x, y): \{\{x\}, \{x, y\}\}$

Mathematische Prozeduren

- ▶ Eine **(mathematische) Prozedur** ist eine Berechnungsvorschrift, die bei der Anwendung auf ein Argument ein Ergebnis liefert.
- ▶ **Mathematische Prozeduren sind mathematische Objekte, die unabhängig von einer Programmiersprache existieren.**
- ▶ Eine mathematische Prozedur ist gegeben durch
 - ▶ den **Namen** der Prozedur,
 - ▶ den **Argumentbereich**,
 - ▶ den **Ergebnisbereich**, und
 - ▶ eine oder mehrere, eventuell rekursive, **definierende Gleichungen**.

Terminierung

Sei p eine Prozedur $X \rightarrow Y$ und $x \in X$.

Wir sagen, dass die **Anwendung** $p x$

- ▶ **mit dem Ergebnis y terminiert** (auch: **für x mit y terminiert** oder **für x das Ergebnis y liefert**), wenn es ein Ausführungsprotokoll gibt, das mit $p x$ beginnt und mit y endet.
- ▶ **divergiert**, wenn es **kein terminierendes** Ausführungsprotokoll gibt, das mit $p x$ beginnt.

Der **Definitionsbereich** einer Prozedur p ist die Menge der **terminierenden Argumente**:

$$\text{Dom } p := \{x \in X \mid p \text{ terminiert für } x\}$$

Terminierungsfunktion für Prozeduren

- ▶ Eine Prozedur **terminiert für** x genau dann, wenn ihre Rekursionsrelation für x terminiert.
- ▶ Eine Prozedur **terminiert** genau dann, wenn ihre Rekursionsrelation terminiert.
- ▶ Eine Prozedur terminiert für ein Argument genau dann, wenn sie für alle **Folgeargumente** terminiert.

- ▶ Eine Funktion $f \in X \rightarrow \mathbb{N}$ heißt **natürliche Terminierungsfunktion** für eine Prozedur p , wenn für jeden Rekursionsschritt (x, x') von p gilt: $f x > f x'$.
- ▶ **Proposition:** Wenn für eine Prozedur eine natürliche Terminierungsfunktion existiert, dann terminiert die Prozedur für alle Argumente.

Ergebnisfunktion

- ▶ Die **Ergebnisfunktion** einer **Prozedur** p :
 $\{(x, y) \mid x \in \text{Dom } p \text{ und die Anwendung von } p \text{ auf } x \text{ liefert } y\}$
- ▶ Eine Prozedur p **berechnet** eine Funktion f , wenn $\text{Dom } f \subseteq \text{Dom } p$ **und** p für jedes Argument $x \in \text{Dom } f$ das Ergebnis fx liefert.
- ▶ Die Prozedur p berechnet f also genau dann, wenn f eine **Teilmenge** der **Ergebnisfunktion** ist.
- ▶ Die **Ergebnisfunktion** ist die **größte Funktion**, die von p berechnet wird.
- ▶ **Proposition:** Sei p eine Prozedur, die eine Funktion f **berechnet**. Dann ist f die **Ergebnisfunktion** von p genau dann, wenn $\text{Dom } f = \text{Dom } p$.

Ergebnissatz, Korrektheitssatz

Satz (Ergebnis)

Die **Ergebnisfunktion** einer Prozedur $p: X \rightarrow Y$ ist eine Funktion $Dom\ p \rightarrow Y$, die die **definierenden Gleichungen** von p für alle $x \in Dom\ p$ erfüllt.

Satz (Korrektheit)

Eine **Prozedur** p **berechnet eine Funktion** f , wenn die folgenden Bedingungen erfüllt sind:

1. $Dom\ f \subseteq Dom\ p$ **und**
2. f erfüllt die **definierenden Gleichungen** von p .

Potenzen

- ▶ **Prozedur p :**

$$p : \mathbb{Z} \times \mathbb{Z} \times \mathbb{N} \rightarrow \mathbb{Z}$$

$$p(a, x, 0) = a$$

$$p(a, x, n) = p(a \cdot x, x, n - 1) \quad \text{für } n > 0$$

- ▶ **Funktion f :**

$$f \in \mathbb{Z} \times \mathbb{Z} \times \mathbb{N} \rightarrow \mathbb{Z}$$

$$f(a, x, n) = a \cdot x^n$$

- ▶ **Anwendung des Korrektheitssatzes:**

- ▶ p terminiert:

Natürliche **Terminierungsfunktion**: $\lambda(a, x, n) \in \mathbb{Z} \times \mathbb{Z} \times \mathbb{N}. n$

daher: $Dom\ p = Dom\ f$

- ▶ Außerdem erfüllt f die definierenden Gleichungen von p :

$$f(a, x, 0) = a \cdot x^0 = a$$

$$f(a, x, n) = a \cdot x^n = a \cdot x \cdot x^{n-1} = f(a \cdot x, x, n - 1) \quad \text{für } n > 0$$

- ▶ **Damit folgt:** p berechnet f .

Fakultäten

▶ **Prozedur p :**

$$p: \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$p(a, 0) = a$$

$$p(a, n) = p(a \cdot n, n - 1) \quad \text{für } n > 0$$

▶ **Funktion f :**

$$f \in \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$f(a, n) = a \cdot \widehat{fac} n$$

▶ **Anwendung des Korrektheitsatzes:**

1. p terminiert:

Natürliche **Terminierungsfunktion:** $\lambda (a, n) \in \mathbb{N}^2. n$

daher: $Dom p = Dom f$

$$fac: \mathbb{N} \rightarrow \mathbb{N}$$

$$fac 0 = 1$$

$$fac n = n \cdot fac(n - 1) \quad \text{für } n > 0$$

\widehat{fac} ist die Ergebnisfunktion der Prozedur fac .

Fakultäten

▶ **Prozedur p :**

$$p: \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$p(a, 0) = a$$

$$p(a, n) = p(a \cdot n, n - 1) \quad \text{für } n > 0$$

▶ **Funktion f :**

$$f \in \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$f(a, n) = a \cdot \widehat{fac} n$$

▶ **Anwendung des Korrektheitsatzes:**

2. Die definierenden Gleichungen von p sind erfüllt:

$$f(a, 0) = a \cdot \widehat{fac} 0 = a$$

Def. f erste
definierende
Gleichung
von
 fac

$$fac: \mathbb{N} \rightarrow \mathbb{N}$$

$$fac 0 = 1$$

$$fac n = n \cdot fac(n - 1) \quad \text{für } n > 0$$

\widehat{fac} ist die Ergebnisfunktion der
Prozedur fac .

Fakultäten

▶ **Prozedur p :**

$$p: \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$p(a, 0) = a$$

$$p(a, n) = p(a \cdot n, n - 1) \quad \text{für } n > 0$$

▶ **Funktion f :**

$$f \in \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$f(a, n) = a \cdot \widehat{fac} n$$

▶ **Anwendung des Korrektheitssatzes:**

2. Die definierenden Gleichungen von p sind erfüllt:

$$f(a, 0) = a \cdot \widehat{fac} 0 = a$$

$$f(a, n) = a \cdot \widehat{fac} n = a \cdot n \cdot \widehat{fac}(n - 1) = f(a \cdot n, n - 1) \quad \text{für } n > 0$$

Def. f

zweite
definierende
Gleichung
von
 fac

$$fac: \mathbb{N} \rightarrow \mathbb{N}$$

$$fac 0 = 1$$

$$fac n = n \cdot fac(n - 1) \quad \text{für } n > 0$$

\widehat{fac} ist die Ergebnisfunktion der Prozedur fac .

Gaußsche Formel

Gaußsche Formel: $0 + 1 \cdots + n = \frac{n}{2}(n + 1)$ für $n \in \mathbb{N}$

▶ Linke Seite: **Prozedur** *sum*:

$$\textit{sum} : \mathbb{N} \rightarrow \mathbb{N}$$

$$\textit{sum} 0 = 0$$

$$\textit{sum} n = \textit{sum}(n - 1) + n \quad \text{für } n > 0$$

▶ Rechte Seite: **Funktion** *f*:

$$f \in \mathbb{N} \rightarrow \mathbb{N}$$

$$f n = \frac{n}{2}(n + 1)$$

▶ **Anwendung des Korrektheitssatzes:**

1. *sum* **terminiert**: natürliche Terminierungsfunktion $\lambda n \in \mathbb{N}. n$
daher: $\textit{Dom sum} = \textit{Dom f}$

Gaußsche Formel

Gaußsche Formel: $0 + 1 \cdots + n = \frac{n}{2}(n + 1)$ für $n \in \mathbb{N}$

▶ Linke Seite: **Prozedur** sum :

$$sum: \mathbb{N} \rightarrow \mathbb{N}$$

$$sum\ 0 = 0$$

$$sum\ n = sum(n - 1) + n \quad \text{für } n > 0$$

▶ Rechte Seite: **Funktion** f :

$$f \in \mathbb{N} \rightarrow \mathbb{N}$$

$$f\ n = \frac{n}{2}(n + 1)$$

▶ **Anwendung des Korrektheitssatzes:**

2. f erfüllt die definierenden Gleichungen von sum :

$$f\ 0 = \frac{0}{2}(0 + 1) = 0$$

$$f\ n \stackrel{\text{Def. } f}{=} \frac{n}{2}(n + 1) = \left(\frac{n-1}{2} + 1\right)n$$

$$\stackrel{\text{Def. } f}{=} \frac{n-1}{2}(n-1+1) + n = f(n-1) + n \quad \text{für } n > 0$$

Def. f

Erweiterungssatz

- ▶ Eine Prozedur p **erweitert** eine Prozedur q , wenn **jede Anwendungsgleichung** für q bis auf den Prozedurnamen eine Anwendungsgleichung für p ist.
- ▶ **Beispiel:** fac' erweitert fac .

$$fac: \mathbb{N} \rightarrow \mathbb{N}$$

$$fac\ 0 = 1$$

$$fac\ n = n \cdot fac(n-1) \quad \text{für } n > 0$$

$$fac' : \mathbb{Z} \rightarrow \mathbb{Z}$$

$$fac'\ x = \text{if } x = 0 \text{ then } 1 \text{ else } x \cdot fac'(x-1)$$

Satz (Erweiterung)

Wenn eine Prozedur p eine Prozedur q erweitert, gilt

1. Der **Argumentbereich** von q ist eine Teilmenge des Argumentbereichs von p .
2. Jedes **Ausführungsprotokoll** für q ist ein **Ausführungsprotokoll** für p (bis auf den Prozedurnamen).
3. **Terminiert** q für x mit dem **Ergebnis** y , so gilt dies auch für p .

Semantische Äquivalenz

- ▶ Zwei Prozeduren heißen **semantisch äquivalent**, wenn sie **denselben Argumentbereich** und **dieselbe Ergebnisfunktion** haben.
- ▶ **Proposition:** Wenn sich zwei Prozeduren **wechselseitig erweitern**, dann sind sie semantisch äquivalent. **Die Umkehrung gilt nicht.**
- ▶ **Beispiele:**

- ▶ semantisch äquivalent, wechselseitige Erweiterung

$$fib : \mathbb{N} \rightarrow \mathbb{N}$$

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

$$fib' : \mathbb{N} \rightarrow \mathbb{N}$$

$$fib'\ n = n \quad \text{für } n < 2$$

$$fib'\ n = fib'(n-1) + fib'(n-2) \quad \text{für } n \geq 2$$

- ▶ semantisch äquivalent, **keine wechselseitige Erweiterung**

$$p : \mathbb{N} \rightarrow \mathbb{N}$$

$$pn = 0$$

$$q : \mathbb{N} \rightarrow \mathbb{N}$$

$$qn = \text{if } n = 0 \text{ then } 0 \text{ else } q(n-1)$$

Kapitel 10

Induktive

Korrektheitsbeweise

Induktion

Behauptung $\forall n \in \mathbb{N}: n < 2^n$.

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

Sei $n = 0$. Dann $n = 0 < 1 = 2^0 = 2^n$.

Sei $n = 1$. Dann $n = 1 < 2 = 2^1 = 2^n$.

Sei $n \geq 2$. Durch Induktion haben wir einen Beweis für $n - 1 < 2^{n-1}$. Also gilt:

$$\begin{aligned} 2^n &= 2 \cdot 2^{n-1} \\ &> 2(n-1) \\ &= n + (n-2) \\ &\geq n \end{aligned}$$

Induktion für $n - 1$

$$n \geq 2$$

Induktion

Behauptung $\forall n \in \mathbb{N}: n < 2^n$.

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

Sei $n = 0$. Dann $n = 0 < 1 = 2^0 = 2^n$.

Sei $n = 1$. Dann $n = 1 < 2 = 2^1 = 2^n$.

Sei $n \geq 2$. Durch Induktion haben wir einen Beweis für $n - 1 < 2^{n-1}$. Also gilt:

$$\begin{aligned} 2^n &= 2 \cdot 2^{n-1} \\ &> 2(n-1) \\ &= n + (n-2) \\ &\geq n \end{aligned}$$

Induktion für $n - 1$

$n \geq 2$



- Beweis durch **Induktion über ...**
- **Induktion für...**

Induktion

Behauptung Jede natürliche Zahl $n \geq 2$ kann als Produkt von Primzahlen dargestellt werden.

Beweis Durch Induktion über $n \geq 2$. Wir unterscheiden zwei Fälle.

n ist eine Primzahl. Dann ist die Behauptung trivial.

n ist keine Primzahl. Dann gibt es $a, b \in \mathbb{N}$ mit $n = a \cdot b$ und $2 \leq a, b < n$. Mit Induktion für a und b folgt, dass a und b als Produkte von Primzahlen darstellbar sind. Also ist $n = ab$ als Produkt von Primzahlen darstellbar.

Vollständige Induktion

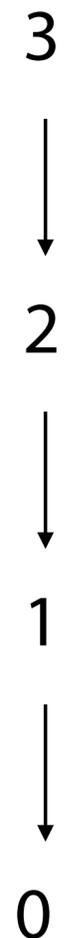
- ▶ Sei $\forall n \in \mathbb{N}: A(n)$ eine **allquantifizierte Aussage** über die natürlichen Zahlen.
- ▶ Um die allquantifizierte Aussage zu beweisen, zeigen wir
 - ▶ den **Induktionsanfang**: $A(0)$
 - ▶ den **Induktionsschritt**: $\forall n \in \mathbb{N}: A(n) \Rightarrow A(n+1)$.
 $A(n)$ heißt **Induktionshypothese**

▶ Vollständige Induktion:

$$\underbrace{(A(0))}_{\text{Induktionsanfang}} \wedge \underbrace{(\forall n \in \mathbb{N}: A(n) \Rightarrow A(n+1))}_{\text{Induktionsschritt}} \Rightarrow \forall n \in \mathbb{N}: A(n)$$

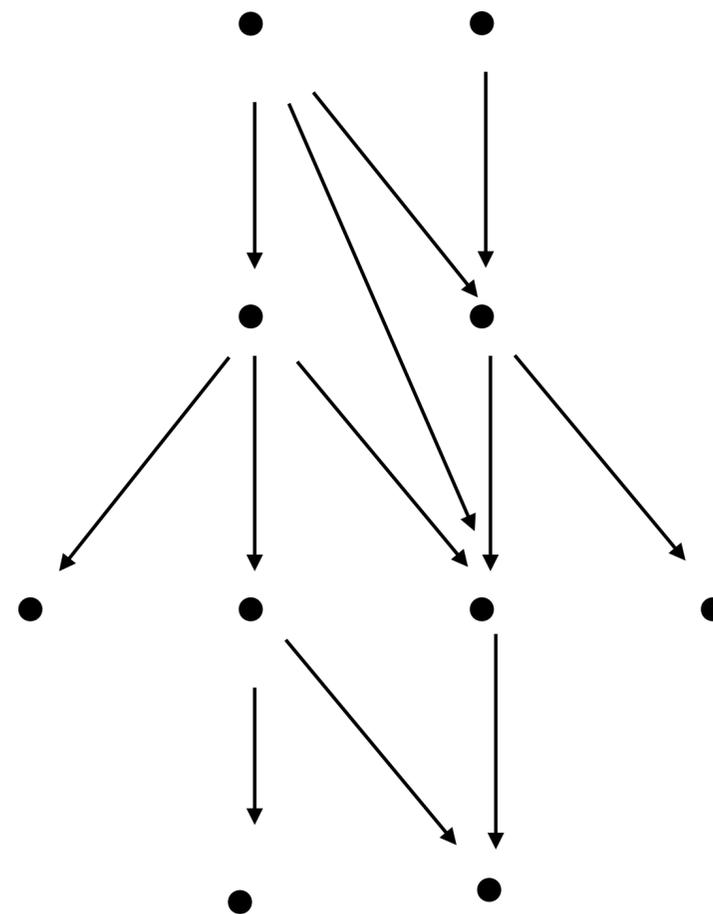
Vollständige vs. wohlfundierte Induktion

Vollständige Induktion



$$(A(0) \wedge \forall n \in \mathbb{N}: A(n) \Rightarrow A(n+1)) \\ \Rightarrow \forall n \in \mathbb{N}: A(n)$$

Wohlfundierte Induktion



$$(\forall x \in X (\forall y \in X: x > y \Rightarrow A(y)) \Rightarrow A(x)) \\ \Rightarrow \forall x \in X: A(x)$$

Wohlfundierte Induktion (Noethersche Induktion)

- ▶ Sei $\forall x \in X: A(x)$ eine **allquantifizierte Aussage** über eine Menge X .
- ▶ Eine **Induktionsrelation** $>$ ist eine **terminierende Relation** auf X .
Um die allquantifizierte Aussage zu beweisen, zeigen wir den **Induktionsschritt**:
für jedes Argument x folgt aus der Tatsache, dass für jedes kleinere Argument y $A(y)$ gilt, dass $A(x)$ gilt.

▶ Wohlfundierte Induktion:

$$\underbrace{(\forall x \in X (\forall y \in X: x > y \Rightarrow A(y)) \Rightarrow A(x))}_{\text{Induktionsschritt}} \Rightarrow \forall x \in X: A(x)$$

Induktionsschritt



Amalie „Emmy“ Noether
(1882 – 1935)

Natürliche vs. strukturelle Induktion

- ▶ Wenn es sich bei der Induktionsrelation um Ter handelt, sprechen wir von **natürlicher Induktion**.

$$Ter := \{(x, y) \in \mathbb{N}^2 \mid x > y\}$$

- ▶ Wenn es sich bei der Induktionsrelation um eine **strukturelle Relation** handelt, sprechen wir von **struktureller Induktion**.
Dazu später mehr.



Ist der folgende Beweis für die Behauptung "für alle n : $2^n = 1$ " korrekt?

Beweis durch Induktion über $n \geq 0$. Wir unterscheiden zwei Fälle.

Sei $n=0$: Dann $2^0 = 1$.

Sei $n > 0$: Dann $2^n = (2^{n-1} \times 2^{n-1}) / 2^{n-2} = (1 \times 1) / 2^{n-2} = 1 / 1 = 1$.

A: Ja



B: Nein, Argument für $n=0$ ist falsch



C: Nein, Argument für $n=1$ ist falsch



D: Nein, Argument für $n=2$ ist falsch



Beispiel: *sum*

$$\text{sum}: (\mathbb{N} \rightarrow \mathbb{N}) \times \mathbb{N} \rightarrow \mathbb{N}$$

$$\text{sum}(f, n) = f\ 0 \quad \text{für } n < 1$$

$$\text{sum}(f, n) = \text{sum}(f, n-1) + f\ n \quad \text{für } n \geq 1$$

Behauptung: $\forall n \in \mathbb{N}: \text{sum}(f, n) = \sum_{i=0}^n f\ i$

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

Sei $n=0$. $\text{sum}(f, 0) = f\ 0 = \sum_{i=0}^0 f\ i$ Definition *sum*

Sei $n > 0$. $\text{sum}(f, n) = \text{sum}(f, n-1) + f\ n$ Definition *sum*

Induktion für $n-1$ $= \sum_{i=0}^{n-1} f\ i + f\ n$
 $= \sum_{i=0}^n f\ i$

Bestimmte Iteration

$$\text{iter}: (\mathbb{N} \times X \times (X \rightarrow X)) \rightarrow X$$

$$\text{iter}(0, x, f) = x$$

$$\text{iter}(n, x, f) = \text{iter}(n-1, f x, f) \quad \text{für } n > 0$$



Schrittfunktion

$$\text{iter}(3, x, f) = \text{iter}(2, f x, f)$$

$$= \text{iter}(1, f(f x), f)$$

$$= \text{iter}(0, f(f(f x)), f)$$

$$= f(f(f x))$$



Terminiert die mathematische Prozedur iter?

$iter: (\mathbb{N} \times X \times (X \rightarrow X)) \rightarrow X$

$iter(0, x, f) = x$

$iter(n, x, f) = iter(n-1, f x, f)$ für $n > 0$

A: Ja, für alle Argumente ✓

B: Ja, für manche (aber
nicht alle) Argumente ✗

C: Nein ✗

Bestimmte Iteration

$$\text{iter}: (\mathbb{N} \times X \times (X \rightarrow X)) \rightarrow X$$

$$\text{iter}(0, x, f) = x$$

$$\text{iter}(n, x, f) = \text{iter}(n-1, f x, f) \quad \text{für } n > 0$$

↓
Schrittfunktion

Wichtiger Unterschied zur konkreten Prozedur *iter* aus Vorlesung 5:

- ▶ Das Argument f der mathematischen Prozedur ist eine **Funktion**.
- ▶ Im Gegensatz dazu wird der konkreten Prozedur eine **Prozedur** übergeben. Wenn die Schrittprozedur divergiert, divergiert auch das konkrete *iter*. **Die mathematische Prozedur terminiert immer!**

Bestimmte Iteration

$$\text{iter}: (\mathbb{N} \times X \times (X \rightarrow X)) \rightarrow X$$

$$\text{iter}(0, x, f) = x$$

$$\text{iter}(n, x, f) = \text{iter}(n-1, f x, f) \quad \text{für } n > 0$$

Behauptung: (Vertauschungseigenschaft von *iter*)

$$\forall n \in \mathbb{N} \quad \forall x \in X \quad \forall f \in X \rightarrow X: \text{iter}(n+1, x, f) = f(\text{iter}(n, x, f)).$$

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

Sei $n=0$. Sei $x \in X$ beliebig, sei $f \in X \rightarrow X$ beliebig.

$$\begin{aligned} \text{iter}(n+1, x, f) &= \text{iter}(1, x, f) = \text{iter}(0, f x, f) = f x \\ &= f(\text{iter}(0, x, f)) = f(\text{iter}(n, x, f)) \end{aligned}$$

Sei $n > 0$. Sei $x \in X$ beliebig, sei $f \in X \rightarrow X$ beliebig.

$$\begin{aligned} \text{iter}(n+1, x, f) &= \text{iter}(n, f x, f) && \text{Definition } \textit{iter} \\ &= f(\text{iter}(n-1, f x, f)) && \text{Induktion für } n-1 \\ &= f(\text{iter}(n, x, f)) && \text{Definition } \textit{iter} \end{aligned}$$

www.prog1.saarland