Kapitel 3
Rekursion und Induktion

Abschnitt 3.1

Rekursive Definitionen

Rekursive Definitionen von Funktionen über den natürlichen Zahlen

Eine Funktion $f : \mathbb{N} \to Y$ (für eine beliebige Menge Y) lässt sich wie folgt definieren:

Rekursionsanfang. Definiere $f(0)$.

Rekursionsschritt. Definiere $f(n+1)$ aus $f(n)$.

Man spricht von einer rekursiven (oder auch induktiven) Definition.

Rekursive Definitionen von Funktionen über den natürlichen Zahlen

Eine Funktion $f : \mathbb{N} \to Y$ (für eine beliebige Menge Y) lässt sich wie folgt definieren:

Rekursionsanfang. Definiere $f(0)$.

Rekursionsschritt. Definiere $f(n+1)$ aus $f(n)$.

Man spricht von einer rekursiven (oder auch induktiven) Definition.

Varianten

- Im Rekursionsanfang definiert man nicht nur $f(0)$, sondern auch andere Werte.

Rekursive Definitionen von Funktionen über den natürlichen Zahlen

Eine Funktion $f : \mathbb{N} \to Y$ (für eine beliebige Menge Y) lässt sich wie folgt definieren:

Rekursionsanfang. Definiere $f(0)$.

Rekursionsschritt. Definiere $f(n+1)$ aus $f(n)$.

Man spricht von einer rekursiven (oder auch induktiven) Definition.

Varianten

- Im Rekursionsanfang definiert man nicht nur $f(0)$, sondern auch andere Werte.
- Im Rekursionsschritt greift man zur Definition von $f(n+1)$ nicht nur auf $f(n)$, sondern auch auf andere Werte $f(i)$ für $i \leq n$ zurück, möglicherweise auf mehrere.

Beispiel 3.1 (Fakultät)

Die Fakultätsfunktion $f : \mathbb{N} \to \mathbb{N}$ ist rekursiv wie folgt definiert:

$$\begin{aligned} f(0) &:= 1 \\ f(n+1) &:= f(n) \cdot (n+1) \qquad \text{für } n \in \mathbb{N}. \end{aligned}$$

Beispiel 3.1 (Fakultät)

Die Fakultätsfunktion $f : \mathbb{N} \to \mathbb{N}$ ist rekursiv wie folgt definiert:

$$\begin{aligned} f(0) &:= 1 \\ f(n+1) &:= f(n) \cdot (n+1) \qquad \text{für } n \in \mathbb{N}. \end{aligned}$$

Dann gilt $f(n) = 1 \cdot 2 \cdot \ldots \cdot n$ für alle $n \in \mathbb{N}_+$
Man schreibt $f(n) =: n!$.

Beispiel 3.1 (Fakultät)

Die Fakultätsfunktion $f : \mathbb{N} \to \mathbb{N}$ ist rekursiv wie folgt definiert:

$$\begin{aligned} f(0) &:= 1 \\ f(n+1) &:= f(n) \cdot (n+1) \qquad \text{für } n \in \mathbb{N}. \end{aligned}$$

Dann gilt $f(n) = 1 \cdot 2 \cdot \ldots \cdot n$ für alle $n \in \mathbb{N}_+$
Man schreibt $f(n) =: n!$.

Beispiel 3.2 (Fibonacci Folge)

Wir definieren $f : \mathbb{N} \to \mathbb{N}$ durch

$$\begin{aligned} f(0) &:= 1 \\ f(1) &:= 1 \\ f(n+2) &:= f(n) + f(n+1) \qquad \text{für } n \in \mathbb{N}. \end{aligned}$$

Beispiel 3.1 (Fakultät)

Die Fakultätsfunktion $f : \mathbb{N} \to \mathbb{N}$ ist rekursiv wie folgt definiert:

$$\begin{aligned} f(0) &:= 1 \\ f(n+1) &:= f(n) \cdot (n+1) \qquad \text{für } n \in \mathbb{N}. \end{aligned}$$

Dann gilt $f(n) = 1 \cdot 2 \cdot \ldots \cdot n$ für alle $n \in \mathbb{N}_+$
Man schreibt $f(n) =: n!$.

Beispiel 3.2 (Fibonacci Folge)

Wir definieren $f : \mathbb{N} \to \mathbb{N}$ durch

$$\begin{aligned} f(0) &:= 1 \\ f(1) &:= 1 \\ f(n+2) &:= f(n) + f(n+1) \qquad \text{für } n \in \mathbb{N}. \end{aligned}$$

Man nennt die Folge $f(0), f(1), f(2), \ldots$ die Fibonacci Folge.

Beispiel 3.3 (Unäre Kodierung)

Die Funktion $f : \mathbb{N} \to \{1\}^*$ sei wie folgt definiert:

$$\begin{aligned} f(0) &:= \varepsilon, \\ f(n+1) &:= f(n)1 \qquad \text{für } n \in \mathbb{N}. \end{aligned}$$

Beispiel 3.3 (Unäre Kodierung)

Die Funktion $f : \mathbb{N} \to \{1\}^*$ sei wie folgt definiert:

$$\begin{aligned} f(0) &:= \varepsilon, \\ f(n+1) &:= f(n)1 \qquad \text{für } n \in \mathbb{N}. \end{aligned}$$

Dann ist $f(n) = 1^n = \underbrace{1 \ldots 1}_{n \text{ mal}}$.

Beispiel 3.3 (Unäre Kodierung)

Die Funktion $f : \mathbb{N} \to \{1\}^*$ sei wie folgt definiert:

$$\begin{aligned} f(0) &:= \varepsilon, \\ f(n+1) &:= f(n)1 \qquad \text{für } n \in \mathbb{N}. \end{aligned}$$

Dann ist $f(n) = 1^n = \underbrace{1 \ldots 1}_{n \text{ mal}}$.

Beispiel 3.4 (Binäre Kodierung)

Die Funktion $f : \mathbb{N} \to \{0,1\}^*$ sei wie folgt definiert:

$$\begin{aligned} f(0) &:= 0, \\ f(1) &:= 1, \\ f(n+1) &:= \begin{cases} f\big(n/2\big)1 & \text{falls } n \text{ gerade,} \\ f\big((n+1)/2\big)0 & \text{sonst,} \end{cases} \qquad \text{für } n \in \mathbb{N}_+. \end{aligned}$$

Dann ist $f(n)$ die Binärkodierung von n.

Rekursive Definition von Mengen

Eine rekursive Definition (oder auch induktive Definition) einer Menge X besteht aus:

Rekursive Definition von Mengen

Eine rekursive Definition (oder auch induktive Definition) einer Menge X besteht aus:

Basisregeln der Form

$$„x \in X“$$

Rekursive Definition von Mengen

Eine rekursive Definition (oder auch induktive Definition) einer Menge X besteht aus:

Basisregeln der Form

$$\text{„}x \in X\text{“}$$

Rekursiven Regeln der Form

$$\text{„Wenn } x_1, \ldots, x_k \in X, \text{ dann } x \in X\text{“},$$

wobei x von $x_1, \ldots, x_k$ abhängt.

Rekursive Definition von Mengen

Eine rekursive Definition (oder auch induktive Definition) einer Menge X besteht aus:

Basisregeln der Form

$$\text{„}x \in X\text{“}$$

Rekursiven Regeln der Form

$$\text{„Wenn } x_1, \ldots, x_k \in X, \text{ dann } x \in X\text{“},$$

wobei x von $x_1, \ldots, x_k$ abhängt.

Die definierte Menge X ist dann die Menge aller Elemente, deren Zugehörigkeit zu X durch endlichmaliges Anwenden der Regeln gezeigt werden kann.

Beispiel 3.5: Zweierpotenzen

Die Menge $Z \subseteq \mathbb{N}$ sei rekursiv wie folgt definiert:

Basisregel.

$$1 \in Z.$$

Rekursive Regeln. Für alle $n \in \mathbb{N}$:

$$\text{Wenn } n \in Z, \text{ dann } 2n \in Z.$$

Es ist leicht zu sehen, dass

$$Z = \{2^k \mid k \in \mathbb{N}\}.$$

Beispiel 3.6: Natürliche Zahlen in Binärdarstellung

Sei Σ_{ASCII} das ASCII-Alphabet. Wir verwenden für Symbole aus Σ_{ASCII} einen Schreibmaschinenfont (`A`,`B`,`...`,`a`,`b`,`...`,`0`,`1`,`...`,`(`,`)`,`...`).

Beispiel 3.6: Natürliche Zahlen in Binärdarstellung

Sei Σ_{ASCII} das ASCII-Alphabet. Wir verwenden für Symbole aus Σ_{ASCII} einen Schreibmaschinenfont ($\texttt{A}, \texttt{B}, \ldots, \texttt{a}, \texttt{b}, \ldots, \texttt{0}, \texttt{1}, \ldots, \texttt{(}, \texttt{)}, \ldots$).

Die Menge $\texttt{BIN} \subseteq \Sigma^*_{\text{ASCII}}$ sei wie folgt rekursiv definiert:

Basisregeln.

$$\texttt{0}, \texttt{1} \in \texttt{BIN}.$$

Rekursive Regeln.
Für alle $w \in \Sigma^*_{\text{ASCII}} \setminus \{\texttt{0}\}$:

$$\text{Wenn } w \in \texttt{BIN}, \text{ dann } w\texttt{0}, w\texttt{1} \in \texttt{BIN}.$$

Beispiel 3.6: Natürliche Zahlen in Binärdarstellung

Sei Σ_{ASCII} das ASCII-Alphabet. Wir verwenden für Symbole aus Σ_{ASCII} einen Schreibmaschinenfont ($\texttt{A},\texttt{B},\ldots,\texttt{a},\texttt{b},\ldots,\texttt{0},\texttt{1},\ldots,\texttt{(},\texttt{)},\ldots$).

Die Menge $\texttt{BIN} \subseteq \Sigma^*_{\text{ASCII}}$ sei wie folgt rekursiv definiert:

Basisregeln.

$$\texttt{0}, \texttt{1} \in \texttt{BIN}.$$

Rekursive Regeln.
Für alle $w \in \Sigma^*_{\text{ASCII}} \setminus \{\texttt{0}\}$:

$$\text{Wenn } w \in \texttt{BIN}, \text{ dann } w\texttt{0}, w\texttt{1} \in \texttt{BIN}.$$

Die Menge $\texttt{BIN}$ lässt sich allerdings auch leicht nichtrekursiv definieren durch:

$$\texttt{BIN} = \{\texttt{0}\} \cup \{\texttt{1}\}\{\texttt{0},\texttt{1}\}^*.$$

Beispiel 3.7: Listen

Die Menge $\texttt{LIST(BIN)} \subseteq \Sigma^*_{\text{ASCII}}$ sei rekursiv wie folgt definiert:

Beispiel 3.7: Listen

Die Menge $\texttt{LIST(BIN)} \subseteq \Sigma^*_{\text{ASCII}}$ sei rekursiv wie folgt definiert:

Basisregel.

$$\texttt{nil} \in \texttt{LIST(BIN)} \qquad \text{(die leere Liste)}$$

Beispiel 3.7: Listen

Die Menge $\texttt{LIST(BIN)} \subseteq \Sigma^*_{\text{ASCII}}$ sei rekursiv wie folgt definiert:

Basisregel.

$$\texttt{nil} \in \texttt{LIST(BIN)} \qquad \text{(die leere Liste)}$$

Rekursive Regeln.
Für alle $v \in \texttt{BIN}$ und $w \in \Sigma^*_{\text{ASCII}}$:

Wenn $w \in \texttt{LIST(BIN)}$, dann $\texttt{cons}(v,w) \in \texttt{LIST(BIN)}$.

Beispiel 3.7: Listen

Die Menge $\texttt{LIST(BIN)} \subseteq \Sigma^*_{\text{ASCII}}$ sei rekursiv wie folgt definiert:

Basisregel.

$$\texttt{nil} \in \texttt{LIST(BIN)} \qquad \text{(die leere Liste)}$$

Rekursive Regeln.
Für alle $v \in \texttt{BIN}$ und $w \in \Sigma^*_{\text{ASCII}}$:

Wenn $w \in \texttt{LIST(BIN)}$, dann $\texttt{cons}(v,w) \in \texttt{LIST(BIN)}$.

Beispiele

$$\texttt{nil} \in \texttt{LIST(BIN)},$$
$$\texttt{cons(23,cons(0,cons(113,nil)))} \in \texttt{LIST(BIN)}.$$

Beispiel 3.8: Klammerausdrücke

Sei $\Sigma = \{(,)\}$. Wir definieren rekursiv die Menge $L_K \subseteq \Sigma^*$ aller „korrekten Klammerausdrücke"', also aller Wörter über Σ^*, in denen jede sich öffnende Klammer geschlossen wird.

Basisregel.

$$\varepsilon \in L_K.$$

Rekursive Regeln.
Für alle $k \in \mathbb{N}_+$ und alle $w_1, \ldots, w_k \in \Sigma^*$:

$$\text{Wenn } w_1, \ldots, w_k \in L_K, \text{ dann } (w_1 w_2 \ldots w_k) \in L_K.$$

Beispiel 3.8: Klammerausdrücke

Sei $\Sigma = \{(,)\}$. Wir definieren rekursiv die Menge $L_K \subseteq \Sigma^*$ aller „korrekten Klammerausdrücke"', also aller Wörter über Σ^*, in denen jede sich öffnende Klammer geschlossen wird.

Basisregel.

$$\varepsilon \in L_K.$$

Rekursive Regeln.
Für alle $k \in \mathbb{N}_+$ und alle $w_1, \ldots, w_k \in \Sigma^*$:

$$\text{Wenn } w_1, \ldots, w_k \in L_K, \text{ dann } (w_1 w_2 \ldots w_k) \in L_K.$$

Beispiele

$$\varepsilon, \quad (()()), \quad ()(())((())) \in L_K$$
$$((), \quad ())(\notin L_K$$

Beispiel 3.9

Sei Σ^* ein Alphabet. Dann lässt sich Σ^* rekursiv wie folgt definieren.

Basisregel.

$$\varepsilon \in \Sigma^*$$

Rekursive Regeln.
Für alle w und alle $a \in \Sigma$:

$$\text{Wenn } w \in \Sigma^*, \text{ dann } wa \in \Sigma^*.$$

Ableitungen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.

Ableitungen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.

Frage

Was genau bedeutet es, dass die Zugehörigkeit eines Elementes x zu X „durch endlichmaliges Anwenden der Regeln in $\mathcal{D}$ gezeigt werden kann“?

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.

Frage

Was genau bedeutet es, dass die Zugehörigkeit eines Elementes x zu X „durch endlichmaliges Anwenden der Regeln in $\mathcal{D}$ gezeigt werden kann“?

Definition 3.10

Eine Ableitung eines Elementes z in $\mathcal{D}$ ist eine Folge $(y_1, \ldots, y_n)$, so dass

- $y_n = z$;
- für alle $i \leq n$ gibt es
  - eine Basisregel „$x \in X$“ in $\mathcal{D}$, so dass $y_i = x$,
  - oder eine rekursive Regel „Wenn $x_1, \ldots, x_k \in X$, dann $x \in X$“ in $\mathcal{D}$ und $j_1, j_2, \ldots, j_k < i$, so dass $y_{j_1} = x_1, y_{j_2} = x_2, \ldots, y_{j_k} = x_k$ und $y_i = x$.

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.

Frage

Was genau bedeutet es, dass die Zugehörigkeit eines Elementes x zu X „durch endlichmaliges Anwenden der Regeln in $\mathcal{D}$ gezeigt werden kann“?

Definition 3.10

Eine Ableitung eines Elementes z in $\mathcal{D}$ ist eine Folge $(y_1, \ldots, y_n)$, so dass

- $y_n = z$;
- für alle $i \leq n$ gibt es
  - eine Basisregel „$x \in X$“ in $\mathcal{D}$, so dass $y_i = x$,
  - oder eine rekursive Regel „Wenn $x_1, \ldots, x_k \in X$, dann $x \in X$“ in $\mathcal{D}$ und $j_1, j_2, \ldots, j_k < i$, so dass $y_{j_1} = x_1, y_{j_2} = x_2, \ldots, y_{j_k} = x_k$ und $y_i = x$.

Jetzt können wir formal die von $\mathcal{D}$ definierte Menge X definieren durch

$$X := \{z \mid z \text{ hat eine Ableitung in } \mathcal{D}\}.$$

Beispiel 3.11

Eine Ableitung von `(()(()))` in der rekursiven Definition von L_K (vgl. Beispiel 3.8):

$$\big(\varepsilon, \quad \texttt{()}, \quad \texttt{(())}, \quad \texttt{(()(()))}\big).$$

Beispiel 3.11

Eine Ableitung von `(()(()))` in der rekursiven Definition von L_K (vgl. Beispiel 3.8):

$$\big(\varepsilon,\quad \texttt{()},\quad \texttt{(())},\quad \texttt{()(())}\big).$$

Eine verständlichere Darstellung dieser Ableitung ist:

1.	ε,	Basisregel
2.	`()`,	Rek. Regel auf 1
3.	`(())`,	Rek. Regel auf 2
4.	`()(())`,	Rek. Regel auf 2 und 3.

Beispiel 3.11

Eine Ableitung von `(()(()))` in der rekursiven Definition von L_K (vgl. Beispiel 3.8):

$$\big(\varepsilon,\quad \texttt{()},\quad \texttt{(())},\quad \texttt{()(())}\big).$$

Eine verständlichere Darstellung dieser Ableitung ist:

1.	ε,	Basisregel
2.	`()`,	Rek. Regel auf 1
3.	`(())`,	Rek. Regel auf 2
4.	`()(())`,	Rek. Regel auf 2 und 3.

Beispiel 3.12

Eine Ableitung von 16 in der rekursiven Definition der Menge Z der Zweierpotenzen (vgl. Beispiel 3.5):

$$(1, 2, 4, 8, 16).$$

Definition 3.13

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.
Eine Menge Y ist abgeschlossen unter $\mathcal{D}$, wenn sie folgende Bedingungen erfüllt:

(i) Für alle Basisregeln „$x \in X$“ in $\mathcal{D}$ gilt $x \in Y$.

(ii) Für alle rekursiven Regeln „Wenn $x_1, \ldots, x_k \in X$, dann $x \in X$“ in $\mathcal{D}$ gilt:

$$\text{Wenn } x_1, \ldots, x_k \in Y, \text{ dann } x \in Y.$$

Definition 3.13

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.
Eine Menge Y ist abgeschlossen unter $\mathcal{D}$, wenn sie folgende Bedingungen erfüllt:

(i) Für alle Basisregeln „$x \in X$“ in $\mathcal{D}$ gilt $x \in Y$.

(ii) Für alle rekursiven Regeln „Wenn $x_1, \ldots, x_k \in X$, dann $x \in X$“ in $\mathcal{D}$ gilt:

$$\text{Wenn } x_1, \ldots, x_k \in Y, \text{ dann } x \in Y.$$

Beispiel 3.14

Die Mengen $\texttt{BIN}$, Σ^*_{ASCII} und

$$\{\texttt{A}, \texttt{B}\}^*\texttt{BIN}$$

sind abgeschlossen unter der rekursiven Definition der Menge $\texttt{BIN}$.

Die kleinste abgeschlossene Menge

Satz 3.15

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.

1. *X ist abgeschlossen unter $\mathcal{D}$.*
2. *Für jede unter $\mathcal{D}$ abgeschlossene Menge Y gilt $X \subseteq Y$.*

Also ist X die kleinste unter $\mathcal{D}$ abgeschlossene Menge.

1. Für alle Basisregeln „$x \in X$“ in $\mathcal{D}$ ist

$$(x)$$

eine Ableitung von x in $\mathcal{D}$, also gilt $x \in X$.
Betrachten wir eine rekursive Regel „Wenn $x_1, \ldots, x_k \in X$, dann $x \in X$“ in $\mathcal{D}$, und nehmen wir an, dass $x_1, \ldots, x_k \in X$. Für $1 \leq i \leq k$ sei

$$(y_{i1}, \ldots, y_{in_i})$$

eine Ableitung von x_i in $\mathcal{D}$.
Dann ist

$$(y_{11}, \ldots, \underbrace{y_{1n_1}}_{=x_1}, y_{21}, \ldots, \underbrace{y_{2n_2}}_{=x_2}, y_{31}, \quad \ldots \quad , \underbrace{y_{(k-1)n_{k-1}}}_{=x_{k-1}}, y_{k1}, \ldots, \underbrace{y_{kn_k}}_{=x_k}, x)$$

eine Ableitung von x in $\mathcal{D}$. Also gilt $x \in X$.

2. Sei Y abgeschlossen unter $\mathcal{D}$. Um zu zeigen, dass $X \subseteq Y$ zeigen wir, dass jedes beliebiges $z \in X$ auch in Y liegt.

Sei also $z \in X$, und sei

$$y_1, \ldots, y_n$$

eine Ableitung von z in $\mathcal{D}$.
Wir zeigen per Induktion über $i, 1 \leq i \leq n$, dass $y_i \in Y$. Insbesondere ist dann $z = y_n \in Y$.

Induktionsanfang: $i = 1$.
Dann enthält $\mathcal{D}$ eine Basisregel „$x \in X$“ mit $x = y_1$.
Weil Y abgeschlossen ist gilt $y_1 = x \in Y$.

Induktionsschritt: $i \to i + 1$, wobei $i + 1 \leq n$.
Wenn $\mathcal{D}$ eine Basisregel „$x \in X$“ mit $x = y_{i+1}$ enthält, so gilt wie im Induktionsanfang $y_{i+1} = x \in Y$.
Sonst enthält $\mathcal{D}$ eine rekursive Regel „Wenn $x_1, \ldots, x_k \in X$, dann $x \in X$“, und es gibt $j_1, j_2, \ldots, j_k \leq i$, so dass $y_{j_1} = x_1, y_{j_2} = x_2, \ldots, y_{j_k} = x_k$ und $y_{i+1} = x$.
Nach Induktionsannahme sind $y_{j_1}, y_{j_2}, \ldots, y_{j_k} \in Y$.
Weil Y abgeschlossen ist, gilt dann auch $y_{i+1} = x \in Y$. □

Rekursive Definition von Funktionen über rekursiv definierten Mengen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X, und sei Y eine beliebige Menge.

Rekursive Definition von Funktionen über rekursiv definierten Mengen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X, und sei Y eine beliebige Menge.

Eine Funktion $f : X \to Y$ lässt sich wie folgt definieren.

Rekursive Definition von Funktionen über rekursiv definierten Mengen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X, und sei Y eine beliebige Menge.

Eine Funktion $f : X \to Y$ lässt sich wie folgt definieren.

Rekursionsanfang.
Für alle Basisregeln „$x \in X$“ in $\mathcal{D}$, definiere $f(x)$.

Rekursive Definition von Funktionen über rekursiv definierten Mengen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X, und sei Y eine beliebige Menge.

Eine Funktion $f : X \to Y$ lässt sich wie folgt definieren.

Rekursionsanfang.
Für alle Basisregeln „$x \in X$“ in $\mathcal{D}$, definiere $f(x)$.

Rekursionsschritt.
Für alle rekursiven Regeln „Wenn $x_1, \ldots, x_k \in X$, dann $x \in X$“, definiere $f(x)$ aus $f(x_1), \ldots, f(x_k)$.

Rekursive Definition von Funktionen über rekursiv definierten Mengen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X, und sei Y eine beliebige Menge.

Eine Funktion $f : X \to Y$ lässt sich wie folgt definieren.

Rekursionsanfang.
Für alle Basisregeln „$x \in X$“ in $\mathcal{D}$, definiere $f(x)$.

Rekursionsschritt.
Für alle rekursiven Regeln „Wenn $x_1, \ldots, x_k \in X$, dann $x \in X$“, definiere $f(x)$ aus $f(x_1), \ldots, f(x_k)$.

Wir sagen: f ist rekursiv (oder induktiv) über $\mathcal{D}$ definiert.

Rekursive Definition von Funktionen über rekursiv definierten Mengen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X, und sei Y eine beliebige Menge.

Eine Funktion $f : X \to Y$ lässt sich wie folgt definieren.

Rekursionsanfang.
Für alle Basisregeln „$x \in X$“ in $\mathcal{D}$, definiere $f(x)$.

Rekursionsschritt.
Für alle rekursiven Regeln „Wenn $x_1, \ldots, x_k \in X$, dann $x \in X$“, definiere $f(x)$ aus $f(x_1), \ldots, f(x_k)$.

Wir sagen: f ist rekursiv (oder induktiv) über $\mathcal{D}$ definiert.

Oft geben wir die rekursive Definition $\mathcal{D}$ von X nicht explizit an, sondern nehmen an, dass sie aus dem Kontext bekannt ist. Wir sagen dann auch:f ist rekursiv (oder induktiv) über den Aufbau von X definiert.

Beispiel 3.16 (Länge eines Wortes)

Sei Σ ein Alphabet. Wir definieren die Funktion $|\cdot| : \Sigma^* \to \mathbb{N}$ rekursiv über die rekursive Definition von Σ^* aus Beispiel 3.9.

Beispiel 3.16 (Länge eines Wortes)

Sei Σ ein Alphabet. Wir definieren die Funktion $|\cdot| : \Sigma^* \to \mathbb{N}$ rekursiv über die rekursive Definition von Σ^* aus Beispiel 3.9.

Rekursionsanfang. $|\varepsilon| := 0$.

Beispiel 3.16 (Länge eines Wortes)

Sei Σ ein Alphabet. Wir definieren die Funktion $|\cdot| : \Sigma^* \to \mathbb{N}$ rekursiv über die rekursive Definition von Σ^* aus Beispiel 3.9.

Rekursionsanfang. $|\varepsilon| := 0$.

Rekursionsschritt. Für alle $w \in \Sigma^*$ und $a \in \Sigma$:

$$|wa| := |w| + 1.$$

Beispiel 3.16 (Länge eines Wortes)

Sei Σ ein Alphabet. Wir definieren die Funktion $|\cdot| : \Sigma^* \to \mathbb{N}$ rekursiv über die rekursive Definition von Σ^* aus Beispiel 3.9.

Rekursionsanfang. $|\varepsilon| := 0$.

Rekursionsschritt. Für alle $w \in \Sigma^*$ und $a \in \Sigma$:

$$|wa| := |w| + 1.$$

Beispiel 3.17 (Länge einer Liste)

Wir definieren die Funktion $\mathrm{lg} : \texttt{LIST(BIN)} \to \mathbb{N}$ rekursiv über den Aufbau von `LIST(BIN)`:

Beispiel 3.16 (Länge eines Wortes)

Sei Σ ein Alphabet. Wir definieren die Funktion $|\cdot| : \Sigma^* \to \mathbb{N}$ rekursiv über die rekursive Definition von Σ^* aus Beispiel 3.9.

Rekursionsanfang. $|\varepsilon| := 0$.

Rekursionsschritt. Für alle $w \in \Sigma^*$ und $a \in \Sigma$:

$$|wa| := |w| + 1.$$

Beispiel 3.17 (Länge einer Liste)

Wir definieren die Funktion $\mathrm{lg} : \texttt{LIST(BIN)} \to \mathbb{N}$ rekursiv über den Aufbau von `LIST(BIN)`:

Rekursionsanfang. $\mathrm{lg}(\texttt{nil}) := 0$.

Beispiel 3.16 (Länge eines Wortes)

Sei Σ ein Alphabet. Wir definieren die Funktion $|\cdot| : \Sigma^* \to \mathbb{N}$ rekursiv über die rekursive Definition von Σ^* aus Beispiel 3.9.

Rekursionsanfang. $|\varepsilon| := 0$.

Rekursionsschritt. Für alle $w \in \Sigma^*$ und $a \in \Sigma$:

$$|wa| := |w| + 1.$$

Beispiel 3.17 (Länge einer Liste)

Wir definieren die Funktion $\mathrm{lg} : \texttt{LIST(BIN)} \to \mathbb{N}$ rekursiv über den Aufbau von `LIST(BIN)`:

Rekursionsanfang. $\mathrm{lg}(\texttt{nil}) := 0$.

Rekursionsschritt. Für alle $w \in \texttt{LIST(BIN)}$ und $v \in \texttt{BIN}$:

$$\mathrm{lg}\left(\texttt{cons}(v, w)\right) := \mathrm{lg}(w) + 1.$$

Beispiel 3.18 (Wert einer Binärzahl)

Wir definieren die Funktion $\mathrm{val}_{\mathtt{BIN}} : \mathtt{BIN} \to \mathbb{N}$ rekursiv über den Aufbau von `BIN`:

Beispiel 3.18 (Wert einer Binärzahl)

Wir definieren die Funktion $\mathrm{val}_{\mathtt{BIN}} : \mathtt{BIN} \to \mathbb{N}$ rekursiv über den Aufbau von $\mathtt{BIN}$:

Rekursionsanfang. $\mathrm{val}_{\mathtt{BIN}}(0) := 0$, $\mathrm{val}_{\mathtt{BIN}}(1) := 1$.

Beispiel 3.18 (Wert einer Binärzahl)

Wir definieren die Funktion $\mathrm{val}_{\mathtt{BIN}} : \mathtt{BIN} \to \mathbb{N}$ rekursiv über den Aufbau von $\mathtt{BIN}$:

Rekursionsanfang. $\mathrm{val}_{\mathtt{BIN}}(0) := 0$, $\mathrm{val}_{\mathtt{BIN}}(1) := 1$.

Rekursionsschritt. Für alle $v \in \mathtt{BIN} \setminus \{0\}$:

$$\mathrm{val}_{\mathtt{BIN}}(v0) := 2\,\mathrm{val}_{\mathtt{BIN}}(v),$$
$$\mathrm{val}_{\mathtt{BIN}}(v1) := 2\,\mathrm{val}_{\mathtt{BIN}}(v) + 1.$$

Beispiel 3.18 (Wert einer Binärzahl)

Wir definieren die Funktion $\mathrm{val}_{\mathtt{BIN}} : \mathtt{BIN} \to \mathbb{N}$ rekursiv über den Aufbau von $\mathtt{BIN}$:

Rekursionsanfang. $\mathrm{val}_{\mathtt{BIN}}(0) := 0$, $\mathrm{val}_{\mathtt{BIN}}(1) := 1$.

Rekursionsschritt. Für alle $v \in \mathtt{BIN} \setminus \{0\}$:

$$\mathrm{val}_{\mathtt{BIN}}(v0) := 2\,\mathrm{val}_{\mathtt{BIN}}(v),$$
$$\mathrm{val}_{\mathtt{BIN}}(v1) := 2\,\mathrm{val}_{\mathtt{BIN}}(v) + 1.$$

Beispiel 3.19 (Summation über eine Liste)

Wir definieren die Funktion $\mathrm{sum} : \mathtt{LIST(BIN)} \to \mathbb{N}$ rekursiv über den Aufbau von $\mathtt{LIST(BIN)}$:

Beispiel 3.18 (Wert einer Binärzahl)

Wir definieren die Funktion $\mathrm{val}_{\mathtt{BIN}} : \mathtt{BIN} \to \mathbb{N}$ rekursiv über den Aufbau von $\mathtt{BIN}$:

Rekursionsanfang. $\mathrm{val}_{\mathtt{BIN}}(0) := 0$, $\mathrm{val}_{\mathtt{BIN}}(1) := 1$.

Rekursionsschritt. Für alle $v \in \mathtt{BIN} \setminus \{0\}$:

$$\mathrm{val}_{\mathtt{BIN}}(v0) := 2\,\mathrm{val}_{\mathtt{BIN}}(v),$$
$$\mathrm{val}_{\mathtt{BIN}}(v1) := 2\,\mathrm{val}_{\mathtt{BIN}}(v) + 1.$$

Beispiel 3.19 (Summation über eine Liste)

Wir definieren die Funktion $\mathrm{sum} : \mathtt{LIST(BIN)} \to \mathbb{N}$ rekursiv über den Aufbau von $\mathtt{LIST(BIN)}$:

Rekursionsanfang. $\mathrm{sum}(\mathtt{nil}) := 0$.

Beispiel 3.18 (Wert einer Binärzahl)

Wir definieren die Funktion $\mathrm{val}_{\mathtt{BIN}} : \mathtt{BIN} \to \mathbb{N}$ rekursiv über den Aufbau von $\mathtt{BIN}$:

Rekursionsanfang. $\mathrm{val}_{\mathtt{BIN}}(0) := 0$, $\mathrm{val}_{\mathtt{BIN}}(1) := 1$.

Rekursionsschritt. Für alle $v \in \mathtt{BIN} \setminus \{0\}$:

$$\mathrm{val}_{\mathtt{BIN}}(v0) := 2\,\mathrm{val}_{\mathtt{BIN}}(v),$$
$$\mathrm{val}_{\mathtt{BIN}}(v1) := 2\,\mathrm{val}_{\mathtt{BIN}}(v) + 1.$$

Beispiel 3.19 (Summation über eine Liste)

Wir definieren die Funktion $\mathrm{sum} : \mathtt{LIST(BIN)} \to \mathbb{N}$ rekursiv über den Aufbau von $\mathtt{LIST(BIN)}$:

Rekursionsanfang. $\mathrm{sum}(\mathtt{nil}) := 0$.

Rekursionsschritt. Für alle $w \in \mathtt{LIST(BIN)}$ und $v \in \mathtt{BIN}$:

$$\mathrm{sum}\left(\mathtt{cons}(v, w)\right) := \mathrm{val}_{\mathtt{BIN}}(v) + \mathrm{sum}(w).$$

Beispiel 3.20: Arithmetische Terme

Die Menge $\texttt{ART} \subseteq \Sigma^*_{\text{ASCII}}$ der arithmetischen Terme sei rekursiv wie folgt definiert:

Beispiel 3.20: Arithmetische Terme

Die Menge $\texttt{ART} \subseteq \Sigma^*_{\text{ASCII}}$ der arithmetischen Terme sei rekursiv wie folgt definiert:

Basisregeln.
Für alle $v \in \texttt{BIN}$:

$$v \in \texttt{ART}$$

Beispiel 3.20: Arithmetische Terme

Die Menge $\texttt{ART} \subseteq \Sigma^*_{\text{ASCII}}$ der arithmetischen Terme sei rekursiv wie folgt definiert:

Basisregeln.
Für alle $v \in \texttt{BIN}$:

$$v \in \texttt{ART}$$

Rekursive Regeln.
Für alle $t, u \in \Sigma^*_{\text{ASCII}}$:

- Wenn $t \in \texttt{ART}$, dann $\texttt{-}t \in \texttt{ART}$.
- Wenn $t, u \in \texttt{ART}$, dann $(t\texttt{+}u) \in \texttt{ART}$.
- Wenn $t, u \in \texttt{ART}$, dann $(t\texttt{*}u) \in \texttt{ART}$.
- Wenn $t, u \in \texttt{ART}$, dann $(t\texttt{-}u) \in \texttt{ART}$.

Beispiel 3.20: Arithmetische Terme

Die Menge $\texttt{ART} \subseteq \Sigma^*_{\text{ASCII}}$ der arithmetischen Terme sei rekursiv wie folgt definiert:

Basisregeln.

Für alle $v \in \texttt{BIN}$:

$$v \in \texttt{ART}$$

Rekursive Regeln.

Für alle $t, u \in \Sigma^*_{\text{ASCII}}$:

- Wenn $t \in \texttt{ART}$, dann $\texttt{-}t \in \texttt{ART}$.
- Wenn $t, u \in \texttt{ART}$, dann $(t\texttt{+}u) \in \texttt{ART}$.
- Wenn $t, u \in \texttt{ART}$, dann $(t\texttt{*}u) \in \texttt{ART}$.
- Wenn $t, u \in \texttt{ART}$, dann $(t\texttt{-}u) \in \texttt{ART}$.

Beispiele

$$\begin{aligned} &\texttt{0}, \quad \texttt{---1}, \quad \texttt{((-0+(-(1+101)*11))*1001)} \quad \in \texttt{ART}, \\ &\qquad \texttt{+1}, \quad \texttt{(-0+(-(1+101)*11))*1001} \quad \notin \texttt{ART}. \end{aligned}$$

Beispiel 3.20 (Forts.)

Wir definieren die Funktion $\mathrm{val}_{\mathtt{ART}} : \mathtt{ART} \to \mathbb{Z}$ rekursiv über den Aufbau von $\mathtt{ART}$:

Beispiel 3.20 (Forts.)

Wir definieren die Funktion $\mathrm{val}_{\mathtt{ART}} : \mathtt{ART} \to \mathbb{Z}$ rekursiv über den Aufbau von $\mathtt{ART}$:

Rekursionsanfang. Für alle $v \in \mathtt{BIN}$:

$$\mathrm{val}_{\mathtt{ART}}(v) := \mathrm{val}_{\mathtt{BIN}}(v).$$

Beispiel 3.20 (Forts.)

Wir definieren die Funktion $\mathrm{val}_{\mathtt{ART}} : \mathtt{ART} \to \mathbb{Z}$ rekursiv über den Aufbau von $\mathtt{ART}$:

Rekursionsanfang. Für alle $v \in \mathtt{BIN}$:

$$\mathrm{val}_{\mathtt{ART}}(v) := \mathrm{val}_{\mathtt{BIN}}(v).$$

Rekursionsschritt. Für alle $t, u \in \mathtt{ART}$:

$$\begin{aligned}
\mathrm{val}_{\mathtt{ART}}(\mathtt{-}t) &:= -\,\mathrm{val}_{\mathtt{ART}}(t), \\
\mathrm{val}_{\mathtt{ART}}\big((t\mathtt{+}u)\big) &:= \mathrm{val}_{\mathtt{ART}}(t) + \mathrm{val}_{\mathtt{ART}}(u), \\
\mathrm{val}_{\mathtt{ART}}\big((t\mathtt{*}u)\big) &:= \mathrm{val}_{\mathtt{ART}}(t) \cdot \mathrm{val}_{\mathtt{ART}}(u), \\
\mathrm{val}_{\mathtt{ART}}\big((t\mathtt{-}u)\big) &:= \mathrm{val}_{\mathtt{ART}}(t) - \mathrm{val}_{\mathtt{ART}}(u).
\end{aligned}$$

Beispiel 3.20 (Forts.)

Es gilt

$$\begin{aligned}\mathrm{val}_{\mathtt{BIN}}(101) &= 2\,\mathrm{val}_{\mathtt{BIN}}(10) + 1\\ &= 2(2\,\mathrm{val}_{\mathtt{BIN}}(1)) + 1\\ &= 2(2 \cdot 1) + 1 = 5.\end{aligned}$$

Ähnlich $\mathrm{val}_{\mathtt{BIN}}(1) = 1$ und $\mathrm{val}_{\mathtt{BIN}}(1101) = 13$.

Beispiel 3.20 (Forts.)

Es gilt

$$\begin{aligned}\mathrm{val}_{\mathtt{BIN}}(101) &= 2\,\mathrm{val}_{\mathtt{BIN}}(10) + 1\\ &= 2(2\,\mathrm{val}_{\mathtt{BIN}}(1)) + 1\\ &= 2(2 \cdot 1) + 1 = 5.\end{aligned}$$

Ähnlich $\mathrm{val}_{\mathtt{BIN}}(1) = 1$ und $\mathrm{val}_{\mathtt{BIN}}(1101) = 13$.

Daraus ergibt sich

$$\begin{aligned}\mathrm{val}_{\mathtt{ART}}\big(\mathtt{-(101*(1-1101))}\big) &= -\,\mathrm{val}_{\mathtt{ART}}\big(\mathtt{(101*(1-1101))}\big)\\ &= -\,\mathrm{val}_{\mathtt{ART}}(101) \cdot \mathrm{val}_{\mathtt{ART}}\big(\mathtt{(1-1101)}\big)\\ &= -\,\mathrm{val}_{\mathtt{BIN}}(101) \cdot \big(\mathrm{val}_{\mathtt{ART}}(1) - \mathrm{val}_{\mathtt{ART}}(1101)\big)\\ &= -\,\mathrm{val}_{\mathtt{BIN}}(101) \cdot \big(\mathrm{val}_{\mathtt{BIN}}(1) - \mathrm{val}_{\mathtt{BIN}}(1101)\big)\\ &= -5 \cdot (1 - 13) = 60.\end{aligned}$$

Abschnitt 3.2

Induktive Beweise

Induktionsprinzip für rekursiv definierte Mengen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.

Wir wollen für alle $x \in X$ eine Aussage $A(x)$ beweisen.

Induktionsprinzip für rekursiv definierte Mengen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.

Wir wollen für alle $x \in X$ eine Aussage $A(x)$ beweisen.

Ein Beweis per (vollständiger) Induktion über $\mathcal{D}$ funktioniert wie folgt:

Induktionsprinzip für rekursiv definierte Mengen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.

Wir wollen für alle $x \in X$ eine Aussage $A(x)$ beweisen.

Ein Beweis per (vollständiger) Induktion über $\mathcal{D}$ funktioniert wie folgt:

Induktionsanfang.
Für alle Basisregeln „$x \in X$“ in $\mathcal{D}$, beweise $A(x)$.

Induktionsprinzip für rekursiv definierte Mengen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.

Wir wollen für alle $x \in X$ eine Aussage $A(x)$ beweisen.

Ein Beweis per (vollständiger) Induktion über $\mathcal{D}$ funktioniert wie folgt:

Induktionsanfang.
Für alle Basisregeln „$x \in X$“ in $\mathcal{D}$, beweise $A(x)$.

Induktionsschritt.
Für alle rekursiven Regeln „Wenn $x_1, \ldots, x_k \in X$, dann $x \in X$“, beweise $A(x)$ unter Verwendung der Induktionsannahmen $A(x_1), \ldots, A(x_k)$.

Induktionsprinzip für rekursiv definierte Mengen

Sei $\mathcal{D}$ eine rekursive Definition einer Menge X.

Wir wollen für alle $x \in X$ eine Aussage $A(x)$ beweisen.

Ein Beweis per (vollständiger) Induktion über $\mathcal{D}$ funktioniert wie folgt:

Induktionsanfang.
Für alle Basisregeln „$x \in X$“ in $\mathcal{D}$, beweise $A(x)$.

Induktionsschritt.
Für alle rekursiven Regeln „Wenn $x_1, \ldots, x_k \in X$, dann $x \in X$“, beweise $A(x)$ unter Verwendung der Induktionsannahmen $A(x_1), \ldots, A(x_k)$.

Oft geben wir die rekursive Definition $\mathcal{D}$ von X nicht explizit an, sondern nehmen an, dass sie aus dem Kontext bekannt ist. Wir sprechen dann auch von einem Beweis per Induktion über den Aufbau von X.

Rechtfertigung

Das Induktionsprinzip für rekursiv definierte Mengen lässt sich auf das Induktionsprinzip für die natürlichen Zahlen zurückführen, indem man induktive Beweise über den Aufbau einer rekursiven Definition $\mathcal{D}$ zurückführt auf induktive Beweise über die Länge einer Ableitung in $\mathcal{D}$.

Zur Erinnerung hier noch einmal die rekursive Definition der Menge L_K der korrekten Klammerausdrücke über dem Alphabet $\Sigma = \{(,)\}$.

Basisregel.

$$\varepsilon \in L_K.$$

Rekursive Regeln.
Für alle $k \in \mathbb{N}_+$ und alle $w_1, \ldots, w_k \in \Sigma^*$:

$$\text{Wenn } w_1, \ldots, w_k \in L_K, \text{ dann } (w_1 w_2 \ldots w_k) \in L_K.$$

Wir definieren zwei Funktionen $\mathsf{auf}, \mathsf{zu} : \Sigma^* \to \mathbb{N}$ durch

$$\mathsf{auf}(w) := |w|_(\quad \text{und} \quad \mathsf{zu}(w) := |w|_).$$

Alternativ können wir die Funktionen rekursiv definieren durch

$$\begin{aligned} \mathsf{auf}(\varepsilon) &:= 0, & \mathsf{zu}(\varepsilon) &:= 0, \\ \mathsf{auf}(w() &:= \mathsf{auf}(w) + 1, & \mathsf{zu}(w() &:= \mathsf{zu}(w), \\ \mathsf{auf}(w)) &:= \mathsf{auf}(w), & \mathsf{zu}(w)) &:= \mathsf{zu}(w) + 1. \end{aligned}$$

Für $v, w \in \Sigma^*$ schreiben wir $v \sqsubset w$ („v ist echtes Präfix von w“), wenn $v \sqsubseteq w$ und $v \neq w$.

Satz 3.22

$L_K = \{w \in \Sigma^* \mid \mathsf{auf}(w) = \mathsf{zu}(w)$ *und* $\mathsf{auf}(v) > \mathsf{zu}(v)$ *für alle* $v \sqsubset w$ *mit* $v \neq \varepsilon\}$.

Beweis. „$\subseteq$“: Per Induktion über den Aufbau von L_K zeigen wir für alle $w \in L_K$:

$$\mathsf{auf}(w) = \mathsf{zu}(w) \text{ und } \mathsf{auf}(v) > \mathsf{zu}(v) \text{ für alle } v \sqsubset w \text{ mit } v \neq \varepsilon \qquad (\star)$$

Induktionsanfang: $w = \varepsilon$
Es gilt $\mathsf{auf}(\varepsilon) = \mathsf{zu}(\varepsilon) = 0$, und weil ε keine echten Präfixe hat, auch $\mathsf{auf}(v) > \mathsf{zu}(v)$ für all $v \sqsubset w$.

Induktionsschritt: $w = (w_1 \ldots w_k)$ für ein $k \geq 1$ und $w_1, \ldots, w_k \in L_K$.
Nach Induktionsannahme gilt $(\star)$ für alle w_i.
Damit

$$\mathsf{auf}(w) = 1 + \sum_{i=1}^{k} \mathsf{auf}(w_i) = \sum_{i=1}^{k} \mathsf{zu}(w_i) + 1 = \mathsf{zu}(w).$$

Sein nun $v \sqsubset w$ mit $v \neq \varepsilon$. Dann gibt es ein $i \leq k$, so dass

$$(w_1 \ldots w_{i-1} \sqsubseteq v \sqsubseteq (w_1 \ldots w_{i-1} w_i$$

und damit ein $v' \sqsubseteq w_i$, so dass

$$v = (w_1 \ldots w_{i-1} v'.$$

Nach Induktionsannahme gilt $\mathsf{auf}(v') \geq \mathsf{zu}(v')$. Also

$$\begin{aligned} \mathsf{auf}(v) &= 1 + \sum_{j=1}^{i-1} \mathsf{auf}(w_j) + \mathsf{auf}(v') \\ &\geq 1 + \sum_{j=1}^{i-1} \mathsf{zu}(w_j) + \mathsf{zu}(v') \\ &> \sum_{j=1}^{i-1} \mathsf{zu}(w_j) + \mathsf{zu}(v') \\ &= \mathsf{zu}(v). \end{aligned}$$

„$\supseteq$“: Per Induktion über $n := |w|$ zeigen wir für alle $w \in \Sigma^*$:

$$\text{Wenn } \mathsf{auf}(w) = \mathsf{zu}(w) \text{ und } \mathsf{auf}(v) > \mathsf{zu}(v) \text{ für alle } v \sqsubset w \text{ mit } v \neq \varepsilon, \text{ dann } w \in L_K. \qquad (\star\star)$$

Induktionsanfang: $n = 0$.
Dann ist $w = \varepsilon \in L_K$.

Induktionsschritt: $n > 0$.
Sei $w = a_1 \ldots a_n \in \Sigma^*$, so dass
$\mathsf{auf}(w) = \mathsf{zu}(w)$ und $\mathsf{auf}(v) > \mathsf{zu}(v)$ für all $v \sqsubset w$ mit $v \neq \varepsilon$.
Induktionsannahme: $(\star\star)$ gilt für alle $w' \in \Sigma^*$ mit $|w'| < n$.
Behauptung: $w \in L_K$.

- Es gilt $a_1 = ($, weil $a_1 \sqsubseteq w$ und damit $\mathsf{auf}(a_1) \geq \mathsf{zu}(a_1)$.
- Es gilt $a_n =)$, weil $\mathsf{auf}(a_1 \ldots a_{n-1}) > \mathsf{zu}(a_1 \ldots a_{n-1})$ und $\mathsf{auf}(a_1 \ldots a_n) = \mathsf{zu}(a_1 \ldots a_n)$.
- Seien $1 \leq i_0 < i_1 < \ldots < i_k \leq n$ so, dass für $0 \leq j \leq k$

  $$\mathsf{auf}(a_1 \ldots a_{i_j}) = \mathsf{zu}(a_1 \ldots a_{i_j}) + 1$$

  und für alle $i \in \{1, \ldots, n\} \setminus \{i_0, \ldots, i_k\}$

  $$\mathsf{auf}(a_1 \ldots a_i) > \mathsf{zu}(a_1 \ldots a_i) + 1.$$

  Dann gilt $i_0 = 1$ und $i_k = n - 1$. Für $1 \leq j \leq k$ sei

  $$w_j := a_{i_{j-1}+1} \ldots a_{i_j}.$$

  Dann gilt

  $$w = (w_1 \ldots w_k).$$

  Für $1 \leq j \leq k$ gilt

  $$\begin{aligned} \mathsf{auf}(w_j) &= \mathsf{auf}(a_1 \ldots a_{i_j}) - \mathsf{auf}(a_1 \ldots a_{i_{j-1}}) \\ &= \big(\mathsf{zu}(a_1 \ldots a_{i_j}) + 1\big) - \big(\mathsf{zu}(a_1 \ldots a_{i_{j-1}}) + 1\big) \\ &= \mathsf{zu}(a_1 \ldots a_{i_j}) - \mathsf{zu}(a_1 \ldots a_{i_{j-1}}) \\ &= \mathsf{zu}(w_j). \end{aligned}$$

  Sei nun $v \sqsubset w_j$ mit $v \neq \varepsilon$. Dann existiert ein $i \in \{i_{j-1} + 1, \ldots, i_j - 1\}$, so dass $v = a_{i_{j-1}+1} \ldots a_i$. Es gilt

  $$\begin{aligned} \mathsf{auf}(v) &= \mathsf{auf}(a_1 \ldots a_i) - \mathsf{auf}(a_1 \ldots a_{i_{j-1}}) \\ &> \big(\mathsf{zu}(a_1 \ldots a_i) + 1\big) - \big(\mathsf{zu}(a_1 \ldots a_{i_{j-1}}) + 1\big) \\ &= \mathsf{zu}(a_1 \ldots a_i) - \mathsf{zu}(a_1 \ldots a_{i_{j-1}}) \\ &= \mathsf{zu}(v). \end{aligned}$$

  Nach Induktionsannahme gilt dann $w_j \in L_K$.

  Damit auch $w = (w_1 \ldots w_k) \in L_K$ nach der rekursiven Regel der Definition von L_K.

□

Wir definieren rekursiv eine Nachfolgerfunktion $N : \mathtt{BIN} \to \Sigma_{\mathrm{ASCII}}^*$:

Rekursionsanfang. $N(0) := 1$, $N(1) := 10$.

Rekursionsschritt. Für alle $v \in \mathtt{BIN} \setminus \{0\}$:

$$N(v0) := v1,$$
$$N(v1) := N(v)0.$$

Behauptung 1

Für alle $v \in \mathtt{BIN}$ *gilt* $N(v) \in \mathtt{BIN} \setminus \{0\}$ *und* $\mathrm{val}_{\mathtt{BIN}}(N(v)) = \mathrm{val}_{\mathtt{BIN}}(v) + 1$.

Beweis.

Induktion über den Aufbau von BIN.

Induktionsanfang.

$v = 0$: $N(0) = 1 \in \mathtt{BIN} \setminus \{0\}$ und $\mathrm{val}_{\mathtt{BIN}}(1) = 1 = \mathrm{val}_{\mathtt{BIN}}(0) + 1$.

$v = 1$: $N(1) = 10 \in \mathtt{BIN} \setminus \{0\}$ und $\mathrm{val}_{\mathtt{BIN}}(10) = 2 = \mathrm{val}_{\mathtt{BIN}}(1) + 1$.

Induktionsschritt.

$v = v'0$ für ein $v' \in \mathtt{BIN} \setminus \{0\}$: Dann ist $N(v) = v'1 \in \mathtt{BIN} \setminus \{0\}$, und es gilt

$$\begin{aligned}\mathrm{val}_{\mathtt{BIN}}(N(v)) &= \mathrm{val}_{\mathtt{BIN}}(v'1)\\ &= 2\,\mathrm{val}_{\mathtt{BIN}}(v') + 1\\ &= \mathrm{val}_{\mathtt{BIN}}(v'0) + 1\\ &= \mathrm{val}_{\mathtt{BIN}}(v) + 1.\end{aligned}$$

$v = v'1$ für ein $v' \in \mathtt{BIN} \setminus \{0\}$: Dann ist $N(v) = N(v')0$.

Nach Induktionsannahme ist $N(v') \in \mathtt{BIN} \setminus \{0\}$, also ist $N(v) \in \mathtt{BIN} \setminus \{0\}$.
Außerdem gilt

$$\begin{aligned}\mathrm{val}_{\mathtt{BIN}}(N(v)) &= \mathrm{val}_{\mathtt{BIN}}(N(v')0)\\ &= 2\,\mathrm{val}_{\mathtt{BIN}}(N(v'))\\ &= 2\big(\mathrm{val}_{\mathtt{BIN}}(v') + 1\big)\\ &= \big(2\,\mathrm{val}_{\mathtt{BIN}}(v') + 1\big) + 1\\ &= \mathrm{val}_{\mathtt{BIN}}(v) + 1.\end{aligned}$$

□

Als nächstes definieren wir eine Additionsfunktion $A : \mathtt{BIN} \times \mathtt{BIN} \to \Sigma_{\mathrm{ASCII}}^*$. Wir verallgemeinern hier unser Rekursionsschema auf zweistellige Funktionen, indem wir die Rekursion verschachteln.

Rekursionsanfang.

- Definition $A(0, v)$ für alle $v \in \mathtt{BIN}$:
$$A(0, v) := v.$$
- Definition $A(1, v)$ für alle $v \in \mathtt{BIN}$:
$$A(1, v) := N(v).$$

Rekursionsschritt. Für alle $u' \in \mathtt{BIN} \setminus \{0\}$:

- Rekursive Definition $A(u'0, v)$ für alle $v \in \mathtt{BIN}$.

  Rekursionsanfang.
  $$A(u'0, 0) := u'0, \qquad A(u'0, 1) := u'1.$$

  Rekursionsschritt. Für alle $v' \in \mathtt{BIN} \setminus \{0\}$:
  $$A(u'0, v'0) := A(u', v')0,$$
  $$A(u'0, v'1) := A(u', v')1.$$
- Rekursive Definition $A(u'1, v)$ für alle $v \in \mathtt{BIN}$.

  Rekursionsanfang.
  $$A(u'1, 0) := u'1, \qquad A(u'1, 1) := N(u'1).$$

  Rekursionsschritt. Für alle $v' \in \mathtt{BIN} \setminus \{0\}$:
  $$A(u'1, v'0) := A(u', v')1,$$
  $$A(u'1, v'1) := N(A(u', v'))0.$$

Behauptung 2

Für alle $u, v \in \mathtt{BIN}$ *gilt* $A(u, v) \in \mathtt{BIN}$ *und*

$$\mathrm{val}_{\mathtt{BIN}}(A(u, v)) = \mathrm{val}_{\mathtt{BIN}}(u) + \mathrm{val}_{\mathtt{BIN}}(v).$$

Wir verzichten auf einen Beweis.